HOPEX Customization

User Guide

MEGA

a Bizzdesign company

HOPEX Aquila 6.2

HOPEX Customization Desktop

WORKING IN HOPEX CUSTOMIZING DESKTOP

This section presents the **HOPEX Customizing** desktop and explains how to customize this desktop.

The points covered are:

Connecting to HOPEX Customizing Desktop

The **HOPEX Customizing** desktop (Windows Front-End) is dedicated to **HOPEX** product customization.

Profiles

Profiles dedicated to customization are:

- **HOPEX Customizer**
  The HOPEX Customizer profile gives access to all HOPEX Power Studio features. It enables to customize the following large sets of elements:
  - Metamodel
  - User Interfaces
  - Documents
- **HOPEX Customizer Publisher**
  The HOPEX Customizer Publisher enables to customize:
  - Documents

> ☛ *For detailed information regarding customization, see HOPEX Power Studio documentation.*

Accessing HOPEX Customizing Desktop

You can start **HOPEX** from your workstation:

- directly with **HOPEX** installed on your machine.
- via a server or another machine to which you can connect.

When you connect to **HOPEX**, you can:

- create a private workspace (if you do not already have one).

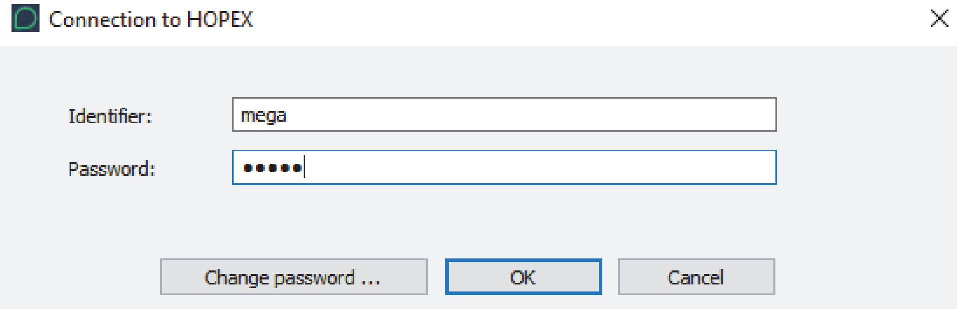
  > ☛ *You can only have one private workspace open in the same environment.*

  > ☛ *The private workspace is open for each user on both his/her repository and on the system repository. A private workspace always exists on system repository even if the user is not using any of the system repository objects.*
- resume work in your private workspace

To connect to **HOPEX Customizing** desktop:

1. Double-click the **HOPEX.exe** file .
   The connection dialog box appears.
2. In the **Identifier** field, enter your identifier.

3. In the **Password** field, enter your password.



Connection to HOPEX

Identifier: mega

Password: •••••

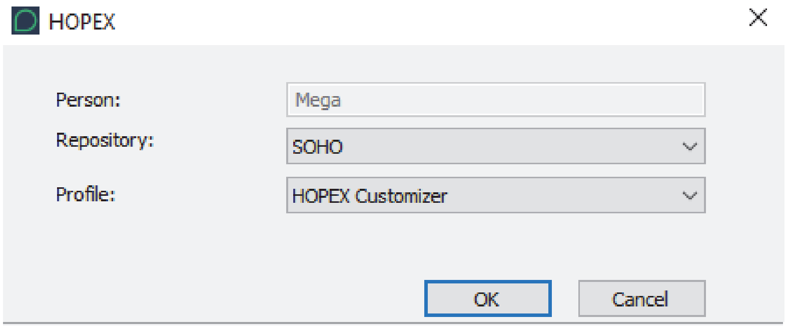
4. Click **OK**.
   You are authenticated, your name appears in the **Person** field.
5. In the **Repository** field, select your work repository.

   ☛ *If you already have a private workspace open, the repository is automatically selected and this field is grayed. To change repository, you must first dispatch or discard your current private workspace.*

   ☛ *The drop-down menu* ▾ *opens the list of repositories available in the environment.*

   ☛ *If you can access only one repository, this is automatically taken into account.*
6. In the **Profile** field, click the arrow and select the profile with which you want to work.

   ☛ *If you have only one profile available, this is automatically taken into account.*



HOPEX

Person: Mega

Repository: SOHO

Profile: HOPEX Customizer

7. Click **OK**.
   Your desktop appears.

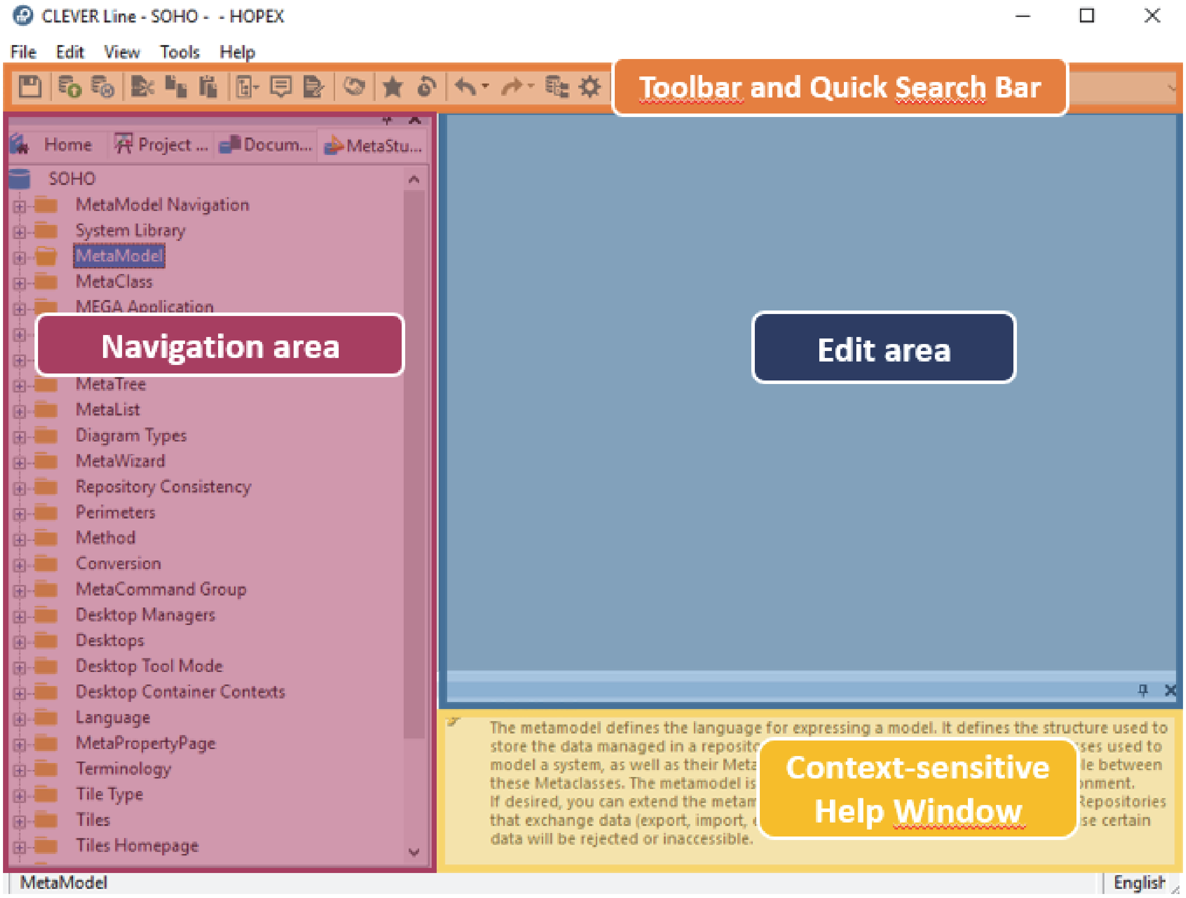
HOPEX Customizing Desktop Description

By default, the **HOPEX** Customizing desktop (Windows Front-End) includes the following elements:

- Toolbar and Quick Search Bar
- Navigation Area
- Edit Area

You can also display the following information:

- Help Window
- Properties Window
- Comment Window



The metamodel defines the language for expressing a model. It defines the structure used to store the data managed in a reposito ses used to model a system, as well as their Meta le between these Metaclasses. The metamodel is onment.
If desired, you can extend the metam Repositories that exchange data (export, import, e se certain data will be rejected or inaccessible.

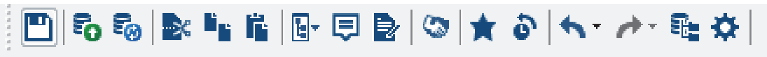
☛ *The object types available depend on the **HOPEX** products you have at your disposal.*

☛ *For more details on organizing and customizing desktop windows, see Handling HOPEX Customizing Desktop.*

Toolbar and Quick Search Bar

The toolbar enables fast access to:

- main objects.
- certain actions available from the desktop.



The search bar enables quick searches in the entire repository.

☛ *See Quick Search.*

The following table describes the toolbar buttons.

Icon	Definition	Description
💾	**Save All**	Saves all your actions in your desktop Shortcut of **File > Save All**
	Dispatch	Dispatches your work in the **HOPEX** repository. This action allows other users to access your work Shortcut of **File > Dispatch**
	Refresh	Updates your desktop with the latest modifications made by other repository users Shortcut of **File > Refresh**
✂	**Cut**	Cuts an object in the navigation or edit areas Shortcut of **Edit > Cut**
	Copy	Copies an object in the navigation or edit areas Shortcut of **Edit > Copy**
	Paste	Pastes an object in the edit area Shortcut of **Edit > Paste**
	Navigation window	Opens a new navigation window Shortcut of **View > Navigation Windows** See Navigation Area.
	Properties	Opens the properties window of the selected object Shortcut of **View > Edit Windows > Properties** See Properties Window.
	Comment	Opens the comment window in the edit area Shortcut of **View > Edit Windows > Comment** See Comment Window.
★	**Favorites**	Adds the **Favorites** tab in the navigation area Shortcut of **View > Favorites** See Managing Favorites.

Icon	Definition	Description
🕒	**Deep history**	Adds the **History** tab of actions executed in **HOPEX** in the navigation area Shortcut of **View > History** See Using the History.
↶	**Undo**	Cancels chronologically the latest actions executed. You can delete actions singly or several at one time Shortcut of **Edit > Undo** See Undoing or Redoing Actions on Objects.
↷	**Redo**	Restores chronologically the latest actions canceled. You can redo actions singly or several at one time Shortcut of **Edit > Redo** See Undoing or Redoing Actions on Objects.
▦	**Explore**	Opens the explore window Shortcut of **Diagram > Explore** See The Explorer.
⚙	**Options**	Opens the options window of the current user Shortcut of **Tools > Options**
🔍	**Query**	Open the advanced query tool. Shortcut of **Tools > Query** See The Advanced Search (Windows Front-End).

Navigation Area

The navigation area contains tabs that allow you to navigate in the repository and easily access objects.

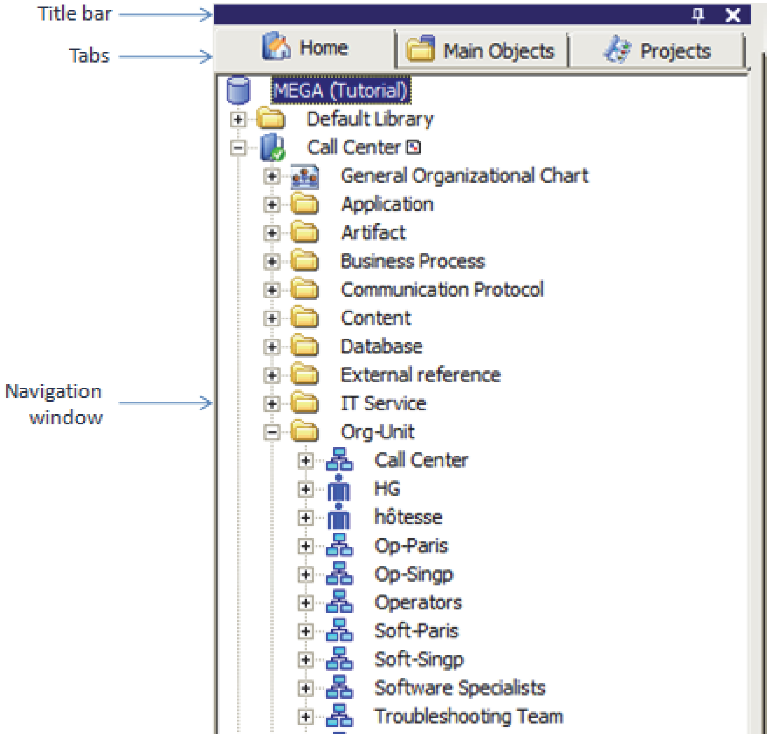
The navigation area comprises:

- an active navigation window, including the tree used to access repository objects.
- tabs, which allow navigation in the navigation area windows.

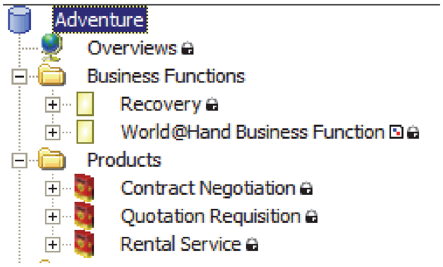
  ☛ *At first connection, the **Home** and **Projects** tabs are present by default. You can add or delete tabs according to your needs.*

- a title bar, which enables organization of the display of windows in the desktop.

  ☛ *You can for example move or dock a window.*



At access to the repository in read-only, a padlock alongside an object indicates that it cannot be modified.



Edit Area

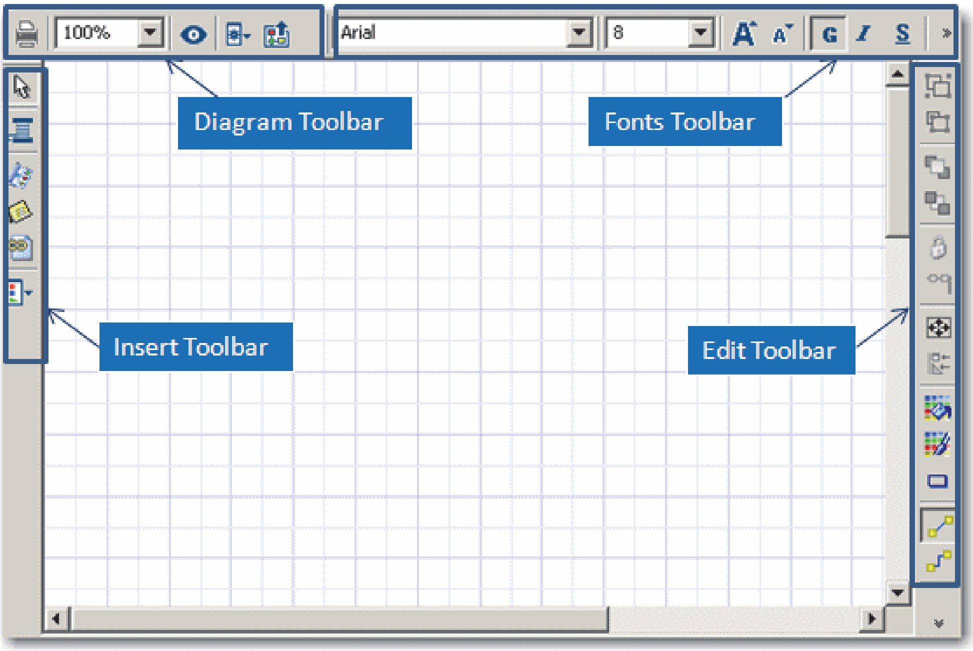
The edit area is mainly used to open and edit diagrams.

The start page in the edit area allows you to:

- build your diagrams using the diagram creation wizard.
- create your projects based on methods proposed by **HOPEX**.
- rapidly access the different **HOPEX** modules.

> ☛ *At first connection, the edit area opens on the home page. To reopen the home page, in the* ***HOPEX*** *menu bar, select* ***?. > Start Page****.*

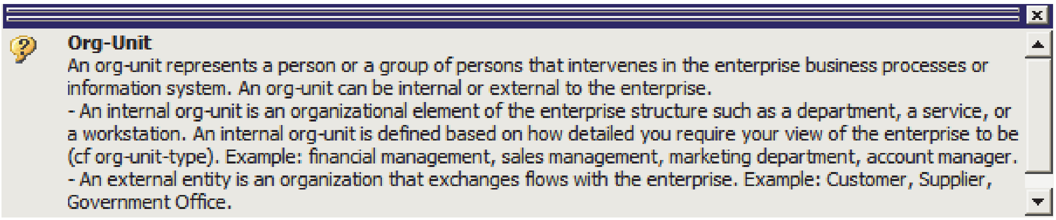
Here is the diagram editor:



Help Window

The context-sensitive help window provides basic information on each concept or repository object type selected.

To display the context-sensitive help window, in the **HOPEX Windows Front-End** menu bar, select **View > Help Window**.



Org-Unit
An org-unit represents a person or a group of persons that intervenes in the enterprise business processes or information system. An org-unit can be internal or external to the enterprise.
- An internal org-unit is an organizational element of the enterprise structure such as a department, a service, or a workstation. An internal org-unit is defined based on how detailed you require your view of the enterprise to be (cf org-unit-type). Example: financial management, sales management, marketing department, account manager.
- An external entity is an organization that exchanges flows with the enterprise. Example: Customer, Supplier, Government Office.

☛ *The title bar enables customization of window display in the desktop. You can for example move or dock a window. For more details on window display, see Positioning Windows.*

Properties Window

The properties window presents all the information relating to the selected object. It allows you to consult and modify information on this object.

To display the properties window:

- Click **Properties** or select **View > Edit Windows > Properties**.

☺ *Displaying the comment window is useful when you want to successively enter the comments of several objects.*
So, when you select an object in a navigation window, in a diagram or in a query result dialog box, you can easily modify its properties in this window docked in the workspace, then move on to those of another object.
This saves you having to activate the pop-up menu of the object each time you wish to access its properties.

For more details on object properties, see Defining Object Properties.

Comment Window

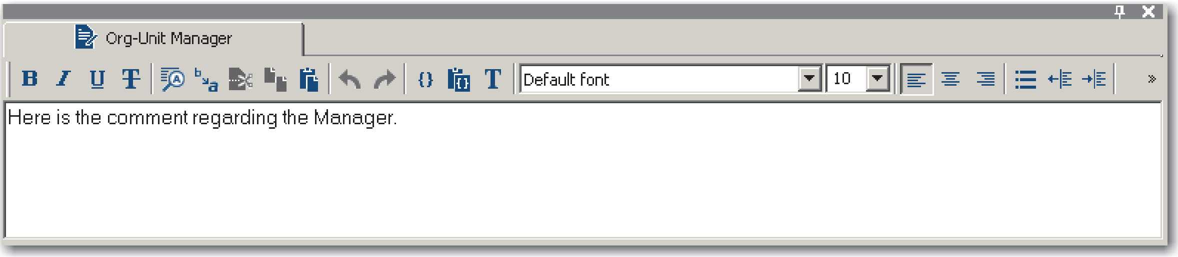
The comment window enables specification of information on the selected object.

To display the comment window:

- Click **Comments** or select **View > Edit Windows > Comment**.

☺ *Displaying the comment window is useful when you want to successively enter the comments of several objects.*
When you click an object in a navigation window, in a diagram, or in a query result dialog box, the comment of the selected (active) object appears in this window. You can therefore easily modify the comment of the object then pass to that of another object.

For more details on how to enter the comment of an object, see Naming Objects.



Here is the comment regarding the Manager.

☛ *The title bar enables customization of window display in the desktop. You can for example move or dock a window. For more details on window display, see Positioning Windows.*

HANDLING HOPEX CUSTOMIZING DESKTOP

See:

- Organizing your HOPEX Customizing Desktop
- Using the Properties Window
- Using the Comment Window
- Checking Spelling in Texts
- Modifying the Data Language

Organizing your HOPEX Customizing Desktop

The **HOPEX Customizing** desktop (Windows Front-End) comprises various windows enabling repository access:

- a navigation window that contains navigators dedicated to objects, projects, diagrams, utilities, etc.
- an Edit windows (Comment and Properties)

You can customize the display of these windows.

To organize your desktop, you can:

- add or close a window
- define position of a window
- create folders in navigation trees

Adding Windows

To add a window in your desktop (Windows Front-End):

1. In the **HOPEX** menu bar, select:
   - **View > Navigation Windows**
   - **View > Edit Windows**
2. Select the window you want to add.

Positioning Windows

You can organize your desktop as you like. You can show/hide/move navigation and edit windows.

You can:

- float, dock, fix, auto hide or enlarge navigation windows.
- merge several windows and access these via tabs.
- integrate a navigation window in the main edit window.

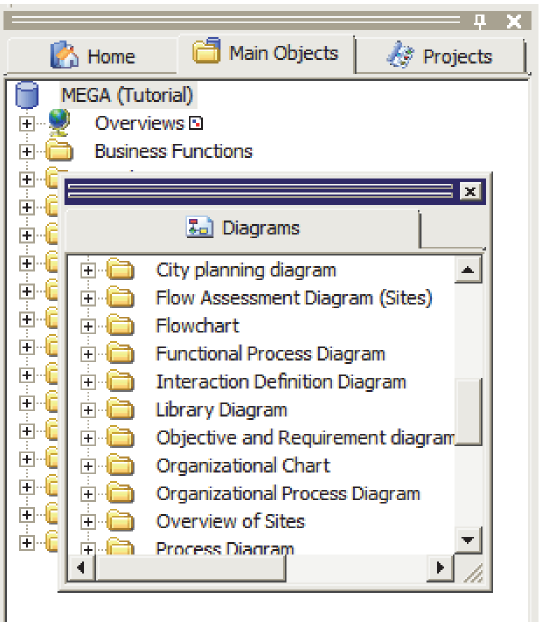
Floating a window

You can detach a window then move it to make it "float" on the desktop.

You can, for example, detach the navigation tabs (such as **Home** and **Main Objects**)

To float a window:

1. Right-click the navigation area title bar and clear **Fixed**.



☛ *To float a window, you can also clear the* ***Docked*** *option from the window title bar pop-up menu.*

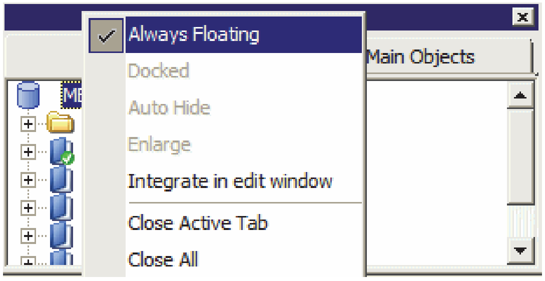
2. Select the window and drag it to where you want.

☛ *Position the window outside "sensitive areas" (when the mouse pointer passes close to sensitive areas, the window will dock in the desktop, which is a situation you do not want here).*

Permanently floating a window

You may decide that a window should always float and never be docked. To do this:

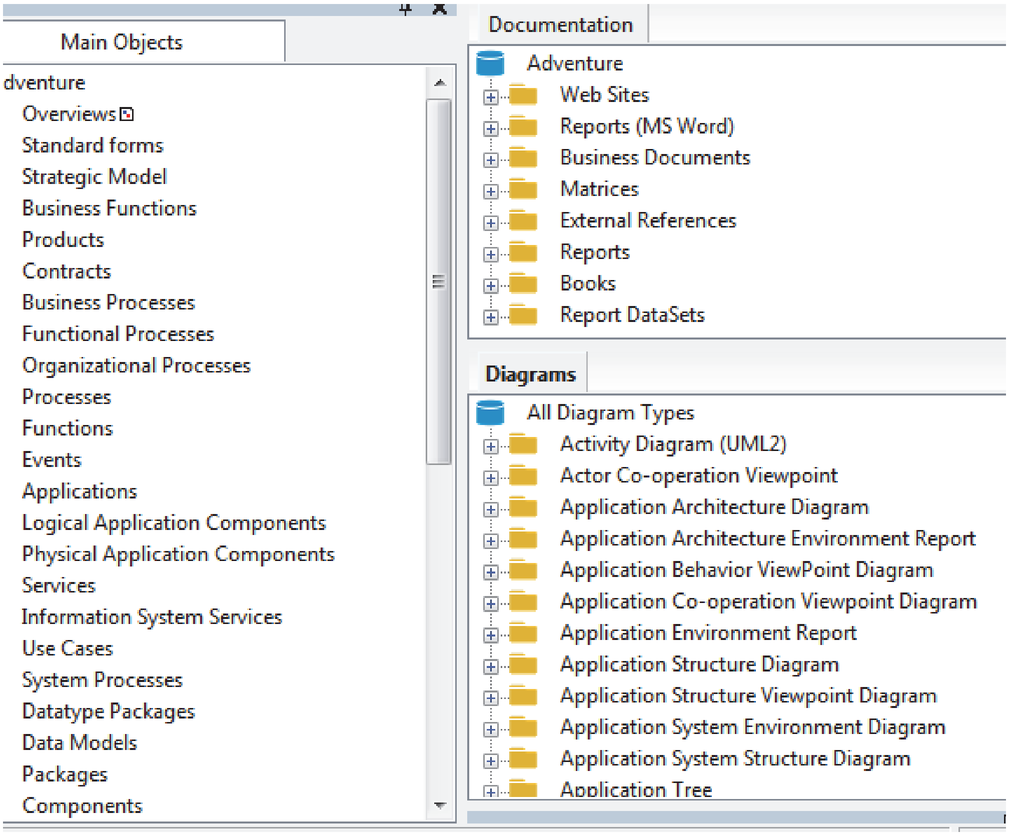
- Right-click the pop-up menu of the floating window title bar, then click **Always Floating**.



Docking a window

You can dock windows in the desktop. A window is said to be **Docked**, as distinct from floating, when it is integrated in the desktop.

To dock a window:



1. Select the floating window and move it in the desktop.
   When the mouse pointer passes close to a sensitive area, a colored frame appears indicating future positioning of the window when docked.
2. When you have found a satisfactory position, release the mouse button.
   The window is docked.

> ☛ *To dock a window, you can also select **Docked** in the window title bar pop-up menu.*

Fixing a window

You can fix a docked window so that it always remains in the same position in the desktop.

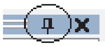
To fix a window:

- In the pop-up menu of the floating window title bar, select **Fixed**.

> ☛ *If the **Fixed** command is cleared, the window can become floating, in other words not fixed to the edge of another window.*

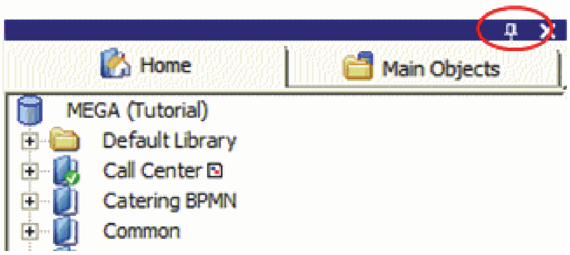
Automatically hiding a window

A pushpin in window title bars facilitates switching to "Auto Hide" mode. This function allows you to increase viewable display and editing space by auto hiding navigation and edit windows.



To automatically hide a window:

- In the title bar, click the pushpin icon .



The window is automatically hidden and is positioned at the edge of the application.

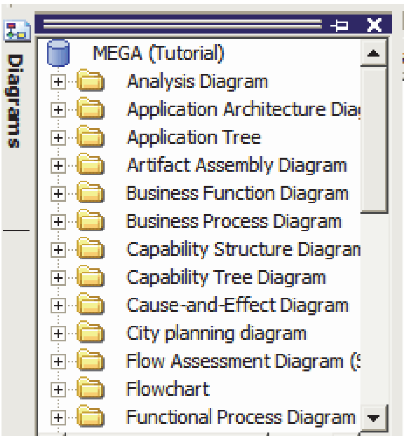
To access an auto-hidden window:

- Click the tab of the hidden window.
  The window appears.

To hide a window again:

- Click its icon.

  ☛ *Clicking outside the active window also has the effect of hiding it.*


Diagrams
MEGA (Tutorial)
Analysis Diagram
Application Architecture Dia
Application Tree
Artifact Assembly Diagram
Business Function Diagram
Business Process Diagram
Capability Structure Diagran
Capability Tree Diagram
Cause-and-Effect Diagram
City planning diagram
Flow Assessment Diagram (
Flowchart
Functional Process Diagram

To deactivate auto hide:

- Click on the pushpin in the title bar.
  The window reappears and can be used.

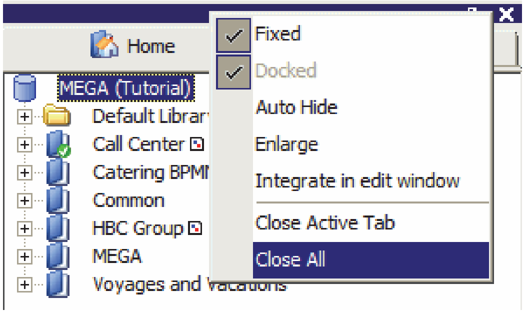
Closing a tab

To close a tab:

- Right-click the tab concerned and select **Close**.

To simultaneously close all tabs of the navigation area:

1. Right-click in the navigation title bar.
2. Select **Close All**.



Integrating a navigation tab in the edit window

To integrate a navigation tab in the edit window:

- Right-click the tab you want to move and select **Integrate in edit window**.
  The tab is moved into the edit window.

> ☛ *To reintegrate the tab in the navigation area, right-click the tab and select* ***Remove from edit window****.*

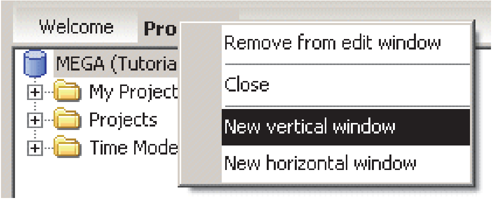
Creating several edit windows

When you have moved a navigation window into the edit window, you can organize this desktop as you wish:

- create new windows
- move tabs in edit windows
- return tabs to the navigation area.

To create a new edit window:

1. In the edit window, right-click the desired tab and select:
   - **New vertical window** or
   - **New horizontal window**.



The new window appears.

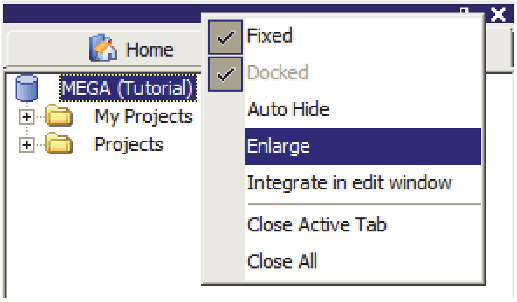
> ☛ *When you have created several edit windows, you can move a tab from one window to another by selecting **Move in previous window** (or **next window**).*

Enlarging a navigation window

You can enlarge a navigation window so that it occupies the entire desktop.

To enlarge a navigation window

1. Right-click the title bar.
2. Select **Fixed**.
3. Select **Enlarge**.



The window now occupies the entire desktop.

Organizing Folders in Navigation Trees

Folders are used in encyclopedic trees to organize the objects in the tree. For this, **HOPEX** offers two possibilities:

- you can create and name your folders
- you can use the encyclopedic classification options

Creating Folders in Navigation Trees

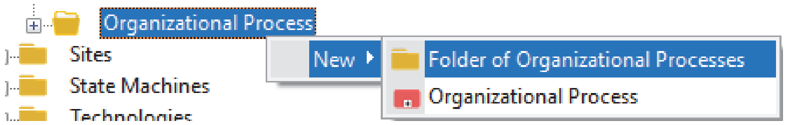
To find your objects more easily in navigators you can create your own groups.

To do this, **HOPEX** proposes creating object folders.

> You can for example create folders grouping your objects to classify them according to your own criteria.

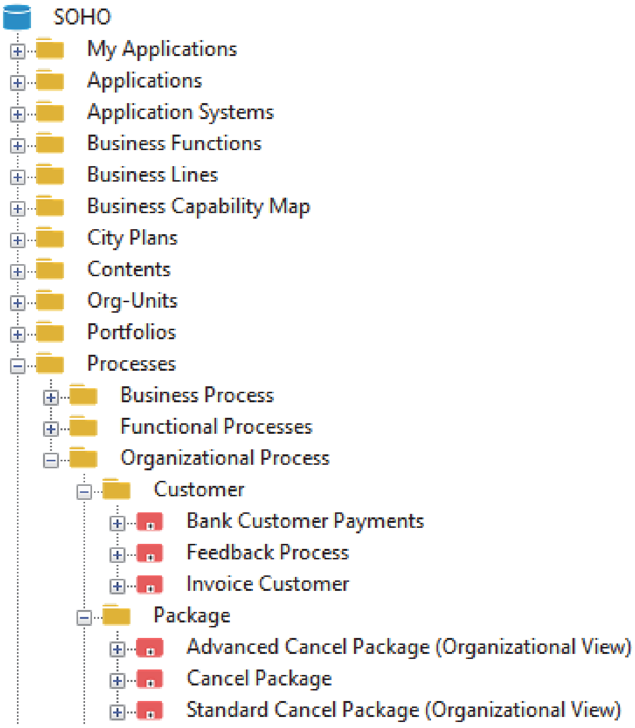
To create folders:

1. Access a navigation tree.
2. Right-click the folder of the object type for which you want to create a folder and select **New > Folder of <Name of object type>**



☛ *Once your modification is dispatched, to delete the folder, you must have the rights to modify the dispatched objects from a private workspace (**Options > Repository**).*

3. In the folder created, you can:
   - create new objects
   - move the objects already created to find them more easily.

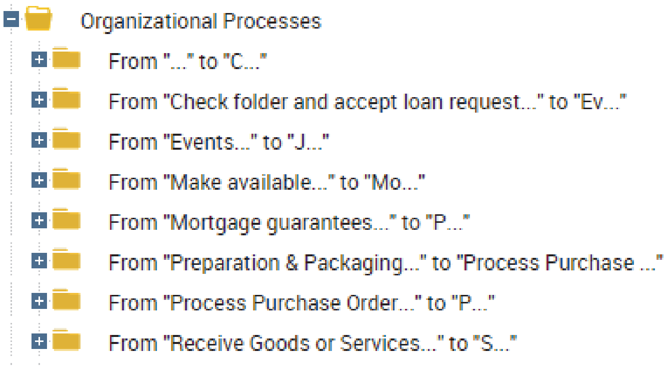


Using the encyclopedic classification options

The display time of a folder that contains a large number of objects can be long.

To reduce this time, you can use these options to classify your objects in alphabetical order in "volumes" represented by folders created and updated automatically by **HOPEX**.

The name attributed to each volume comprises the name of the first object in the volume and the first letter of the last object.



Organizational Processes

- From "..." to "C..."
- From "Check folder and accept loan request..." to "Ev..."
- From "Events..." to "J..."
- From "Make available..." to "Mo..."
- From "Mortgage guarantees..." to "P..."
- From "Preparation & Packaging..." to "Process Purchase ..."
- From "Process Purchase Order..." to "P..."
- From "Receive Goods or Services..." to "S..."

To modify these options:

1. In the menu bar, select **Tools > Options**.
2. In the tree on the left, click **Workspace**.
3. On the right side, complete the following fields:
   - **Trigger threshold for Encyclopedic Classification in the Browser**: this is the number of objects from which automatic classification is triggered within each folder concerned. By default this perspective is set to 200.

     ☛ *If the threshold is equal to zero, the encyclopedic classification is disabled.*
   - **Number of Elements per Encylopedic Volume**, by default set to 200.

Using the Properties Window

Accessing object properties

To access an object properties in **HOPEX**:

1. Connect to **HOPEX**.

   ☛ *See Accessing HOPEX Customizing Desktop.*
2. Right-click the object (for example in a tree, result window, or diagram) and select **Properties**.
   The properties window appears.

Displaying the Properties window on a permanent basis

You can choose to display the Properties windows in **HOPEX** on a permanent basis so as to view immediately the properties of an object.

To display the object property window in **HOPEX**:

1. Connect to **HOPEX**.

   ☛ *See Accessing HOPEX Customizing Desktop.*

2. In the **HOPEX** menu bar, select **View > Edit Windows > Properties**.
3. Select an object.
   Its properties appear in the **Properties** window.

Using the Comment Window

You can modify or enter object comments in the desktop comment window.

☺ *Displaying the comment window is useful when you want to successively enter the comments for several objects.*
When you click an object in a navigation window, in a diagram, or in a query result dialog box, the comment of the selected (active) object appears in this window. You can therefore easily modify the comment for the object then move to the comment for another object.

Displaying the comment window

To display the comment window:

1. Select **View > Edit Windows > Comment**.
   The comment window appears. It displays the comment of the active object.
2. You can modify/enter a comment.

Inserting an object in a comment

You can insert an object in a comment:

- by drag & drop

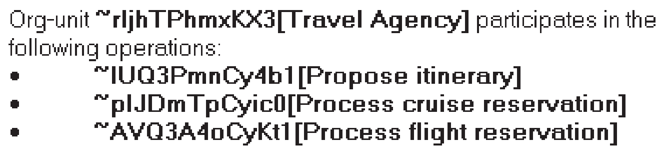
  ```
  Example: select an object in the explorer and drop it into
  the comment.
  ```

- with the **Insert a field** button of the comment window.

Viewing the identifier of an object in a comment

To view the identifier of objects inserted in comments:

❱ In the object comment windows, click **Hide/Show Fields**. {}



Org-unit **~rljhTPhmxKX3[Travel Agency]** participates in the following operations:

- **~IUQ3PmnCy4b1[Propose itinerary]**
- **~pIJDmTpCyic0[Process cruise reservation]**
- **~AVQ3A4oCyKt1[Process flight reservation]**

Deleting a field in a comment

To delete a field in a comment:

1. Click {} to view the field.
2. Select the corresponding field code.
3. Delete the selected code.

Copying/pasting text

You can copy/paste text in a comment.

By default the text is pasted without formatting (RTF or HTML). To format the text, use the HOPEX text-editor available in the comment window.

Pasting text with formatting does not guarantee formatting consistency in deliverables (reports, reports (MS Word) or Web sites) restituting this text.

To remove/keep the text formatting while pasting text:

1. Access the **Options**.
2. In the **Options** tree, select **Compatibility > Windows Front-End > Text Editing**.
3. In the right pane:
   - To remove text formatting while pasting text, keep **Paste without formatting**.

     ☛ *Default value (recommended).*
   - To keep text formatting while pasting text, clear **Paste without formatting**.

     ☛ *This mode is not recommended. In case of troubles, manually unformat the text, then use the HOPEX text-editor to format it.*

Checking Spelling in Texts

The **Spelling** button ✓ starts the spelling checker.

☛ *By default, the dictionary of the current language is used. Dictionaries in French, English and Italian (format .tlx) are provided in the "System" folder. You can customize dictionaries using the "userdic.tlx" file.*

Modifying the Data Language

To modify the data language in HOPEX:

1. In the **HOPEX** menu bar, select **Tools > Language**.
2. Select the language.
The data (translated) appear directly in the selected language.

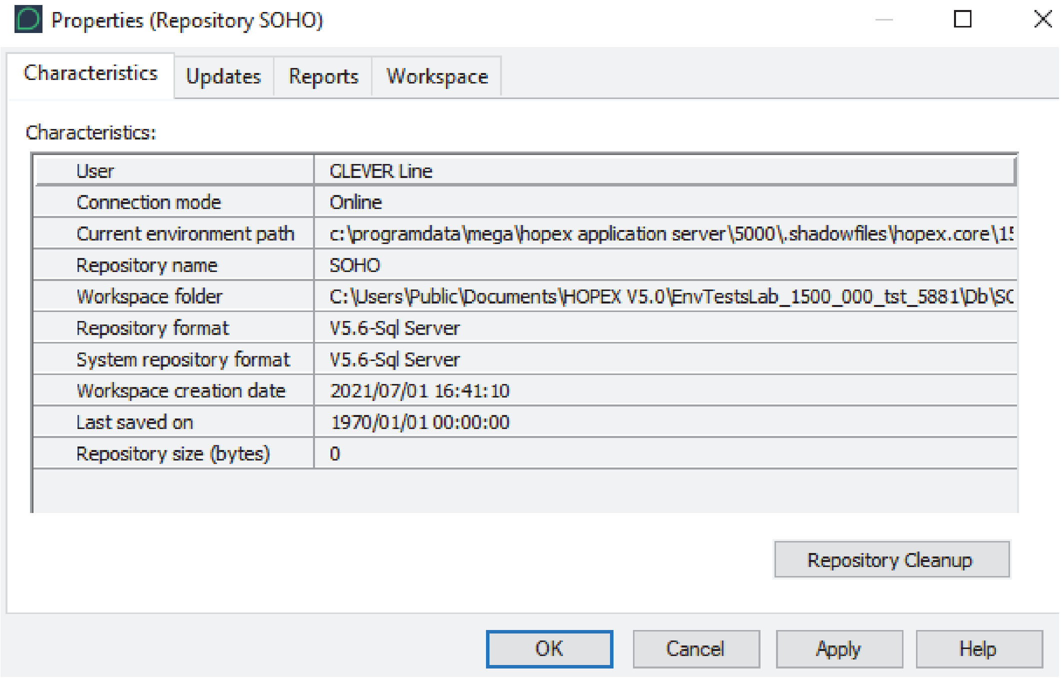
Accessing your Workspace Properties

To consult your workspace properties:

1. Connect to **HOPEX**

   ☛ *See Connecting to HOPEX Customizing Desktop.*

2. In the **HOPEX** menu bar, select **File > Properties**.
   The properties dialog box of your workspace appears.



User	GLEVER Line
Connection mode	Online
Current environment path	c:\programdata\mega\hopex application server\5000\.shadowfiles\hopex.core\1
Repository name	SOHO
Workspace folder	C:\Users\Public\Documents\HOPEX V5.0\EnvTestsLab_1500_000_tst_5881\Db\SC
Repository format	V5.6-Sql Server
System repository format	V5.6-Sql Server
Workspace creation date	2021/07/01 16:41:10
Last saved on	1970/01/01 00:00:00
Repository size (bytes)	0

The workspace properties dialog box provides the following information on the current workspace:

- current **User**
- information on the current repository: its **Name**, **Backup logfile**, **Format**, **Workspace creation date**, **Last saved on**.
- **Repository Cleanup** option.

  ☛ *For more information on repositories, see Managing Repositories.*

HANDLING UPDATES

During their modeling work, users make additions to a **HOPEX** repository within their workspace: they create objects, links between objects, diagrams, etc.

Updates corresponding to user actions can be viewed in detail.

You can back up all modifications made to a repository from a private workspace in a private workspace log, which can be exported in the form of a command file.

The following points are detailed here:

- Viewing Updates
- Viewing the Dispatches and their Content
- Exporting Updates
- Exporting Your Private Workspace Log

Viewing Updates

The repository property windows shows the updates made in the repository.

> ☛ *You can also view the updates from the dispatch tree (**Repository Activity > Repository Dispatch** navigation window*

The repository property window shows the updates performed:

- on the current **HOPEX** repository (by default), or
- on the system repository.

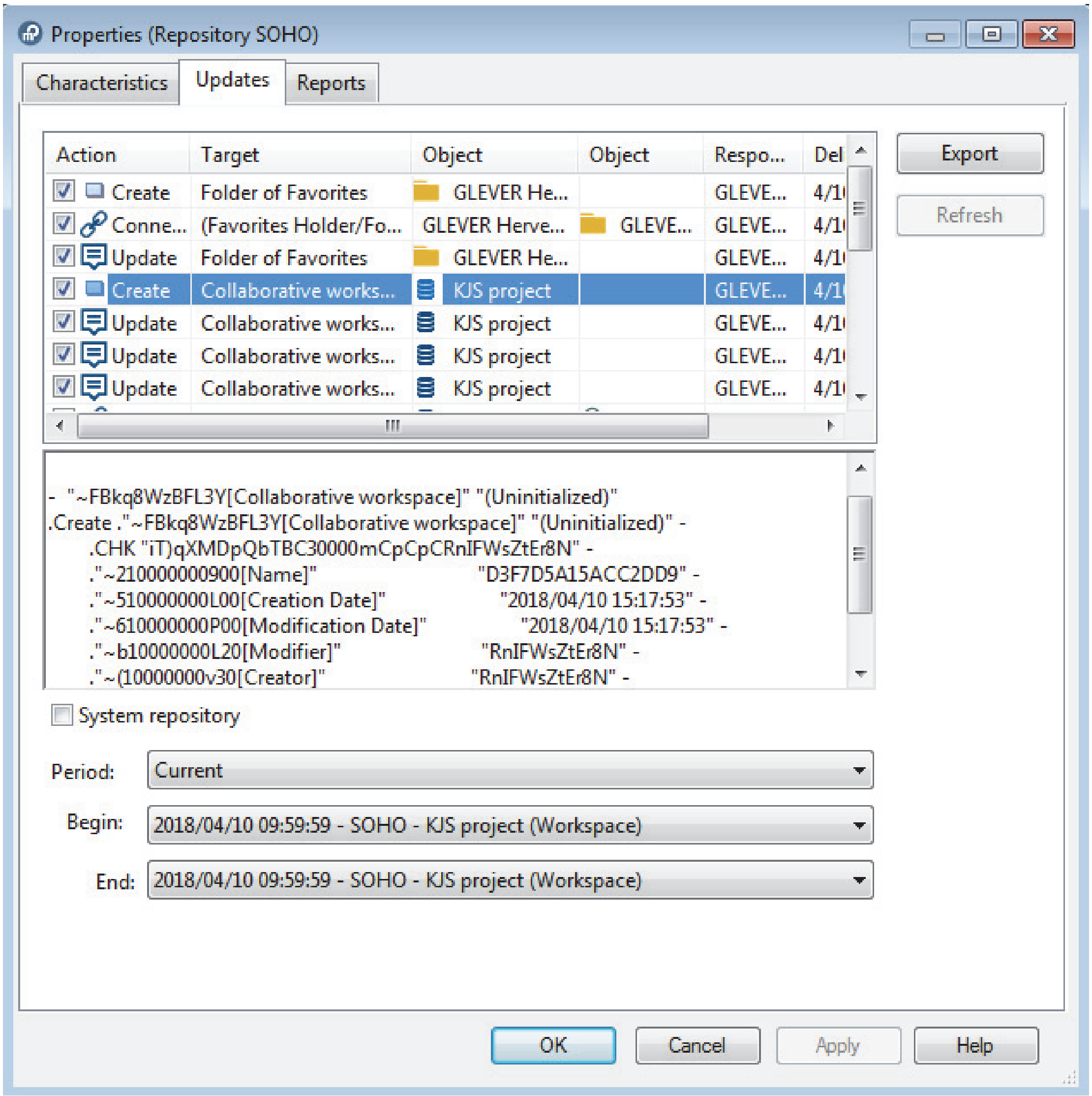
By default, the updates displayed are those you have performed:

- in your current private workspace
- on the **HOPEX** repository

You can also display the updates dispatched on the current repository within the interval defined by two dispatches.

The Updates window

☛ *To display the Updates window, see Displaying your current updates on the HOPEX repository, or Displaying your current updates on the system repository.*



Action	Target	Object	Object	Respo...	Del
☑ Create	Folder of Favorites	GLEVER He...		GLEVE...	4/1(
☑ Conne...	(Favorites Holder/Fo...	GLEVER Herve...	GLEVE...	GLEVE...	4/1(
☑ Update	Folder of Favorites	GLEVER He...		GLEVE...	4/1(
☑ Create	Collaborative works...	KJS project		GLEVE...	4/1(
☑ Update	Collaborative works...	KJS project		GLEVE...	4/1(
☑ Update	Collaborative works...	KJS project		GLEVE...	4/1(
☑ Update	Collaborative works...	KJS project		GLEVE...	4/1(

```
-  "~FBkq8WzBFL3Y[Collaborative workspace]" "(Uninitialized)"
.Create ."~FBkq8WzBFL3Y[Collaborative workspace]" "(Uninitialized)" -
      .CHK "iT)qXMDpQbTBC30000mCpCpCRnIFWsZtEr8N" -
      ."~210000000900[Name]"                    "D3F7D5A15ACC2DD9" -
      ."~510000000L00[Creation Date]"              "2018/04/10 15:17:53" -
      ."~610000000P00[Modification Date]"            "2018/04/10 15:17:53" -
      ."~b10000000L20[Modifier]"                  "RnIFWsZtEr8N" -
      ."~(10000000v30[Creator]"                  "RnIFWsZtEr8N" -
```

□ System repository

Period: Current

Begin: 2018/04/10 09:59:59 - SOHO - KJS project (Workspace)

End: 2018/04/10 09:59:59 - SOHO - KJS project (Workspace)

☛ *To sort the updates, click the column header.*

Indicated for each action are:

- the **Action** type performed

  `Example: "Create", "Connect", "Update".`

  ☛ *See Command File Syntax for more information on operators.*

- the type of the object concerned (**Target**)
- the name of the **Object** concerned
- the name of the second **Object** concerned in the case of a "Connect", "Disconnect" or "Change" action
- the person **Responsible** for the action
- the **Delivery date** of the action

  `Format: <D/MM/YYYY h:mm:ss AM/PM>`

  If you have not dispatched your work yet, the date indicated is the execution date of the action.

- The name of the **Dispatch** that contains the action, in the following format:

  `<YYYY/MM/DD hh:mm:ss> <repository> <responsible>`

  With:

  - <YYYY/MM/DD hh:mm:ss>: workspace date and creation time
  - <Repository>: name of the current repository
  - <responsible>: name of the person or of the collaborative workspace responsible for the action

  If you have not dispatched your work yet:

  `<your name> (workspace)`

  `<name of your collaborative workspace> (collaborative workspace)`

- (when you select the line of a command), the complete text of the update is displayed in the window lower pane.

  ☛ *If needed, to copy the command and paste it in a text file for example, roll the mouse over the whole text, then press <CTRL> + <C> and paste the text in a text file.*

Displaying your current updates on the HOPEX repository

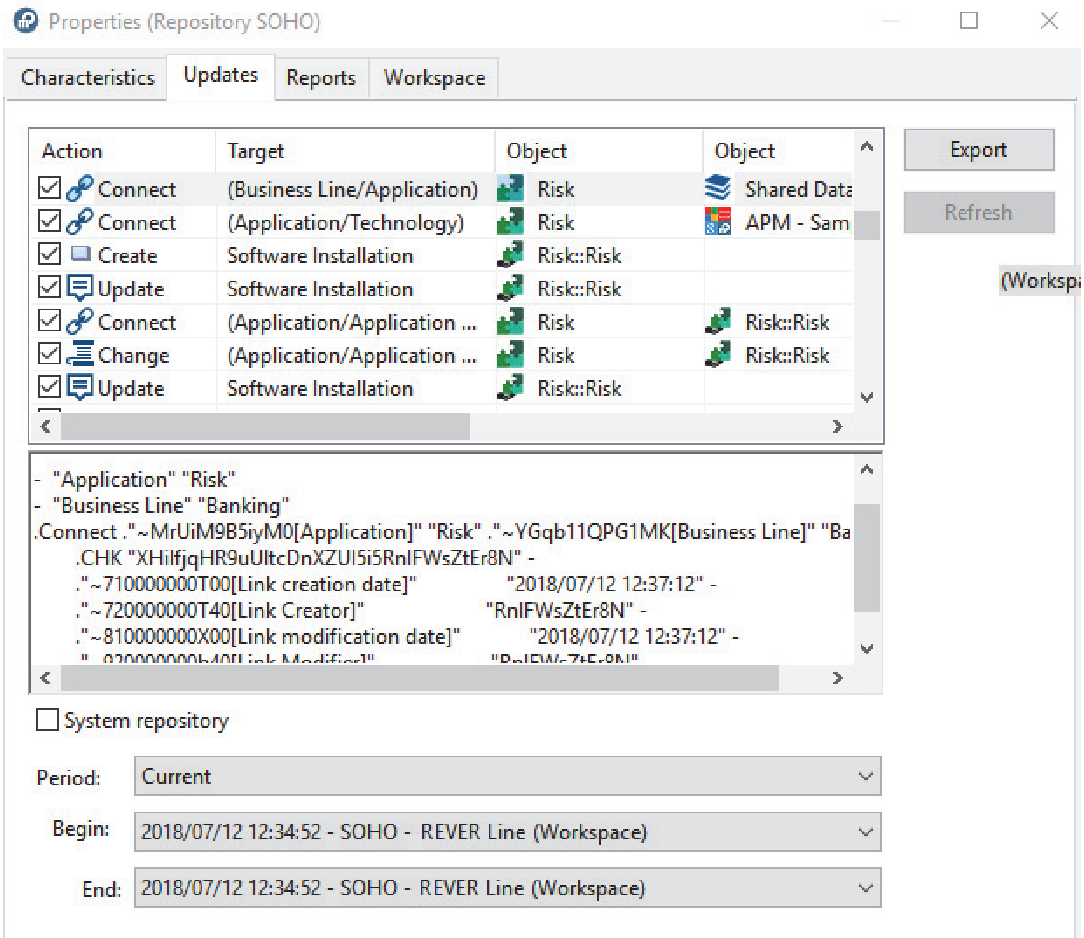
To view the updates you have made in your current workspace:

1. In the **HOPEX** menu bar, select **File > Properties**.

   ☛ *This process may take some time if there are many updates.*

   The property window of the repository in which you are working appears.

2. Select the **Updates** tab.
   By default, the **Updates** tab shows in chronological order the actions you have performed in your private workspace on the **HOPEX** repository



Action	Target	Object	Object
☑ Connect	(Business Line/Application)	Risk	Shared Data
☑ Connect	(Application/Technology)	Risk	APM - Sam
☑ Create	Software Installation	Risk::Risk	
☑ Update	Software Installation	Risk::Risk	
☑ Connect	(Application/Application ...	Risk	Risk::Risk
☑ Change	(Application/Application ...	Risk	Risk::Risk
☑ Update	Software Installation	Risk::Risk	

```
-  "Application" "Risk"
-  "Business Line" "Banking"
.Connect ."~MrUiM9B5iyM0[Application]" "Risk" ."~YGqb11QPG1MK[Business Line]" "Ba
     .CHK "XHilfjqHR9uUltcDnXZUl5i5RnlFWsZtEr8N" -
     ."~710000000T00[Link creation date]"           "2018/07/12 12:37:12" -
     ."~720000000T40[Link Creator]"               "RnlFWsZtEr8N" -
     ."~810000000X00[Link modification date]"         "2018/07/12 12:37:12" -
     ."~920000000b40[Link Modifier]"              "RnlFWsZtEr8N"
```

☐ System repository

Period: Current

Begin: 2018/07/12 12:34:52 - SOHO - REVER Line (Workspace)

End: 2018/07/12 12:34:52 - SOHO - REVER Line (Workspace)

☛ *Actions you have performed in your private workspace on the system repository are not shown. To display the updates performed on the system repository, see Displaying your current updates on the system repository.*

Displaying your current updates on the system repository

By default, the **Updates** tab shows the actions you have performed in your workspace on the **HOPEX** repository.

☛ *The display of updates is exclusive (**HOPEX** repository or system repository).*

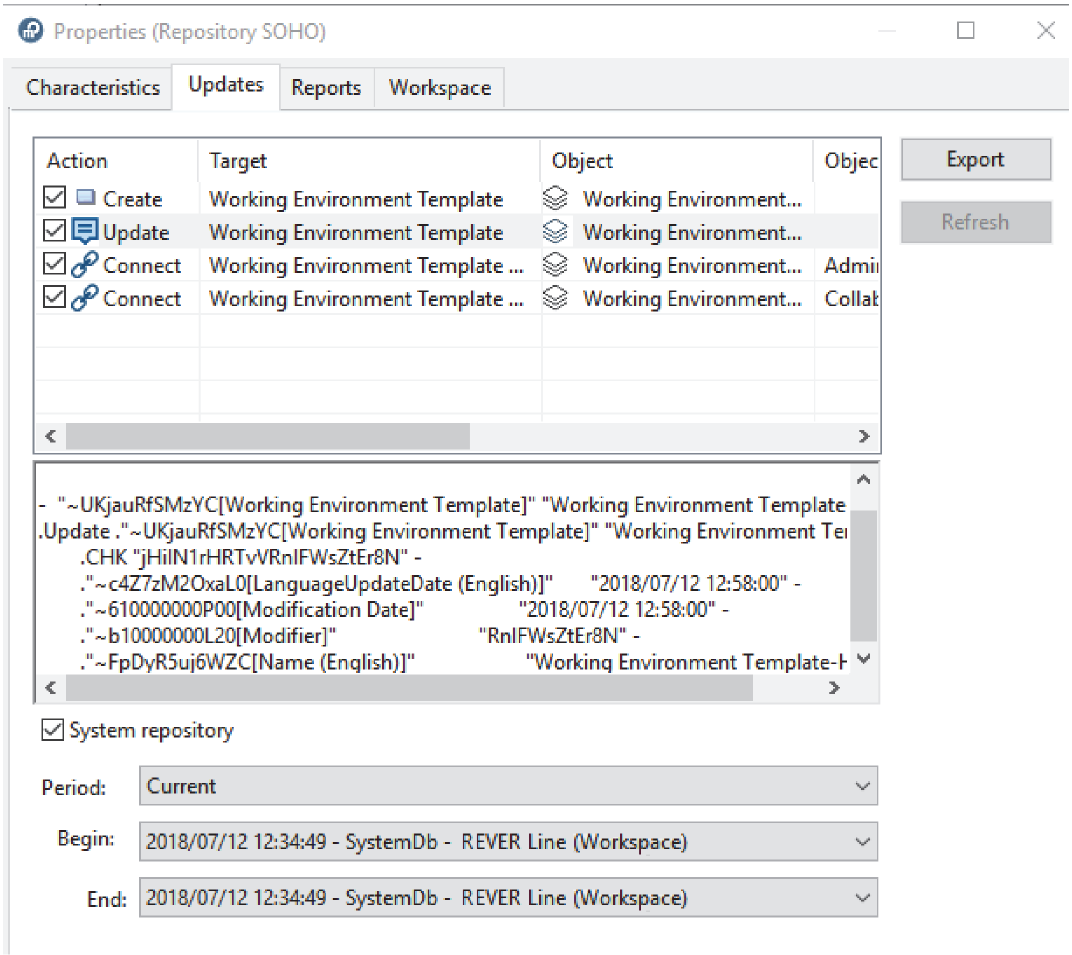
To display your current updates on the system repository:

1. In the **HOPEX** menu bar, select **File > Properties**.

   ☛ *This process may take some time if there are many updates.*

   The property window of the repository in which you are working appears.
2. Select the **Updates** tab.

3. Select the **System repository** check box.
   The **Updates** tab display in chronological order the updates performed on the objects of the system repository.



Action	Target	Object	Objec
☑ Create	Working Environment Template	Working Environment...	
☑ Update	Working Environment Template	Working Environment...	
☑ Connect	Working Environment Template ...	Working Environment...	Admi
☑ Connect	Working Environment Template ...	Working Environment...	Collat

```
-  "~UKjauRfSMzYC[Working Environment Template]" "Working Environment Template
.Update ."~UKjauRfSMzYC[Working Environment Template]" "Working Environment Te
      .CHK "jHilN1rHRTvVRnIFWsZtEr8N" -
      ."~c4Z7zM2OxaL0[LanguageUpdateDate (English)]"      "2018/07/12 12:58:00" -
      ."~610000000P00[Modification Date]"              "2018/07/12 12:58:00" -
      ."~b10000000L20[Modifier]"                  "RnIFWsZtEr8N" -
      ."~FpDyR5uj6WZC[Name (English)]"              "Working Environment Template-H
```

☑ System repository

Period: Current

Begin: 2018/07/12 12:34:49 - SystemDb - REVER Line (Workspace)

End: 2018/07/12 12:34:49 - SystemDb - REVER Line (Workspace)

Viewing updates dispatched on the repository over a period of time

By default, the **Updates** tab shows the actions you have performed in your workspace on the **HOPEX** repository. You can display the updates made over a period defined by two dispatches.

To view the updates made over a period of time:

1. In the **HOPEX** menu bar, select **File > Properties**.

   ☛ *This process may take some time if there are many updates.*

   The property window of the repository in which you are working appears.
2. Select the **Updates** tab.
3. Select (/Clear) **System Repository** to display the system repository (/ **HOPEX** repository) updates.

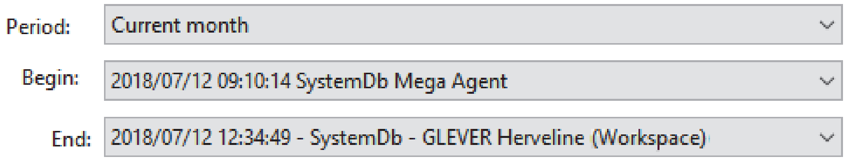
4. In the **Period** field, select the period you are interested in.

    `E.g.: Today, Current week, Current month, From the beginning.`

    The selected period defines the list of dispatches available in the **Begin** drop-down list.
5. In the **Begin** field, select the dispatch from which you want to display the updates.

    ☛ *The first item corresponds to the first dispatch in the repository.*
6. In the **End** field, select the dispatch until which you want to display the updates.

    ☛ *You can select your current workspace, your updates are included in the display.*



Period: Current month

Begin: 2018/07/12 09:10:14 SystemDb Mega Agent

End: 2018/07/12 12:34:49 - SystemDb - GLEVER Herveline (Workspace)

7. Click **Refresh**.
    The selected repository log appears as a list of actions displayed in chronological order.

Viewing the Dispatches and their Content

☛ *See also Viewing the Repository Update Log.*

The **Repository Dispatches** folder contains private workspaces dispatched in the current repository and in the system repository. Dispatches are listed by day, week and month.

To display private workspaces dispatched and the content of their updates:

1. From **HOPEX** (Windows Front-End), in the **Repository Activity > Repository Dispatch** navigation window, expand the repository file concerned:
    - <Current **HOPEX** repository name> or
    - System repository (**SystemDb**)
2. Right-click the dispatch concerned and select **Properties**.
    Private workspaces dispatched on the current repository and system repository are contained in the repository folder.
3. Select the **Updates** tab.

Exporting Updates

In the repository update log window, the **Export** button enables export of selected commands in MGL or XMG format.

To export updates:

1. Access the repository update log.
2. Select the updates to be exported.
3. Click **Export**.
4. Select the export format:
   - *.mgl: text format
   - *.xmg: MEGA XML format.
5. (If needed) Modify the export file name and save folder proposed as default (the **Browse** button ... enables to browse the folder tree and select the folder in which the file will be saved).
6. Click **Export**.
   The file is exported and saved in the specified folder.

   The **Execution Report** appears.
7. (Optional) Click **Open result file** to view the file.
8. Click **OK** to close the window.

Exporting Your Private Workspace Log

You can create an export file and save it in your work folder.

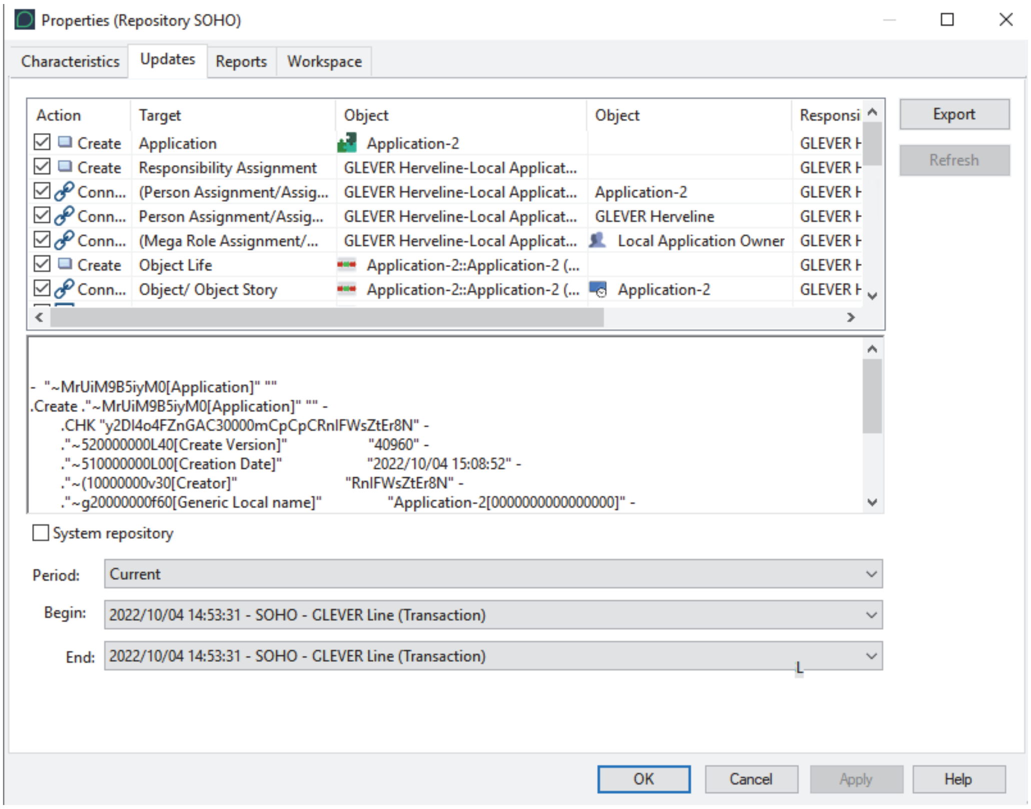
The export file can be exported in format:

- **logfile text** (.mgl).
  Name format of the exported file is "LOGmmdd.mgl", where "mmdd" represents logfile export date month and day.
- **XML MEGA** (.xmg)
  The exported file is in the form of an XML file containing commands or data (objects and links).

To export the work done in the current private workspace in the form of a command file:

1. In the **HOPEX** menu bar, select **File > Properties**.

2. Display the **Updates** tab.



Properties (Repository SOHO)

Action	Target	Object	Object	Responsi
☑ Create	Application	Application-2		GLEVER H
☑ Create	Responsibility Assignment	GLEVER Herveline-Local Applicat...		GLEVER H
☑ Conn...	(Person Assignment/Assig...	GLEVER Herveline-Local Applicat...	Application-2	GLEVER H
☑ Conn...	Person Assignment/Assig...	GLEVER Herveline-Local Applicat...	GLEVER Herveline	GLEVER H
☑ Conn...	(Mega Role Assignment/...	GLEVER Herveline-Local Applicat...	Local Application Owner	GLEVER H
☑ Create	Object Life	Application-2::Application-2 (...		GLEVER H
☑ Conn...	Object/ Object Story	Application-2::Application-2 (...	Application-2	GLEVER H

```
- "~MrUiM9B5iyM0[Application]" ""
.Create ."~MrUiM9B5iyM0[Application]" "" -
      .CHK "y2Dl4o4FZnGAC30000mCpCpCRnlFWsZtEr8N" -
      ."~520000000L40[Create Version]"              "40960" -
      ."~510000000L00[Creation Date]"               "2022/10/04 15:08:52" -
      ."~(10000000v30[Creator]"                  "RnlFWsZtEr8N" -
      ."~g20000000f60[Generic Local name]"            "Application-2[0000000000000000]" -
```

☐ System repository

Period: Current

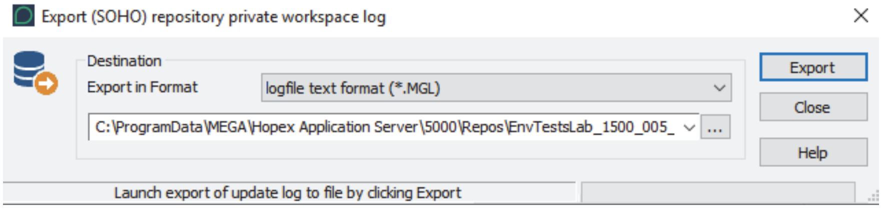
Begin: 2022/10/04 14:53:31 - SOHO - GLEVER Line (Transaction)

End: 2022/10/04 14:53:31 - SOHO - GLEVER Line (Transaction)

3. Select all of your actions.

   ☛ *See* *Viewing Updates* *for more details.*

4. click **Export**.



Export (SOHO) repository private workspace log

Destination

Export in Format: logfile text format (*.MGL)

C:\ProgramData\MEGA\Hopex Application Server\5000\Repos\EnvTestsLab_1500_005_

Launch export of update log to file by clicking Export

5. Select the export format.
6. (Optional) If necessary, modify the data export file name and save folder proposed as default (the **Browse** button ... enables to browse the folder tree and select the folder in which the file will be saved).
7. Click **Export**.
   The **Execution Report** appears.
8. Click **OK**.
   The file is exported and saved in the specified folder.

You can subsequently import this logfile, for example into a new private workspace.

2

WORKING IN HOPEX CUSTOMIZING DESKTOP

See:

- Editing Objects
- Using Favorites
- Using History
- Managing Options
- Saving Sessions
- Saving Your Data
- Exiting HOPEX

Editing Objects

HOPEX offers an editor, that is a data entry tool, for certain object types. This editor enables simple display and modification of the objects concerned, offering in the same interface the advantages of both navigator and properties dialog box.

You can run the editor on the following object types:

- Method
- Project
- Project type
- MetaClass (i.e.: object type, accessible via the **Utilities** tab)

To run the editor in **HOPEX**:

1. Right-click the object and select **Edit**.
   The editor opens on the selected object.

   ☛ *To view the editor with its two panes (tree and properties), click the **Details** button in the toolbar* ☰.

   ☛ *If the editor shows only the tree, click the **Details** button in the toolbar* ☰.

2. To validate modifications carried out in the properties dialog box of an object (right pane of editor), press the <Enter> key.

Using Favorites

Accessing favorites

To access favorites:

1. In **HOPEX** toolbar, click **Favorites** ★ (or in the menu bar select **Display > Favorites**).
   In the edit area, the **Hierarchical view** tab displays the favorites tree.

2. Expand the folders concerned to access the favorites.

Adding an object to favorites

To add an object to your Favorites list:

1. Connect to **HOPEX**.

   ☛ *See Accessing HOPEX Customizing Desktop.*

2. Right-click the object you want to add to your favorites and select **Add to Favorites**.

Using History

Accessing the History

The history contains all the objects you have accessed (e.g.: display of their properties pages) or modified.

To access the history:

1. In the **HOPEX** toolbar, click **History** (or in the menu bar select **Display > History**).
   The history tree displays the objects ordered by date.
2. Expand the folder of a date.
   Objects are classified by object type (MetaClass).
3. Expand the folder for each type of object to display the objects.

Accessing the History of an Object

The history of an object displays:

- the object creation date and its author
- the date of last modification of the object and its author
- all modifications made for the object in chronological order
  - the author of modifications
  - which attributes were modified
  - the date of each modification

To display the history of an object:

1. Right-click the object and select **Manage > History**.

   ☛ *With the HOPEX Customizer profile (Web Front-End and Windows Front-End), in the object properties select **General > History**.*

   The chronology of the object modifications is displayed.

Managing Options

User level options

To configure your desktop and access certain functionalities, you may need to configure the user level options.

To configure the (user level) options:

1. In the menu bar, select **Tools > Options**.
   The user level options window opens.
2. Modify the options as needed.

   ☛ *For more details on options, see the **HOPEX Administration - Supervisor** guide, "Managing Options" chapter.*

   ☛

Managing Windows Front-End user inactivity

User inactivity management is taken into account if the **Automatic Session Timeout** option (options: **Workspace > Desktop**) is selected.

☛ *See Activating/Deactivating user inactivity management.*

To manage inactivity of Windows Front-End users:

1. Access **Options** at the environment level.

   ☛ *See Accessing Options.*
2. In the **Options** tree, select values for options:
   - **Workspace > Desktop > Duration of inactivity before closing HOPEX**
   - **Compatibility > Windows Front-End > Workspace > Period of inactivity requiring reauthentication**

   When this duration has been reached, the Windows Front-End user is disconnected and **HOPEX** closes without warning.

   ☛ *So that the user is warned of disconnection, the **Period of inactivity requiring authentication** must be specified and its value less than that of the **Duration of inactivity before closing HOPEX**.*

Opening the Trace File

To open the trace file from **HOPEX**:

1. Connect to **HOPEX**.

   ☛ *See Connecting to HOPEX Customizing Desktop.*
2. In the **HOPEX** menu bar, select **Help > Technical Support > Error Logfile > Edit**.

   ☛ *Alternative: in the **HOPEX** menu bar, select **Help > About HOPEX**, then in **System Information**, select **Error Log > Edit**.*

Working in HOPEX Customizing Desktop

Saving Sessions

📖 *A session is the period during which a user is connected to a repository. A session begins when the user authenticates and ends when he/she exits HOPEX. Sessions and private workspaces can overlap. When you dispatch, refresh or discard a private workspace, a new private workspace is created in the same session. Conversely, a user can keep his/her private workspace when exiting a session.*

To save modifications you have made in your *session* since the last save:

❱ In the **HOPEX** menu bar, click **Save** 💾.

☛ *These modifications are not saved in the repository. To save your modifications in the repository, you must dispatch these modifications, see Dispatching Your Work.*

Saving Your Data

Your data is preserved in the repositories.

You should regularly perform backups of your repository to avoid loss of your data due to unforeseen circumstances such as hard disk failure.

You should keep these backups in a safe location, on a different site from that containing the data.

Backup frequency will vary depending on how quickly the data changes. Weekly backups are generally sufficient when one user is involved. Backups should be more frequent when multiple users are working every day. You should back up your repository before and after important modifications are made to it.

For more details on repository back-up, see chapter "Managing Repositories" in the **HOPEX Administration - Supervisor** guide.

☛ *You can also export the log of your private workspace if required, see Exporting Your Private Workspace Log.*

💣 **Remember to regularly back up your data and to keep several backups.**

Exiting HOPEX

Before exiting **HOPEX**, remember to save your data, see:

- Saving Your Data.
- Exporting Your Private Workspace Log.

If one does not already exist, a private workspace is created when you request repository access: **HOPEX** saves a view of the state of the repository at the moment you connect.

You can add your own modifications without being disturbed by any concurrent modifications of other users: in the case of a network installation, each **HOPEX** user has his/her own view of the repository.

To save your latest modifications at any time, in the **HOPEX** toolbar, click **Save All** 💾 or in the menu bar select **File > Save All**.

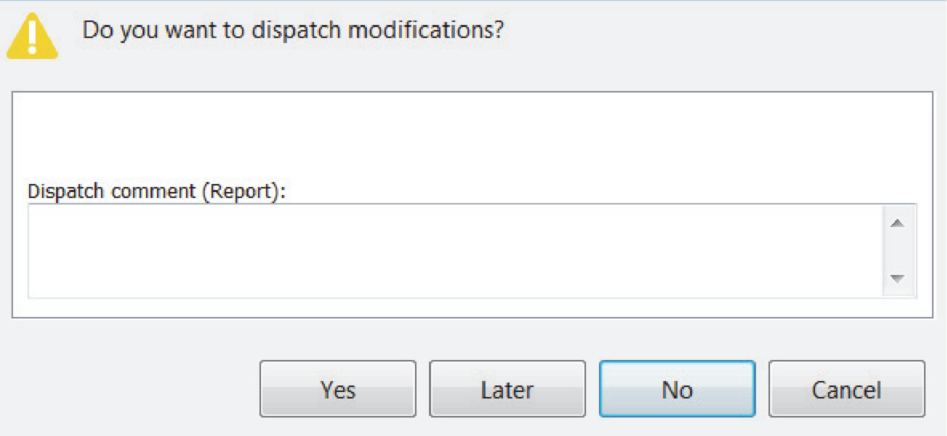
When you exit **HOPEX**, you close your session. You can:

- save in the repository the modifications you have made in your private workspace
- keep the modifications you have made in your private workspace

  ☛ *These modifications will remain awaiting validation, subsequent modification, or deletion.*
- cancel modifications you have made.

To exit your work session:

1. In the **HOPEX** menu bar, select **File > Exit**.
   A dialog box appears.



Do you want to dispatch modifications?

Dispatch comment (Report):

2. (Optional) In the **Dispatch comment (Report)** frame, enter a comment to remind you of modifications made in your private workspace.

   ☛ *This comment is added in the dispatch properties, see Viewing the Dispatches and their Content.*
3. (Optional) Select a design task and select **Complete task** if necessary.
4. Click:
   - **Yes**
     Modifications you have made in your private workspace are saved in the repository.

     ☺ *In order to work effectively as a team, it is recommended that you dispatch frequently and regularly. Other users can update their own* *private workspace without dispatching their work (menu **File > Refresh**).*

☛ *This exit mode also allows the user to select a different repository the next time he/she logs in.*

- **No**

All modifications you made since your last dispatch will be lost. You can use this option if you want to view data quickly and exit without impacting the repository.

☛ *Modifications to your desktop are also lost.*

- **Later**

This option allows you to keep your changes without impacting the repository. You can open your session later and continue working but other users are not yet seeing the changes you have made.

☛ *Click **Cancel** to not exit your private workspace.*

SEARCHING OBJECTS

- ✓ Introduction to the Search
- ✓ Quick Search
- ✓ The Advanced Search
- ✓ Editing Queries
- ✓ Query Results
- ✓ Handling Candidate Objects
- ✓ Configuring Query Tool Options

Introduction to the Search

To perform searches on objects in the repository, **HOPEX Customizing** desktop (Windows Front-End) offers:

- a search bar for a quick search by object type.

  ☛ *See Quick Search.*

- an advanced search tool, for an advanced search through a wizard or the possibility to enter a query using a repository query language.

  ☛ *See The Advanced Search.*

Searching from the workspace

In your **HOPEX Customizing desktop** (Windows Front-End), you can search objects via:

- the search toolbar to run a quick search

  ☛ *See Quick Search.*

- the advanced search tool (click 🔍) which offers a number of search modes:
  - Assistant

    ☛ *See Querying Using the Query Assistant.*

  - Recent queries

    ☛ *See Using Recent Queries.*

  - Registered queries

    ☛ *See Editing Queries.*

Searching from another HOPEX tool

The search tool can be called from numerous HOPEX windows or tools (e.g.: explorer, diagram).

For example it is called from a diagram when you want to add an object.

When run from another **HOPEX** tool, the search toolbar includes the following tabs in addition to the three standard modes:

- Candidates

  ☛ *See Handling Candidate Objects).*

- Result

  ☛ *See Query Results.*

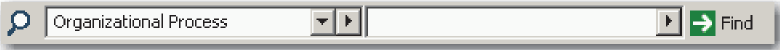
For more details on the different query modes, see Querying Using the Query Assistant.

QUICK SEARCH

Search Toolbar

With **HOPEX** (Windows Front-End), you can perform quick searches by object type.

With the search toolbar at the top of the workspace, you can perform the more common searches.



☛ *To run more complex searches, click 🔍 or see The Advanced Search to access the advanced search.*

Objects Being Search for

Searching for a single object type

To perform a search on a single object type:

1. Access the search toolbar.
2. In the first field, select an object type from the drop-down list.

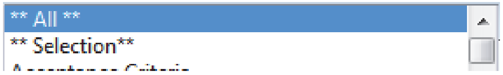
   `Example: "Org-Unit" to search for all the org-units in the repository`

3. Click **Find** ➔ or press [Enter].
   Objects are listed in the result window.

Searching for all object types

To perform a search on all object types:

1. Access the search toolbar.
2. In the first field, select "** All **" (the first value in the drop-down list).



3. In the second field, define the search type.

   ☛ *See Search Type.*

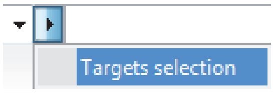
4. Click **Find** ➔ or press [Enter].
   Objects are listed in the result window.

Searching for several object types

To perform a search on a limited number of object types:

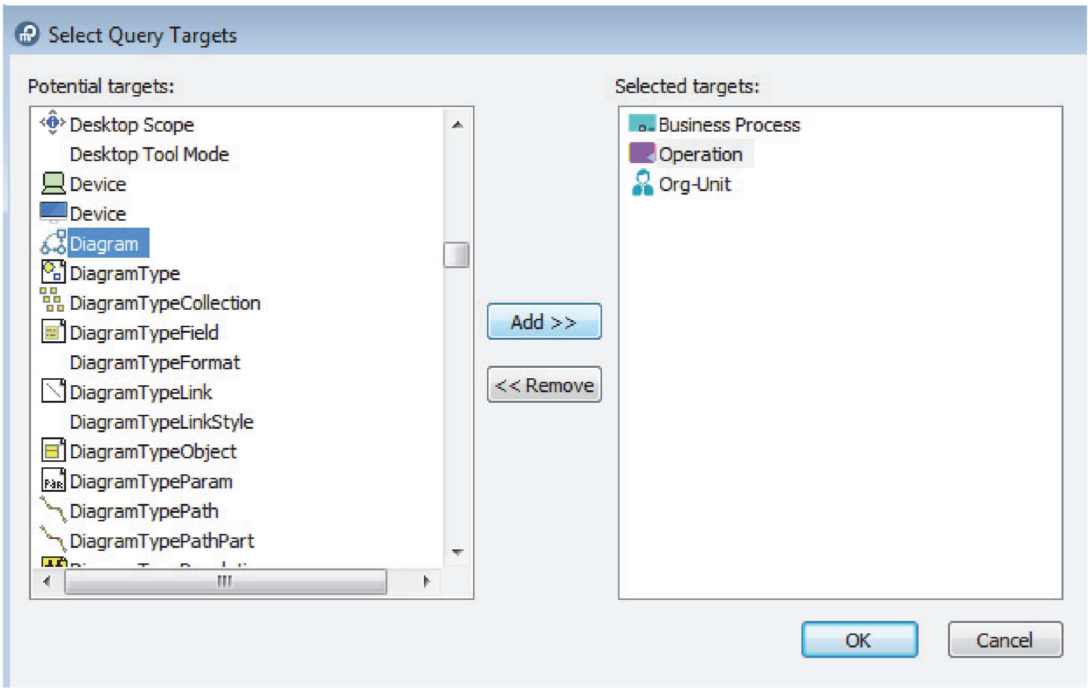
1. Access the search toolbar.

2. In the first field, click the right arrow and select **Targets Selection**.

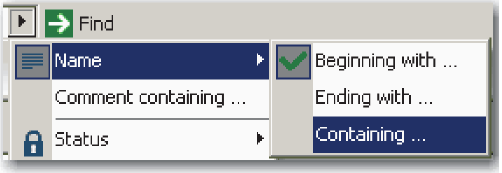


3. In the **Select Query Targets** dialog box, select the object types on which you want to run a search, then click **Add**.

   ☛ *To select several object types at once, use the [Ctrl] key.*



4. Click **OK**.
   The first field, for object type selection, automatically displays "Selection".
5. In the second field:
   - enter the character string to be searched,
   - click the arrow to define whether the search is run on the name and/or comment



6. Click **Find** or press [Enter].
   Objects are listed in the result window. The result is heterogeneous (multitarget).

Search Type

☛ *See also Quick Search.*

You can perform searches as a function of characteristics common to all object types.

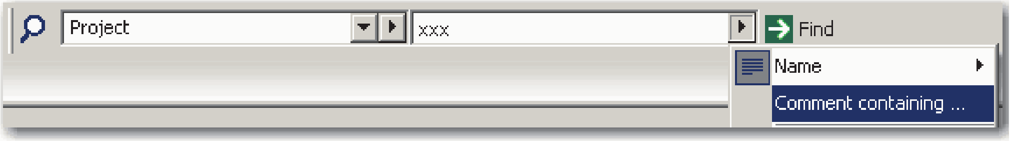
Searching in the object comment

To search for a character string in object comment:

1. Access the search toolbar.
2. In the first field, select the object type.

   `Example: Project`

3. In the second field:
   - enter the character string to be searched for.
   - click the arrow and select **Comment Containing**.



4. Click **Find** ➔.
   Objects are listed in the result window.

Searching in the object names

You can search for objects of which the short name:

- begins with,
- ends with, or
- contains a character string.

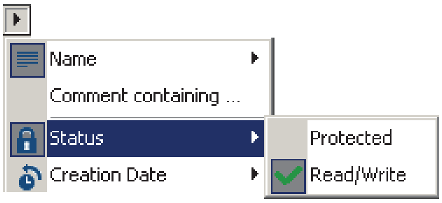
Searching as a function of object status

You can perform a search using object status as search criteria: "read/write" (modifiable) or "protected" (read-only).

This criterion can be combined with another.

`For example, you can search for modifiable projects whose names begin with a given string chain.`

In this case, both criteria are indicated graphically, which allows you to immediately see on which criteria the search is based.



Searching by date

You can perform a search by **Creation Date** or **Modification Date**. This date can be:

- **Before**
- **On**
- **After**

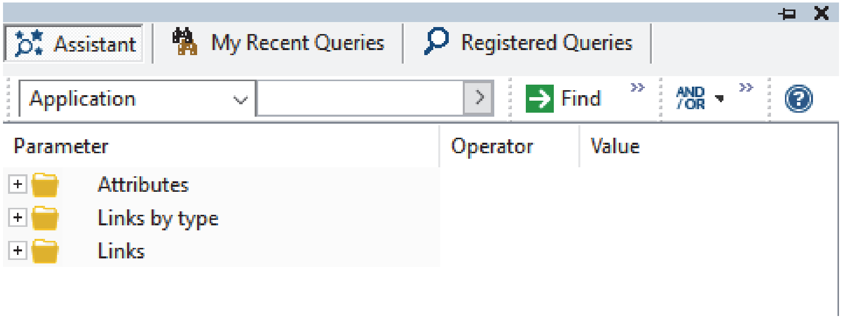
Searching by absolute identifier

You can perform a search using an absolute identifier.

THE ADVANCED SEARCH

☛ *See also Quick Search.*

Advanced Search Tool Description



The advanced search tool (Query Tool) is available:

- from the workspace, using the search toolbar or 🔍.
- from other **HOPEX** tools such as diagrams, the explorer, etc.

Different modes, each corresponding to a tab, are available depending on how the search is run (from the workspace or from another tool).

Use the advanced search tool to enter queries, which define:

- the objects and links browsed.
- the conditions on links and objects making up the repository browsing path.

The subject of the search is called "target".

A query can be written:

- using a wizard (assistant)

  ☛ *See Querying Using the Query Assistant.*

- directly using the repository query language (ERQL)

  ☛ *ERQL (Entity Relationship Query Language) enables you to define queries using entity/relationship formalism.*
  *Syntax of this language is described in **HOPEX Power Studio - Query Syntax**.*

Querying Using the Query Assistant

The **Assistant** tab enables creation of multi-parameter searches without needing to know ERQL language.

A search bar similar to that presented in Quick Search section, enables selection of the object type you want to search for.

☛ *The search bar in the advanced search tool does not enable searches from a selection of object types as it is the case for the workspace search bar.*

You can perform a simple search using this bar, or perform a search using the parameters that appear in the tree.

To query using the assistant:

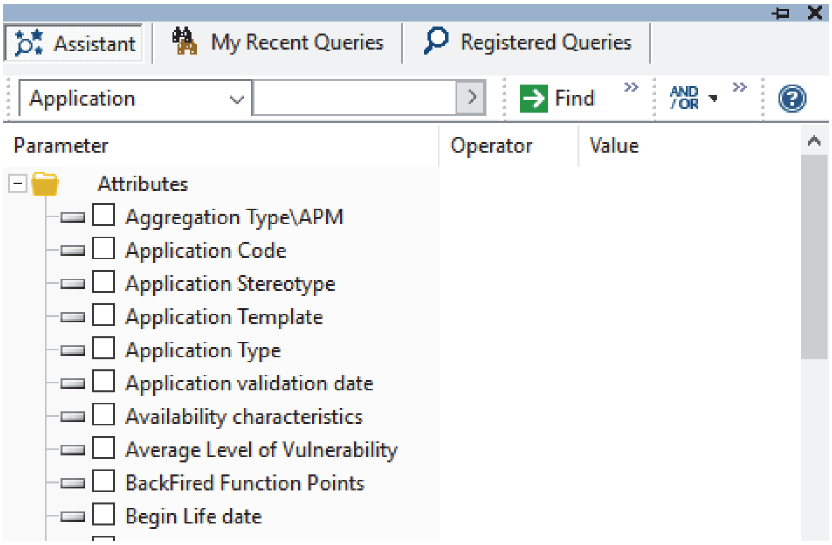
1. Access the advanced search tool.

   ☛ *See Advanced Search Tool Description.*

   The advanced search tool appears and displays the **Assistant** tab.

2. In the search bar, in the first field, select the type of object to query.

   `Example: Application.`



3. In the parameter field, select the parameter concerned by the search.

   ☛ *See Using the query assistant tree.*

   ☛ *See Query parameters.*

4. In the **Operator** field, define the operator to be applied.

   ☛ *See Query operators.*

5. (If necessary) In the **Value** field, define the parameter value.

   ☛ *See Query parameter values.*

6. (If needed) Display the query code.

   ☛ *See Displaying the query code.*

7. Click **Find** →.
   The result list is displayed.

   ☛ *See Query Results.*

   ☛ *To save the query, see Saving a query from the assistant.*

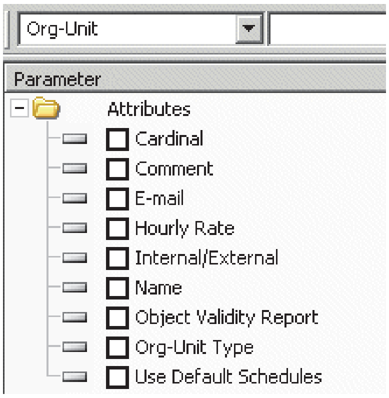
Using the query assistant tree

For each target object type, the query assistant proposes:

- its **Attributes**
- the **Links** to the types of objects that can be connected.

Target object type characteristics

You can use characteristics (attributes) of the target object type as parameters (for example, name and comment, which are valid characteristics for all object types).



Org-Unit

Parameter

Attributes
- □ Cardinal
- □ Comment
- □ E-mail
- □ Hourly Rate
- □ Internal/External
- □ Name
- □ Object Validity Report
- □ Org-Unit Type
- □ Use Default Schedules

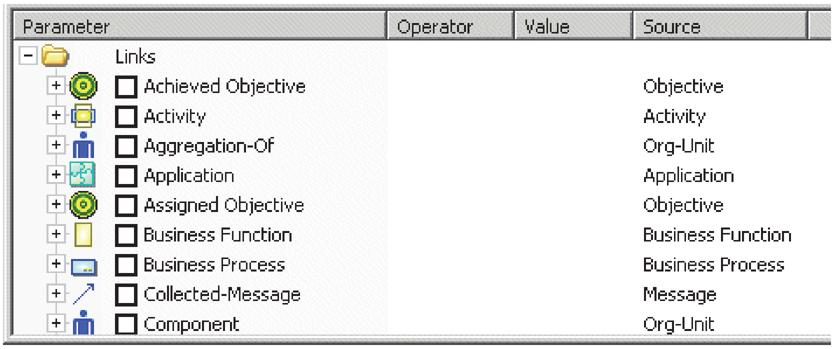
Links

You can also use link characteristics as parameters.

Object types that can be connected to the target object type appear under the "Source" heading.

☛ *To sort the links by object type, click this* ***Source*** *column.*

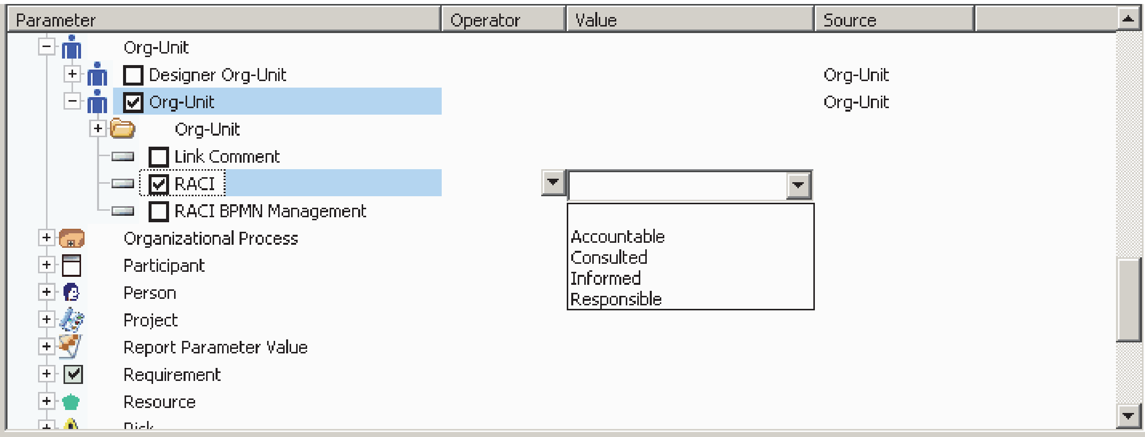
Each link groups the characteristics of the link and associated objects.



Parameter	Operator	Value	Source
Links			
☐ Achieved Objective			Objective
☐ Activity			Activity
☐ Aggregation-Of			Org-Unit
☐ Application			Application
☐ Assigned Objective			Objective
☐ Business Function			Business Function
☐ Business Process			Business Process
☐ Collected-Message			Message
☐ Component			Org-Unit

☛ *The **Links by type** folder shows the links sorted by the object type concerned.*

"Involvement", "Follow-up Policy", "RACI", "Performer status" are properties of the link between Org-Unit and Organizational Process in the example below.

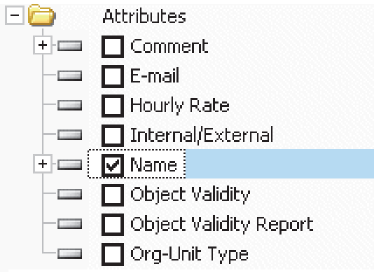


Parameter	Operator	Value	Source
Org-Unit			
☐ Designer Org-Unit			Org-Unit
☑ Org-Unit			Org-Unit
Org-Unit			
☐ Link Comment			
☑ RACI		Accountable Consulted Informed Responsible	
☐ RACI BPMN Management			
Organizational Process			
Participant			
Person			
Project			
Report Parameter Value			
Requirement			
Resource			
Risk			

Query parameters

The **Parameters** column contains the names of the query parameters (characteristics, links, associated objects). Check boxes enable indication of the parameters that define the query.

- Select (or clear) the parameters you want to take into account to build your query.



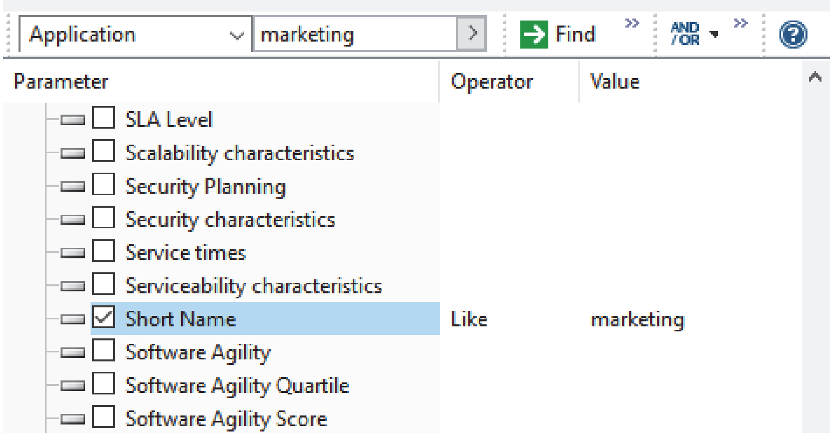
Attributes
- □ Comment
- □ E-mail
- □ Hourly Rate
- □ Internal/External
- ☑ Name
- □ Object Validity
- □ Object Validity Report
- □ Org-Unit Type

Selected links or properties appear highlighted to facilitate the query build and readability.

☛ *To reset all the parameters, click (**Remove all query criteria**).*

When you enter a name in the second field of the assistant query bar, the **Name** parameter is initialized using this value.

☛ *Check that you have selected one of the options offered by the arrow in the second field (e.g.: **Name starting with**, **Name containing**).*



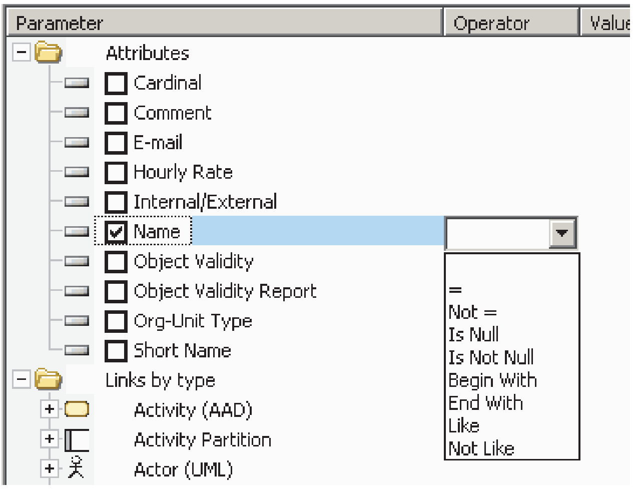
Parameter	Operator	Value
□ SLA Level		
□ Scalability characteristics		
□ Security Planning		
□ Security characteristics		
□ Service times		
□ Serviceability characteristics		
☑ Short Name	Like	marketing
□ Software Agility		
□ Software Agility Quartile		
□ Software Agility Score		

By default, the tree includes the main search parameters only. To extend the list of proposed parameters to:

- the administration properties, see Displaying administration properties.
- (If you work in a multilingual environment) translations, see Displaying translation properties.

Query operators

Operators are defined as a function of the selected parameter.



And, Or, and Not Operators

In the assistant toolbar of the advanced search tool, the AND/OR button enables use of the "And", "Or", and "Not” operators.

These operators concern all parameters selected in the assistant.

The parameters concerned by these operators are highlighted in a color that varies according to the operator (blue for "And", green for "Or", orange for "Not").

- ***And***
  The **And** operator means **ALL conditions must be fulfilled**.

```
Select [Project] Where [Progress percentage] ="100%"
     And [Project manager names] ="Brown"
     And [Project diagram] Is Not Null
```

- ***Or***
  The **Or** operator means **AT LEAST ONE condition must be fulfilled**.

```
Select [Org-Unit] Where [Message-Sent]
     Or [Message-Received]
     Or [Message-Collected]
```

- ***Not***
  The **Not** operator means **NO condition must be fulfilled**.

```
Select [Org-Unit] Where Not [Message-Sent]
     And Not [Message-Received]
     And Not [Message-Collected]
```

Inherited objects

> ☛ *For more details on inherited objects, see Object Variations.*

In case of use of object variations (Activating Variations), by default, the **Inherited Objects** option is selected. The code generated by the query assistant includes a keyword mentioning the inheritance.

Examples of query codes taking account of inheritance:

```
Select [IT Service] Inherited Where [Defining-Application] =
'Myapplication V2.0"
```

This query retrieves all services defined for the "Myapplication V2.0" application , including those inherited.

```
Select application inherited Where [Functionality] = &func
```

This query retrieves all applications that carry the "My func" functionality either directly or via inheritance.

Deactivating inheritance in the query assistant

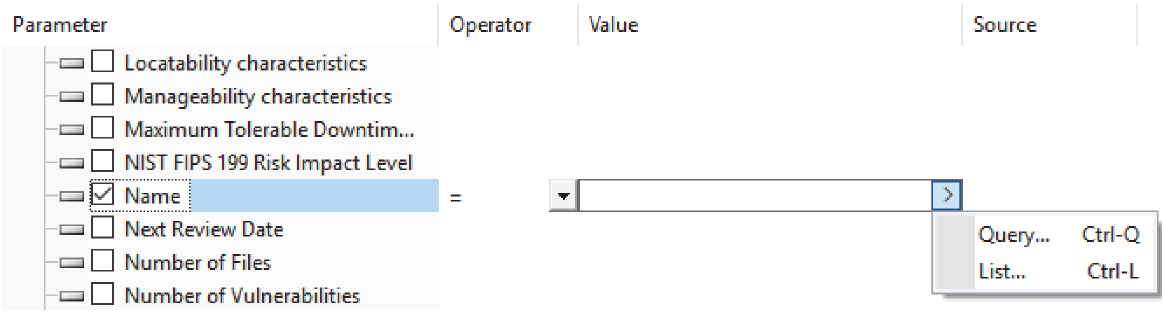
To deactivate inheritance in the query assistant:

1. In the query assistant, click AND/OR.
2. Clear **Inherited Objects**.

> ☛ *The **Inherited Objects** option of the assistant window impacts only the code generated by this assistant.*

Query parameter values

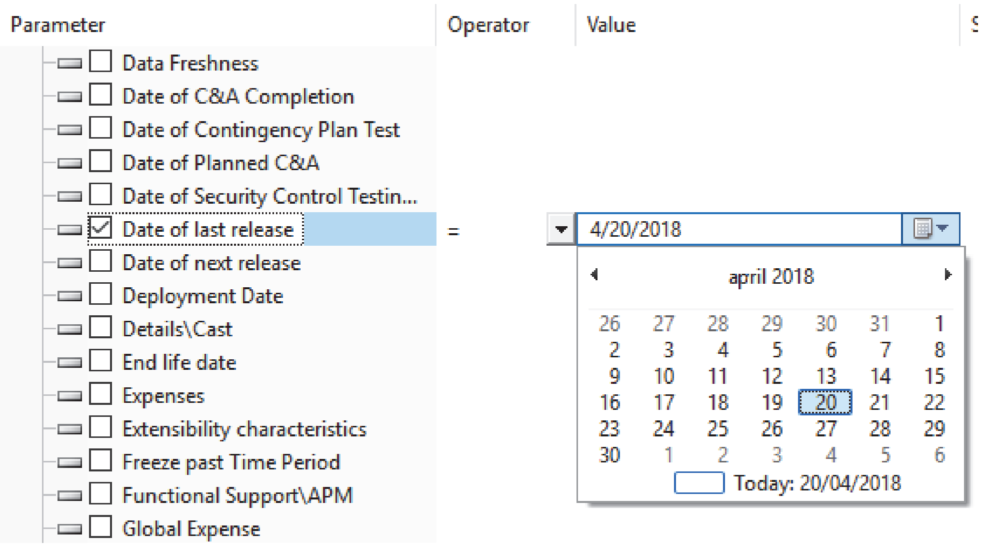
The **Value** field enables to enter or search for the parameter value.


Parameter
Operator
Value
Source
☐ Locatability characteristics
☐ Manageability characteristics
☐ Maximum Tolerable Downtim...
☐ NIST FIPS 199 Risk Impact Level
☑ Name
=
☐ Next Review Date
☐ Number of Files
☐ Number of Vulnerabilities
Query... Ctrl-Q
List... Ctrl-L

Depending on the parameter type, some tools help you enter its value.

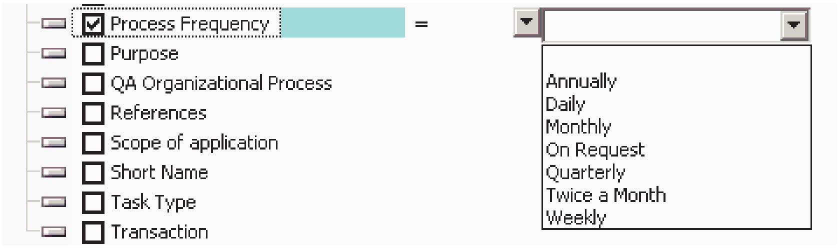
For example:

- to enter a date, a calendar is proposed.



Parameter	Operator	Value
□ Data Freshness		
□ Date of C&A Completion		
□ Date of Contingency Plan Test		
□ Date of Planned C&A		
□ Date of Security Control Testin...		
☑ Date of last release	=	4/20/2018
□ Date of next release		
□ Deployment Date		
□ Details\Cast		
□ End life date		
□ Expenses		
□ Extensibility characteristics		
□ Freeze past Time Period		
□ Functional Support\APM		
□ Global Expense		

- to enter a frequency, a list of values is proposed.



Parameter	Operator	Value
☑ Process Frequency	=	
□ Purpose		
□ QA Organizational Process		
□ References		
□ Scope of application		
□ Short Name		
□ Task Type		
□ Transaction		

Displaying the query code

The query code is built as you progressively select the boxes and complete the fields.

Values entered in the fields are kept when you clear the box (so as to note the parameter corresponding to the query code - you can then quickly reintegrate it by rechecking the box).

By default, the query code is not visible in **Assistant** mode.

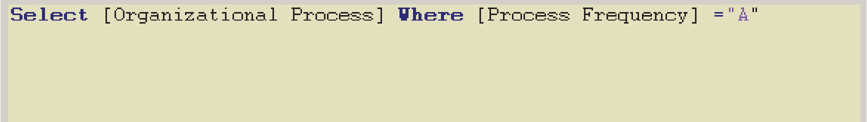
To display the query code:

› In the assistant toolbar of the advanced search tool, click **Show** ▦ > **Query Code**.
The query code is displayed at the bottom of the **Assistant** tab.
You cannot modify this code.
If needed, you can copy (Ctrl+C) the query code to use it elsewhere or to save it to create a new query.

☛ *See Saving a query from the assistant.*

Saving a query from the assistant

When **Show > Query Code** is selected, the query code appears at the bottom of the assistant.



```
Select [Organizational Process] Where [Process Frequency] ="A"
```

You can save it by creating a query.

To create a query from the code obtained in the assistant:

1. Display or create a query.

   ☛ *See Querying Using the Query Assistant.*

2. In the assistant menu bar, click **Save as** .

   The query creation window opens.
3. In the **Name** field, enter a name for your query.
4. In the **Stereotype** field, select the query type.

   ☛ *See Query type.*

5. Click **Next** to reread the code of your query.
6. Click **Finish**.
   Your query is available in the **Registered Queries** tab.

Using Recent Queries

In the advanced search tool, **My Recent Queries** tab provides fast access to the latest queries you have executed.

Accessing your recent queries

To access the last queries you have launched:

1. Open the advanced query tool.

   ☛ *See Searching from the workspace.*

2. Select the **My Recent Queries** tab.
3. By default the object type on which your last query dealt with is selected; if needed, in the drop-down list, select another object type.
4. Select a query in the list.
   The query code is displayed at the bottom of the list.

Defining recent queries

My Recent Queries mode uses the "Temporary Query" object type (or MetaClass) to temporarily memorize your queries.

They contain the final code that produced the objects listed in the result tab. These objects are specific to each user.

A recent query is not intended to be kept indefinitely. If you want to keep it, you must save it.

☛ *See Saving a query from the assistant.*

Configuring the number of recent queries

For each user, a defined number of recent queries is memorized.

When this number is reached, the oldest recent queries are deleted, except if they are among your favorites.

To configure the number of memorized recent queries:

1. In **HOPEX** menu bar, select **Tools > Options**.
2. Select the **Queries** folder.
3. In the right pane, in the **Number of Recent Queries Kept** field, modify the value.

☛ *The default value is 50.*

Examples of recent queries

A recent query is a query that has been resolved and is directly executable. If the query includes parameters, the recent query contains the values of these parameters.

Query	Parameter values	Temporary query code memorization
Select [Project] Where [Short name] ="&[Short name]	HBC for &[Short name]	SELECT [Project] WHERE [Short name] ="HBC"
Select [Project] Where [Start date] = "&Project.Start date" And [Project manager names] = "&Project.Project manager names"	"15/02/2004" for "&Project. Start date" and "Brown" for &Project.Project manager names"	Select [Project] Where [Start date] = ""15/02/2004"" And [Project manager names] = ""Brown""
Select [Organizational Process] Where [Name] Like "Com#"		Select [Organizational Process] Where [Name] Like "Com#"

EDITING QUERIES

Registered Queries mode facilitates access to queries by target object type and update. You can also create your own queries from this mode.

Using a Repository Query

Queries can be of two types:

- ***Internal***: includes “technical” queries used in standard supplies (descriptors, report templates (MS Word), matrices, etc.), “Name” and "*".
- ***Usual***: includes queries programmed by users (proposed by default) as well as the queries "*" and “Name”.

To access repository queries:

1. Open the advanced search tool.

   ☛ *See Searching from the workspace.*

2. In the **Registered Queries** tab, select an object type in the drop-down list.

To filter queries according to their type:

1. In the assistant toolbar of the advanced search tool, click **Show** ▦ **> Query type**.
   The **Display Queries** window opens.
2. Select the query types you want to be displayed.

   ```
   Example: “Internal query”.
   ```

3. Click **OK**.
   The list of registered queries is updated depending on the type selected.

Creating a query

To create a query:

1. Open the advanced search tool.

   ☛ *See Searching from the workspace.*

2. Click the **Registered Queries** tab.
3. In the drop-down list, select the object type for which you want to create a query.
4. Click **Create New query**.
   Your query is initialized as follows:

   ```
   "Select xx where", where "xx" is the name of the target object.
   ```

5. Enter your query code.
   HOPEX offers help when you are in doubt on the possibilities offered by the ERQL syntax.

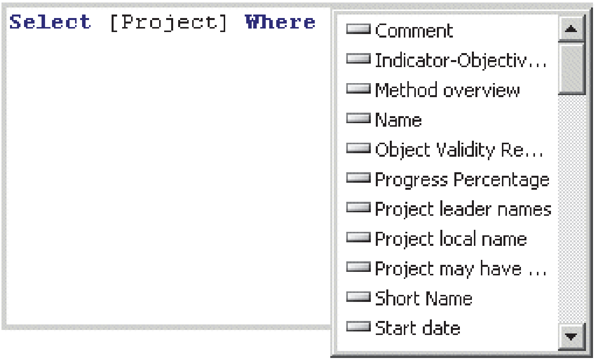
   ☛ *For more details, see Using Help When Entering Queries.*

   ☛ *You can analyze your query code before execution by clicking* ✓ *.*

Using Help When Entering Queries

To use help during query code build:

1. Press <Ctrl> + <Space>.
   A drop-down list proposes the possibilities for completion of your code.



```
Select [Project] Where
```

   As in **Assistant** mode, properties proposed in this drop-down list are the most common. To add other properties, modify the display parameters, see:
   - Displaying administration properties
   - Displaying translation properties
2. Select an object type to complete your query.

Displaying administration properties

To display the administration properties:

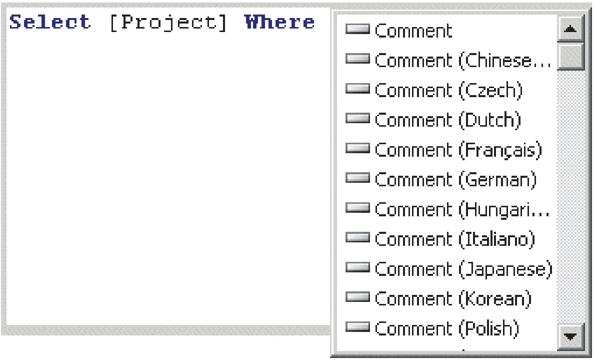
1. Open the advanced search tool.
   ☛ *See Searching from the workspace.*
2. Click the **Registered Queries** tab.
3. In the assistant toolbar of the advanced search tool, click **Show** **> See administration properties**.
   The drop-down list is updated and now includes the administration properties.

Displaying translation properties

To display language-dependent properties:

1. Open the advanced search tool.
   ☛ *See Searching from the workspace.*
2. Click the **Registered Queries** tab.

3. In the assistant toolbar of the advanced search tool, click **Show** ▦ **> See translation properties**.
   The drop-down list is updated and now includes translations of each characteristic in the different languages.



```
Select [Project] Where
```

- Comment
- Comment (Chinese...
- Comment (Czech)
- Comment (Dutch)
- Comment (Français)
- Comment (German)
- Comment (Hungari...
- Comment (Italiano)
- Comment (Japanese)
- Comment (Korean)
- Comment (Polish)

Pasting Object Names in Queries

☛ *See also Quick Search.*

An object name can be inserted in a query in the form of a field.

In this way, when you enter a query of type:

```
Select [Org-Unit] Where name = xxx
```

you can copy the object name and paste it in the query.

So when you modify the object name you have copied, the query will be automatically updated.

To copy an object name, right-click the object (in a diagram or in the navigator) and select **Copy**.

Query Results

After having executed a query by clicking **Find**, the **Result** tab appears.

To select all objects in the result tab:

❱ Press the <Ctrl> and <A> keys simultaneously.

Two display modes are proposed:

- **Reinitialize** mode
- **Cumulative** mode

Constituting Results

Reinitializing results

By default, the result tab content is reinitialized at execution of each query. In other words, the result tab is emptied and filled at execution of each query.

☛ ***Reinitialize*** *is the only available mode when a search is performed from the search bar.*

Cumulating results

You can cumulate the results of several queries in the same tab.

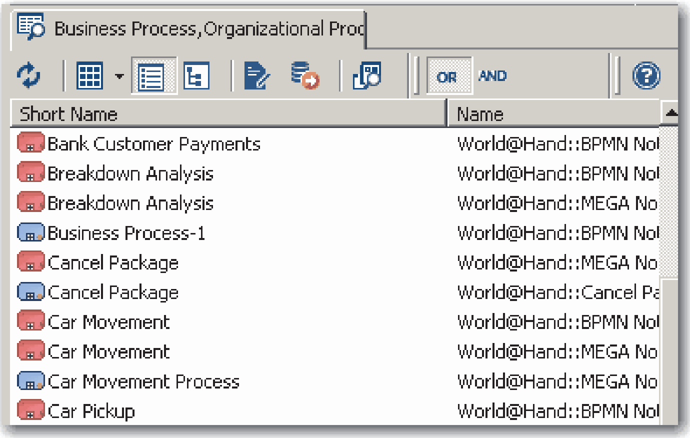
☛ *The workspace search toolbar does not allow result cumulation. Content is systematically reinitialized.*

To get the **Cumulative** mode:

- In the assistant toolbar of the advanced search tool, click > **Cumulative**.

  ☛ *To get a cumulative result, do not close the result window after running queries.*

  In **Cumulative** mode, the result tab can list objects of heterogeneous types (for example, processes and org-units).



Short Name	Name
Bank Customer Payments	World@Hand::BPMN Not
Breakdown Analysis	World@Hand::BPMN Not
Breakdown Analysis	World@Hand::MEGA No
Business Process-1	World@Hand::BPMN Not
Cancel Package	World@Hand::MEGA No
Cancel Package	World@Hand::Cancel Pa
Car Movement	World@Hand::BPMN Not
Car Movement	World@Hand::MEGA No
Car Movement Process	World@Hand::MEGA No
Car Pickup	World@Hand::BPMN Not

Displaying objects common to several result sets

When the cumulative mode is selected, to display objects common to several sets of results, click OR.

☛ *This button appears only when Cumulative mode is selected.*

Displaying Results of the Advanced SearchTool

Creating a new tab in the result tab

You can create a new tab so as to avoid mixing results.

To create a new tab in the result window:

1. Perform a search.

   ☛ *See Querying Using the Query Assistant.*

2. In the search result window, click **Keep result in a new window** .
   Your result appears in a separate tab, allowing it to be kept in case of re-execution of a query.

Customizing result tab columns

By default, the result tab presents the following columns for each object:

- short name
- long name

You can customize these columns. Each column corresponds to an object property.

To customize columns of the search result window:

1. Perform a search.

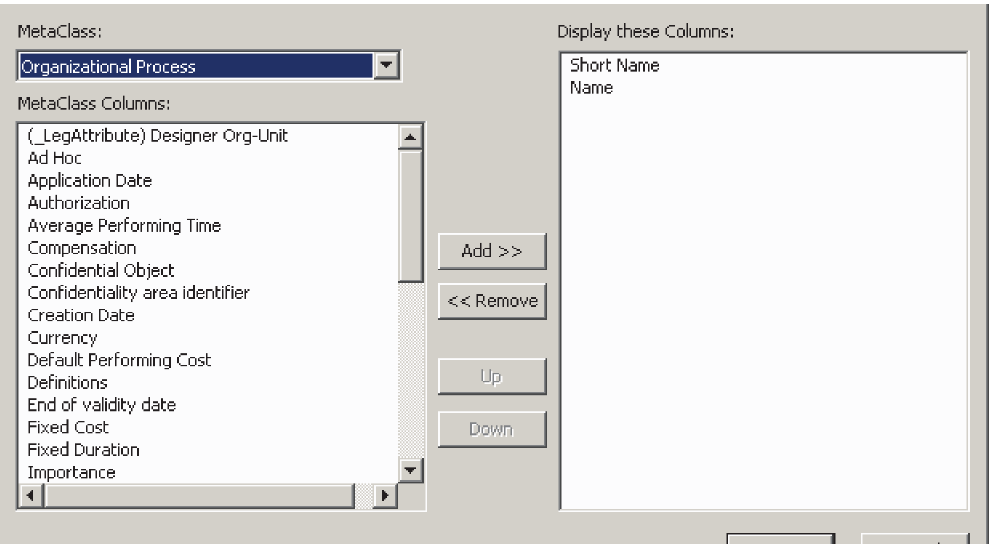
   ☛ *See Querying Using the Query Assistant.*

2. In the search results window, click **Show** ▦ **> Customize columns**.

   ☛ *The **Use Generic Objects** option presents the objects resulting from the query as abstract MetaClass instances.*

   *For more details on abstract MetaClasses, see "Abstract Metamodel" in **HOPEX Power Studio - Studio**.*

   The **Column Display Customization** dialog box allows you to add or remove columns for a given MetaClass.



MetaClass: Organizational Process

MetaClass Columns:

(_LegAttribute) Designer Org-Unit
Ad Hoc
Application Date
Authorization
Average Performing Time
Compensation
Confidential Object
Confidentiality area identifier
Creation Date
Currency
Default Performing Cost
Definitions
End of validity date
Fixed Cost
Fixed Duration
Importance

Display these Columns:

Short Name
Name

Each new column corresponds to a MetaClass characteristic.

☛ *If you want to sort result window results by object type, add the MetaClass column.*

☛ *For object types in a namespace, the short name is also indicated by default.*

Memorizing columns

To store in memory the columns to be displayed when executing future queries:

1. Access the user options (**Tools > Options**).
2. In the Options tree, select **Query**.
3. In the right pane, select the **Result: Keep displayed columns** option.
   The next time you perform a query, the result window proposes the same columns as those previously defined.

Reinitializing columns

The list of columns displayed depends on the object types of the query.

To reinitialize the columns:

1. Access the user options (**Tools > Options**).
2. In the Options tree, select **Query**.
3. In the right pane, select the **Result: Reinitialization of result list columns** option.
   After each query execution, columns corresponding to characteristics not connected to queried targets are removed.

Displaying results as a list

By default, results appear in the form of a list ☰.

When you click **Display result objects list**, duplicates are eliminated (i.e. duplicates of objects that appear in several recent queries).

```
For example, if the "Urgent Order" organizational process appears in
several recent queries (ie. it appears several times in the query tree),
it will appear only once in the list.
```

Display result objects list *is equivalent to* OR in the result window.

Displaying queries that produced the result

You can display results as a tree, which enables viewing of recent queries that produced result object collections.

To display query results as a tree:

1. Click **Display tree of recent queries and result objects** .
2. Double-click the recent queries to show the result objects.
   You obtain a tree enabling identification of the different queries executed with their result sets.

Displaying objects common to several result sets

The AND button is used to restrict the view to only those objects common to all recent queries.

☛ OR *and* AND buttons only appear when Cumulative mode is selected.

Refreshing result window content

You can rerun execution of queries that produced the list of objects in the result tab.

To refresh results:

- In the query results window, click **Refresh** ⟳ .

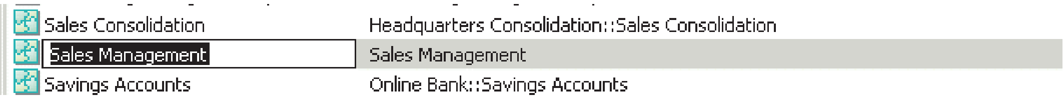
Modifying Objects in the Result Window

You can use the query result window to update the characteristics of objects it contains.

Modifying an object

To modify an object from the result window:

- In the result window, right-click the object and select **Modify**.
  You can then modify the selected characteristic, for example the object name.



Sales Consolidation	Headquarters Consolidation::Sales Consolidation
Sales Management	Sales Management
Savings Accounts	Online Bank::Savings Accounts

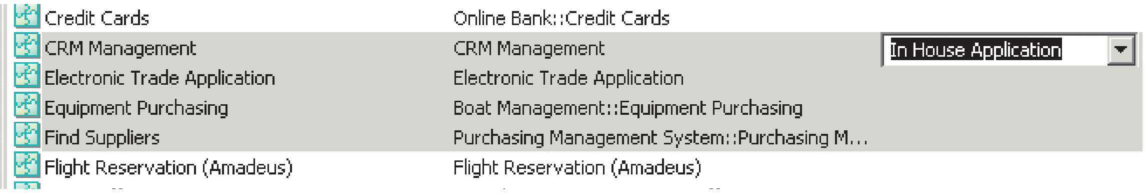
Modifying a selection of objects

You can also modify the characteristics of a selection of objects (for example, assign the same characteristic value to a selection of objects).

To display columns corresponding to the different object characteristics, see Customizing result tab columns.

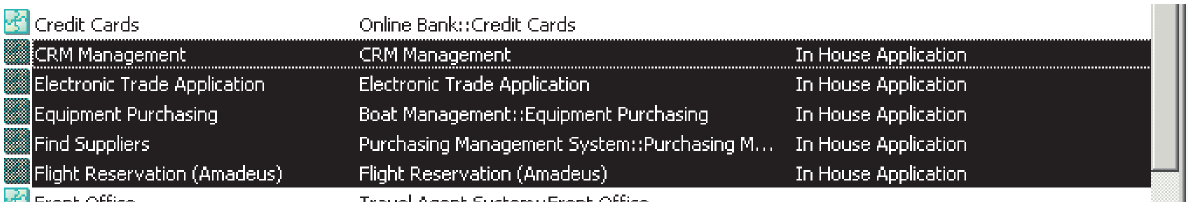
To modify a selection of objects from the result window:

1. Select the objects you want to modify.
2. Click the object characteristic you wish to modify and in the pop-up menu of the selection, select **Modify**.
3. Then modify the characteristic value.



Credit Cards	Online Bank::Credit Cards	
CRM Management	CRM Management	In House Application
Electronic Trade Application	Electronic Trade Application	
Equipment Purchasing	Boat Management::Equipment Purchasing	
Find Suppliers	Purchasing Management System::Purchasing M...	
Flight Reservation (Amadeus)	Flight Reservation (Amadeus)	

4. Press "Enter" to validate.
   The data is updated for all objects in the selection.



Credit Cards	Online Bank::Credit Cards	
CRM Management	CRM Management	In House Application
Electronic Trade Application	Electronic Trade Application	In House Application
Equipment Purchasing	Boat Management::Equipment Purchasing	In House Application
Find Suppliers	Purchasing Management System::Purchasing M...	In House Application
Flight Reservation (Amadeus)	Flight Reservation (Amadeus)	In House Application

Search and Confidentiality Result

☛ *If reading access management is activated, see the **HOPEX Administration - Supervisor** guide.*

Certain modeling projects may be confidential or contain confidential or sensitive data (Example: costs, risks, controls). The administrator may therefore decide to hide certain objects so that they are only visible to authorized users.

Visibility of objects by a user is managed in **HOPEX** by reading accesses. A reading access level is assigned to each user, which restricts his/her data visibility.

> ☛ *For more details on data reading access, see the **HOPEX Administration - Supervisor** guide.*

The search tool operates in dynamic mode. Searches take into account the reading access area of the user. Depending on the reading access area, the search results can therefore vary and display different numbers of objects.

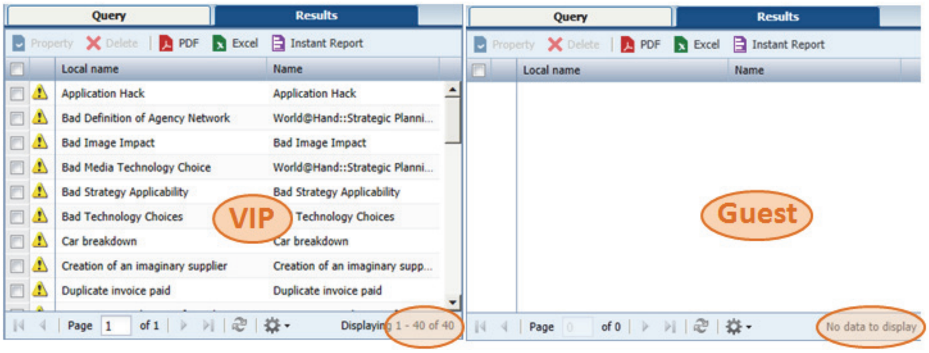
Objects not available in diagrams due to their reading access level appear in search results, but cannot be accessed by the user.

Search example

Take the example of a search for risks with two distinct users and their associated reading access level:

- "Guest" (standard level)
- "VIP" (maximum level)

Search on the **Risk** object returns 0 objects for the "Guest" user and 40 objects for the "VIP" user.



Local name	Name
Application Hack	Application Hack
Bad Definition of Agency Network	World@Hand::Strategic Planni...
Bad Image Impact	Bad Image Impact
Bad Media Technology Choice	World@Hand::Strategic Planni...
Bad Strategy Applicability	Bad Strategy Applicability
Bad Technology Choices	Technology Choices
Car breakdown	Car breakdown
Creation of an imaginary supplier	Creation of an imaginary supp...
Duplicate invoice paid	Duplicate invoice paid

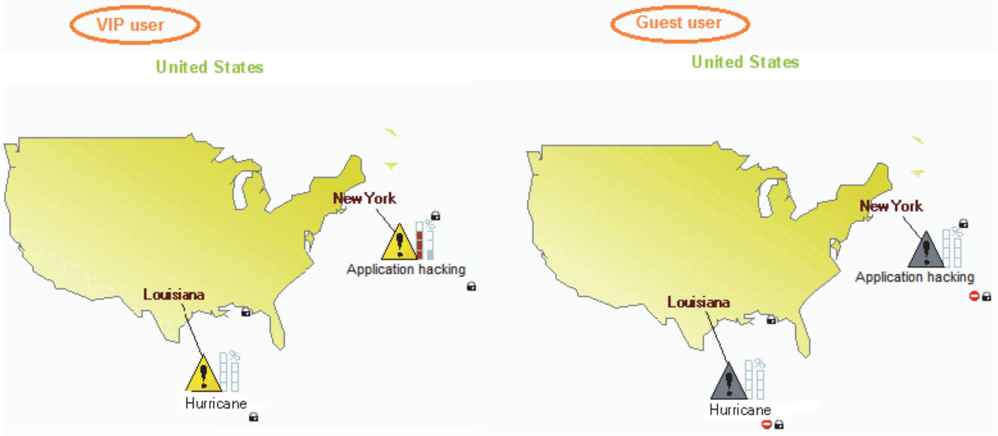
Page 1 of 1 Displaying 1 - 40 of 40

Local name	Name

Page 0 of 0 No data to display

On the following "Geographical Risk Map":

- for the "Guest" user, risks are grayed and their details are not accessible.
- having the maximum reading access level, the "VIP" user can access all risks in the diagram.



Generating Analysis Reports

To generate a report from objects selected:

1. In the query results, select the objects.

   ☛ *To select several objects, use the [Ctrl] key.*

2. Click **Generate Report From Selected Objects** .

   The report is displayed. The table columns are the same as those of the list at the time of report generation.

   ☛ *See Customizing result tab columns.*

   This tab allows you to quickly display the list of MetaClasses present in the result dialog box, and the corresponding number of objects.

3. (if needed) Click **Send To** to send by e-mail the report you have generated.

   ☛ *To modify the css style sheet of the query report, see Style sheet location.*

Handling Candidate Objects

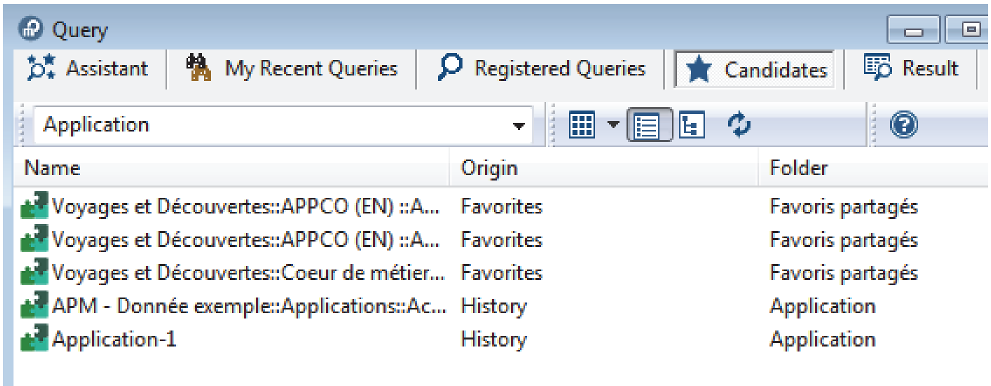
The **Candidates** tab of the advanced search tool appears when the advanced search tool is run from another tool such as the explorer, a diagram, etc.

It presents, by object type:

- objects from History
- objects from Favorites
- objects included in the **HOPEX** clipboard.

These are objects recently or currently used. This mode enables fast access to objects handled. It is useful when you call the query tool from a function that does not allow use of the navigator.

> ☛ *This tab does not appear if you run the query tool from the desktop (you therefore have access to the navigator, enabling you to easily handle repository objects and in particular your favorites and history).*



Name	Origin	Folder
Voyages et Découvertes::APPCO (EN) ::A...	Favorites	Favoris partagés
Voyages et Découvertes::APPCO (EN) ::A...	Favorites	Favoris partagés
Voyages et Découvertes::Coeur de métier...	Favorites	Favoris partagés
APM - Donnée exemple::Applications::Ac...	History	Application
Application-1	History	Application

To filter the object display as a function of their source:

1. In the search window toolbar, click **Show** ▦.
2. Clear the options from sources that you do not want to display:
   - Favorite objects
   - Recent objects
   - Clipboard

Configuring Query Tool Options

Accessing the Query Tool Configuration Options

To configure the query tool:

1. (**HOPEX Windows Front-End**) In the menu bar, select **Tools > Options**.
   The user options window opens.

   ☛ *To parameterize the query tool in **HOPEX Web Front-End**, click **Settings > Options** from the **Main Menu***
2. In the Options tree, select **Queries** folder.
   Configure the query tool in the right pane.
3. Select (or clear) the required options.

   ☛ *See Query Tool Configuration Options.*

Query Tool Configuration Options

You can modify, for example, the configuration of the following options:

Displaying the query tool and its tabs

You can define whether or not to display:

- the query tool: **Display Query Wizard** option
- tabs constituting the query tool:
  - queries: **Display Registered queries page** option
  - recent queries: **Display Recent Queries** option
  - candidate objects: **Candidate objects display** option

☛ *For more details on the different query modes and query tool tabs, see Querying Using the Query Assistant.*

Query target synchronization

The **Query target synchronization** option enables activation or deactivation of query target synchronization between workspace query bar and query tool.

- **Always**
  This value enables permanent synchronization of workspace query bar and query tool: if a target is selected in one of these devices, it will also be automatically selected in the other.
- **Sometimes (if the tool is not hidden)**
  The workspace query bar and the query tool are synchronized except if the query tool is hidden (**Auto Hide command**).
- **Never**
  This value deactivates all synchronization between workspace query bar and query tool.

Style sheet location

Report: Style sheet location option defines the style sheet location.

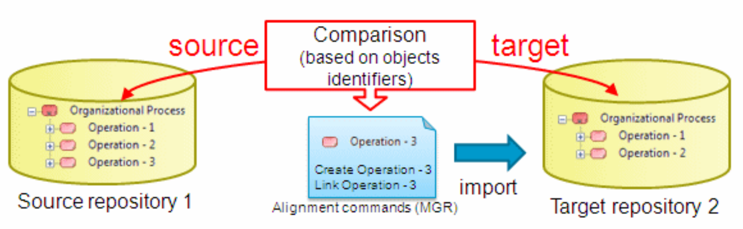
WINDOWS FRONT-END SPECIFIC FEATURES

This section describes features that are available with **HOPEX Windows Front-End** only.

MANAGING REPOSITORY TRANSFERS

Presented here are use cases of the alignment functionality, as well as modes of transfer to a production repository.

HOPEX enables transfer of objects from their original repository (source) to a destination repository (target).



☛ *For more details on the Compare and Align feature underlying the alignment feature, see the **HOPEX Administration - Supervisor** guide.*

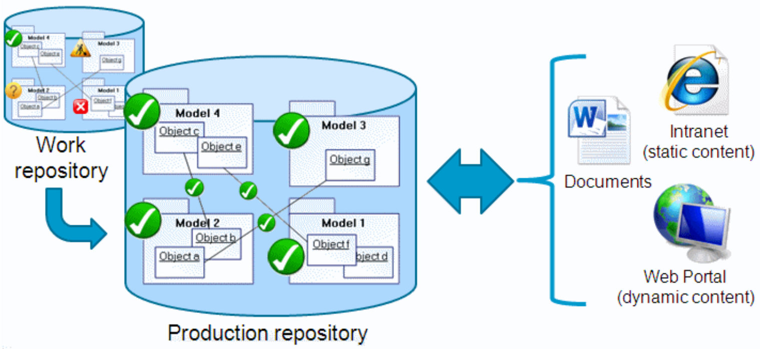
Alignment Use Case

In teamworking, it can be difficult to work on a single repository, projects and diagrams not all having the same production cycles. Certain diagrams are in course of creation, while others can be already validated or requiring update.

Implementing a Production Repository

It may be necessary to install a production repository so as to have permanently available models ready for dispatch.

This production repository is supplied by work carried out in the work repository when it is ready for dispatch.



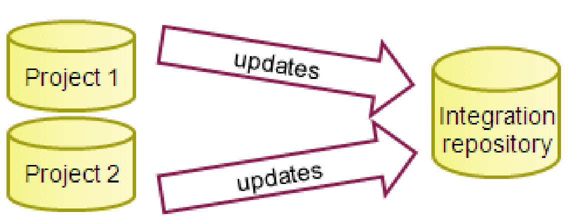
Transfering Objects Between Repositories

It can sometimes be necessary to transfer objects between repositories in a context other than that of the production repository.

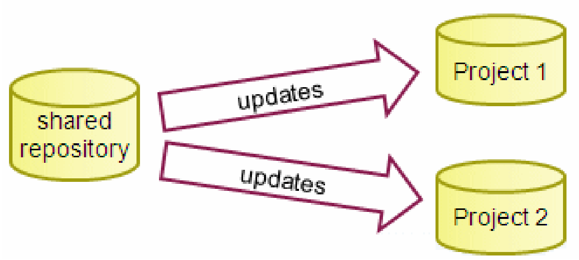
> ☛ *In these contexts, it can be necessary to define strict modeling rules to avoid conflicts.*

Some examples:

- Distinct projects and integration repository
  Modeling projects are initiated in distinct repositories. They are then transferred to an integration repository.



- Shared models
  Models are created in a single repository. They are then used in projects, each of which results in creation of a repository.



Modes of Transfer to a Production Repository

There are several possible modes of transfer of objects to a production repository.

Organizational and Technical Choices

Organizational choices impose constraints and influence the choice of transfer mode. Certain questions should be asked:

- Are modelers responsible for dispatch of their work?
- Is validation necessary before dispatch of models?
  Large teams generally require a validation step, which is not necessarily required by small teams.
- Which event triggers object transfer or validation?
- Who triggers transfer (modeler or validator)? From which repository (source repository or target repository)?

Depending on the selected organization, use of Compare and Align can be supplemented by:

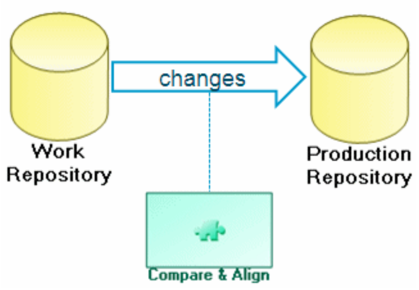
- design tasks
- validation workflow
- definition of perimeters, using the **HOPEX Power Studio** technical module.

  ☛ *For more details, see the technical article "Customizing Perimeters".*
- import/export files

Transfering Objects in Push Mode

"Push" mode allows the user to transfer modifications from the work repository. It is not necessary to connect to the production repository.

"Push" mode without validation

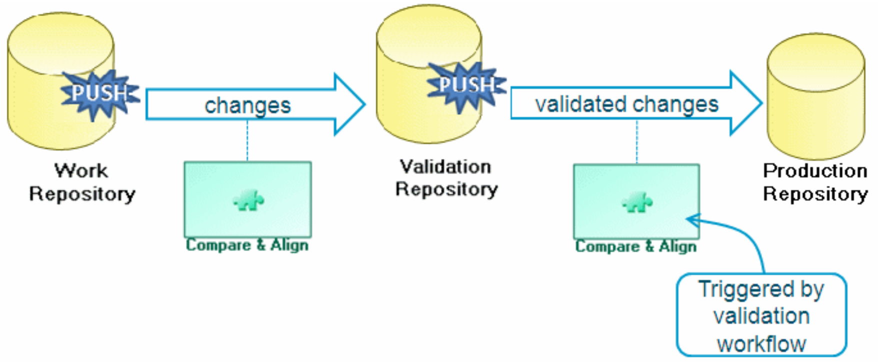


Variants are possible:

- transfer triggered at dispatch
- transfer with simultaneous dispatch

"Push" mode with validation

Data transfer by compare and align can be automatically triggered with installation of a validation workflow



Transfering Objects in Pull Mode

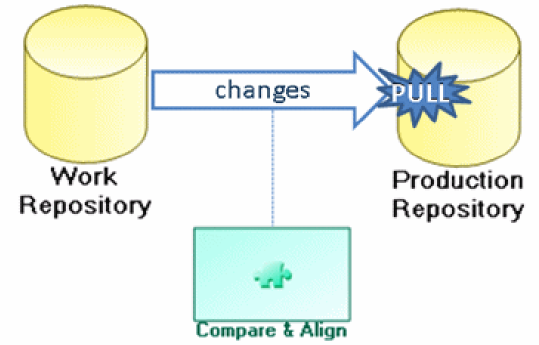
"Pull" mode requires connection to the production repository. Modifications are imported in a private workspace from the target repository.

If transfer is under the responsibility of users working in the source repository, this mode can prove to be restrictive. In this context, users connected to the source repository must:

- dispatch their private workspace in the work repository (source repository) and exit **HOPEX**
- open a private workspace in the target repository
- import objects from the target repository
- dispatch the private workspace in the target repository
- open a new private workspace in the work repository (source repository) to continue modeling work

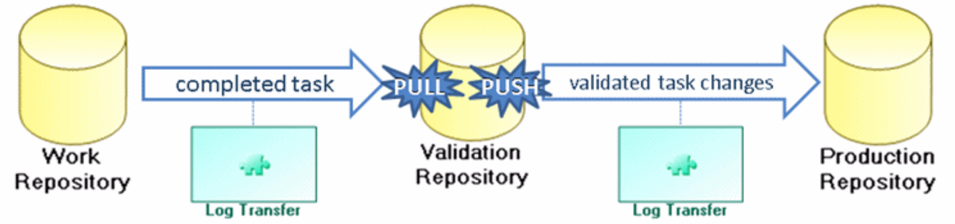
If however the transfer is under the responsibility of users controlling the transfer result before dispatch, "pull" mode has the advantage of allowing private workspace discard if necessary.

"Pull" mode with validation



- A model cannot be modified while it is awaiting validation
- These modifications are saved in the private workspace. The private workspace is dispatched if the model is validated.

"Pull" mode with use of design tasks

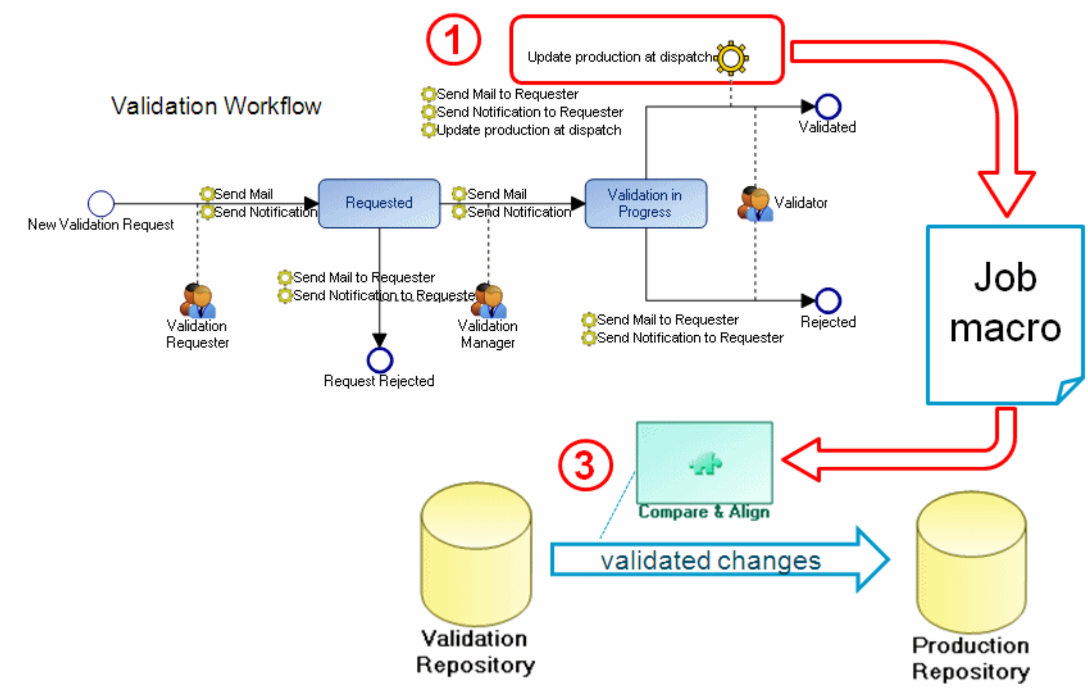


It may be necessary to check dependencies between tasks before export/import of the logfile.

Automatic alignment

The production repository is updated at the time of dispatch (1).

A "job" macro (2) automatically triggers alignment (3)



Transfering Objects

HOPEX allows you to transfer validated objects from one repository to another.
By default, only libraries can be transferred.

> ☛ *Other object types can however be transferred. To do this, you must first connect the corresponding MetaClass to the "Transfer Candidate Object" abstract MetaClass.*

Principle

Validated objects can be automatically transferred to another repository.

> ☛ *For more details on validation, see "Using the Validation Request Workflow".*

To be able to automatically transfer validated objects, you must first:

- create a transfer
- add the object to the transfer

Creating a Transfer

To create a transfer:

1. In the **Repository Activity** navigation window, expand the **Transfer Management** folder then the **Transfer Folder** sub-folder.
2. Right-click the **Not Scheduled Transfers** sub-folder and select **New Transfer**.

3. In the creation of transfer dialog box, select a transfer template if required and click **Next**.
4. In the transfer definition dialog box, enter the necessary elements if the transfer template has not already specified these.
   - source repository
   - target repository
   - transferred object
5. Click **Next**.
6. If required, select a person to be informed in the **Informed User** frame.
7. Click **Finish**.
8. In the final dialog box, select the transfer execution mode:
   - Execute at scheduled date (specifying date/time in the **Scheduled Transfer Date Time** field)
   - Execute as soon as possible

   ☛ *You can schedule transfer later. See "Scheduling a Transfer".*
9. Click **OK**.

Scheduling a Transfer

When the transfer has been created, you must schedule it.

To schedule a transfer (if not scheduled at the time of its creation):

1. In the **Repository Activity** dialog box, expand the sub-folder **Transfer Management > Transfer Folder > Not Scheduled Transfers**.
2. Right-click the transfer concerned and select **Schedule Transfer**.
3. In the dialog box that appears, select:
   - execution mode "Execute at Scheduled Date and Time"
   - the date and time concerned
4. Click **Finish**.
   When scheduled, the transfer appears under the **Scheduled Transfers** sub-folder.

Adding an Object to a Transfer

To add an object to a transfer:

1. Right-click the object to be transferred and select **Add to Transfer**.
2. In the list that appears, select the required transfer and click **OK**.

   ☛ *You can schedule the transfer at object validation.*

Transfering the Object

The object can be transferred:

- if it has been validated
- if it has been added to a transfer

The object is transferred:

- immediately if the transfer has not been scheduled
- at a time specified in transfer scheduling

☛ *The transfer is executed in the name of the person who saved the transfer.*

Managing Transfers and Transfer Templates

To view the transfers:

1. In the **Repository Activity** navigation window, expand the **Transfer Management** folder.
2. In the **Transfer Folder** sub-folder, you can view:
   - not scheduled transfers
   - scheduled transfers
   - executed transfers

To create a transfer template:

1. In the **Repository Activity** navigation window, expand the **Transfer Management** folder.
2. Right-click the **Transfer Templates** folder and select **New > Transfer Template**.
3. Indicate:
   - a source repository
   - a target repository
   - the object to be transferred
4. Click **Finish**.

☛ *You can now use this transfer template when creating a transfer.*

USING DESIGN TASKS

A design task enables assignment of execution of work to be done in the modeling repository. The design task assures traceability and check of work carried out by modelers by associating dispatched modifications with a motive.

> 🕮 *A motive is an object connected to a design task that justifies or explains why the design task has been created. The request for change is an example of task motive.*

Management of design tasks integrated in **HOPEX** improves control and analysis of repository modifications. In particular it is used to:

- check user dispatch rights (the user can be obliged to associate private workspace modifications with a design task).
- identify modifications to be transferred from one repository to another.

Prerequisites for Using Design Tasks

To be able to use design tasks, make sure you have:

- Activated task management.
- Activated the logfile of the repository and/or the system repository for which you want to keep history of modifications associated with tasks.

> ☛ *For more details on repository log activation, see the **HOPEX Administration-Supervisor** guide, "Managing Repositories", "Managing Logfiles".*

To activate task management:

1. From the administration application, open the desired environment.
2. Right-click the environment and select **Options > Modify**.
3. In the **Collaborative Environment > Change Management** option group, select the "Enabled" value for the **Associate modifications with a design task** option.
   Task management is activated.

> ☛ *The option can also be defined at user level.*

The **Associate modifications with a design task,** option has three values:

- if task management is **Disabled**, the user cannot dispatch his/her private workspace as a task
- if task management is **Enabled,** the user can use tasks, and can choose whether or not to associate a task with the work dispatched
- if task management is **Mandatory**, the user can only dispatch his/her work by associating it with a task.

When dispatching the private workspace as a task, the user associates all dispatched modifications with the assigned task.

Accessing Design Tasks

To access design tasks:

- From the navigation tree, select **Collaboration > Overview > My Tasks**.

HOPEX enables you to view:

- **tasks to complete**: these are tasks assigned to the current user
- **tasks to validate**: these are tasks that the current user must validate, or not
- **requested tasks**: these are all design tasks assigned to the current user

Creating Design Tasks

To create a design create:

1. See "Accessing Design Tasks".
2. In **Collaboration > Collaboration Overview > My Tasks**, select **Requested Tasks** in the drop-down menu and click New.
   The dialog box for creating a design task appears.
3. Enter the name of the design task.
4. If applicable, specify the design task motive and due date and the EA project connected to the design task, then click **OK**.

   ☛ *A motive is an object connected to a design task that justifies or explains why the design task has been created. The request for change is an example of task motive.*
5. In the dialog box that appears, specify if you want to:
   - create the task and assign it
   - create the task and assign it later
6. In the next dialog box, enter a comment and click **Next**.
7. Select the user who must execute the task and click **Finish**.
   The design task is created and if specified, assigned to the selected user.

Specifying design task motive

Design tasks can be created from:

- a request for change

  📖 *A request for change enables expression of a comment concerning change to be made to an object.*
- a requirement

  📖 *A requirement is a need or expectation explicitly expressed, imposed as a constraint to be met within the context of a project. This project can be a certification project or an organizational project or an information system project.*

These two object types constitute the motive of a design task.

It can be useful, from a request for change, to create and assign a design task to the person responsible for its execution.

To create a design task from its motive (here, a request for change):

1. See Accessing Requests for Change.
2. Expand the navigation folder corresponding to the request for changes.
3. In the page corresponding to your change request, click **Expand** to create a design task.
4. In the dialog box that appears, enter the name of the design task and indicate the task due date.
5. Click **OK**.

6. In the dialog box that appears, specify if you want to:
   - create the task and assign it
   - create the task and assign it later
7. Enter a comment for the design task, then click **Next**.
8. Select the design task assigned user if required, then click **Finish**.

The design task is created.

Assigning and Aborting a Design Task

Assigning a design task later

When you created the design task, you were able to create it without immediately assigning it.

To assign a design task after creation:

1. In the list of required tasks, click .
2. In the right part of the window, click on the button used to move on to the next step in the workflow.

Discarding a design task

If a design task has been created but not immediately assigned, it can be aborted.

To abort an unassigned task:

1. In the list of required tasks, click .
2. In the right part of the window, click on the button used to move on to the next step in the workflow and select **Abandon**.

Dispatching as a Design Task

A user can dispatch work in a private workspace as an assigned design task if the task management function is activated.

When dispatching work in the private workspace as a task, the user associates all dispatched modifications with the assigned task.

To dispatch work in a private workspace as a design task:

1. In the **HOPEX Web Front-End** desktop, save and quit while dispatching.
2. In the dialog box that appears, select the required design task if you have not already done so.
3. If this dispatch contains all the modifications necessary for resolution of the design task, select the **Complete Task** check box.

   ☛ *You can dispatch work from several private workspaces as the same design task. However, you cannot group several design tasks in the same private workspace.*
4. Select **Yes**.

Customizing Documentation

CUSTOMIZING REPORTS (MS WORD)

This section presents customization of reports (MS Word) using report templates (MS Word). The generation mechanism is also presented.

The following points are covered here:

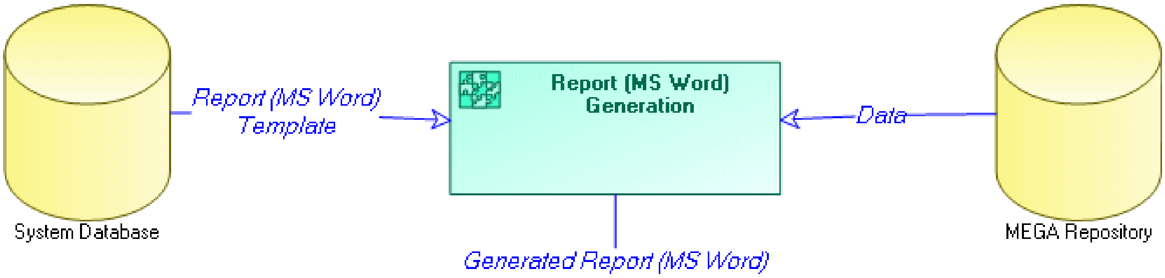
Report (MS Word) Customization Functions

You can use the edit functions of **HOPEX** to produce high quality reports using Microsoft Word for Windows™ (from now on referred to as Word).

Reports (MS Word) are created from report templates (MS Word), in the same way as ***MS Word*** documents are created from ***MS Word*** document templates.

Report (MS Word) storage

These report templates (MS Word) are stored in the **MEGA** system repository. This system repository contains all standard supplies that enable **MEGA** operation, for example report templates (MS Word), queries, Web site templates, etc.


System Database
Report (MS Word) Template
Report (MS Word) Generation
Data
MEGA Repository
Generated Report (MS Word)

Explained here is how you can create and customize report templates (MS Word) to produce reports (MS Word) matching your company standards.

Use of MS Word

To effectively handle reports (MS Word), you do not need to have an in-depth knowledge of ***MS Word***. However, if you want to modify the content or format of reports (MS Word) and report templates (MS Word), you must be familiar with the styles, document templates, and fields as used in MS Word.

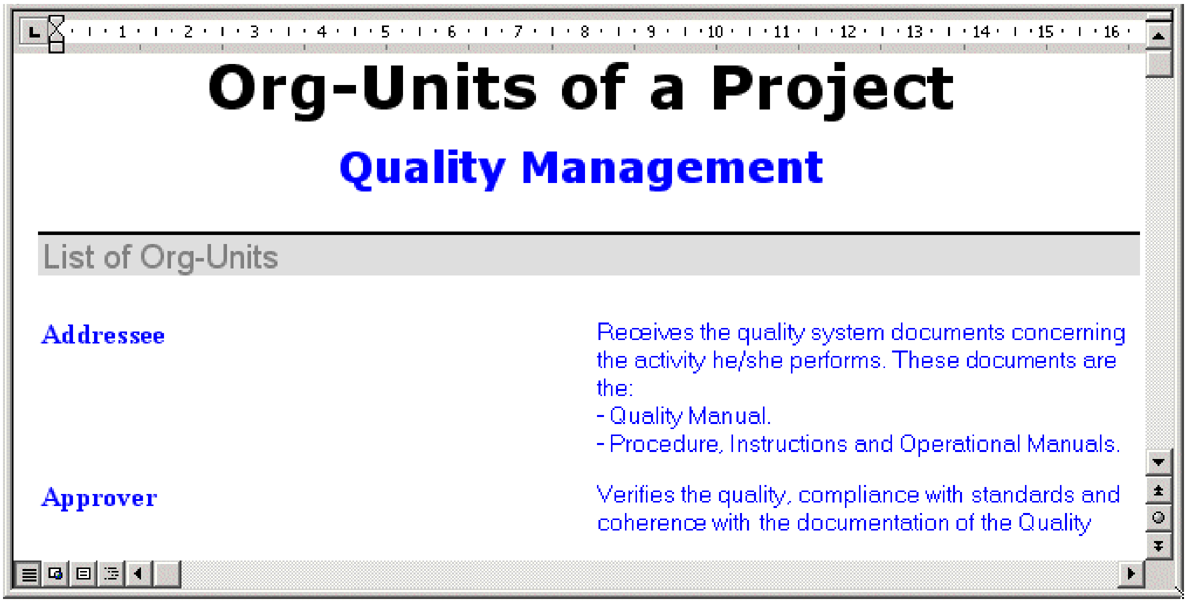
The purpose of this guide is not to teach you all about ***MS Word***, but to offer a few guidelines on this software.

☛ *You can use* ***HOPEX*** *with word processing software other than MS Word, but you will be able to execute RTF descriptors only (see Customizing RTF Descriptors).*

Contents of Generated Reports (MS Word)

Generated reports (MS Word) consists of two parts:

- Texts that you can modify directly in MS Word.
- Description of the objects in the **HOPEX** repository. This text is in blue.



Org-Units of a Project

Quality Management

List of Org-Units

Addressee Receives the quality system documents concerning the activity he/she performs. These documents are the:
- Quality Manual.
- Procedure, Instructions and Operational Manuals.

Approver Verifies the quality, compliance with standards and coherence with the documentation of the Quality

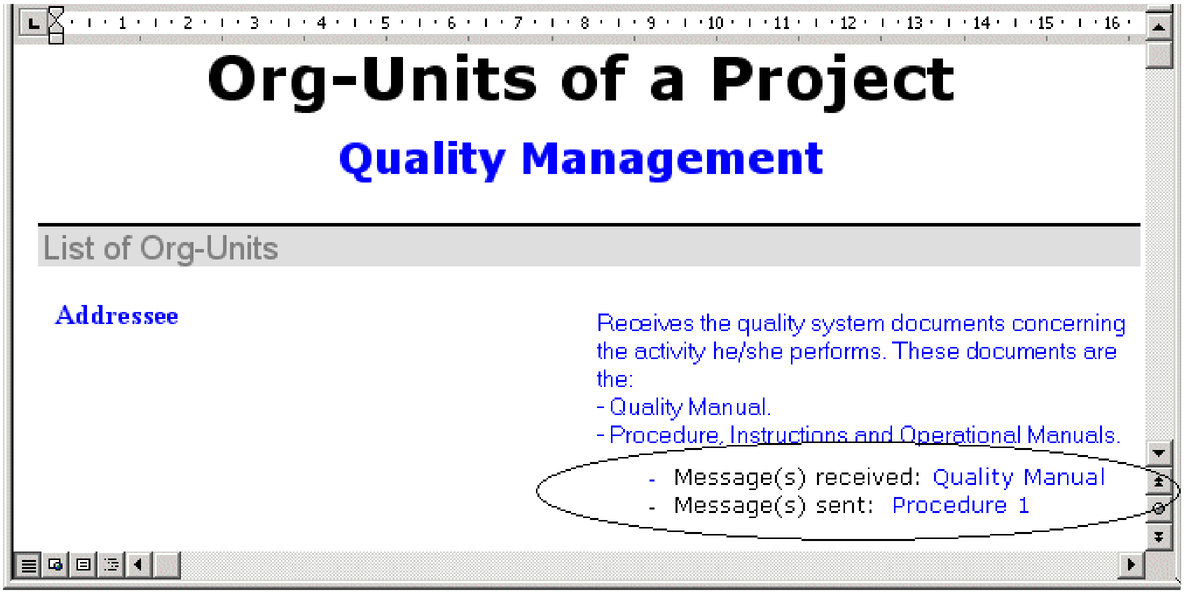
To insert descriptions of **HOPEX** repository objects into documents:

1. Select objects to be inserted in the report (MS Word).
2. Select the format for describing these objects.

In the above example, the selected objects are the various org-units involved in the "Order Management" project.

Each org-unit name is in bold characters and its comment appears next to it. The text is formatted using ***MS Word*** styles.

An object description can include other objects:



Org-Units of a Project

Quality Management

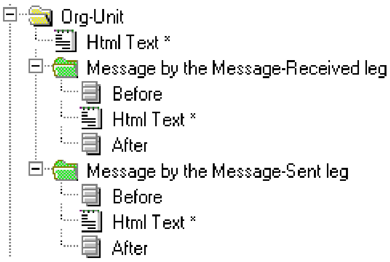
List of Org-Units

Addressee

Receives the quality system documents concerning the activity he/she performs. These documents are the:
- Quality Manual.
- Procedure, Instructions and Operational Manuals.

- Message(s) received: Quality Manual
- Message(s) sent: Procedure 1

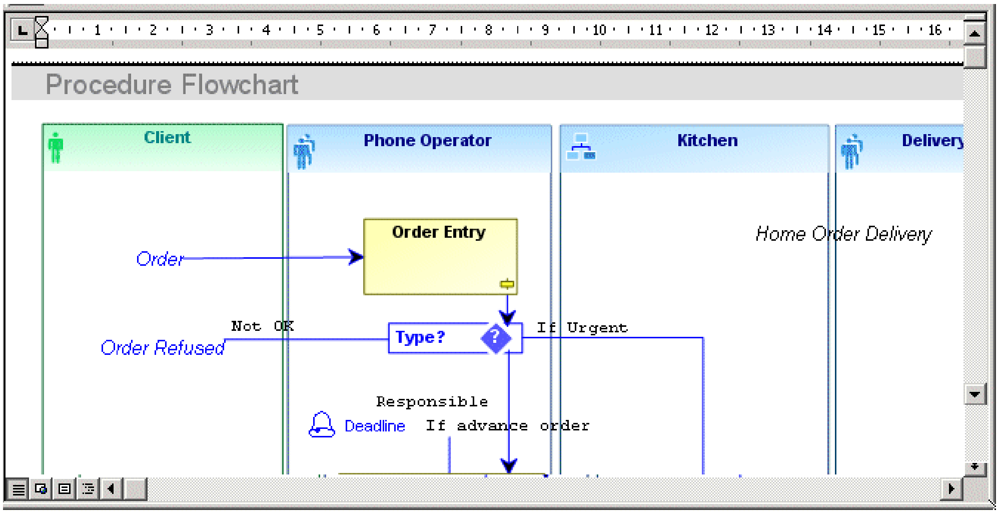
In the above example, the description of the org-unit contains its name and comment plus the messages the org-unit sends and receives.



```
Org-Unit
  Html Text *
  Message by the Message-Received leg
    Before
    Html Text *
    After
  Message by the Message-Sent leg
    Before
    Html Text *
    After
```

In this case, the object description, or descriptor, is represented by a menu tree structure with text that explains each included object.

An object descriptor can also contain a drawing.



Report (MS Word) Generation Steps

For each main object type in **HOPEX** (business process, organizational process, application, database, etc.), one or several report templates (MS Word) are available.

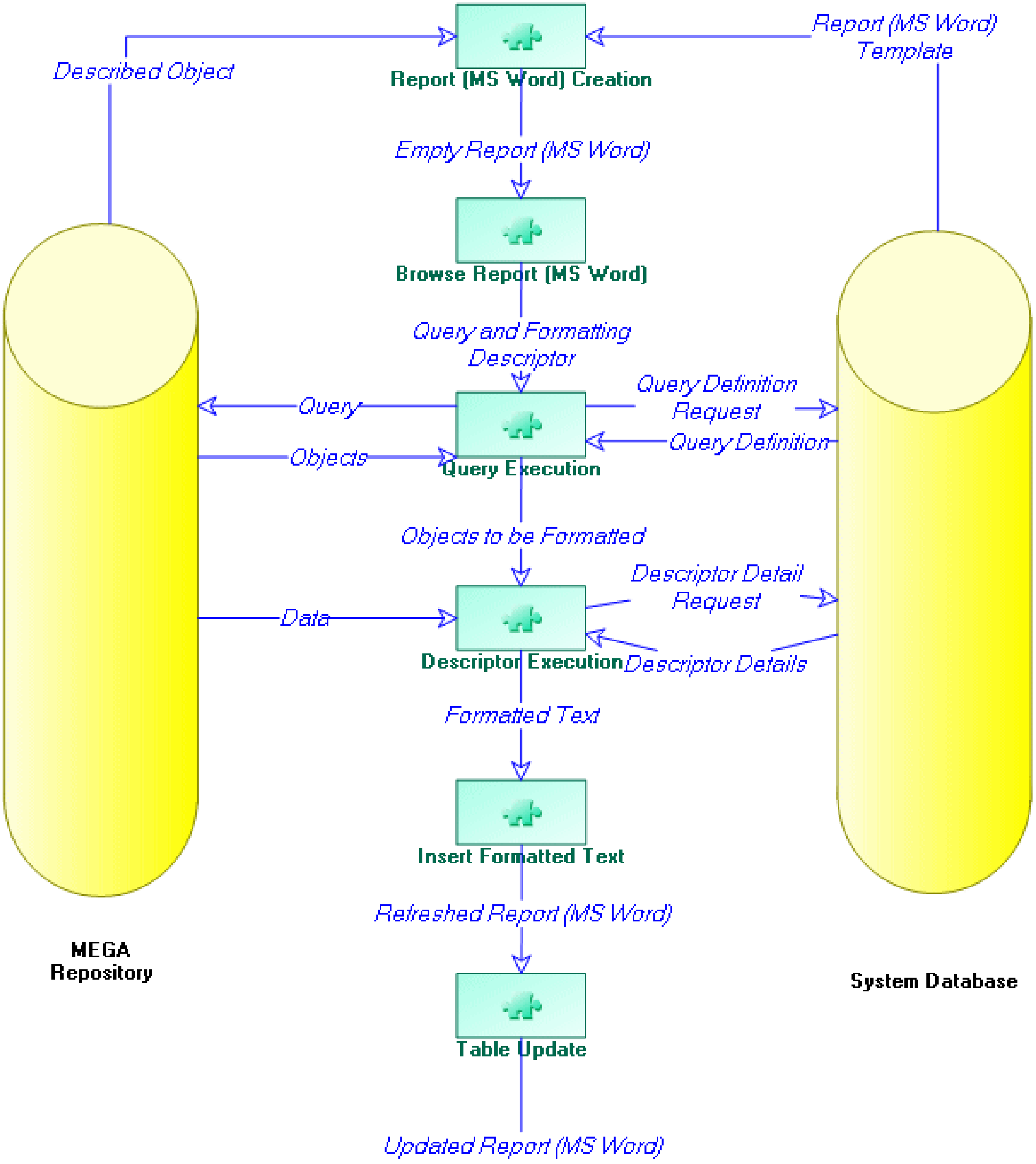
> For a process for example, there is a general report template (MS Word), one specific to risk analysis and one specific to process simulation.

The generation of a report (MS Word) in **HOPEX** is as follows:

1. **HOPEX** searches for report templates (MS Word) to be proposed as a function of the type of object to be described and the products available to the user.
2. If several report templates (MS Word) exist, **HOPEX** asks the user to select the appropriate report template (MS Word).
3. The report (MS Word) is automatically created from a copy of the selected report template (MS Word).
4. **HOPEX** browses the report (MS Word) to find queries specified in the report template (MS Word).
5. For each query found, the program searches for its definition in the system repository, then executes this in the **HOPEX** repository.
6. **HOPEX** formats each query result object using the descriptor specified in the report template (MS Word). This descriptor is found in the system repository.
7. The formatted text is inserted in the report (MS Word).
8. The program processes the next query.

9. When all queries have been processed, the program updates the table of contents and index if these exist.

☛ *Note that for .docx reports generated on **HOPEX Web Front-End**, the table of contents need to be updated manually.*



Creating a Report Template (MS Word)

You can create as many reports (MS Word) as you want, and modify them as many times as you want.

However, if you want to produce several reports (MS Word) of the same type, it is recommended that you use a report template (MS Word) to create each report (MS Word).

A report template (MS Word) provides the framework of a report (MS Word). it is fleshed out with data from the repository when creating a report (MS Word). It contains texts and report template (MS Word) elements which allow you to rapidly create reports (MS Word) associated with an object.

You can create a report template (MS Word) even if you do not have the **HOPEX Power Studio** technical module.

> ☛ *The standard report templates provided are protected. You must duplicate them if you wish to modify or create new templates from those proposed.*

In the following example you will create a new report template (MS Word) describing the list of org-units in a project with their comments.

To create a report template (MS Word) from an existing one:

1. From **HOPEX**, select the **Utilities** navigation window.

   > ☛ *To access the **Utilities** window, select **View > Navigation Windows > Utilities**.*

2. From the **Report Templates (MS Word)** folder, right-click the report template (MS Word) from which you want to create your new template and select **Duplicate**.

   `Example: "Standard".`

   - The "Standard" report template (MS Word) contains style and macro definitions only. Use this template to create a new blank report template (MS Word).
   - If you select a different report template (MS Word), your new report template (MS Word) will start with the framework of this template. You can then modify the new report template (MS Word) as desired.

   The **Duplicate an Object (Report Template (MS Word))** dialog box appears.

3. (Optional) Modify the **Name** of the report template (MS Word) to be created.
4. In the **Strategy** field, select **Reuse descriptors and queries**.

   > ☛ *For more details on duplication of report templates (MS Word), see Duplicating Descriptors.*

5. Click **OK**.
   When duplication has been completed, the element corresponding to the new report template (MS Word) is created.

To open the report template (MS Word) you created:

- Right-click the report template (MS Word) you created and select **Open**.
  The MS-Word application is displayed in the foreground.

Modifying a Report Template (MS Word)

Report template (MS Word) modifications can apply to page format, headers and footers and accompanying text entered in MS Word: in this case they concern only MS Word, and are made as in normal use of this software.

☛ *See Entering text in your MS Word report template*

To display the name of a project, that is to say the name which should be instanced in each report (MS Word), you need to insert a "report template (MS Word) element".

☛ *See Inserting Report Template (MS Word) Elements*

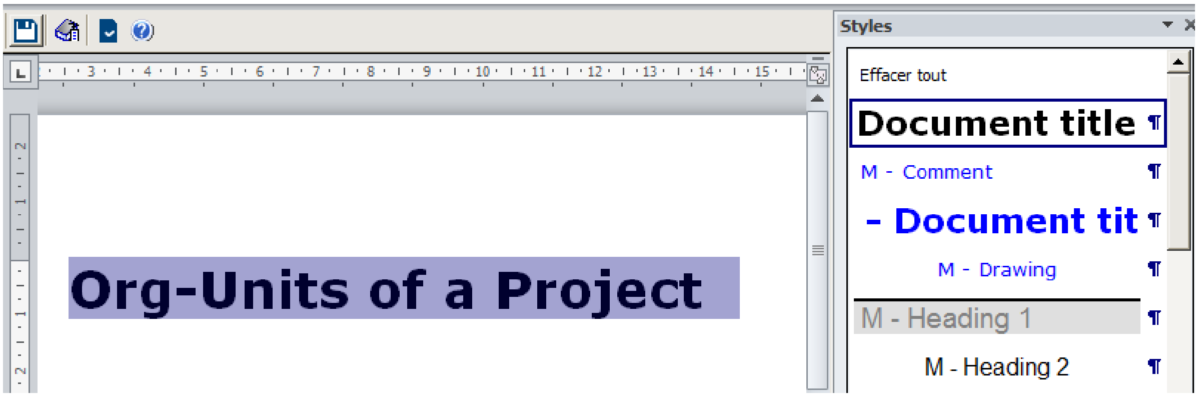
Entering text in your MS Word report template

To add a title:

1. In the report template, enter the title.

   `Example: "Org-Units of a Project"`

2. Select the title (for example: "org-Units of a Project") and in the MS Word **Styles** apply "Document title" style.



Inserting Report Template (MS Word) Elements

Report template (MS Word) elements query the repository; they consist of:

- a descriptor (see Customizing RTF Descriptors)
- a query (see the **HOPEX Common Features** guide).

Report template (MS Word) elements can be inserted in report templates (MS Word): they are transformed into report (MS Word) elements when creating a report (MS Word).

To insert report template (MS Word) elements in a report template (MS Word):

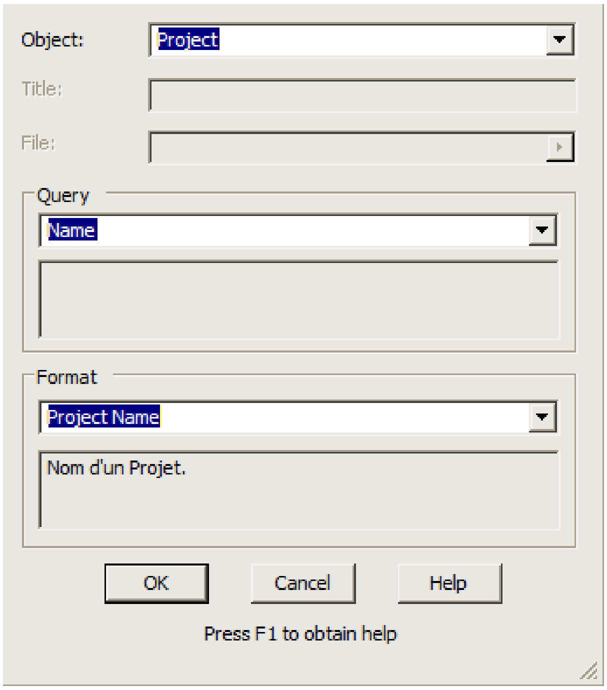
1. Position the cursor where you want to insert the "report template (MS Word) element.

2. In the edit toolbar, click **Insert a Report template (MS Word) Element**.
   The **Insertion of Report Template (MS Word) Element** dialog box opens.
3. In the **Object** field, select the type object to which the report template (MS Word) element relates.

   `Example: “Project”.`
4. In the **Query** field, select the query criterion for of the desired project.

   `Example: "Name" which will search for projects by name.`
5. In the **Format** field, select the format of the result with a description.

   `Example: The "Project Name" descriptor enables display of the project name.`



Object: Project

Title:

File:

Query

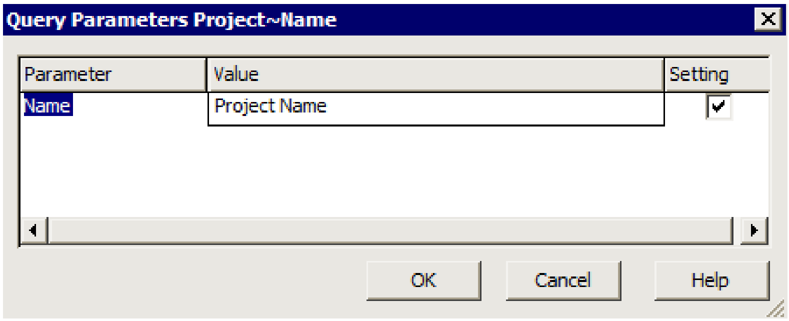
Name

Format

Project Name

Nom d'un Projet.

6. Click **OK**.
   An intermediate dialog box appears where input is required in our example:



Query Parameters Project~Name

Parameter	Value	Setting
Name	Project Name	☑

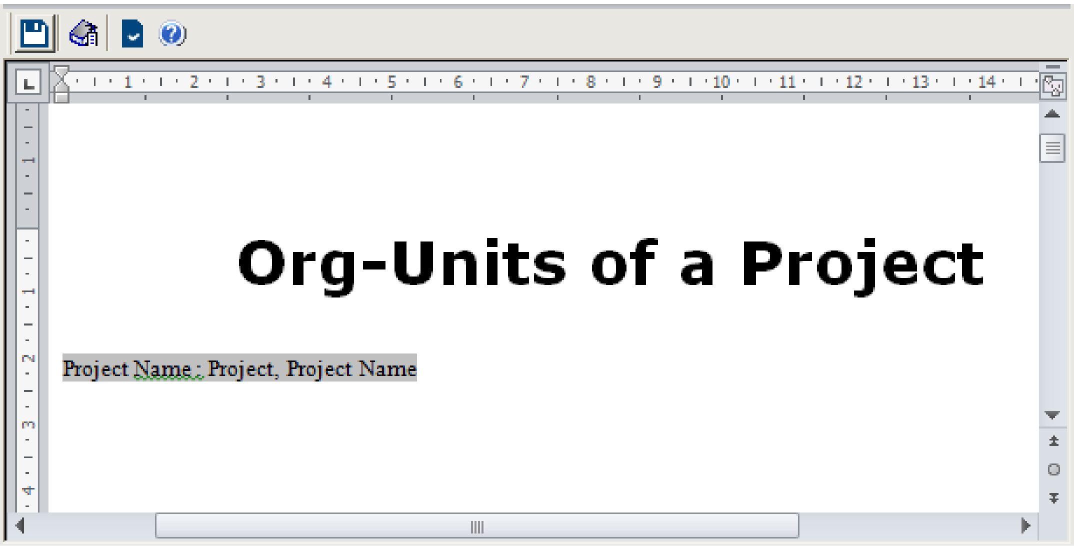
   The software will request a value for this setting when it creates a report (MS Word) based on this template.

7. In the **Value** filed, enter the value.

   ```
   Example: "Project Name".
   ```

   ☺ *If you clear the **Setting** check box, the value you enter will be used as the setting value when the report (MS Word) is created.*

8. Click **OK**.
   The first report template (MS Word) element has been inserted in the document template.

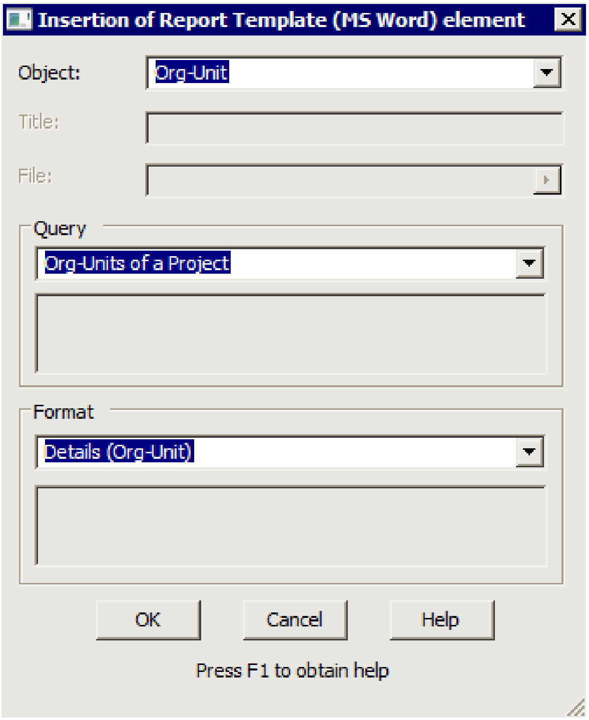


Org-Units of a Project

Project Name : Project, Project Name

9. Enter the title of the next paragraph (for example: "List of Org-Units"), and apply the "Heading 1" style.

10. Insert a second report template (MS Word) element, which will present the “Org-Units of a Project” with the “Details (Org-Unit)” descriptor.



Insertion of Report Template (MS Word) element

Object: Org-Unit

Title:

File:

Query

Org-Units of a Project

Format

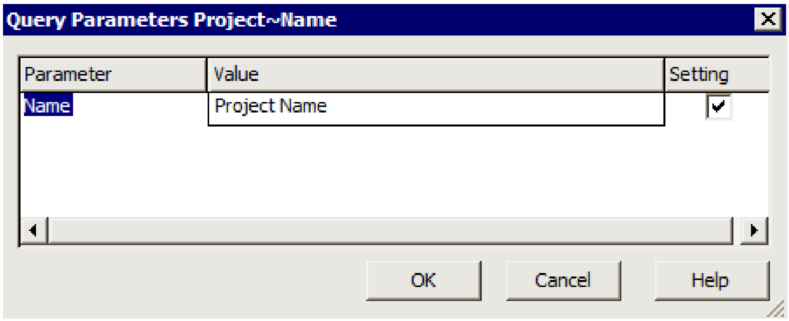
Details (Org-Unit)

11. Complete the **Query** and **Format** fields and click **OK**.

12. In its project text box, enter the name of the setting value which will be requested when the report is processed.

> ☛ *Several report template (MS Word) elements can use the same setting. For this to be possible, give the same name to the settings associated with each of these elements.*

`Example: enter “Project Name“.`



Query Parameters Project~Name

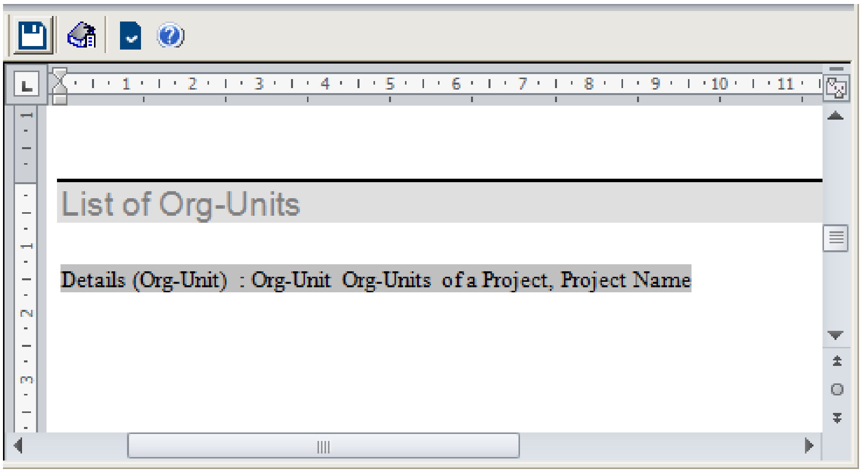
Parameter	Value	Setting
Name	Project Name	☑

13. Click **OK**.

> ☛ *Insertion of a report template element is by insertion of a report element in MS-Word, followed by the name of the descriptor used (here,*

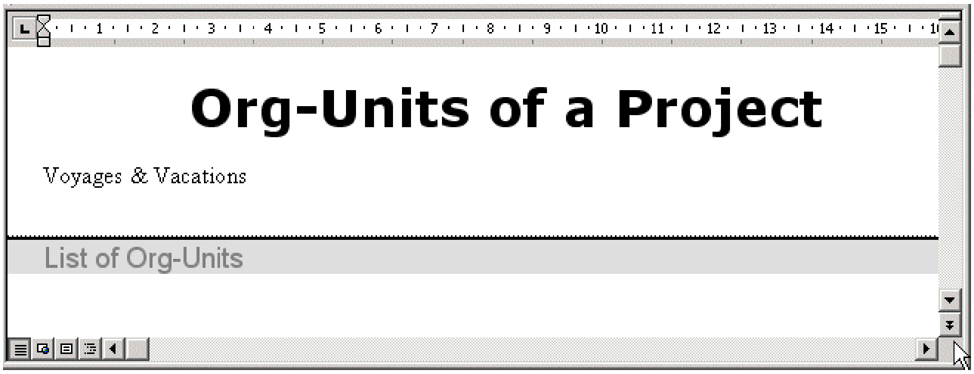
"Details (Org-Unit)") followed by the names of the query and the setting (in this example, "Org-Units of a Project", Project Name"):

Report template (MS Word) element 'Details (Org-Unit): Org-Units of a Project, Project Name'


List of Org-Units

Details (Org-Unit) : Org-Unit Org-Units of a Project, Project Name

14. In the edit toolbar, click **Save**.
You can now create reports (MS Word) from your new report template (MS Word). T


Org-Units of a Project

Voyages & Vacations

List of Org-Units

Deleting a Report Template (MS Word) Element

To delete of a report template (MS Word) element:

- Delete the corresponding line.

Replacing a Report Template (MS Word) Element

You can replace a report template (MS Word) element by a new element.

☛ *The standard report templates provided are protected. If you want to modify them, you must first duplicate them. The same applies for descriptors.*

To delete a report template (MS Word) element and insert the new element:

1. Open the report template (MS Word).
2. Delete the report template (MS Word) element by deleting its code.
3. From the document edit toolbar, click to insert a new report template (MS Word) element.

Backing Up a Report Template (MS Word)

To create a backup copy for use in another environment, you need to extract the report template (MS Word) together with the queries and descriptors it uses.

To extract a report template (MS Word):

1. In the navigator, right-click the report template and select **Properties**.

   `For example: "Org-Units of a Project" (MS Word)`
2. Select the **Documented object** tab.
3. Make sure that the documented **Object type** appears.

   `Example: “Project”`

   The **Refresh** button updates the links between the report template (MS Word) and its components (queries and descriptors).

   ☺ *If queries or descriptors have been modified or renamed, we recommend that you click the* ***Refresh*** *button before extraction. When in doubt, always click this button before extraction.*
4. If necessary, specify the **Category** to which the object type is attached (a category is a set of objects of the same type).

   ☛ *A category is a group of objects of the same type. For example, “Organizational Chart” is a category of the “Diagram” type. A type may have several categories.*
5. Right-click the report template (MS Word) and select **Manage > Export**. The **Export HOPEX objects** dialog box opens.
6. Click **Export**.

PRIVATE AND PUBLIC REPORT TEMPLATES (MS WORD)

At the time of its creation, a new report template (MS Word) is private and is therefore visible only to its creator.

This report template (MS Word) can be made public so that it will be available to other users.

> ☛ *For more details, see To make a private report template (MS Word) public:.*

To access report templates (MS Word):

1. In **HOPEX**, open the **Utilities** navigation window.
2. Expand folders **Report Template (MS Word)** and **Private**.
   The **Private** folder contains private report templates (MS Word) of the current user.

   A private report template (MS Word) is owned by the user that created it. You can see this by exploring the report template (MS Word).

To explore the report template (MS Word) and see its creator:

1. Right-click the report template (MS Word) and click **Explore**.
2. In the window that opens, expand the **User** folder.
   The Report Template (MS Word) creator is displayed in the right pane of the window.

Making report templates (MS Word) available

Making a private report remplate (MS Word) public

To make a private report template (MS Word) public:

1. In the **Utilities** navigation window, expand the **Report Template (MS Word)** and **Private** folders.
2. Right-click the desired report template (MS Word) and select **Manage > Make Public**.
   The report template (MS Word) is moved to the global list of report templates (MS Word).

   > ☛ *A report template (MS Word) that has been made public can be returned to private status.*

Making a report remplate (MS Word) private

To make a report template (MS Word) private:

- Select the report template (MS Word) and drag it into the **Private** folder.
  When a report template (MS Word) has been made public, other users of the environment can reuse or modify this report template (MS Word).

Deleting Private Report Templates (MS Word)

Private report templates (MS Word) and descriptors can be deleted at repository cleanup.

To delete all private report templates (MS Word) and descriptors:

1. From **HOPEX** menu bar, select **File > Properties**.
2. In the **Characteristics** tab, click **Repository Cleanup**.
   The repository cleanup dialog box presents groups of elements that can be deleted. The private report templates (MS Word) and descriptors proposed are those of the current user.

   ☺ *To display the list of private report templates (MS Word) and descriptors, click **View**.*
3. Click **OK**.
   Selected elements are deleted.

To delete private report templates (MS Word) one-by-one:

1. Right-click the report template (MS Word) name and select **Delete**.
   A dialog box asks you to confirm deletion.
2. Click **Delete**.
   The report template (MS Word) is deleted.

CUSTOMIZING REPORT (MS WORD) SENDING

HOPEX enables sending of reports (MS Word) for review or validation. This function is accessible via the commands **Send for Review** and **Send for Validation** in the report (MS Word) properties dialog box.

Information concerning recipients is based on content of the **Distribution** tab of the report (MS Word) properties dialog box.

> ☛ *Rules concerning the recipients list are described in the **HOPEX Common Features** guide, chapter "Generating documentation", paragraph "Distributing a report (MS Word) for review or validation".*

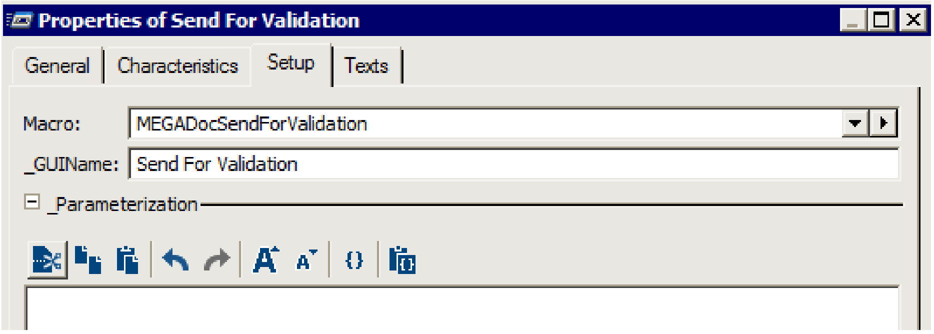
To customize report (MS Word) sending:

1. Explore the **Report (MS Word)** MetaClass.
2. Right-click the Command and select **Properties**.

   `For example: "Send for Validation" command.`
3. From the **Setup** tab, you can access macros.
   The macro "MEGADocSendForValidation" calls code that executes standard code.
4. In the **Macro** field, click the right-oriented arrow and select **Create** to create a new macro.

   `For example: "MyDocSendForValidation".`

> ☛ *In the option (**Tools > Options > Repository**), the **Authorize MEGA data modification** option value must be "Authorize".*



Macro: MEGADocSendForValidation

_GUIName: Send For Validation

5. Open the **Properties** of this new macro.
6. Update the script in the **VB Script** tab of the macro properties and click **OK**.

> ☛ *For more information on these functions, see the **All about Starting with APIs** technical article.*

> ☛ *To call a VB script of this type, you must use a .dll (cdo.dll) and deploy it on all workstations. The user must have adequate administration rights for a script to be executed automatically on his or her workstation.*

FAQs on Report Templates

How do I customize the logo in generated reports (MS Word)?

To customize the logo in generated reports (MS Word):

1. Double-click the icon of the report (MS Word) of which you want to change the logo.
2. In the opened report (MS Word), select **ViewHeader and footer**.
3. Use the Word header and footer toolbar to select the header containing the logo.
4. Click the default logo and then select **Insert > Picture**.
5. A dialog box appears. Select the image you want and then click **OK**.

The initial logo is replaced by the one you selected.

☺ *If you customize the logo in a report template (MS Word), the reports (MS Word) that you generate from this template will automatically present your logo. This means you avoid having to insert your logo in every report (MS Word) you create.*

☛ *Note that standard report templates (MS Word) are read-only. To make changes to them, you must first make copies of them. To do this, create a report template (MS Word) based on the desired standard template.*

How do I replace the company name with the name of my own organization in the report (MS Word) cover page?

The name that appears in the report (MS Word) is the one indicated when the product was installed.

To modify the company name:

1. Run the **Administration** application by double-clicking the corresponding icon.
2. In the pop-up menu of the site (navigation tree root, named "HOPEX"), select **Options > Modify**.
3. In the Options window, expand the **Installation** folder and double-click the **Company information** icon.
4. Type the name of your organization in the **Company Name** text box. This name will now appear in your reports (MS Word).
5. Click **OK** and close the **Administration** application.

CUSTOMIZING RTF DESCRIPTORS

You can customize the *descriptors* provided at installation to suit your specific needs if you have the **HOPEX Power Studio** technical module.

You can also create new descriptors to:

- produce files in RTF format that can be used by most word processors.
- insert the produced files in *report (MS Word) elements* or *report template (MS Word)* elements used in your reports (MS Word) and report templates (MS Word).

See:

✓ Creating RTF Descriptors
✓ Defining descriptors
✓ Executing Descriptors

CREATING RTF DESCRIPTORS

RTF Descriptor Categories

Descriptors are either:

- ***Contextual*** descriptors, which are designed to be used in a report template (MS Word).
- ***Elementary*** descriptors, which can be used in a report template (MS Word) or processed alone.
- ***Customized*** descriptors that you create yourself to meet your specific needs.
- ***Private*** descriptors

Creating a Descriptor

☛ *To create a descriptor from an existing descriptor, see Duplicating Descriptors.*

To create a descriptor:

1. From **HOPEX Customizing** desktop, open the **Utilities** navigation window.

   ☛ *To access the **Utilities** window, select **View > Navigation Windows > Utilities**.*

2. Expand the **Descriptors** and **Rtf Format** folders.
3. Right-click the descriptor category folder corresponding to the descriptor category you want to create and select **New > Descriptor**.

   ☛ *See RTF Descriptor Categories.*

   Example: **Customized** category.

   The **Create Descriptor** dialog box opens.

4. In the **Name** field, enter the name of your descriptor.
5. In the **Described object** field, select the described object, for example Org-Unit.
   The descriptor can also relate to an abstract MetaClass.

   ☛ *For more details on the abstract metamodel, see **HOPEX Power Studio**, "Managing the Metamodel", "Abstract Metamodel".*

   ☛ *If the name you entered is already used by another descriptor, **OK** is unavailable (gray).*

6. Click **OK**.

Descriptors and Inherited Objects

If the global option concerning variations is activated (**Activate variations** options in **Options > Business Process and Architecture Modeling**) you can take account of inheritance.

> ☛ *For more details on inheritance and object variations, see the **HOPEX Common Features** documentation ("Handling Repository Objects", "Object Variations").*

When inheritance is taken into account repository paths specified in descriptors (via queries or MetaAssociationEnds) also include inherited objects.

You can define to take account of inheritance:

- globally at descriptor level
- specifically at the level of a given path

To take account of inheritance:

1. Right-click the descriptor, and select **Properties**.
2. In the **Characteristics** tab, select the **Take account of inherited objects** check box.
   Inheritance is now taken into account in the descriptor, except if you specify a different behavior on certain elements of the structure.

DEFINING DESCRIPTORS

The object descriptor structure is made up of several linked groups organized in a tree. Each group relates to an object and specifies the query or link, which enables passage from the previous object to this object.

Text is the basic element used to define what is displayed for each of the objects in a generated report (MS Word).

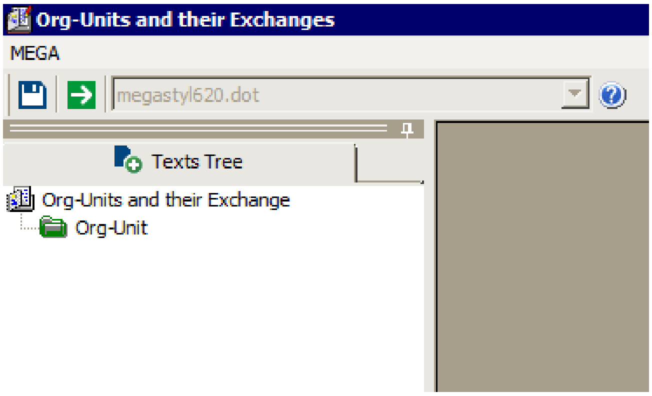
Accessing the descriptor structure

To access the descriptor structure:

1. From the **Utilities** navigation window, double-click the created descriptor.

   ☛ *See Creating a Descriptor.*

   The descriptor editing window appears and displays the **Texts Tree** tab.



To create the descriptor structure, see:

- Adding Text to a Descriptor
- Creating Groups

Adding Text to a Descriptor

Text is the basic element used to define what is displayed for each of the objects in a generated report (MS Word).

> Example: you can add text to the “Org-Unit” object to edit its name and comment.

To add text to a descriptor:

1. Access the descriptor structure.

   ☛ *See Accessing the descriptor structure.*

2. Right-click the folder representing the described object (Example: **Org-Unit**) and select **New > Text**.

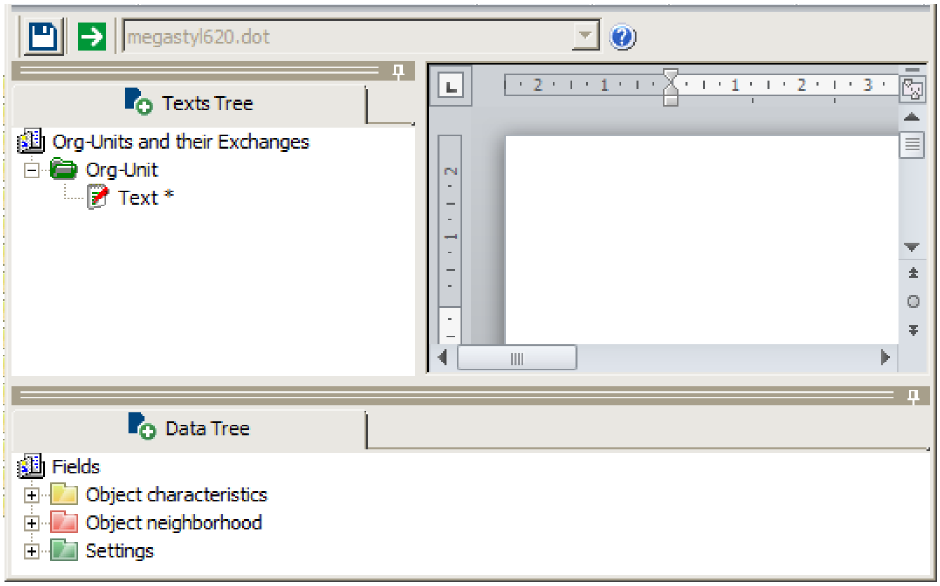
   `Example: **Org-Unit**.`

   Text 📝 is added in the **Texts Tree** tab.

   ☛ *The* 📝 *icon indicates which text is currently being edited.*

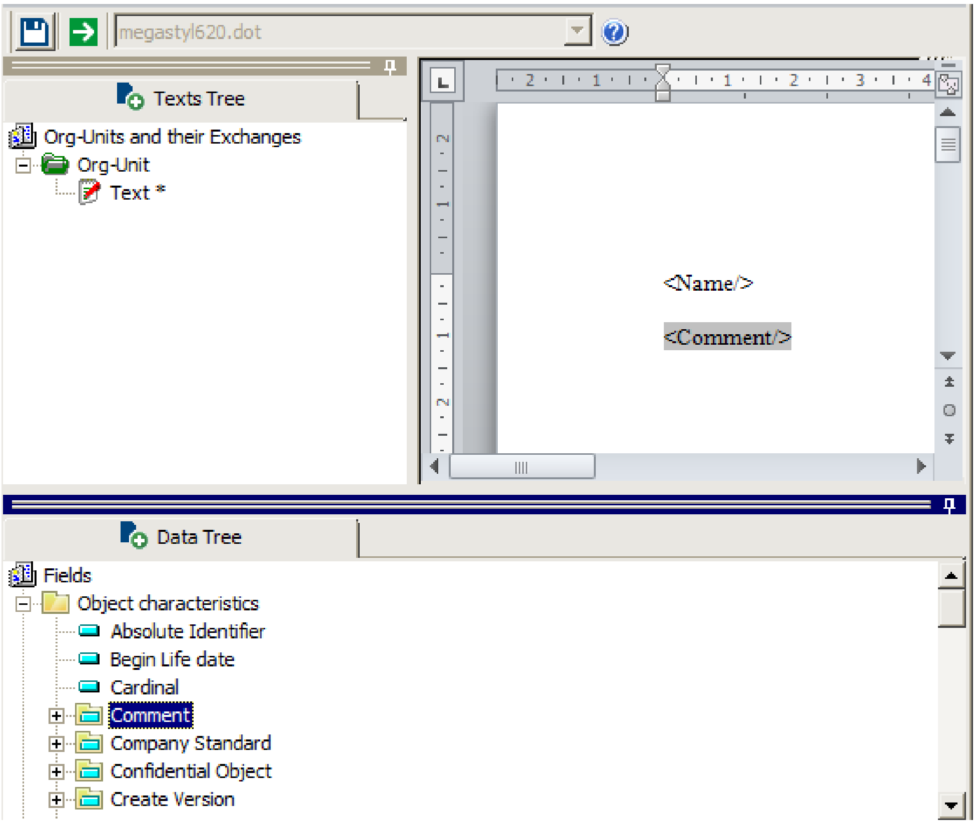
   The following panes appear in the descriptor editing window:

   - an MS Word pane (top right pane) to edit the text.
   - a **Data Tree** tab (bottom pane), which enables you to select a field and drag and drop it for edition in MS Word pane.



3. In the **Data Tree** tab select the field you want to insert in the text.
4. Drag and drop the field in the top-right pane.

   Example: to insert the **Name** and **Comment** fields in the text area, in the **Object characteristics** folder, successively select the **Name** and **Comment** fields and drag-and-drop them into the desired location in the text area.



5. (Optional) Customize your document format by applying Word styles.

   ☛ *These styles have the prefix M- and are based on the M-Normal style that is similar to the Word Normal style. Note that the M-Normal style text is in blue.*

   ☛ ***Fields** from **Object characteristics** folder cannot be manually edited. You must insert these in the text by drag-and-drop as explained above, or by copy/paste.*

6. Click **Save**.
7. In the **Texts Tree** tab, right-click **Text** and select **Close**.
   The text is automatically saved and the icon returns to its original form.

Creating Groups

The object descriptor structure is made up of several linked groups organized in a tree. Each group relates to an object and specifies the query or link, which enables passage from the previous object to this object.

```
Example: You can display the messages sent by each org-unit.
```

To create a group:

1. Right-click the folder representing the described object (Example: **Org-Unit**) and select **New** > **Group**.
   The **Group Properties** dialog box opens.
2. (Optional) In the **Title** field, enter a title.

   ☛ *In most cases, you may choose not to give a **Title** to the group, since the object and query names provide sufficient information to identify the group.*
3. You can select a registered query/MetaAssociationEnd or embed a query. Select either:
   - **Query or MetaAssociationEnd** to select a registered query.

     📖 *A MetaAssociationEnd is the extremity of a MetaAssociation linking the MetaAssociation to a MetaClass.*

     ☛ *Concrete links as well as generic links are proposed in the list of MetaAssociationEnds.*

     `Example: querying organizational processes linked to an operation, you will see all links connecting these two object types, including the "Behavior" generic link that will find the organizational process describing behavior of the operation.`

   In the **Described object** field, select the described object.

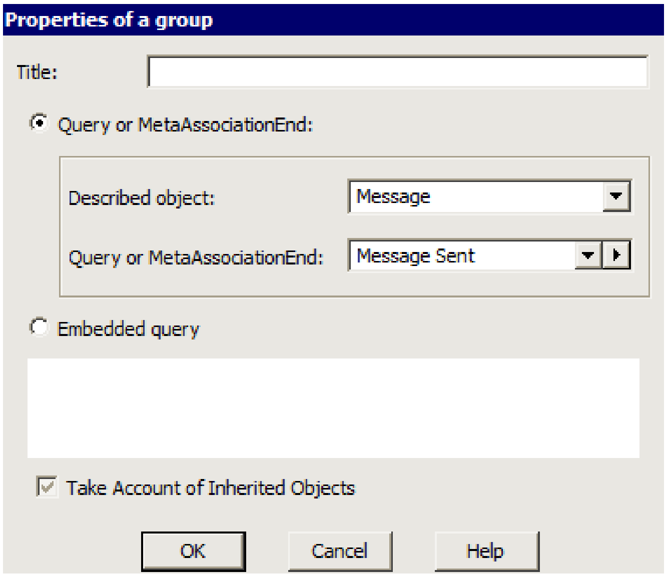
     `Example: "Message" object`

   In the **Query or MetaAssociationEnd** field, select the query/MetaAssociationEnd.

     `Example: "Message-Sent".`
   - **Embedded query** to embed a query that you do not want to save in the repository. If you have a specific need to write a query that will never be reused elsewhere. Enter your query code directly in the corresponding frame.

     ☛ *For more details on queries, see **HOPEX Common Features**.*

4. If the general option on variations is activated, the **Take Account of Inherited Objects** option is displayed.
   You can choose to:
   - follow default behavior defined at descriptor level (see Descriptors and Inherited Objects): keep the check box grayed: ☑ (or ■ according to the selected Windows theme).
   - activate or deactivate the option at group level (by selecting or clearing the check box).



Properties of a group

Title:

● Query or MetaAssociationEnd:

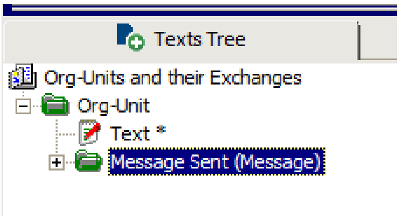
Described object: Message

Query or MetaAssociationEnd: Message Sent

○ Embedded query

☑ Take Account of Inherited Objects

5. Click **OK**.
   The created group is displayed under the described object folder.



Adding Text to a Group

You can add text:

- before the object
- to the object
- after the object

To add text to a group:

1. Expand the group folder.

   ☛ *See Creating Groups.*

2. Right-click **Before** (/**After** according to where you want to add text) and select **Text** to add text at the beginning (/end).

   Example: select **Before** to add text at the beginning of the list of messages sent by an org-unit.

3. In MS Word pane, enter the text.

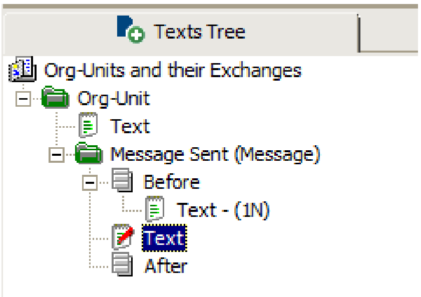
   Example: "Messages sent" that you want to display before the list of messages.

4. Apply a style to your text.

   Example: "M-List"

   See Managing Formats and Styles for more information on styles.

5. Right-click **Text** and select **Close** to close the Text editor panes.
6. To add text to the object (Example: "Message sent"), right-click the object and select **New > Text**.



   Another **Text** icon appears in the tree.

7. From the **Data Tree** tab, in the **Object characteristics** folder, select **Name** and drag and drop it into MS Word pane.

   Example: Add **Name** and **Comment** object characteristics

8. (optional) Customize the layout of your document using the styles provided by **HOPEX** and MS Word.
9. Right-click **Text** and select **Close** to close the Text editor panes.
10. (optional) Create other groups.

    Example: Create a group that displays the messages received by an org-unit.

    ☛ *Do not forget to save your work from time to time (select* ***Descriptor > Save****).*

Specifying text execution conditions

You can define execution conditions on a "Before" text. You can vary the text according to the number of objects resulting from execution of the query.

Execution condition can be:

- **No object**: the text is displayed if there is no object

  `Example: “No message sent”`

- **A single object**: the text is displayed if there is only one object

  `Example: “Message sent”`

- **One object or more**: the text is displayed if one or more objects

  `Example: "Message(s) sent”`

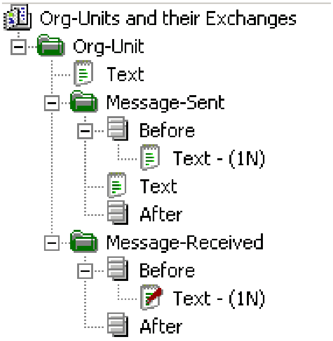
- **More than 1 object**: The text is displayed when there are several objects

  `Example: “Messages sent”`

To define text execution conditions:

1. From the descriptor editing window, under the **Before** folder, right-click **Text** and select **Properties**.
2. Select the **Execution** tab.
3. Select the execution condition.
   The text is edited depending on the number of objects found on execution of the query.

   ```
   Example: The text is edited depending on the number of
   messages sent by the org-unit
   ```



```
Org-Units and their Exchanges
  Org-Unit
    Text
    Message-Sent
      Before
        Text - (1N)
      Text
      After
    Message-Received
      Before
        Text - (1N)
      After
```

☛ *To reorder the descriptor text and groups see Defining the Object Order in Reports (MS Word).*

Executing Descriptors

Executing a descriptor

To execute a descriptor:

1. Access the descriptor editing window.

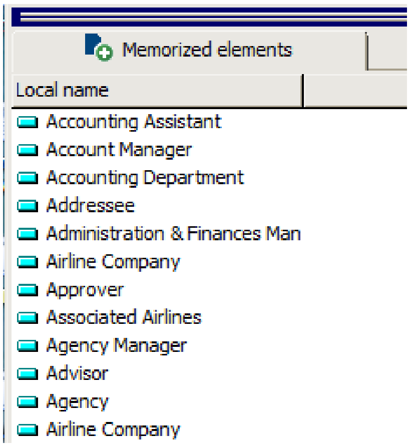
   ☛ *See Accessing the descriptor structure.*

2. From the descriptor editing window, select **MEGA > Process**.
   The query dialog box appears.

3. Click Find.

4. In the dialog box that opens, select the objects that interest you and click **OK**.
   A message box shows the progress of descriptor processing.

   Objects selected are memorized and displayed in the **Memorized elements** tab.



Accounting Assistant
Account Manager
Accounting Department
Addressee
Administration & Finances Man
Airline Company
Approver
Associated Airlines
Agency Manager
Advisor
Agency
Airline Company

The result is displayed in an MS Word window:

Re-executing the descriptor from memorized objects

HOPEX memorizes objects used in the last execution of the query. These objects appear in the **Memorized Elements** tab (see Executing a descriptor).

To re-execute a descriptor from memorized object:

1. (optional) If needed, from the descriptor editing window, delete an object from the **Memorized Elements** tab list (right-click the object and select **Remove**).
   This set of objects enables time saving when executing and testing the descriptor.

2. Click **Run descriptor** and select **Execute on all memorized elements**.
   The reports execute and the RTF format document is regenerated.

ADVANCED FEATURES OF RTF DESCRIPTORS

Previous chapters detail the basic principles of customizing reports (MS Word) and RTF descriptors. This chapter introduces advanced functions that you can use to refine your descriptors.

☛ *These descriptor customization functions are available only with the **HOPEX Power Studio** technical module.*

The following points are covered here:

MODIFYING DESCRIPTORS

Duplicating Descriptors

☛ *A number of standard descriptors are created by the installation program. These standard descriptors are read-only so as to avoid conflicts when you upgrade your version of **HOPEX**. If you want to make changes to a standard descriptor, you must duplicate it.*

To duplicate a descriptor:

1. From **HOPEX**, open the **Utilities** navigation window.

   ☛ *To access the **Utilities** window, from **HOPEX** menu bar select **View > Navigation Windows > Utilities**.*

2. In the **Descriptors** folder, right-click the descriptor you want to duplicate and select **Duplicate**.
   The **Duplicate an Object (Descriptor)** dialog box appears.
3. In the **Strategy** field, keep **Duplicate** value if you want to modify descriptor content (example: query).

   ☛ *Select **Reuse descriptors and queries** if you only want to reuse the descriptor content.*

4. (Optional) In the **Prefix**/**Suffix** field, modify the prefix/suffix used by default to create the name of duplicated descriptors.

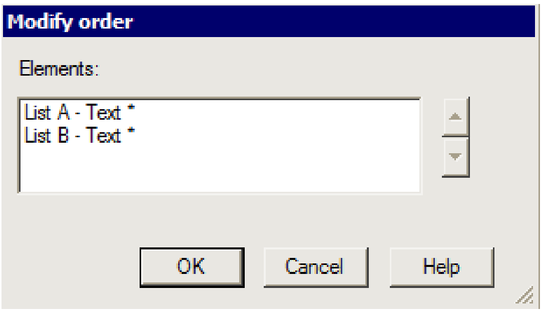
   ☛ *By default: **(Duplicate)** suffix.*

5. (Optional) Modify the **Name** of the duplicated descriptor.
6. (If needed) In the **Library** pane, select **Create duplicates in another library (for objects stored in a library)** and select the library in the drop down list.

Modifying the Order of Groups and Texts in Descriptors

To move groups and texts within the descriptor structure:

1. From the **Utilities** navigation window, right-click the descriptor item you want to reorganize (example: a group) and select **Reorganize**.
   The **Modify order** dialog box appears.



List A - Text *
List B - Text *

2. Select the element to be moved.

   ```
   Example: "List B - Text"
   ```

3. Use the up and down arrows to move the selected element.
4. Click **OK**.
   The item order is modified.

Using Block Types

The followings points are covered here:

- Using Macros
- Adding a Diagram
- Adding existing descriptors
- Creating descriptors directly in the descriptor editor tree
- Using Groups

Using Macros

HOPEX enables addition of macros in descriptors. You can therefore communicate with Word via VB Script and use APIs to access the repository.

Adding a macro

To add a macro:

1. Right-click a group and select **New > Macro**.
   The **Create Macro** dialog box opens.
2. Enter the macro name and click **OK**.
   A VB Script tab appears.

Macro principle

The macro is called by the "Generate" method from the following code:

```
Explicit Option
Sub Generate(oObject, oContext, sUserData, sResult)
End  Sub
```

- oObject: the current object in the descriptor
- oContext: macro generation context which supports the following properties:
  - StyleSheet: report (MS Word) style sheet address
  - Current: current object on which macro executes
  - Parent: parent object
  - Root: descriptor entry point object
  - GenerationMode: rtf generation mode
  - IsConfidential: returns a boolean
- sResult: string in which RTF generated by the macro is written

Macro availability

A macro can relate to one, several or all the MetaClasses.

The MetaClass to which the macro relates should appear in the **Characteristics** tab of the macro properties.

☛ *If no MetaClass is indicated in the macro properties, this means that the macro relates to all the MetaClasses.*

Macros example

Two macros are provided as standard. You can configure these by first duplicating them.

- ***External Reference Content***: manages insertion of content of external references in the report (MS Word). Various formats are proposed:
  - .txt, .ini, .log, .xml: files are read in text format.
  - .rtf, .wri, .doc, .xls, .bmp, .gif, .jpg, .png, .tif: files are read/converted to .rtf or image format.

  ☛ *In **HOPEX Web Front-End**, you cannot reproduce the content of external references.*
- ***Business process hypothesis achievement matrix:*** enables creation of a matrix type table.

Macro structure

The following code gives the structure enabling opening of a Word session in VB, creating an empty report (MS Word) and saving its content in RTF. Word APIs then enable writing of text, tables, etc.

```
Sub RTFGenerate(mgoContext, strResultRtf)
  Dim oWordApplication
  Set oWordApplication = CreateObject("Word.Application")
      Dim oWordDocument
      Set oWordDocument = oWordApplication.Documents.Add
' Enter your code here
mgoContext.GetRoot.SaveWordDocumentAsRTF oWordDocument,
strResultRtf
  oWordApplication.Quit
End  Sub
```

SaveWordDocumentAsRtf is an _Operator that enables Word document content backup in RTF.

Taking inheritance into account in macros

Regarding inheritance, you can:

- explicitly request by code the inclusion of inherited objects
- specify a behavior for the macro by fixing its execution context

To specify behavior at macro level:

- In the macro properties, **Characteristics** tab, in the **_ExecutionOptions** field, specify the execution options:
  - not specified: the macro does not take inheritance into account
  - "Variation Inheritance": the macro takes inheritance into account
  - "Inherits caller options": the macro behaves like the caller macro

When you choose to take inheritance into account, the getCollections and getSelections functions of the macro include inherited objects.

Behavior regarding inheritance is the same for the following two expressions:

- myApplication.getCollection("Service")
- myApplication.getCollection("Service", " @inherited ")

☛ *In macros, even if the **Variation Inheritance** execution option is selected, the "Inheriting" parameter (which shows objects that are inheriting) is deactivated. To take this parameter into account, it must be indicated specifically in the getCollection (see API documentation).*

Adding a Diagram

You can define:

- Report (MS Word) diagram presentation
- Report (MS Word) diagram orientation

To add a diagram:

1. Add a new group.

   ☛ *See Creating Groups.*

2. From the **Properties of a group** dialog box:
   - in the **Described object** field select "Diagram"
   - in the **Query of MetaAssociationEnd** select "Diagram".
3. Click **OK**.
4. Right-click the group and select **Insert > Diagram**.
   The icon representing a diagram appears in the tree.

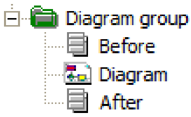
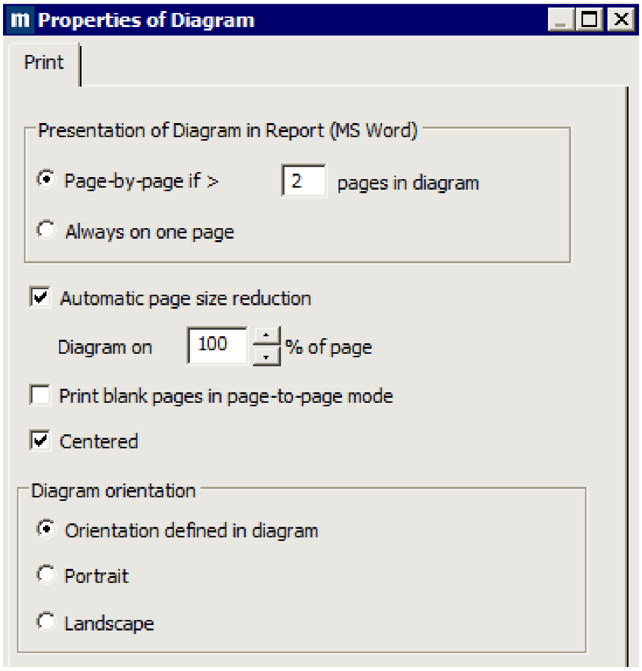


Diagram group
- Before
- Diagram
- After

5. Right-click the diagram icon and select **Properties**.



Properties of Diagram

Print

Presentation of Diagram in Report (MS Word)

● Page-by-page if > 2 pages in diagram

○ Always on one page

☑ Automatic page size reduction

Diagram on 100 % of page

□ Print blank pages in page-to-page mode

☑ Centered

Diagram orientation

● Orientation defined in diagram

○ Portrait

○ Landscape

6. In the **Presentation of Diagram in Report (MS Word)** you can define if you want your diagram to be displayed on one or several pages in the generated report (MS Word):
   - By default, the **Page by page** option is selected if your drawing needs more than 2 pages. You can modify the setting for the number of pages required to activate this option.
   - The **All Diagram on one Page** option scales the entire drawing to fit on a single page. In this case the size is automatically reduced.
7. You can reduce the diagram size (in **% of page**).
8. Select the **Print blank pages in page-to-page mode** option so that unused pages are inserted in the diagram.
9. By default the diagram is centered, unselect **Centered** if you do not want the diagram to be centered.
10. In the **Diagram orientation** pane, define the orientation of the diagram at generation.
    - **Portrait** for a vertically oriented diagram
    - **Landscape** for a horizontally oriented diagram
    - **Orientation defined in diagram** for a diagram oriented as specified in the diagram editor (**Diagram > Page Setup**).

Adding existing descriptors

To add an existing descriptor to the descriptor you are currently building:

1. Right-click the group in which the descriptor is to be inserted, and select **Insert > Descriptor**.
   The dialog box proposes all descriptors that relate to the same object type.

   ```
   For example if you want to add an existing descriptor from
   the "Org-Unit" group, all descriptors with "Org-Unit" entry
   point can be selected.
   ```

2. Select a descriptor and click **OK**.
   The added descriptor appears in the descriptor tree.

   ☛ *In our example, we added a descriptor directly under "Org-Unit". If you want descriptors corresponding to another type of object to appear, such as "Message", you must first create a group corresponding to this object type and then add a descriptor under this group.*

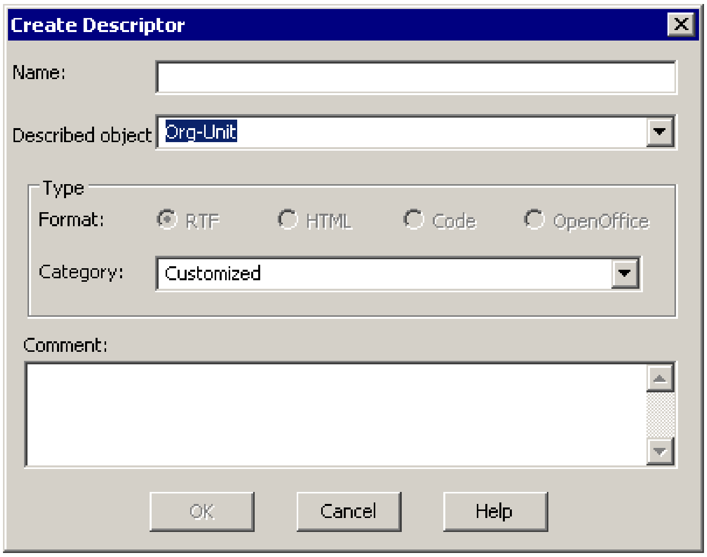
To disconnect it from your descriptor, right-click the added descriptor and select **Disconnect**.

☛ *Introduction (**Before**) and Conclusion (**After**) texts of a descriptor are not executed if the descriptor is called from another descriptor.*

Creating descriptors directly in the descriptor editor tree

To create a descriptor directly from the descriptor editor tree:

- Right-click the descriptor root object type or group and select **New > Descriptor**.
  The **Create Descriptor** dialog box opens.
  The descriptor can relate to the root object (for example: "Org-Unit"), or to the abstract MetaClass from which the root object is derived.



Create Descriptor

Name:

Described object: Org-Unit

Type

Format: ● RTF ○ HTML ○ Code ○ OpenOffice

Category: Customized

Comment:

For more details, see Customizing RTF Descriptors.

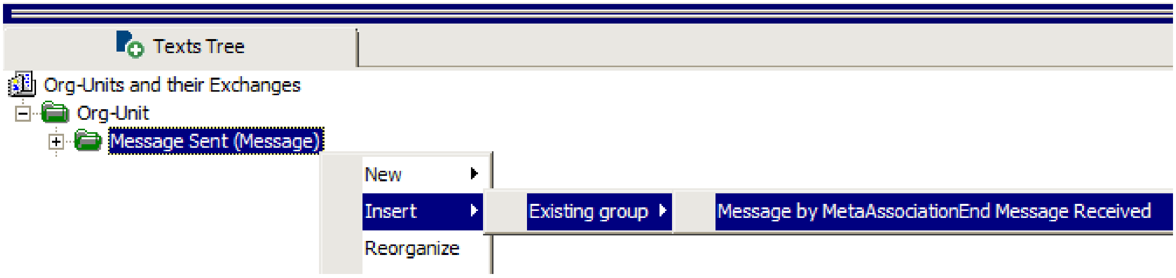
Using Groups

Reusing an existing group

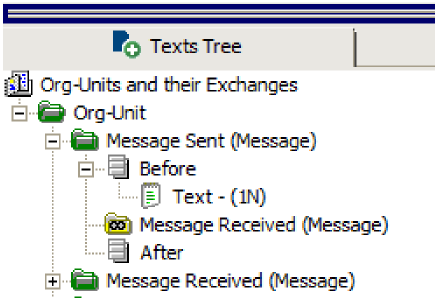
You can reuse an existing group in your descriptor if this group relates to the same object type.

To reuse an existing group:

1. Right-click the group that relates to the same object type as the group you want to add and select **Insert > Existing Group** then the name of the group you want to add.
   Groups relating to the same object type are proposed.


Texts Tree
Org-Units and their Exchanges
Org-Unit
Message Sent (Message)
New
Insert
Reorganize
Existing group
Message by MetaAssociationEnd Message Received

The group connected in this way remains linked to the reference group. An icon is added to the tree structure, but you cannot modify these elements (you can modify the group in which it was initially created).


Texts Tree
Org-Units and their Exchanges
Org-Unit
Message Sent (Message)
Before
Text - (1N)
Message Received (Message)
After
Message Received (Message)

Deleting a Group

To delete a group:

- Right-click the group and select **Delete**.
  All dependent elements are deleted, except for the connected groups and descriptors, which remain in their initial location.

  ☛ *When in doubt, you can begin by deleting the texts and then check the results before deleting the entire group.*

Modifying a group properties

To modify a group properties:

1. Right-click the group and select **Properties**.
2. In the group properties you can modify the query.
3. Click **OK**.

DEFINING THE OBJECT ORDER IN REPORTS (MS WORD)

To sort objects within a group ***in the generated report (MS Word)***:

1. In a group properties, click **Sort**.
   The Sort dialog box of the group opens.
   The default criteria are repository "Order" and "Short Name".
2. To add or remove a sort criterion:
   - in the **Characteristics** list select a characteristic and click **Add** to add a sorting criterion
   - in the **Criteria** list, select a characteristic and click **Remove** to remove a sorting criterion.
3. To order the sort criteria, right-click a characteristic and select **Up** or **Down** as required.
4. If needed, in the **Criteria** list, select an item and in the **Direction** field define the sorting direction ("Ascending" or "Descending").

Examples

- To sort the group objects in alphabetical order, place the “Name” criterion at the top of the **Criteria** list and select **Ascending** order.
- To group the objects by creator, add “Creator Name” characteristic in the **Criteria** list and place it at the top.

To modify the order of objects in reports (MS Word):

1. Check that the descriptor used in your report (MS Word) contains groups relating to a MetaAssociationEnd and not to a query.
   The group should be colored green (not blue).
2. In the diagram, click **Tools > Order**.

> If you want to reproduce the previously defined order in a diagram, you must keep the "Order" criterion at the top of the list. The order can be kept only for green-colored groups, which relate to a MetaAssociationEnd (it cannot be kept for blue-colored groups, which relate to a query).

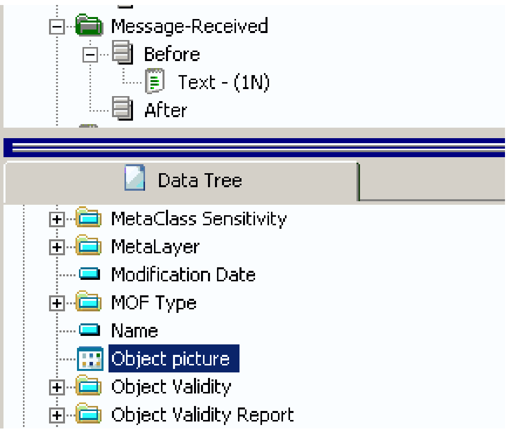
MODIFYING TEXT

The following points are detailed here:

- Adding the Image of an Object
- Showing Metamodel Names
- Component Drag-and-Drop
- Defining Text Formats
- Specifying Text Language (Multilingual Context)
- Inserting Tables into Text

Adding the Image of an Object

The "object image" characteristic allows you to obtain the object image in the generated report (MS Word). This characteristic is available for each object.



In the descriptor editor, the <Object image/> tag is used for this purpose.

Use of this tag simplifies multilingual management by making object type naming optional.

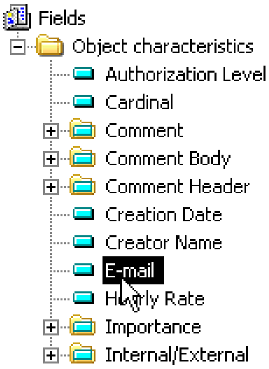
Showing Metamodel Names

You can show metamodel characteristic and link names and object type names in different languages.

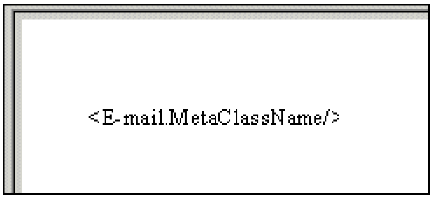
Showing a characteristic name

To show a characteristic name:

1. In the tree, select the icon corresponding to the object characteristic and simultaneously press the <Ctrl> key.



2. Holding the <Ctrl> key down, drag the characteristic name to the text area.
   <Name.MetaClassName> appears in the text area. In the report (MS Word), "Name" appears in the current language.



<E-mail.MetaClassName/>

Showing the object type

To show an object type name:

1. In the "Object Characteristics" folder, select "Object MetaClass Name" and simultaneously press the <Ctrl> key.
2. Holding the <Ctrl> key down, drag-and-drop "Object MetaClass Name" to the text zone.
   <Object MetaClass Name/> appears in the text area. In the report (MS Word), the object type name will appear in the current language.

Summary regarding metamodel names

<Name>	returns object name
<Name.MetaClassName>	returns "Name" in current language
<MetaClassName>	returns object (MetaClass) type name in current language

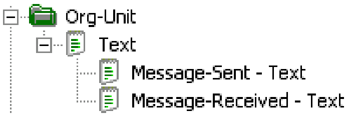
Use of names derived from the metamodel simplifies multilingual management. It avoids multiplication of texts, simplifying maintenance and reducing repository size.

Component Drag-and-Drop

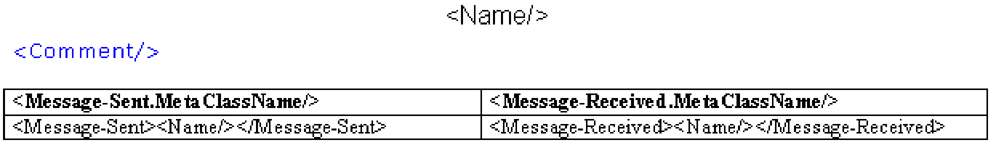
From the descriptor editor you can drag-and-drop object components so as to obtain sub-texts carrying the MetaAssociationEnd name.

To create sub-texts:

1. Drag-and-drop components for example "Message-Sent" and "Message-Received" into the text body of the "Org-Unit" group.



2. In each of the sub-texts, drag-and-drop the "Name" characteristic.
3. Format the text under the group, as below:



<Name/>

<Comment/>

<Message-Sent.MetaClassName/>	<Message-Received.MetaClassName/>
<Message-Sent><Name/></Message-Sent>	<Message-Received><Name/></Message-Received>

☛ *In the properties dialog box of the component texts created, select the **Format** tab, then the **New Line** option.*

This method can prove simpler than adding groups, as described in Using Groups, but it does have limitations. For example, it is not possible to specify text execution conditions as described in Text editing conditions.

Taking inheritance into account in texts with components

If the general option of variations is activated, you can choose to:

- follow default behavior of the descriptor
- or activate/deactivate inheritance

To specify behavior of text regarding inheritance:

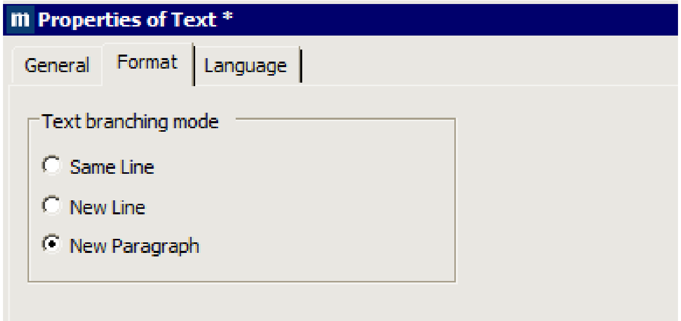
❱ Select or clear the corresponding check box in the text properties.

☛ *The box should remain grayed to follow descriptor default behavior.*

Defining Text Formats

To define a text format:

1. From the descriptor, right-click the text and select **Properties**.
2. Click **Format** tab.



Text branching mode

○ Same Line
○ New Line
● New Paragraph

3. Specify how sequentially linked texts should appear ("text branching mode"), select:
   - **Same Line** for the selected text to appear on the same line as the previous text.
   - **New Line** for the selected text to start a new line without insertion of paragraph spacing between the selected text and the previous text.
   - **New Paragraph** (default) for the selected text to start a new paragraph. This option inserts paragraph spacing between the selected text and the previous text.

Specifying Text Language (Multilingual Context)

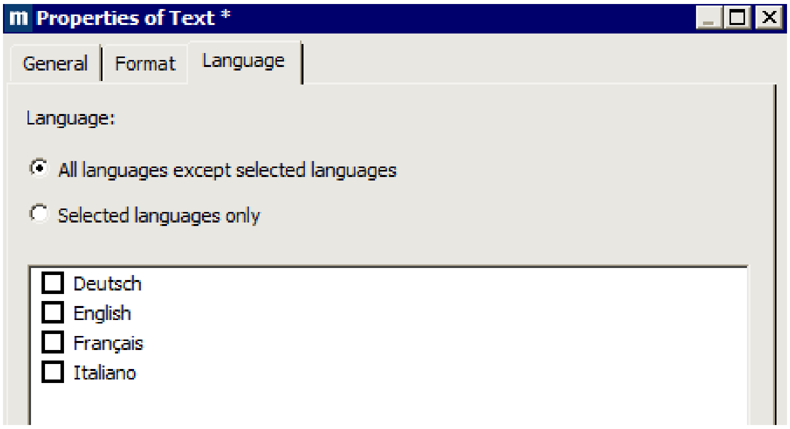
You can edit texts in several languages.

Text editing conditions

To specify in which language text will be edited:

1. From the descriptor, right-click the text and select **Properties**.

2. Select the **Language** tab.
   By default, a text is executed in all languages.
3. Select or remove one or several of the environment languages.



Language:

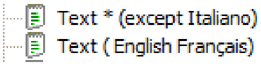
● All languages except selected languages

○ Selected languages only

□ Deutsch
□ English
□ Français
□ Italiano

4. To restrict or extend the choice of languages, you can:
   - specify the languages you want (**Selected languages only** option).
   - exclude unwanted languages (**All languages except selected languages** option).

   In the descriptor menu tree, the languages applied to text are indicated alongside its icon. "*" indicates that all languages have been selected.



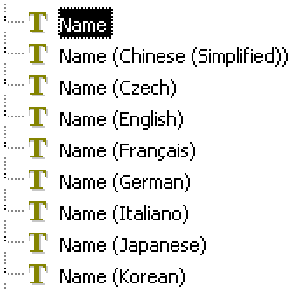
```
Text * (except Italiano)
Text ( English Français)
```

☺ *If you edit the same text in several languages, it can be useful to provide a text with the property **All languages except**. If you add languages at a later stage, all the languages can thus be processed without the risk of any being forgotten.*

Data derived exclusively from MEGA

When you edit texts that contain **HOPEX** data exclusively (not texts that you yourself have entered), it is of no great value to duplicate these texts for each language.

```
Example: enter <&Name&> to edit the name in the current
language (which is not the case if you enter "&Name
(English)&").
```



Name
Name (Chinese (Simplified))
Name (Czech)
Name (English)
Name (Français)
Name (German)
Name (Italiano)
Name (Japanese)
Name (Korean)

☛ *See also Showing Metamodel Names.*

Inserting Tables into Text

If you want your descriptor to appear in tabular form, insert a table into all descriptor texts.

To do this:

1. Access the descriptor structure.

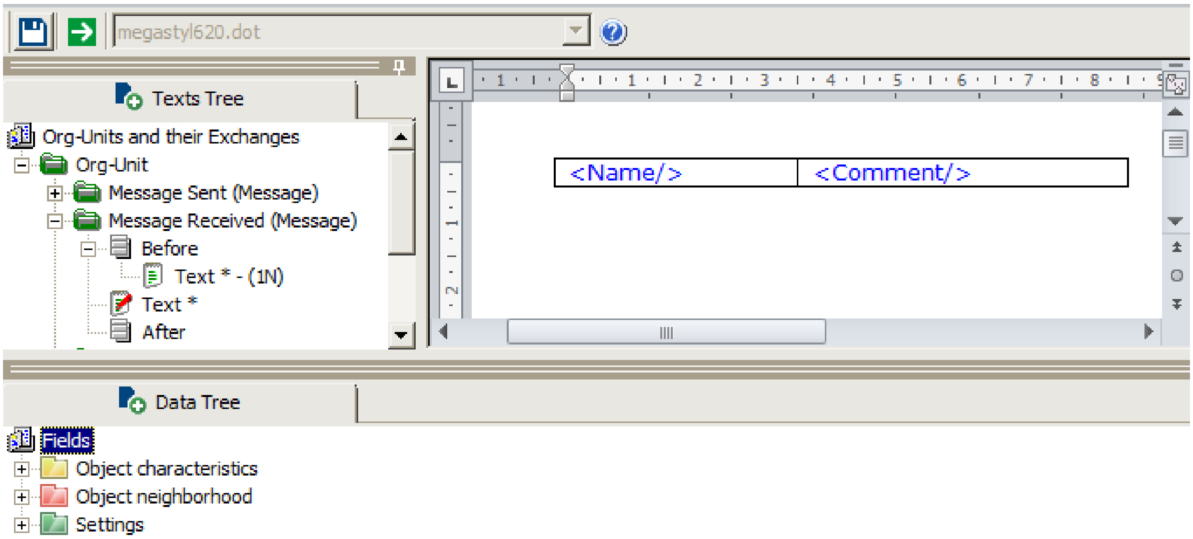
   ☛ *See Accessing the descriptor structure.*

2. Edit the group text.

   ```
   Exemple: "Message-Received" group
   ```

3. From Word menu bar, insert a table and set up the text fields in table format.

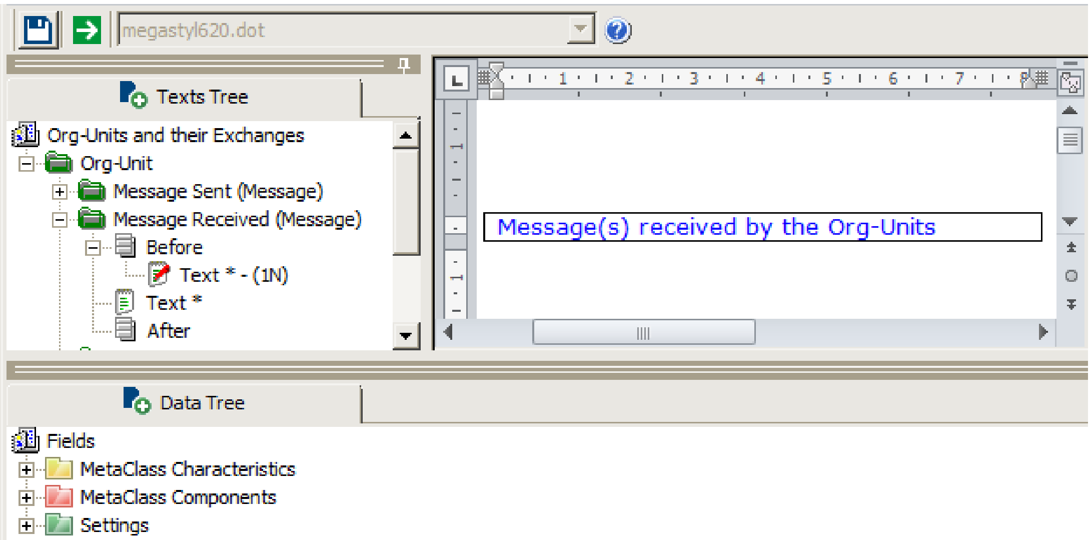
4. Format this text with the “M-Table” style.



<Name/>	<Comment/>

5. Create a table for the “Before” text. Insert a one-row table and enter text describing the contents of the table and format this text with the “M-Table” style.

```
Example: text such as “Message(s) received by the Org-Unit"
```

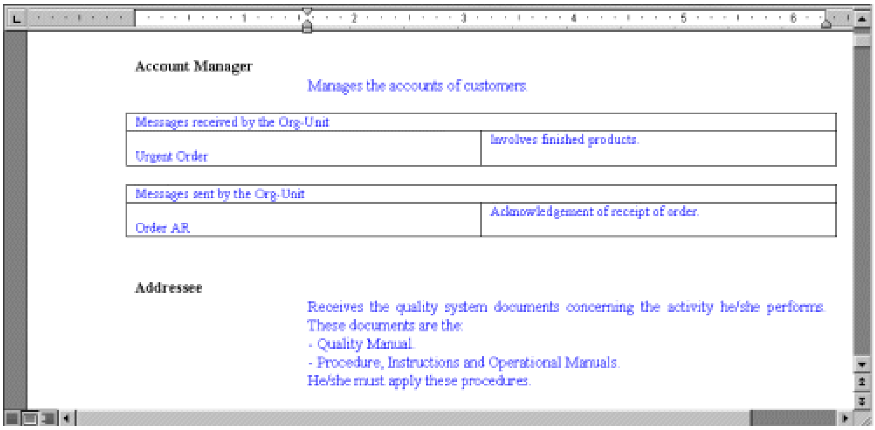


Message(s) received by the Org-Units

6. Right-click the “Text” you modified previously, and select **Properties** to specify the text branching mode.
7. Select the **Format** tab and select **Same Line**, so that there is no break between the tables.

8. Proceed similarly with "Message by the Message-Sent MetaAssociationEnd" group, but copy the tables you just created so that presentation is consistent.

After processing the descriptor, a result similar to the following is displayed:



Account Manager

Manages the accounts of customers.

Messages received by the Org-Unit	
Urgent Order	Involves finished products.

Messages sent by the Org-Unit	
Order AR	Acknowledgement of receipt of order.

Addressee

Receives the quality system documents concerning the activity he/she performs.
These documents are the:
- Quality Manual.
- Procedure, Instructions and Operational Manuals.
He/she must apply these procedures.

☛ *By default, Word 2000 adjusts the columns of the table to fit their contents. For improved legibility of tables generated using **HOPEX**, create the first line of the table in the descriptor editor, then clear the **Automatically resize to fit contents** check box in the table options (in Word, menu **Table > Table Properties**, **Table** tab, **Options** button).*

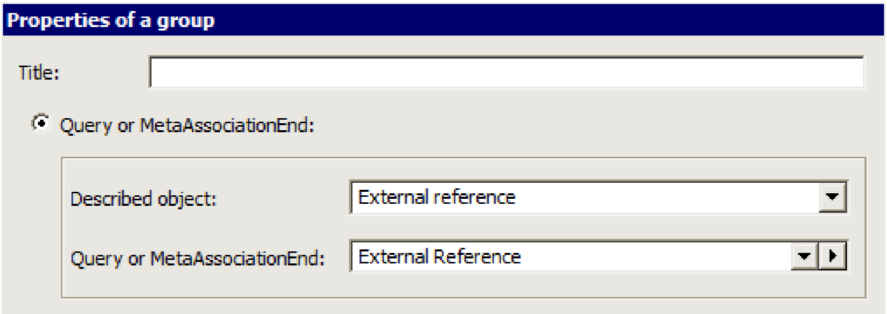
Adding External References in a Descriptor

If you have created external references for certain repository objects, you can include fields corresponding to these references in your descriptors to quote them in generated reports (MS Word).

```
In the "Org-Units and their exchanges" descriptor that you
created earlier, add a group enabling creation of a link to
external references associated with the messages sent.
```

To do add an external reference in a descriptor:

1. in the group (example: "Message-Sent") add a new group.
   The **Group Properties** dialog box opens.



Properties of a group

Title:

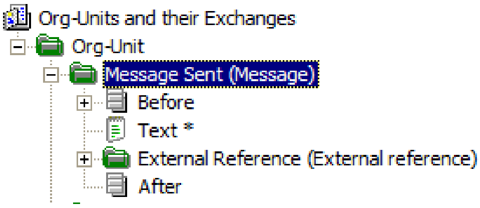
● Query or MetaAssociationEnd:

Described object: External reference

Query or MetaAssociationEnd: External Reference

2. Select **Repository query or leg** and:
   - In the **Described object** field select "External reference".
   - In the **Query or leg** field select "External reference".

     ☛ *Else select **Embedded query** and enter: Select [External reference] where [Message]= "&Message"*
3. Click **OK**.
   An **External Reference** folder is added in the tree.

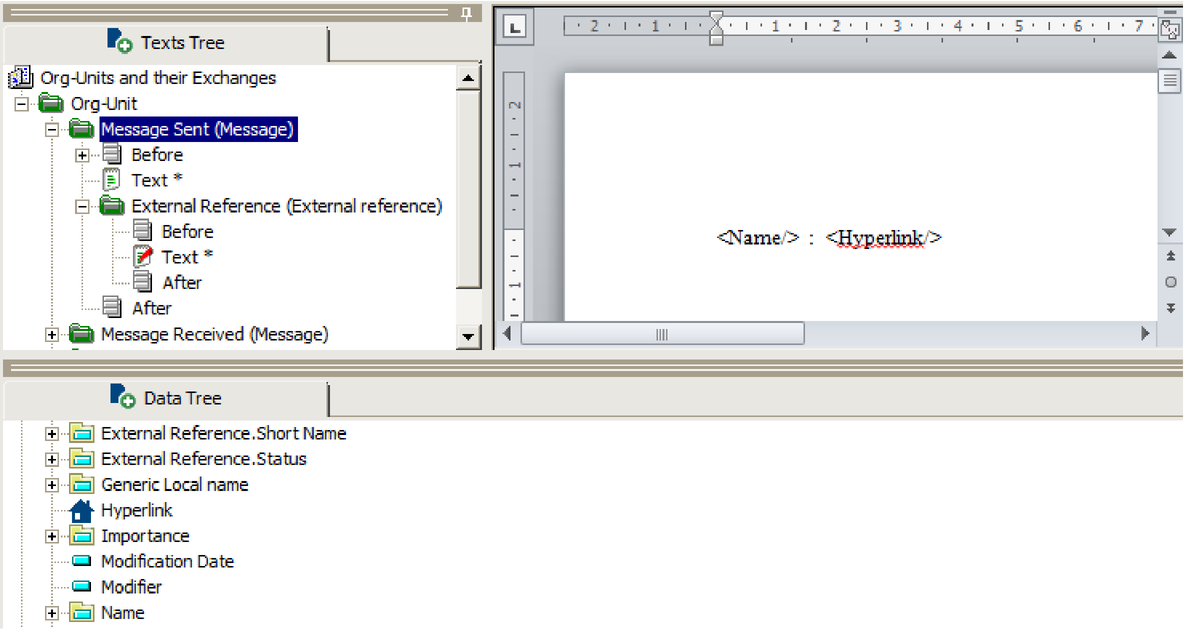


Org-Units and their Exchanges
Org-Unit
Message Sent (Message)
Before
Text *
External Reference (External reference)
After

4. Add a text to this group.

   ☛ *See Adding Text to a Descriptor.*

   An empty report (MS Word) appears.
5. Add the "Name" and "Hyperlink" fields (drag and drop).



Texts Tree

Org-Units and their Exchanges
Org-Unit
Message Sent (Message)
Before
Text *
External Reference (External reference)
Before
Text *
After
After
Message Received (Message)

<Name/> : <Hyperlink/>

Data Tree

External Reference.Short Name
External Reference.Status
Generic Local name
Hyperlink
Importance
Modification Date
Modifier
Name

In the report (MS Word) obtained after processing the descriptor:

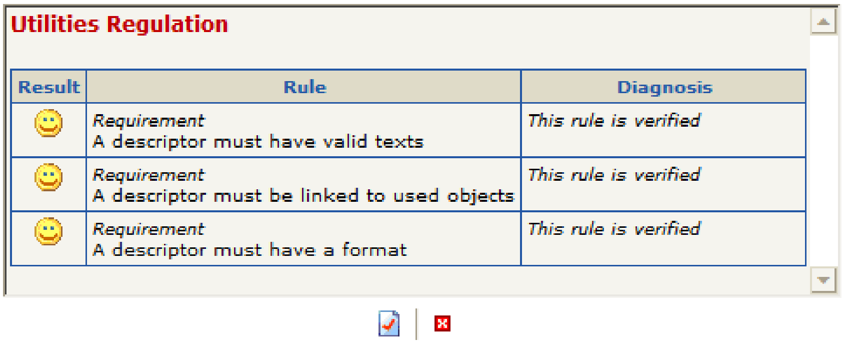
- The "Name" field contains the name you gave to the external reference.
- The "Hyperlink" field indicates either the file path or the URL address, depending on the type of external reference.

  ☛ *The content of the external reference cannot be displayed using this method. To do this, you must use a macro. For more details, see Adding a Diagram.*

CHECKING DESCRIPTOR VALIDITY

To check validity of a descriptor

1. In the descriptor editor menu, select **MEGA > Tools > Check**.
2. Select the **Modeling Regulation**.
3. Click **OK**.
   Errors are displayed. The incorrect field is indicated for each text concerned.



Utilities Regulation

Result	Rule	Diagnosis
☺	*Requirement* A descriptor must have valid texts	*This rule is verified*
☺	*Requirement* A descriptor must be linked to used objects	*This rule is verified*
☺	*Requirement* A descriptor must have a format	*This rule is verified*

☛ *The **Check** menu is available on descriptors, report templates (MS Word) and reports (MS Word).*

MANAGING FORMATS AND STYLES

The following points are covered here:

✓ Reproducing RTF Text Formats

✓ Customizing Styles

☛ *The description of styles in this chapter is useful for those who have the* ***HOPEX Power Studio*** *technical module.*

REPRODUCING RTF TEXT FORMATS

As a reminder, **HOPEX** RTF texts enable formatting of:

- characters (bold, underline, color...)
- paragraphs (bullets, indents, etc.).

Standardizing Formatting

HOPEX ensures consistency of documentation when you generate a report (MS Word): styles take precedence over formats introduced in the RTF text editor.

Reproducing character formatting

The following table indicates how each format is reproduced when generating a report (MS Word) or Web site.

Formatting	Reproduction in report (MS Word)	Reproduction in Web site
Bold	Yes	Yes
Italic	Yes	Yes
Underline	Yes	Yes
Strikethrough	Yes	Yes
Font	No	No
Font size	No	No
MEGA fields	Object name in bold	If objects produce pages, fields are reproduced in the form of links. If not, only the object name is reproduced.

Standardizing formatting - characters

☛ *Formatting of texts generated in tables is ignored in reports (MS Word).*

Reproducing paragraph formatting

Paragraph formatting applies to the complete RTF text, unlike character formatting.

Formatting	Reproduction in report (MS Word)	Reproduction in Web site
Alignment	No	No
Indent	The indent is added to that of the style	The indent is added to that of the style
Bullets	Yes	Yes

Standardizing formatting - paragraphs

☛ *Formatting of texts generated in tables is ignored in reports (MS Word).*

Defining the Formatting in Web sites

When you generate a Web site, you can choose to:

- standardize formatting in web sites, to get consistent results throughout the HTML page (for example: same font, same character size).
- keep RTF formatting of comments entered by users.

You must indicate the generation mode selected for each Web site template.

To define formatting in Web sites:

1. Access the Web Site Template properties.
2. In the **Generation** tab:
   - clear the **RTF font enabled** option, to standardize formating.
   - select the **RTF font enabled** option, to keep RTF text formatting.

Formatting and Objects Not Supported by HOPEX

The following formats are not supported by **HOPEX**, whether executed in RTF descriptors, reports (MS Word), or Web sites:

- Tables
- Pictures
- Word styles (obtained by copy/paste of a Word document)

CUSTOMIZING STYLES

Presentation

Word styles allow you to format object descriptors and reports (MS Word). The style names indicated in object descriptors are included when inserting the descriptors into reports (MS Word) or report templates (MS Word).

It is necessary to use styles to format your reports (MS Word) as the definition of the style 'Normal' style available with Word can vary from workstation to workstation. It is therefore necessary to define specific styles named other than Normal' to control formatting.

Styles

A style is a combination of formats that you can apply to a paragraph of text in a word processor. Styles allow you to systematically apply formats such as fonts, margins, and indents.

HOPEX reports (MS Word) include styles.

The names of the styles provided in the default style sheet are prefixed with “M-”. These styles are based on the “M - Normal” style, which is identical to the MS-Word “Normal” style. The only difference is the character color, which is blue.

☺ *To modify the font used for all the “M -” styles, modify the font of “M - Normal”.*

Style sheet

The styles (character and paragraph format, etc.) are defined in a style sheet (which is a Word document template).

Default style sheet

The default style sheet used by report templates (MS Word) and object descriptors is “Megastyl620.dot”. It is located in the Mega_Std folder. It is recommended to use this style sheet.

☛ *For more information, see Customizing the default style sheet.*

Specifying a style sheet

To change the default style sheet:

1. From **HOPEX** menu bar, select **Tools > Options**.

2. Expand the **Documentation** folder.
3. Select **Reports (MS Word)** and in **MEGA style sheet** field indicate the name of the document template you want to use as the document template style sheet.

Use of the same style sheet is recommended for descriptors, report templates (MS Word) and reports (MS Word).

Structure of HOPEX Styles

HOPEX descriptor styles are most of the time organized logically. The style names indicate their purpose, independently of their format.

```
For example, diagrams are style "M-Drawing".
```

To print the description of styles:

1. In Word, open the document template used (default: "Megastyl620.dot" in the MEGA_STD folder).
2. From the **File** menu, select **Print**.
3. In the **Settings / Document properties** section, select **Styles**.
   The styles used in the style sheets will be printed.

Creating and Modifying Styles

If a style format does not suit you, instead of applying another style you should modify the style provided. This way you will not have to open all the descriptors to change the style name.

Modifying a style

To modify a style:

1. Open the style sheet ("xx.dot", the default being "Megastyl620.dot").
2. Save it in the "Mega_Std" folder of the custom module tree-structure.
3. Modify the style.

Remarks on modification of styles

The default style sheet "Megastyl620.dot" is write-protected in the Mega_Std folder. You can create a new one in the "Mega_Std" folder of the custom module.

Styles are defined hierarchically.

```
For example, "M-Sublevel 2" is based on "M-Sublevel 1".
```

This is why modifications made to one style may impact other styles.

You can modify a style without having to modify the object descriptors; all descriptors whose text uses this style will include the modification. To ensure that modifications to a style are taken into account, they have to be included in the style sheet defined in the user options. All reports (MS Word) defined using report templates (MS Word) or descriptors based on the style sheet will then be modified accordingly.

To modify the orientation (portrait/landscape) or the margins of generated reports (MS Word):

- Modify the document template directly.

Parameterizing the report templates (MS Word) allows you to define which style sheet should be used when creating a report (MS Word) based on this report template (MS Word). The user can modify the template using standard MS-Word template and style manipulation functions.

💣 **Use of frames is not recommended in definition of styles. It is also not recommended to use images in the style sheet (for example in the footer).**

Creating a new style

You can create a new style that can be used in the texts of object descriptors.

When you create a new style, you have to add it to the “Megastyl620.dot” style sheet. It then becomes available to the object descriptors.

💣 **Never insert the style directly into the descriptor. To ensure that the style is properly defined in generated reports (MS Word), it must be included in the style sheet specified in the user options. Avoid using the same style in the body of the text and in a table. If you later reduce the size of this style, cells using this style could become so small they are illegible.**

☛ *Styles provided in the default style sheet are prefixed with "M-". However the new styles you create do not have to be prefixed with "M-".*

Attaching a style sheet

You can attach a style sheet to an environment or to a user:

- The style sheet attached to the environment will be the default for each user you create.
- The style sheet attached to a user will be the default when the user creates a new report template (MS Word) or descriptor.

☛ *After creation, the report (MS Word) is independent and modifications made to the style sheet are not automatically reflected in the report (MS Word).*

☛ *Styles taken into account in a report (MS Word) created from a report template (MS Word) are those contained in this report template (MS Word). Only the names of styles defined in a descriptor are included in the report (MS Word). The details for the styles defined in the descriptor are replaced by those inherited from the report template (MS Word).*

When a style used in a descriptor does not exist in the report template (MS Word) being used, it is replaced by the “Normal” style. For this reason, it is recommended that you create style sheets using “Megastyl620.dot” as the template.

You can define or modify a style by testing it in a descriptor. However, to make this new style or modification available to new report templates (MS Word), you should also define or modify the style in the style sheet attached.

Customizing the default style sheet

It is advised to use the Megastyl620.dot syle sheet provided.

To customize this style sheet:

1. Copy the reference style sheet ("Megastyl620.dot") to the "Mega_Std" folder of the custom module tree-structure.
2. Open the style sheet with MS Word, without running **HOPEX**.
3. Make the style modifications or create a new style.
4. Using **HOPEX**, attach the style sheet to the user, so that it becomes the default style sheet for descriptors.

> ☛ *Attach the style sheet by indicating this style sheet in the* ***Documentation > Reports (MS Word)*** *options.*

Applying styles in a descriptor

It is recommended to attach a style sheet to a descriptor.

To apply or make sure modified styles are available in a descriptor:

1. Edit or update styles as described in Customizing the default style sheet.
2. Open the descriptor and select **MEGA > Properties**.
3. In the descriptor properties, select the desired RTF stylesheet.
   When the descriptor is edited, it is loaded with the appropriate stylesheet.

Synchronizing styles in a report template (MS Word)

It is sometimes necessary to synchronize styles between a report template (MS Word) and the stylesheet.

> ☛ *The prerequisite for the following procedure is to customize the MS Word ribbon and to add a button to call MS Word options.*

To synchronize styles:

1. Open the report template (MS Word).
2. Click the MS Word options button you have added to the ribbon.
3. Select Add-Ins on the left in the MS Word options dialog box.
4. Select Manage "Word Add-ins" and click Go.
5. Click the **Organizer** button.
   The Windows Organizer is loaded.

   Note that:

   - The styles displayed on the left (document Unnamed) are the styles of the report template (MS Word) saved in the system repository
   - The styles displayed on the right will be used to access the RTF stylesheet.
6. Below the right list click **Close File** then browse to get the .dot file of the RTF stylesheet.
7. Add styles from the right list (RTF stylesheet) to the left (Report template (MS Word)).

> ☛ *Note that some styles of the left list cannot be deleted (Normal, Heading 1 for instance).*

Description of Default Styles (MegaStyl620.dot)

Here are some of the styles provided in the default style sheet.

Titles

Title 1 For chapter titles, it is numbered and automatically inserts a page break before.

Title 0 Identical to Title 1, but does not include numbering or page break. For contents table and distribution list.

Title 2 Defines the various sub-chapters. It is centered and does not include numbering.

Title 3 Italic and does not include numbering.

M - Document Title Used on front page. It is centered, in font 26. It takes the report (MS Word) title.

M - Document Title 2 Used on front page. It is centered, in font 22. It is used to supplement the title if necessary.

M - Sublevel
M - Sublevel 2
M - Sublevel 3 Enable creation of additional paragraph titles that do not however appear in the contents table.

Texts

M - Comment Text default format.

M - List Style used for lists. A bullet precedes the beginning of the line.

M - Remark Style used for a comment preceded by a special bullet. Can be used to indicate remarks in a report (MS Word).

Tables

M - Table Default format for a cell.

M - Table Center Format used to center text in cells.

M - TableHeading Title cell format, centered with gray fill.

Drawing

M - Drawing Style for diagrams.

Contents table

TM1	Contents table line style connecting to "Title 1".
TM2	Contents table line style connecting to "Title 2".

Index

Index title	Index title style
Index 1	Generated index data style.

Customizing Business Documents

HOPEX lets you store, classify, reference and update documents whose contents are independent of the repository. Business documents follow particular storage and classification rules; this chapter presents how to customize these.

☛ *For a presentation of business documents, see "Generating Documentation" chapter, in **HOPEX Common Features** guide.*

MANAGING BUSINESS DOCUMENTS PREREQUISITES

To use customization functions of business documents, you must have available certain products or options.

Below is a summary of functions available with each product or option.

Product/Option	Functions
HOPEX Power Studio	Defining documentable objects Customizing macros
Products or solutions	Accessing documentable objects.

DEFINING BUSINESS DOCUMENT METACLASSES

By default, only some MetaClasses instances can be connected to a business document.

An instance can be connected to a Business Document if:

- the instance MetaClass must inherit the "Element with business document" abstract MetaClass, see Using Business Document MetaClass,
- the instance properties must be updated to connect a Business Document, see Updating MetaClass properties.

> ☛ *Access to the **HOPEX** metamodel is only possible with a **HOPEX Power Studio** license.*

Using Business Document MetaClass

To add the possibility of connecting business documents to a specific MetaClass:

1. Right-click "Element with business document" abstract MetaClass and select **Explore**.
   An explorer window opens.
2. Expand the "SubMetaClass" folder.
3. Connect the specific MetaClass to this folder.

Updating MetaClass properties

By default, a Business Document can be connected to an object using the **Characteristics** Properties and the **Attachements** section.

To update the specific MetaClass properties:

1. Open the **MetaPropertyPage** of the MetaClass presenting the instances Characteristics,
2. Open the **Setup** property page of MetaClass Characteristics MetaPropertyPage,
3. Insert the text below:

```
GroupAttachment =
Group(Bar),Name(~bzc)jC04G5r5[Attachments])

Attachments = Item(~LKKM8XSyWvF6[Shared Attachments]),

In(GroupAttachment),Control(SubPage)
```

> ☛ *"Attachments" is a **_Code Template***

> ☛ *"Shared Attachments" is a **MetaPropertypage***

Customizing Web Sites

CREATING A WEB SITE

Every **HOPEX** product enables generation of a Web site from objects in the repository. This allows simple distribution of the content of the **HOPEX** repository via an Intranet. In this way, users can consult the objects of a project (for example, processes) in the form of HTML pages.

The following points are covered here:

About Web site Creation

A Web site can be created in two ways in **HOPEX Windows Front-End**:

- without using a template (you need **HOPEX Power Studio** technical module).
- from a Web site template that serves as a template.
  The pages are generated independently of each other, and the links between the repository objects automatically define the hypertext links.

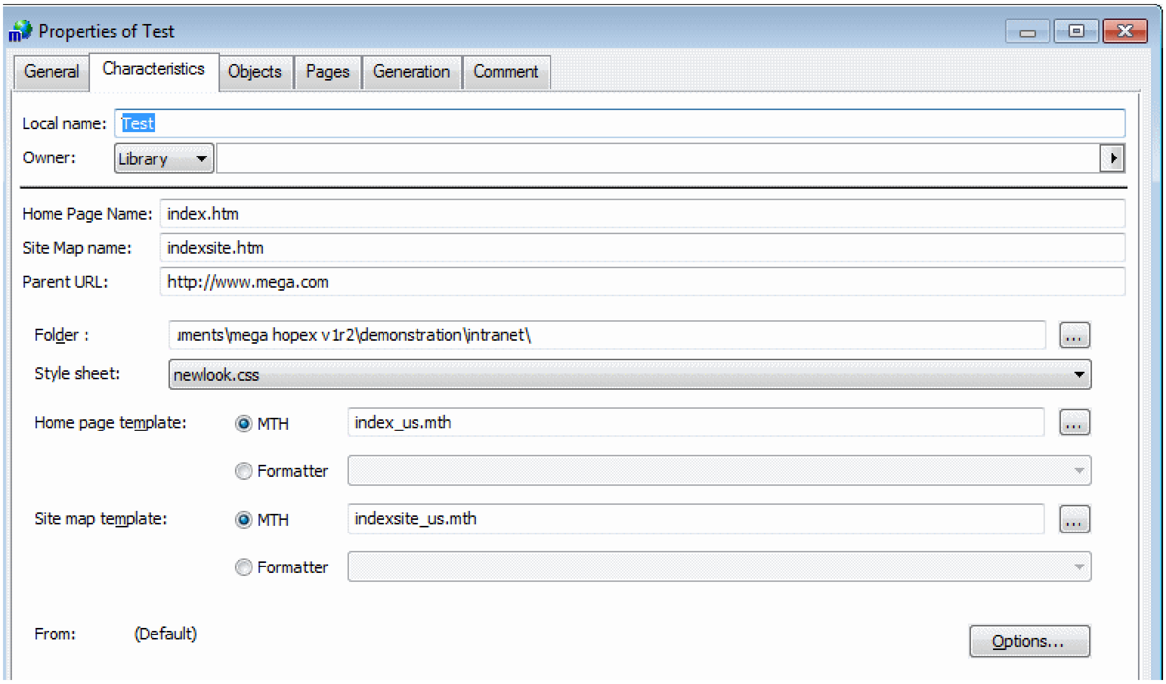
The Web site generator builds the site pages from the repository data and the objects you have chosen to describe.

Defining Web Site Composition

Before generating a Web site, you need to define its composition and page setup by entering its properties.

To define the Web site composition:

1. In **HOPEX Windows Front-End**, right-click the Web site and select **Properties**).
2. Indicate the properties and format of your Web site, see:
   - Defining Objects
   - Adding Pages
   - Adding Index Pages



Local name: Test
Owner: Library
Home Page Name: index.htm
Site Map name: indexsite.htm
Parent URL: http://www.mega.com
Folder : ıments\mega hopex v1r2\demonstration\intranet\
Style sheet: newlook.css
Home page template: ● MTH index_us.mth ○ Formatter
Site map template: ● MTH indexsite_us.mth ○ Formatter
From: (Default)

Defining Objects

You must indicate which main object or main object list you want to display in your Web site. You can add projects, processes, etc. that group basic objects.

> Example: You can build your site from a project. All the component objects of the project will appear in the Web site when you create the corresponding pages.

To define objects you want to display in your Web site:

1. From the Web site **Properties,** select the **Object** tab.

2. To add:
   - objects, see Adding an object
   - object lists, see Adding an object list
3. If you do not want the objects that depend on the main object to appear in the Web site, you need to deactivate the extraction process: clear **Propagation**.

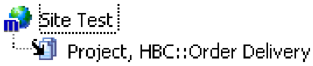
   ☛ *Extraction carried out starting from a project includes the diagrams of this project, with for each diagram the objects represented such as org-units, messages. For detailed information, see the* ***HOPEX Power Supervisor*** *guide.*
4. Click **OK**.

Adding an object

To add an object:

1. From the Web site **Properties,** select the **Object** tab.
2. Click **Add** and select **An object**.

   ☛ *You can also right-click the web site and select the button* ***Add Object****.*
3. In the **Add object** dialog box, select the **Type** of object you want to add.
4. In the **Name** field, click the arrow and select **List**.
5. Select the desired object and click **OK**.
   The object is added in your Web site tree.



Site Test
Project, HBC::Order Delivery

6. (Optional) Add other projects or objects.

   ☛ *You can also open the object properties to obtain information or carry out modifications prior to generation.*

Adding an object list

To add an object list:

1. From the Web site **Properties,** select the **Object** tab.
2. Click **Add** and select **A Query**.

   ☛ *You can also right-click the web site and select the button* ***Add Query****.*
3. In the **Type** field, select the object type to be queried.
4. In the **Query** field, select the query.
   Select:
   - "*" if you want to add all repository objects corresponding to the selected object type, or
   - "[Object collection]" to select only certain objects of a given type.

Adding Pages

A page is a hypertext document that is part of a Web site. You need to add pages for each object to be covered in a page.

To add pages to your Web site:

1. From the Web site **Properties,** select the **Pages** tab.
2. In your Web site tree, right-click the **Pages** folder and select **Add**.
   The **Add Page** dialog box opens.
3. In the **Object** field select the type of object described
4. (optional) If necessary, you can select a **Query** (filter) to restrict the list of objects to be included in a page.

   `Example: Quality Assurance organizational processes`

   ☛ *You cannot use queries that contain variables. For this reason they are not proposed in the list. If no query appears, click* ... .
5. In the **Format** pane, define the HTML page Format:
   - **Header**
   - **Body**

   ☛ *See Text formatting.*
6. Click **OK**.

Text formatting

HTML pages include two parts:

- Header
- Body

Header

An HTML page header contains the title of the page (visible to the user), META information that defines keywords that could be used by search engines, the style sheet, etc.

HOPEX proposes three types of header:

- Standard header
  The Standard header automatically specifies the style sheet, the page title (object name) and the keywords associated with the object.

  If you have **HOPEX** keywords associated with an object (**Keywords** navigation window), these keywords appear in the header of the page describing the object.

  Example: <META NAME="keywords" CONTENT="Order, Delivery">.
- Header defined by an HTML descriptor
  HOPEX offers the possibility of creating HTML Header descriptors that supplement the META information contained in the HTML page header. You can therefore add keywords, the page title and the style sheet using standard HTML tags. If you have created a specific header, you can select it in the **Header** drop-down list

  ☛ *The descriptor should not contain <Head> ... </Head> tags.*
- Header defined by a script formatter
  You can describe a header using a script formatter connected to a VB macro.

  ☛ *See Level 2: Script formatters configuration.*

Body

The body constitutes the visual content of the HTML page. There are different types of format:

- Formatting using HTML descriptors.

  ☛ *See Using HTML Descriptors.*

- Formatting using script formatters

  ☛ *See Level 2: Script formatters configuration.*

Modifying pages

To modify a created page:

1. Right-click the page and select **Properties**.
2. You can modify:
   - its **Name**
   - its **Query**
   - its **Format** (header and body)

Adding Index Pages

Index pages facilitate access to your Web site pages.

Indexes list the objects that appear in the Web site.

You can create index pages from objects already the subject of a page.

If the index is not created from an object page, you can customize it using an HTML descriptor or VB Script. The descriptor should be of category "HTML Specific Body"

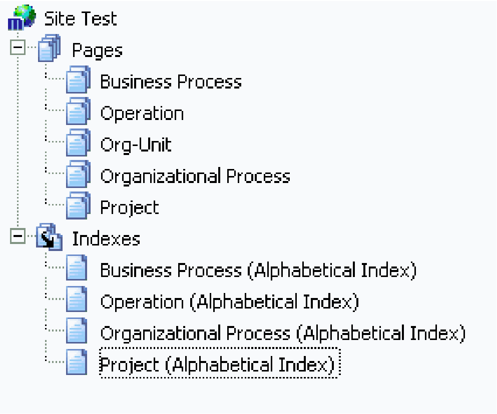
☛ *This descriptor should be created from the "HTML Specific Body" sub-folder of the "HTML Web Site Specific Page" folder in the* ***Utilities*** *navigation window).*

To create an index page from an object:

1. From the Web site **Properties,** select the **Pages** tab.
2. In your Web site tree, right-click the **Indexes** folder and select **Add**.
   The **Add Index Page** dialog box opens.
3. From the **Contents** pane, select the page.

4. From the **Index type** pane, select the type of object that will form the contents of the index.
   The index type allows you to determine the format of your index.

   When your pages and index pages have been created, you obtain a result like this:



- Site Test
  - Pages
    - Business Process
    - Operation
    - Org-Unit
    - Organizational Process
    - Project
  - Indexes
    - Business Process (Alphabetical Index)
    - Operation (Alphabetical Index)
    - Organizational Process (Alphabetical Index)
    - Project (Alphabetical Index)

☛ *When you are defining Web site pages, do not specify two different formats for the same object type: when the page is generated, only the latest format will be taken into account.*

5. Click **OK**.
   The properties of your Web site have been specified.

Contents of a Generated Web Site

A generated Web site contains several types of page:

- Site Map
- Index pages
- Pages specific to each object in the repository Hypertext links in these pages provide access to related pages. The pages are divided into several parts or frames.

Presentation of pages can be customized. For more details, see Configuring Web Sites.

Home page

The Home Page is the main page of a Web site, used as a starting point for accessing the other pages. It indicates the main object described by the site. It also contains the site comment, if one was entered in the Comment tab of the Web site properties dialog box.

To return to the Home Page from another site page:

❱ Click the 🏠 icon.

The Home Page also provides access to the parent URL (represented in the Home Page by the Enterprise Site link).

To enter the parent URL address:

- Select the **Characteristics** tab of the Properties dialog box when you create your Web site.
  You can use the URL address of your company Web site, for example.

  ☺ *If the name of the object or objects you indicated in the **Objects** tab do not appear as a hypertext link, this is because you have not defined the corresponding page.*

Site map

The Site Map provides an overview of the Web site contents. It includes links to the index pages and to the pages describing the site objects.

Index pages

The index pages list the objects by object type.

Pages specific to each object

There are several parts / tabs in pages describing an object in the repository. At the top of the page tabs provide a navigation tool.

Example: A process may have as its components org-units, notes, a diagram, chapters, external references that you can access by clicking on the corresponding link.

☛ *It is also possible to display the parent objects of the object concerned, which are the objects that are linked to it at a higher level in the hierarchy, for example the project(s) containing the organizational process. However, the default is to hide this facility in order to make your pages more readable. We will discover how to display these parent objects in Configuring Web Sites.*

Web Site Folders and Files

Files supplied as standard and those resulting from generation are located in the the following folder: ***<environment path>\Intranet***

☛ *A sub-folder for each Web site is automatically created (if it has not been specified at the time of creation of the Web site).*

In the "Intranet" folder of the environment or in the sub-folder corresponding to the Web site you will find:

- The "Standard" folder that contains the standard images provided for use when generating the Web site.
- The "Pages" folder that contains the generated files ("*.htm" extension).
- The "Images" folder that contains the generated diagrams.

  ☛ *The name of the default folders can me modified. See Modifying the name of Web site folders.*

- Files corresponding to the Home Page and Site Map.

  ☛ *See Home page and site map.*

Viewing or modifying the location of web site files

To view or modify the location of web site files:

- Open the properties of the web site in **HOPEX Windows Front-End**.

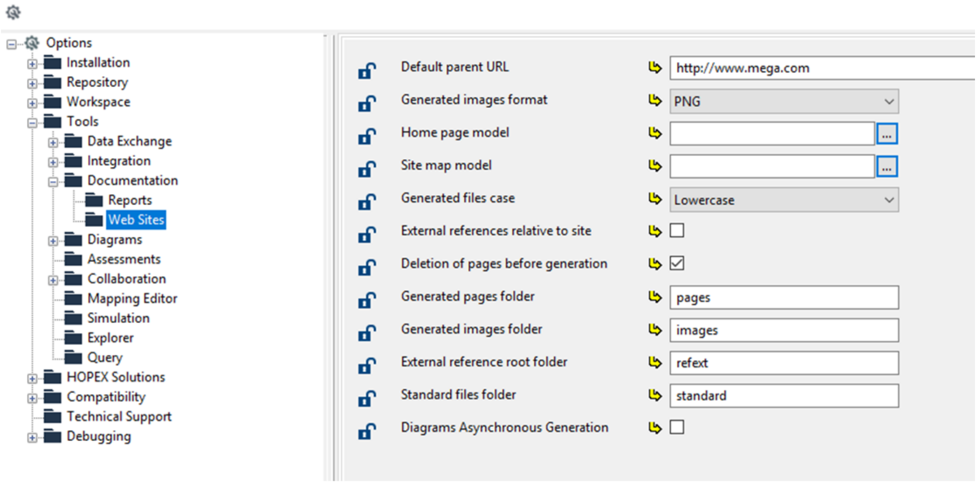
Modifying the name of Web site folders

To modify the name of web site folders:

1. Launch "Administration.exe" and open the site options.

   ☛ *See Modifying options at site level.*

2. Expand the **Tools > Documentation > Web sites** folder.
3. Modify the name of the "Generated pages", "Generated Images" or "Standard files" folders.



Default parent URL: http://www.mega.com
Generated images format: PNG
Home page model:
Site map model:
Generated files case: Lowercase
External references relative to site: □
Deletion of pages before generation: ☑
Generated pages folder: pages
Generated images folder: images
External reference root folder: refext
Standard files folder: standard
Diagrams Asynchronous Generation: □

Home page and site map

File names for the Home Page and Site Map can vary depending on your specifications. If a name does not appear in the **Characteristics** tab of the site properties dialog box, this means that the corresponding page has not been generated.

☛ *By default, the Home Page and the Site Map have the navigation menu on the left. If you want to position this at the top, you must change the Home Page and Site Map page templates. They can also be customized by copying templates in the Mega_Std folder of the has.custom module. For more detailed information, see Modifying Home Page and Site Map Files.*

Using a Web Site Template

A Web site can be created from a Web site template (in the same way as a report (MS Word) can be created from a report template (MS Word)).

The Web site template is not intended for generation but only as a template for building the Web site.

Creating a Web Site Template

In a Web site template, you indicate the type(s) of object to be described in the site as well as the pages generated. This avoids you having to specify these at each new creation of a Web site.

To create a Web site template:

1. In **HOPEX Windows Front-End** workspace, select the **Utilities** window (accessed by selecting **View > Navigation Windows > Utilities**).
2. Right-click the **Web Site Template** folder and select **New > Web site template**.
3. In the **Create Web Site Template** dialog box, indicate its **Name** and click **OK**

☛ You can base a Web site template on another Web site template. To do this, you must duplicate the existing Web site template. See Duplicating a Web Site Template.

The Web site template is placed on the desktop, and in its properties dialog box you must define the described objects and pages as we did for a Web site.

Adding Pages

To add pages:

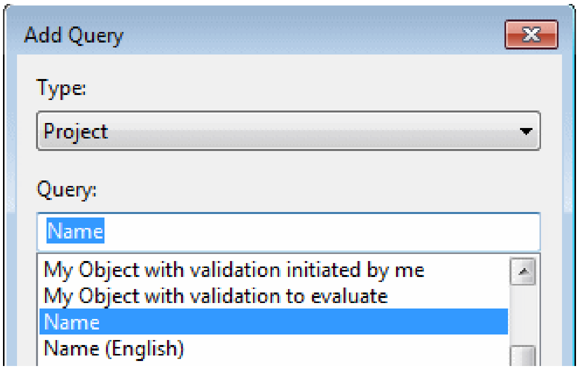
1. In the properties dialog box of the Web site template, select the **Pages** tab.
2. Add a page and/or index page for all the object types that will form a page, as described in the corresponding paragraph for Web site pages Adding Pages.

Selecting Objects

To select objects:

1. In the properties dialog box of the Web site template, select the **Objects** tab.
2. Add a query that will produce a list of objects appearing on the Web site. The **Add Query** dialog box appears.

3. Select the object type and the query, for example the "Project" object type and the "Name" query.



Add Query

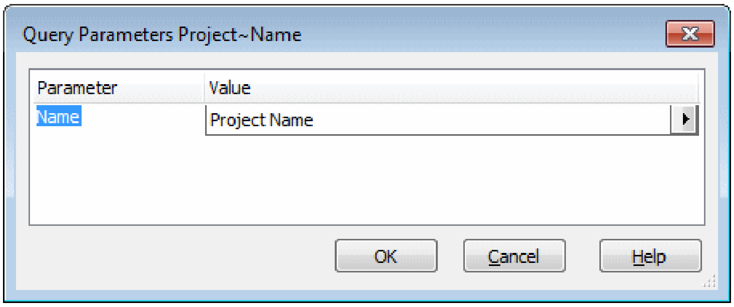
Type: Project

Query: Name

My Object with validation initiated by me
My Object with validation to evaluate
Name
Name (English)

4. Click **OK**.
   When parameters must be resolved, the **Query Parameters** dialog box appears.
   Under the **Value** field, you can indicate:
   - The name of the parameter of which the value will be requested when creating the Web site, for example "Project Name".

     ☛ *Several parameters can be created.*



Query Parameters Project~Name

Parameter	Value
Name	Project Name

   - The name of the object to which the Web site relates. In this case, the selected object is taken into account directly as parameter value when creating the Web site. To select an object, you must clear the **Setting** check box, click in the **Value** field and find the object in question.

     ☛ *You have a choice between parameters and objects and cannot mix the two.*

5. Click **OK**.
   You can now create a Web site from this Web site template.

   ☛ *The Web site template can be configured in the same way as a Web site.*

Creating a Web Site from a Web Site Template

To create a Web site from a Web site template:

1. In the **HOPEX Windows Front-End** workspace, select the **Documentation** window.

   ☛ *To access the **Documentation** window from the **HOPEX** menu bar, select **View > Navigation Windows > Documentation**.*

2. In the tree, right-click the **Web Sites** folder and select **New > Web Site**.
   The **Create Web Site** dialog box appears.
3. Enter the **Name** of the Web site (eg: "Site 1").
4. By default, the Web site is generated in the "Intranet" folder. You can select an alternative folder by clicking the ... button. It is recommended that you create a sub-folder for each Web site generated, for example by creating folders with the same name as your Web site under the "Intranet" folder.

   ☛ *The content of the "Intranet" folder is deleted at each generation. This being so, when you have obtained a satisfactory site you want to keep, remember to copy it to an appropriate folder.*

5. In the **From Web site template** list box, click the ▼ button and select a Web site template.
6. Click **Finish**.
   The new Web site (eg:: "Site 1") is listed in the **Web Sites** folder.
7. Right-click the new Web site (eg: "Site 1") and select **Properties**.
8. In the properties dialog box of the Web site, select the **Objects** tab.
9. To restrict the range of objects to be generated, click the unwanted object types and delete these using the **Delete** button.
10. To add an object to the range of objects to be generated, click **Add** and select **An Object**.
11. In the **Add Object** dialog box, select the object **Type** (eg: "Project".
12. In the **Name** box, select the object (eg: "Customer Loyalty Program") and click **OK**.
    The object appears in the list.
13. Close the Web site properties dialog box.

Creating a Web Site Template from a Web Site

When you have created your Web site, you can save it as a Web site template in order to reuse it.

To create a Web site template from a Web site:

- In the pop-up menu of the Web site, select **Manage > Save as Web Site Template**.

Duplicating a Web Site Template

Duplication of a Web site template enables creation of a Web site template based on an existing Web site template.

To duplicate a Web site template:

- In the pop-up menu of the Web site template, select **Duplicate**.

Two duplication strategies are proposed.

- Partial duplication (**Reuse descriptors, queries and macros**)
  Only the Web site template is duplicated. Descriptors, queries and macros will be the same as those of the source Web site template.

  This is the strategy you will use when you wish to create a Web site template from another Web site template.
- Total duplication (**Duplicate descriptors, queries and macros**)
  Total duplication duplicates the Web site template, descriptors, queries and macros.

CONFIGURING WEB SITES

Here you will discover how to configure your Web site if you find the result you first obtain by default to be unsuitable.

Configuration can be carried out:

- Overall, by modifying the general options of the Web site.
- For each object type, by modifying the parameters of the pages to be generated and by using style sheets.

☛ *Here we shall explain how to configure a Web site. These explanations are also valid for configuration of Web site templates.*

The following points are covered here:

About Web Site Configuration

Web pages are formatted using HTML formatters.

These formatters are of two types:

- HTML descriptors
- script formatters

Level 1: HTML descriptor configuration

☛ *See "Using HTML Descriptors", page 141, and Managing Web Sites: Advanced Functions*

Technologies used

- MEGA: HTML descriptors, knowledge of metamodel
- Web: HTML and CSS

Level 2: Script formatters configuration

These are HTML formatters connected to a VB macro.

The HTML formatter is connected on one hand:

- to one or several _Types, depending on pages in which you wish it to appear:
  - Object HtmlFormatter (object page formatter)
  - Index HtmlFormatter (index page formatter)
  - HomePage HtmlFormatter (home page formatter)
  - SiteMap HtmlFormatter (site map formatter)
- to a VB macro.
  The macro should implement the following method:

```
Public Sub Generate(oObject, oWebContext, strUserData,
strResult)

End  Sub
```

Technologies used

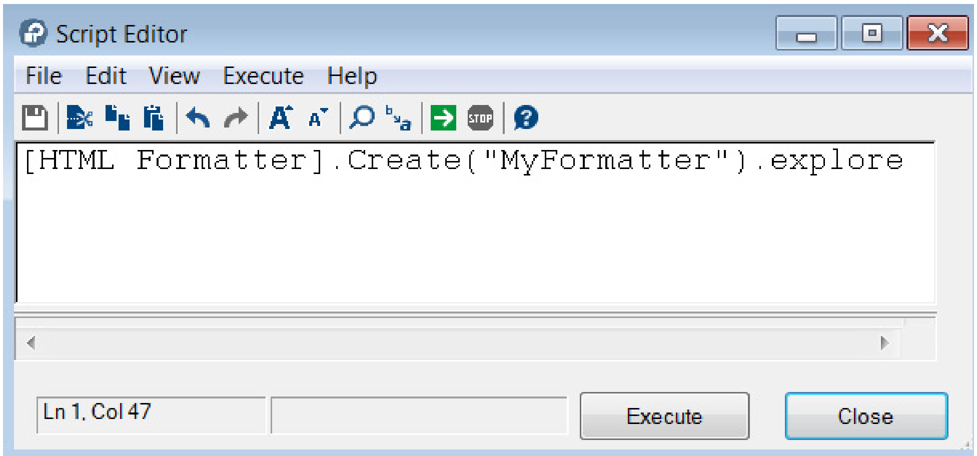
- MEGA: API Script, metamodel knowledge
- Web: HTML, CSS, VB Script

Creating a script formatter

To create a script formatter:

1. From **HOPEX** menu bar, select **Tools > Script Editor**.

2. Create an HTML formatter and click **Execute** ➔.

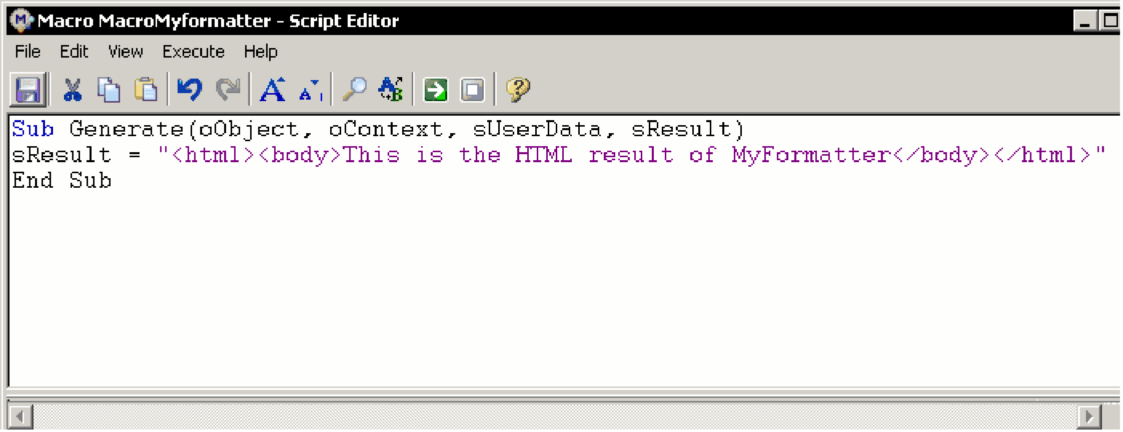


```
[HTML Formatter].Create("MyFormatter").explore
```

3. In the explorer that appears, access the formatter properties.
4. In the **_Type** field, connect the formatter to a _Type.

   `For example: Object HtmlFormatter (an object page formatter).`

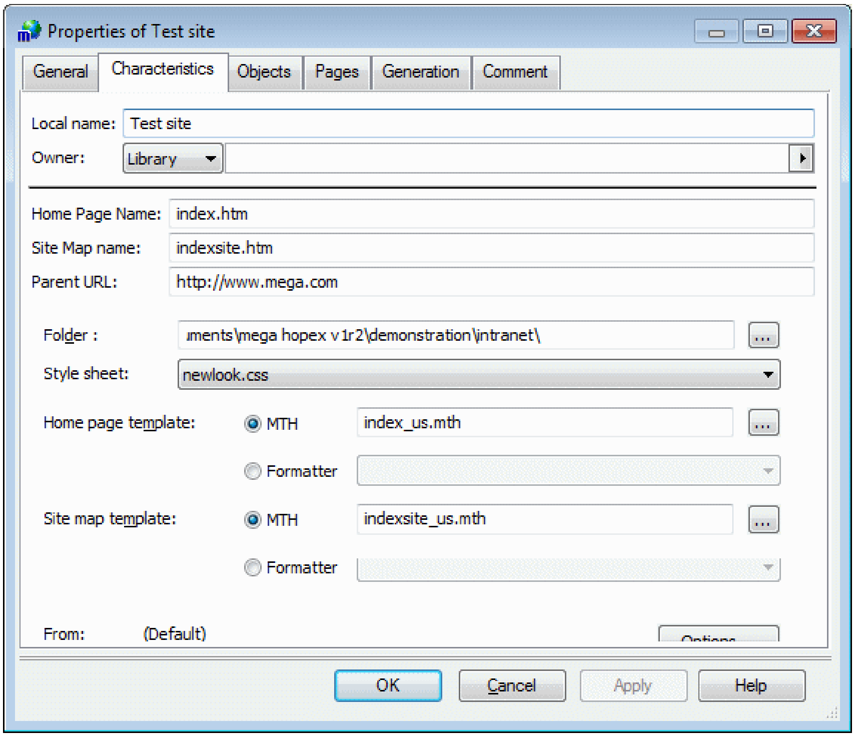
5. In the **Macro** field, click the right-oriented arrow and select **Create macro**.
6. In the **Macro** field, select **Edit**.
7. Enter the macro.


Macro MacroMyformatter - Script Editor
File Edit View Execute Help

```
Sub Generate(oObject, oContext, sUserData, sResult)
sResult = "<html><body>This is the HTML result of MyFormatter</body></html>"
End Sub
```

The formatter is now available in formatting fields of Web site pages.

Modifying Web Site/Web Site Template General Characteristics



Local name: Test site

Owner: Library

Home Page Name: index.htm

Site Map name: indexsite.htm

Parent URL: http://www.mega.com

Folder : ıments\mega hopex v1r2\demonstration\intranet\

Style sheet: newlook.css

Home page template: ● MTH index_us.mth ○ Formatter

Site map template: ● MTH indexsite_us.mth ○ Formatter

From: (Default)

In the Web site properties, **Characteristics** tab, you can enter or modify the default values for:

- The Web site **Folder** (temporary folder used for generation only).
- Names of **Home Page** and **Site Map**.

  ☛ *You can modify the default names of the Home Page and Site Map. You can also leave a box empty. In this case, the corresponding page will not be generated in the generation folder.*
- The **Parent URL**, if the site is generated for insertion into an Intranet. By default, the parent URL (which corresponds to the Enterprise Site link in the Web site Home Page) is “www.mega.com”.
- The **Style Sheet** used ("NewLook.css" by default).

  📖 *A style sheet groups the rules of presentation that can be applied to elements in a Web page. These rules are expressed in Cascading Style*

Sheet (CSS) language. For further information, consult the W3C (World Wide Web Consortium) recommendations (http://www.w3.org).

You can choose a *style sheet* different from that proposed by default if you do not find this suitable.

You can customize the style sheets supplied in the "Mega_Std" folder. To do this, you must copy them into the “Mega_Std” folder of the custom module tree-structure, where you can modify and rename them. You can also create new style sheets and save them in this same folder.

- **Home Page** and **Site Map** templates.
  The home page and site map can be generated:
  - From files with extension **.MTH**, located in the "MEGA_Std" folder in which **HOPEX** was installed: "HomeLeft.mth" for the Home Page, "SiteMap.mth" for the Site Map with navigator at the left.

    ☛ *If you want to customize these files, copy them into the “Mega_Std” folder of the custom module tree-structure and refer to Modifying Home Page and Site Map Files.*
  - From **formatter**s of Descriptor or Script type.

    ☛ *See also Configuring Page Setup for description of the **Options** button.*

    ☛ *The **Options** button is active only with the **HOPEX Power Studio** technical module.*

    ☛ *In a Web site template properties, **Folder** and **Parent URL** fields are not available. The other fields are available if the Web site template is not write-protected.*

Configuring Page Setup

To enter the general options of Web site generation:

1. In the Web site properties, **Characteristics** tab, click **Options**.

   ☛ *The **Options** button is active only with the **HOPEX Power Studio** technical module.*

   The **Options** window opens.
2. In the **Options** window you can enter the characteristics of objects that you want to appear, and configure certain page setup options.

Characteristics

You can:

- Edit **Characteristics not entered** of objects. This option can be useful if you want to display all the characteristics of objects (entered or not).
- Configure the display of object characteristics; show the **General** characteristics of objects (creation date, modification date, name of creator, etc.) and their **Extended** characteristics (concerning technical administration).
- Choose whether or not to display **External references**.
  If you decide to include external references at the time of generation and these references have predefined addresses, you can choose to treat them as being relative to the Web site. For more details, see Integrating External References.

Section titles

- Modify the **Hyperling image** if the standard one is not suitable (a small yellow arrow appears to the right of a title if there is a link to another page).
- Modify the **Return to top image** of the page if the standard one is not suitable in the case of generation without a frame.

Menu presentation

- Display **Menu** or not.
  You can also customize menu display on each Web site page. The option defined at page level takes precedence over that defined at Web site level. See Configuring Web Site Pages.
- Indicate the **Presentation of menu** (with or without a frame) as well as its **Position of menu** (on the left or at the top of the screen).

  ☛ *The position of the navigation menu in the Home Page and Site Map is defined by the ".mth" file. For more detailed information, see Modifying Home Page and Site Map Files.*
- Indicate the **Menu width** located on the left, as a percentage of the total width of the window.

Web site hosting

- Select and specify the name of the **Frame** in which the generated Web site will appear when it forms part of a broader Web site.

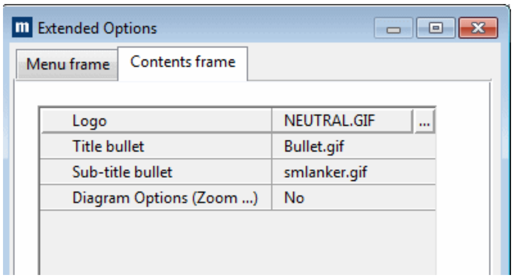
  ☛ *The name of the frame corresponds to the reference given to the target frame in the FRAME tag. Example: <FRAME NAME="Name">*

Modifying Logos, Bullets and Fonts

To modify the logo and bullets of your Web site pages:

1. In the Web site properties, **Characteristics** tab, click **Options**.
2. In the **Options** window, click **Extended options**.
3. In the **Extended options** window:
   - The **Menu Frame** tab allows you to customize the appearance of the navigation menu located at the left of the navigation window: logo, bullets of titles and sub-titles.
   - The **Contents Frame** tab allows you to customize the appearance of the elements that appear in the main part of the page: logo, bullets of titles and sub-titles.

   You can also specify if you wish to zoom in your diagram drawings (**Zoom Image (Internet Explorer)**). In choosing this function, you can select a scale in % in a menu located at the bottom of Web site diagrams or in the diagram pop-up menu (for SVG format). This functionality is only available if the diagram is in a frame isolated from the diagram description.



Logo	NEUTRAL.GIF
Title bullet	Bullet.gif
Sub-title bullet	smlanker.gif
Diagram Options (Zoom ...)	No

*Web site properties, **Characteristics** tab, **Options** button, **Extended options** button*

Images of logo and bullets are located in the "Mega_Std" folder of the installation site. You can modify and copy them into the “Mega_Std” folder of the custom module tree-structure.

If you do not want your logo to appear in generated pages, select the "Neutral.gif" file in the **Logo** text box.

Details concerning background images, colors, fonts and links

- Background images, color of links and the font should be configured in a style sheet. For more details, see Modifying Style Sheet .css.
- If you want to retain fonts used in the RTF comments of objects, you should select the **RTF font enabled** check box in the **Generation** tab of the Web site properties.

Modifying Image Format

You can generate Web site diagram images in the following formats:

- PNG
- Bitmap
- SVG
- JPEG

To modify the default format of generated images:

1. In the workspace, select **Tools > Options**.
2. Expand the **Documentation** folder and select **Web Sites**.
3. In the right pane, in **Generated images format** field, select the required format.
4. Click **OK**.

SVG format

Images generated in SVG (Scalable Vector Graphics) format are of higher quality than those generated in other formats, particularly when using zoom (distortion).

To view images generated in this format, you must install a specific viewer. The **SVG Editor** field indicates the SVG viewer URL address. It will be automatically downloaded when viewing the image. If you prefer to use another viewer, change the URL address.

Using SVG format in generated pages

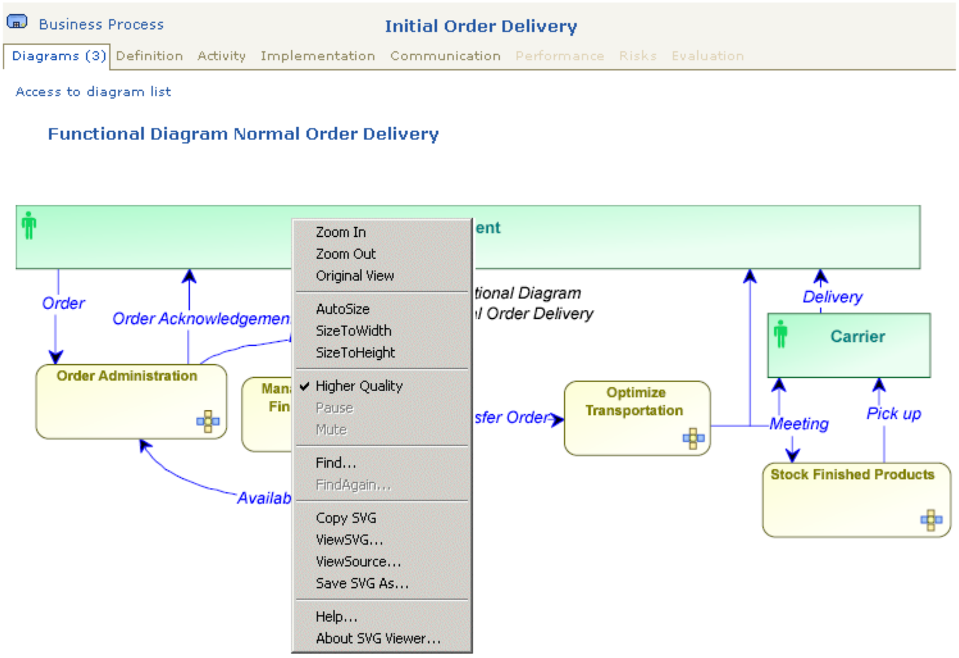
You can zoom in a diagram by activating one of the commands in the pop-up menu.

To zoom in:

- Click **Zoom in**.

To zoom out:

- Click **Zoom out**.



The zoom is carried out from the drawing center point.

To zoom on part of a drawing:

1. Position the cursor on the part of the drawing concerned and press the <Ctrl> key.

   The pointer changes shape:

2. Right-click the mouse button and select the zoom type.

Configuring Web Site Pages

To configure the appearance and content of the pages corresponding to each object type:

1. In the Web site properties, select the **Pages** tab.
2. Select the **Properties** of one of the pages that you have created:

   ```
   Example: You may decide to configure the pages that describe
   processes differently from the pages that describe org-
   units.
   ```

Web Site Page, Characteristics tab

In the **Characteristics** tab of a page properties, you can view and modify:

- the page name (by default this is the object type)
- the query applied
- the page format
  - <Head>
  - <Body>

☛ *For more details on formatting see Text formatting.*

Web Site Page, Menu & Frame Tab

You can define menu and frame display options at Web site level. In this case, the options will apply to all pages generated. See Menu presentation.

You can also modify these display options at page level in the page properties, **Menu & Frame** tab.

MANAGING WEB SITES: ADVANCED FUNCTIONS

Covered here are technical concepts that are useful only if you want to understand the HTML code generated, or modify the appearance of your Web site by using a style sheet. Also explained is how to manage the addresses of the generated elements, so you can create links between sites

The following points are covered here:

HTML Concepts

The purpose of this presentation of HTML language is to describe the basic concepts of HTML (HyperText Markup Language). This is obviously not an exhaustive description of HTML, but it may be useful in better understanding of the code defining the generated site.

For a more detailed description of HTML, there are many references offering every level of detail, and there are many sites accessible online. The site http://www.W3.org/, of the World Wide Web Consortium, provides access to HTML specifications and offers numerous links to sites that discuss HTML and related subjects.

Presentation

Use

The initial purpose of HTML was to distribute information on the Internet, but the technique is now widely used within companies in their Intranets.

HTML is used to describe documents that may contain text, images, and links to other pages called hyperlinks. The information contained in the document is used by a browser, such as Firefox or Microsoft Internet Explorer, to display the document contents in a possibly complex format.

Hyperlinks provide access to other documents. When the user clicks on a hyperlink, the document it points to is displayed in a new window, or in a frame if the window is divided into frames.

Tags

The basic principle of HTML is the use of tags, ranging from simple to complex. An example of a simple tag is the use of <B> to display text in bold:

```
<B>This text is displayed in bold</B>
```

which gives the following result:

This text is displayed in bold

With standard 4.0 compatible navigator versions, the use of formatting tags such as <font> is not recommended. This standard provides for use of styles, either internal to the page or applied in the header (see below). However, these tags are still recognized and enable rapid application of a format when required.

The current recommendation is to replace these tags with <SPAN style :bold> tags combined with </SPAN> tags. However, the evolution of recommended standards, the fact that HTML codes vary as a function of the navigator and the compatibility with previous versions that is required mean that Web site creators have a wide choice for their HTML processing.

Tags are placed between < and >, most often in pairs, as was seen in the above example. The end tag is identical to the start tag, but is preceded by a slash. For certain tags, only the start tag is required, but an unnecessary end tag generally does not cause problems.

Tags can be nested:

```
<B>This text is in bold, <I>now in bold italic,</B> now not
in bold but still italic, </I> now normal.
```

which gives the following result:

This text is in bold, *now in bold italic,* *now not in bold but still italic,* now normal.

Structure of HTML Documents

The basic structure of HTML documents is as follows:

```
<HTML>
<HEAD>
<TITLE>Simple HTML Document</TITLE>
</HEAD>
<BODY>
<P>This is a very simple document.
</BODY>
</HTML>
```

- The tag <HTML>, which begins all HTML documents, indicates to the browser that there are HTML tags to be interpreted. The end tag </HTML> indicates the end of the document.
- The tag <HEAD> defines the header of the document, which contains:
  - Between the <META> tags, keywords used by search engines or for other purposes.
  - The title (<TITLE> tags), which will be displayed as the title of the browser window.
  - The style sheet (<SHEET> tags), a quick way to define a format for displaying various elements in the page.

    ☛ *For more detailed information, see Style Sheets.*
- Next, the <BODY> tag indicates the start of the body of the document, composed of text, images, and multimedia objects.
- The <P> tag, which indicates a new paragraph, is frequently used in the body of a document.

Hyperlinks

Hyperlinks to other locations are indicated by the <A HREF> tags, which contain the address of the text, image or other referenced object. When the user clicks on the hyperlink, its content is then displayed, either by replacing the contents of the browser window or inside a frame in the main window. The referenced element can also be another part of the same document. Hyperlinks are generally displayed in a different color than ordinary text. In addition, when the pointer is on a hyperlink, its shape changes.

A hyperlink is defined as follows:

```
Click <A HREF="//www.mega.com/">here</A> to visit our site.
```

In this example, “here” is underlined to show that it is a link, but the way links are actually displayed depends on browser configuration.

The address appears inside quotation marks. These can be omitted if the address contains no spaces.

> ☺ *Special characters such as quotation marks are coded so that the browser does not interpret them as separators. For example, “&quot;” replaces quotation marks, “&lt;” replaces “<”, “&gt;” replaces “>” (the semicolon is part of the code). Letters with accents are theoretically coded, but most newer browsers recognize them as is.*

It is possible to create several addresses in the same document, to allow jumping to a specific point in a document.

> ☛ *When the location indicated for the jump already appears on the screen, it might be wrongly assumed that the jump has not operated.*

The tag for defining an address is:

```
<A NAME="Complinfor">
```

A jump to this address would be indicated as follows:

```
For more information see <A
HREF="Info.htm#Complinfor">Other Information</A>
```

It is also possible to indicate a relative path (../xxx.htm). (This is a normal slash and not the usual backslash.)

The name of the page can be omitted if the link refers to another point in the same document.

```
For example: HREF="#Complinfor".
```

Other HTML Tags

There are several tags for defining the color of the element concerned. This color can be indicated by using colors predefined in HTML (black, purple) or by their Red/Green/Blue hexadecimal value (#FF0000=red, #00FF00=blue, #0000FF=green, #000000=black; the # can be omitted with certain browsers).

Other tags are used to define text formatting, create lists, tables, etc., as well as to define a document structure using different levels of sub-headers (<H1>, <H2>, …).

Using Frames

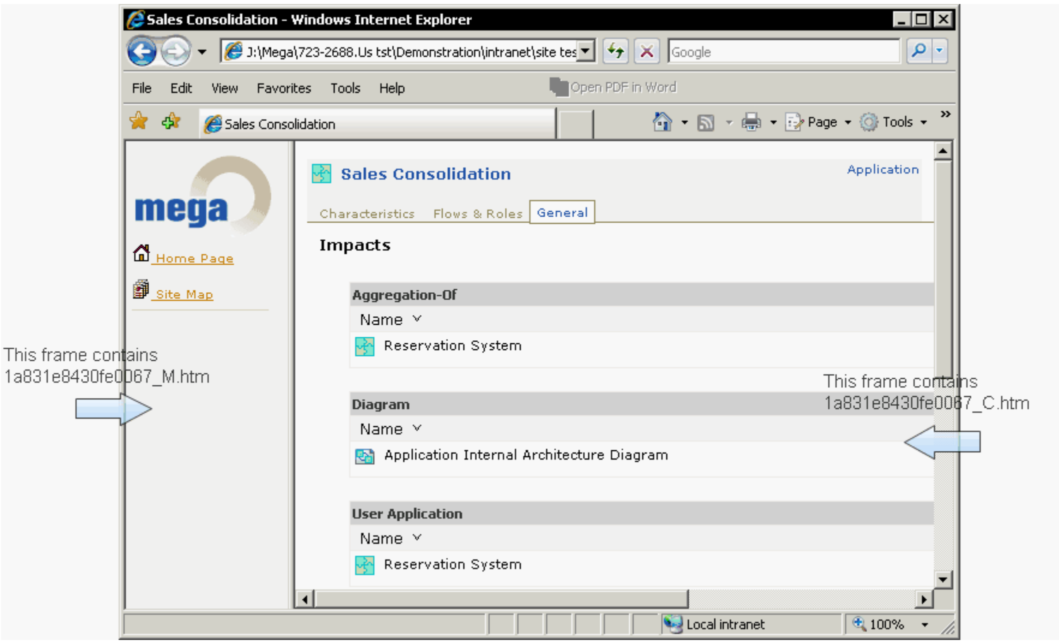
Frames are used to partition the browser window, in order to display several documents simultaneously.

In addition, when a link in the frame is selected, the frame can modify either its content (in the same manner as when the window is not partitioned), or the content of another frame.

Frames are defined in a main HTML document. This document often acts as a container for calls to other documents, indicating where they are to appear. In the example below, the page is divided into two frames, and it calls the documents that are to appear in each frame.

```
<FRAMESET COLS="20%, *">
  <FRAME SRC="1a831e8430fe0067_m.htm" NAME="menuframe">
  </FRAME>
  <FRAME SRC="1a831e8430fe0067_c.htm" NAME="contentframe">
  </FRAME>
</FRAMESET>
```

The pages called can define new frames, which are nested inside the calling frame.



Impacts

Aggregation-Of
Name
Reservation System

Diagram
Name
Application Internal Architecture Diagram

User Application
Name
Reservation System

This frame contains 1a831e8430fe0067_M.htm

This frame contains 1a831e8430fe0067_C.htm

STYLE SHEETS

See:

- Style Sheet Presentation
- Defining Styles
- Styles Available in "Neutral.css"
- Modifying Style Sheet .css

Style Sheet Presentation

Style sheets are a way to quickly modify the appearance of pages without having to change each page individually.

A style sheet defines the appearance of each element in Web pages, based on HTML tags and on the styles specified as parameters in the body of the page.

Options for displaying an element can be defined in several ways:

- By explicitly indicating the format (for example <B>text in bold</B>).
- By defining the style in the HTML tag (for example <TABLE BORDER=1 style="FONT: 10pt Arial">).
- By linking the HTML document to an external style sheet. This allows modification of the appearance of all Web pages that use the style sheet by acting on the style sheet itself.

This last method is used when generating pages; the Web site style sheet (defined in its properties) is listed in the header (<HEAD>) for each page:

```
<LINK REL=stylesheet href="StylSh.Css">
```

Styles for various elements are then specified. For example, the table is defined with the style "StdConstLayTabProp":

```
<TABLE CLASS="StdConstLayTabProp">
```

Note that the keyword indicating the style to be used is CLASS, and the STYLE keyword is used to define local tag formatting.

The parameters indicated in the HTML page are added to those indicated in the style sheet and replace those already defined.

The hierarchy of definitions indicated in style sheets varies as a function of the navigator used. They are not recognized by certain navigators.

Redefinition of a style in a page takes priority over the style sheet definition.

Defining Styles

Styles can be defined in a page header, to avoid repeatedly specifying a particular formatting each time it is used.

<HEAD>

<STYLE>

H1{font: 22pt Arial, Helvetica, sans-serif}

</STYLE>

In the above example, the H1 level header will appear in 22 point Arial font if this font is installed on the computer of the person accessing the page, otherwise in Helvetica, and if neither of these fonts is available, in a sans serif font.

☺ *This possibility can be used in all tags that define which font to use for display.*

In the example below, when the H1 tag indicates that style is “Style1Plus” (<H1 class=Style1Plus>), the color red is added, replacing the color used by default.

```
<STYLE>

H1{font: 22pt Arial, Helvetica, sans-serif}

H1.Style1Plus {color: red}

</STYLE>
```

It is possible to define a style that applies to all tags, by declaring it as follows:

```
.StyleName{formatting}
```

This style can be used in form H1.StyleName, TR.StyleName, etc.

When pages are displayed, priority is given to the format nearest to that of the element concerned. If a style is first defined in a style sheet, then redefined in a header, then defined a third time in the body of the text, it is the final definition that will be taken into account.

☛ *The different definitions will supplement each other if not contradictory: a style defining a color in the style sheet will remain valid if the redefinition concerns, for example, only the font size.*

Style sheets are text files, as are HTML pages. They generally have the extension CSS, for Cascading Style Sheet. They are indicated in the META section as follows:

```
<LINK REL=STYLESHEET TYPE="text/css" HREF="../STANDARD/
NEUTRAL.CSS">
```

Defined elements	Font	Size	Color
Style sheet	Arial	12	Blue
Header (HEAD)		18	
Result	Arial	18	Blue

Example of cascading style operation

Defined elements	Font	Size	Color
Style sheet	Arial		
Header (HEAD)		18	
Tag		size:20	color=red
Result	Arial	20	red

A further example of cascading style operation

Just as there are HTML editors, there is software for modifying style sheets without having to know the formatting codes. These codes are documented in numerous Web sites (including http://www.W3.org/, site for the World Wide Web Consortium).

In pages generated with the HTML generator, default styles are indicated with indication of which object is concerned.

Styles Available in "Neutral.css"

The style sheet "Neutral.Css", found in MEGA_STD, contains the definitions for these styles ***without formatting***. These styles are: (styles prefixed by "Model" are used to format diagrams, those prefixed by "Std" are used for standard formatting)

.ModelConstLayLinePropLine	Properties of components of the diagram, displayed as lines
.ModelConstLayLinePropTable	Properties of components of the diagram, displayed as table
.ModelConstLayLineText	Comments of components of the diagram, displayed as lines

.ModelConstLayTabNameComment	Components of the diagram, displayed as table with their comments
.ModelConstLayTabProp	Components of the diagram, displayed as table with their properties
.ModelDescribedObj	Object described by the diagram
.ModelObjPropByLine	Properties of objects in the diagram, displayed as lines
.ModelObjPropByTable	Properties of objects in the diagram, displayed as table
.ModelObjTableNameComment	Name and comment of objects in the diagram
.ModelObjText	Comment of objects in the diagram
.ModelParentLayLinePropLine	Properties of parents of the diagram, displayed as lines
.ModelParentLayLinePropTable	Properties of parents of the diagram, displayed as table
.ModelParentLayLineText	Comments of parents of the diagram, displayed as table
.ModelParentLayTabNameComment	Parents of the diagram, displayed as table with their comments
.ModelParentLayTabProp	Parents of the diagram, displayed as table with their properties
.ModelProperties	Properties of the diagram
.StdConstLayLinePropLine	Properties of components, displayed as lines
.StdConstLayLinePropTable	Properties of components, displayed as table
.StdConstLayLineText	Comments of components, displayed as lines
.StdConstLayTabNameComment	Components displayed as table with their comments
.StdConstLayTabProp	Components displayed as table with their properties
.StdParentLayLinePropLine	Properties of parents, displayed as lines
.StdParentLayLinePropTable	Properties of parents, displayed as table
.StdParentLayLineText	Comments of parents
.StdParentLayTabNameComment	Parents displayed as table with their comments
.StdParentLayTabProp	Parents displayed as table with their properties
.StdPropLine	Property displayed as lines
.StdPropTable	Property displayed as table
.StdText	Comment

Modifying Style Sheet .css

Several parameters can be defined in style sheet .css:

- Background color
- Links color
- Visited links color
- Text color
- Default font

Modifying contents page parameters

To modify:	Style and parameter to be modified:
Background image	Style 'BODY.Content' or 'BODY', parameter 'BACKGROUND-IMAGE'
Background color	Style 'BODY.Content' or 'BODY', parameter 'BACKGROUND-COLOR'
Links color	Style 'A', parameter 'COLOR'
Visited links color	Style 'A:visited', parameter 'COLOR'
Text color	Style 'BODY.Content' or 'BODY', parameter 'COLOR'
Default font	Style 'BODY.Content' or 'BODY', parameter 'FONT-FAMILY

☛ *'BODY' style applies to all HTML pages, except if another style takes precedence.*

'BODY.Content' style applies to the contents page only.

Modifying menu parameters

To modify:	Parameter to be modified:
Background color	Style 'BODY.Menu' parameter 'BACKGROUND-COLOR'
Links color	Style 'Menu A' parameter 'COLOR'
Visited links color	Style 'Menu A:visited' parameter 'COLOR'
Text color	Style 'BODY.Menu' parameter 'COLOR'
Default font	Style 'BODY.Menu' parameter 'FONT-FAMILY'

Example of .css configuration

Example: image (current repository) without background color

```
BODY.Menu
{
MARGIN-TOP: 10px;
FONT-SIZE: 10px;
BACKGROUND-IMAGE:url(..\standard\HOMEPICTURE.JPG);
MARGIN-LEFT: 5px;
}
```

Example: background color without image

```
BODY.Menu
{
MARGIN-TOP: 10px;
FONT-SIZE: 10px;
BACKGROUND-IMAGE: none;
MARGIN-LEFT: 5px;
BACKGROUND-COLOR: #f5f4ee
}
```

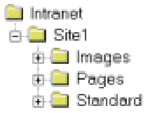
MANAGING WEB SITE FILES

See:

- Generated File and Folder Structure
- Configuring User Options
- Checking File Names
- Integrating External References

Generated File and Folder Structure

The Web site is generated in a folder indicated in the properties of each Web site. For each site, the following structure is generated:



The described structure is generated by the standard sites provided. The generation can be customized if a different structure is required.

- The environment folder includes image files such as bullets and the company logo, which are independent of the repository content, as well as the Web site style sheet (xxx.CSS).
- The "Images" folder contains the diagrams generated from the repository contents.
- The "Pages" folder contains the HTML files that describe the site repository objects, as well as the files required for site navigation.

☛ *For each object, there can be several HTML files; for example, the one that defines the two frames in the main window (menu on the left, and object description on the right), and the two files that define the content of these two frames. Depending on the type of object, there may be additional files; for example, an object described by a diagram will have additional HTML files used to display its drawing, etc.*

To move a site, simply select the root folder for the site structure and copy it to another location. The generated HTML addresses are relative and remain valid wherever the site is placed. Example: BACKGROUND="../STANDARD/BKGROUND.GIF".

Configuring User Options

To configure user options:

1. From **HOPEX** menu bar, select **Tools > Options**.

2. In the **Options of user** window, double-click the "Web Sites" icon.
   You can:
   - specify the **Generated images format** (Bitmap, JPEG, SVG, PNG).
   - modify the **Default parent URL** (http://www.mega.com).
   - specify the templates used for the Home Page and Site Map.

   The Home Page and Site Map are generated from files found in the MEGA_STD folder of the folder in which **HOPEX** was installed. These files have the extension ".mth": "HomeLeft_us.mth" for the Home Page, "SiteMap_us.mth" for the Site Map with navigator at the left. If you want to customize these files, copy them into the "Mega_Std" folder of the custom module tree-structure, and refer to Modifying Home Page and Site Map Files.
   - modify the folder that contains the standard files (for example, the images corresponding to the various object types) and the images and pages generated.
   - process external references as being relative to the Web site.
   - **HOPEX Power Studio - Publisher** offers you the possibility of importing external references into the **HOPEX** site. For more detailed information, see Integrating External References.
   - specify the case for names of generated files (first letter upper or lower-case).
   - delete pages of an earlier generation before any new generation.

Modification of these parameters is valid for all Web sites generated.

Checking File Names

The file "Sitedef.ini" found in the root folder in which the site is generated (**Intranet** is the default) indicates for each object the correspondence between its absolute identifier and the name of the file describing its main page.

For each object in the site, a paragraph like this is created:

```
[245D261330A3002D]
PageName = 245D261330A3002D
Available = Yes
Object = Sales Department Org-Unit
MetaClass = Org-Unit
```

Object identifier

The identifier for the object is given inside brackets. This identifier never changes once the object is created, in order to retain all links and properties even if the object is renamed.

File naming rules

The PageName value indicates the name of the main page. When the site for the object with this absolute identifier is regenerated, this string of characters is used to name the HTML files, followed by a letter and then the HTM extension. PageName is used even if the object name has been changed. This ensures that when a site points to another site, links remain valid (the indicated address remains constant).

Main page of the object	HexaIdAbs of the occurrence
Menu page of the object	HexaIdAbs of the occurrence followed by "_M"
Contents page of the object	HexaIdAbs of the occurrence followed by "_C"
Diagram page associated with the object (displayed in object page)	HexaIdAbs of the occurrence followed by HexaIdAbs of the diagram followed by "_D"
other diagram page (independent of the object page)	HexaIdAbs of the occurrence followed by HexaIdAbs of the diagram

☛ *The generated HTML pages are built of several frames that are called by the main page. There is an HTM file specifying the contents of each frame. For example, for the object "Sales Department Org-Unit", the files generated could be "*245D261330A3002D*.htm", which is the name of the main page describing the object and is fixed, supplemented for example by "*245D261330A3002D*_M.htm", "*245D261330A3002D*_D", etc.*

Respecting specific naming rules

When specific rules must be followed for naming pages, simply perform an initial generation, modify the file names to comply with the rules, and regenerate the site so that the generated file names are in compliance with the rules in effect. For example, simply modifying the above paragraph to look like this means the HTML files will have the prefix A1_DR instead of DIREC2:

```
[245D261330A3002D]
PageName = A1_DR
Available = Yes
Object = Sales Department Org-Unit
MetaClass = Org-Unit
```

The line "Available = Yes" indicates that the object is still present in the site. If the object was included during a previous generation but is no longer part of the site (such as an organizational process now deleted from the project), this line becomes "Available = No".

☛ *As the file "Sitedef.ini" is intended to ensure that file names are retained during successive page generation operations, though an object may no longer be referenced in the site it is still retained in this file. If the object is later reincluded, the page names will remain identical.*

The object type and name are then indicated in the "Object" line (in the example, "Object = Sales Department Org-Unit").

Index page naming rules

An index page comprises a menu page (_M) and a content page (_C).

Menu page

Naming rule	Index page name
Without naming rule (without SiteDef.ini)	ParentPageName_M.htm
With specific naming rule (with Sitedef.ini)	IndexPageIdentifier_M.htm

Content page

Naming rule	Index page name
Without naming rule (without SiteDef.ini)	ParentPageName_C.htm
With specific naming rule (with Sitedef.ini)	IndexPageIdentifier_C.htm

☛ *When several index pages are specified for the same object type (alphabetical index, index by characteristic, etc.), the index pages are suffixed by an order number (for example, xxxxx_C_PIDX1.htm, xxxx_C_PIDX2.htm).*

Image naming rules

Images are named as follows:

- HexaIdAbs of described object _HexaIdAbs of diagram_I.png (or .bmp, .svg, .jpg depending on image format)

Integrating External References

HOPEX Power Studio - Publisher offers you the possibility of importing external references into the **HOPEX** site.

☛ *To do this, you must create external references with predefined folder (see the **HOPEX Common Features** guide for more detailed information).*

External reference integration standard mode

To integrate external references in the **HOPEX** site:

1. Verify that the external references are specified using a predefined folder.
2. Indicate that the address of external references is relative to the Web site.
   From **HOPEX** menu bar, select **Tools > Options**, then double-click the **Web Sites** icon and select **External references relative to site**.
3. Generate the Web site.

4. Start the "XRefCopy.vbs" file in the "Intranet" folder to copy the external references in a sub-folder of the site.
   By default, this sub-folder is called REFEXT. You can modify it in the user configuration (in the workspace, select **Options > Tools** then double-click the **Web Sites** icon).
   The external references are now accessible from the Web site.

External reference management configuration

External reference path generation can be configured if the standard method described above is not suitable.

You can carry out this parameterization using a macro, connecting it to the Web site or Web site template by the "External reference manager" link.

To do this:

1. In the pop-up menu of the Web site, select **Explore**.
2. In the empty "External reference manager" folder, click **Connect**.
   The Query window appears.
3. Select the required macro.

Macro content

If no macro is connected to the Web site, the "ExternalReferenceManager" macro provided by default is taken. This macro carries out processing described in External reference integration standard mode.

You can create your own macros. The macro must contain the following methods:

Start of generation

This method indicates start of generation.

```
Sub StartExternalReference(oWebGenCtx)
End  Sub
```

Method called for external references

```
Sub ExternalReferencePathGet(oExternalReference,
oWebGenCtx,enFilter , strResultPath)
```

This method is called for each external reference. It retrieves the path of the external reference to be displayed in strResultPath and manages copy.

oExternalReference	External reference object
oWebGenCtx	Generation context

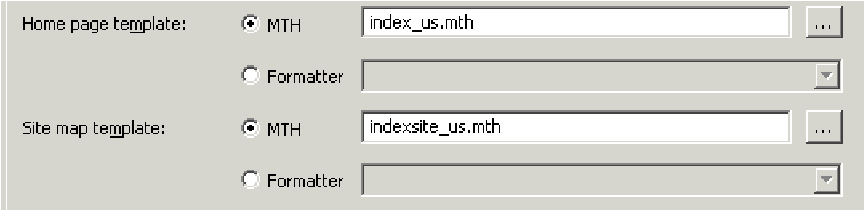
enFilter	enFilter = 1, relative path
	enFilter = 0, absolute path

End of generation

```
Sub StopExternalReference()
End  Sub
```

Modifying Home Page and Site Map Files

You can define the format of home page and site map using a formatter or an .mth file.



Home page template: ● MTH index_us.mth ...
○ Formatter
Site map template: ● MTH indexsite_us.mth ...
○ Formatter

Web site properties dialog box

Formatter

You can define the home page and site map using a descriptor or a VB script.

The descriptor must be category HTML Web Site Specific Page.

The script formatter must be of _type Homepage or SiteMap, depending on whether applied to home page or site map.

See About Web Site Configuration.

.mth file

It is possible to customize files (templates) used for generation of the Home Page and Site Map. These files are found in the MEGA_STD folder.

They contain the basic presentation structure of Home Page and Site Map. It is possible to indicate the name of the files used in the options of the environment and of each user.

To modify these files, it is necessary to copy them into the "Mega_Std" folder of the custom module tree-structure, as for all customizing of elements proposed as standard. In the same way, the new files of this type must be placed in this same folder.

To modify the setup proposed, it is recommended that the existing files be copied and their contents modified either using an HTML editor or in HTML code.

Parameters used in generation files

These parameters correspond either to values entered at configuration, or, particularly for page addresses, to values obtained on generation.

The addresses indicated are relative addresses (..//) and not absolute addresses (X:Path//filename).

COMPLETESITEMAP	Name of site map pointing to site map address
COMPLETEHOMEPAGE	Name of home page pointing to home page address
CONTENTBULLET1	Contents frame title bullet
CONTENTBULLET2	Contents frame sub-title bullet
CONTENTLOGO	Contents frame logo
DESCRIBEDOBJECTLIST	Displays described objects list: objects found in objects tab
HIGHTURL	Site parent URL
HOMEPAGE	Home page address
INDEXPAGELIST	Index pages list
INDEXPAGELISTTITLE	Displays index pages (+ links to pages) and the names of associated MetaClasses
INDEXTITLELIST	Displays index pages (+ links to pages)
MENUBULLET1	Menu frame title bullet
MENUBULLET2	Menu frame sub-title bullet
MENULOGO	Menu frame logo
OBJECTPAGELIST	Displays list of all objects producing a page
SITECOMMENT	Text of site comment
SITEMAP	Address of page containing Site Map
SITENAME	Site name
STYLESHEET	Style sheet used in site pages
PAGES	Example syntax: [MEGA param=pages object=7fce816c457d008a] -> Recovers URL of object page 7fce816c457d008a
STANDARD	[MEGA param=standard file=MyFile.png] -> Recovers URL of MyFile.png found in standard directory
IMAGE	[MEGA param=image file=MyFile.png] - > Recovers URL of MyFile.png found in image directory

CHECKING WEB SITE GENERATION

See:

- Generation Report
- Checking Web Site Validity

Generation Report

If generation has been successfully completed, an .MGR file is available in the sub-folder corresponding to the user in "\SysDb\USER".

This report file contains:

- Information relating to generation
  - The generated Web site
  - Definition of the list of objects producing a page
- Information on generation of each page
  - Home page
  - Site map
  - Object pages in the form Page generated =, MetaClass =, Object =, Date =

If generation has not been successfully completed, the report is stored in a temporary file name with beginning CRD in the temporary file of the workstation. You will find here information concerning the object at fault.

Checking Web Site Validity

You can check validity of your Web site:

- Before starting its generation.
- After migration or import of data relating to the Web site.

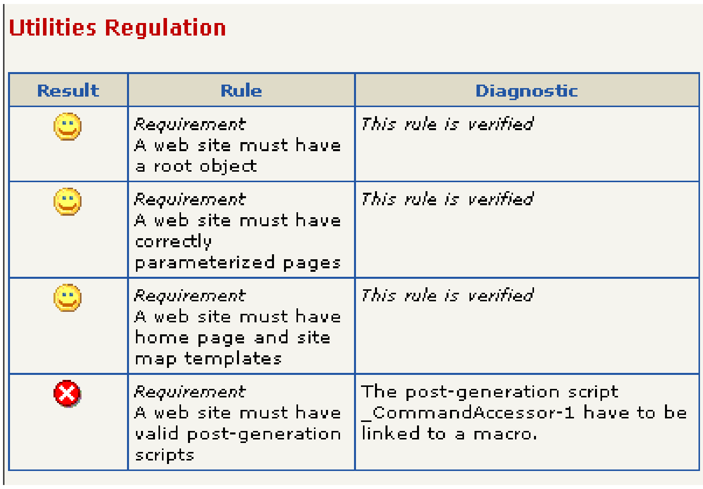
To do this:

- In the pop-up menu of the Web site, select **Check > Regulation with propagation**.

The following checks are carried out:

- Existence of .mth templates
- Page configuration
- Presence of at least one object in the **Objects** tab of the Web site.
- Post-generation script configuration

The check result is displayed:



Utilities Regulation

Result	Rule	Diagnostic
☺	*Requirement* A web site must have a root object	*This rule is verified*
☺	*Requirement* A web site must have correctly parameterized pages	*This rule is verified*
☺	*Requirement* A web site must have home page and site map templates	*This rule is verified*
✗	*Requirement* A web site must have valid post-generation scripts	The post-generation script _CommandAccessor-1 have to be linked to a macro.

USING HTML DESCRIPTORS

MEGA allows you to create HTML descriptors that can be used to replace the standard formats.

HTML DESCRIPTORS OVERVIEW

Accessing HTML descriptors

To access HTML descriptors:

1. In **MEGA**, select the **Utilities** window.

   ☛ *The **Utilities** window is accessed by selecting **View > Navigation Windows > Utilities**.*

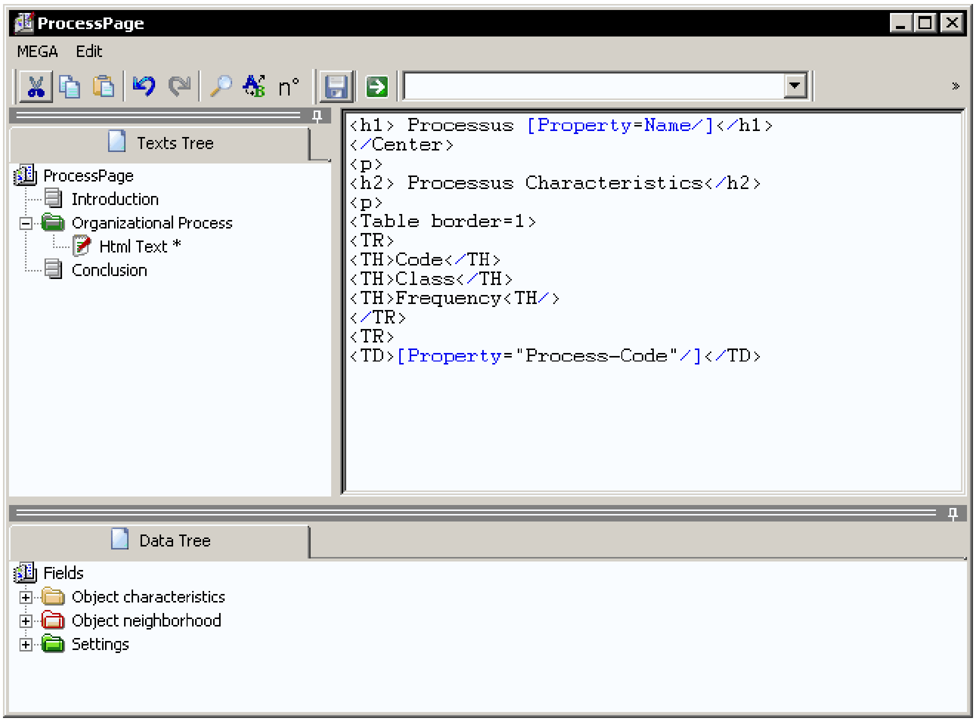
2. Expand the "Descriptor" folder, then the "HTML" sub-folder.
   You can access descriptors of object pages or specific pages (index, home page, site map).
3. To access descriptors of object pages, expand the "HTML Object Page" sub-folder.
4. Select HTML <Body> or HTML <Head> according to the type of descriptor you want to create (body or header).
   You can, for example, create your descriptor in the "HTML Personal" sub-folder.

   ☛ *The body constitutes the visual content of the HTML page. The head contains the page title, META information defining keywords that can be used by search engines, the style sheet, etc.*

HTML descriptor editor

In the case of HTML descriptors, the descriptor editor functions in much the same way as described in the sections covering RTF editing (see Customizing RTF Descriptors). There are, however, important differences:

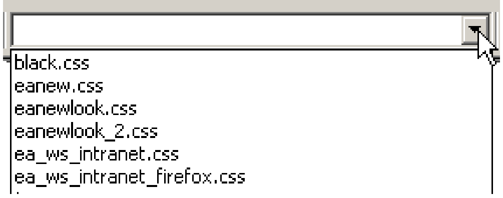
- Texts titled “HTML source” are written in a specific editor similar to WordPad.
- The codes corresponding to repository data differ from those used in RTF descriptors.



```
<h1> Processus [Property=Name/]</h1>
</Center>
<p>
<h2> Processus Characteristics</h2>
<p>
<Table border=1>
<TR>
<TH>Code</TH>
<TH>Class</TH>
<TH>Frequency<TH/>
</TR>
<TR>
<TD>[Property="Process-Code"/]</TD>
```

Example of an HTML descriptor

Formatting is carried out using HTML tags or "CSS" style sheets. By default, **MEGA** provides a set of style sheets. You can apply a style to the content of a descriptor by selecting the content in question and by selecting the style sheet in the field at the top of the Edit window.



As for RTF descriptors, elements from the repository are inserted in the text by selection from the **Fields** menu tree, then by drag-and-drop into the text.

It is possible to include standard HTML tags (for example <BR> to add a line break) and text in the content of the descriptor, as well as to complement elements using the appropriate syntax.

For description of syntax, see:

- "Syntax of HTML Sources
- Tags Specific to HTML Descriptors.

Icon specific to the HTML descriptor editor

Icon n° enables movement to a given line.

Taking account of inheritance in HTML descriptors

The "Inherited" parameter allows us to take into account or to ignore inherited objects.

Possible values for this parameter are:

- On
- Default: default behavior when the "Inherited' parameter is not specified in the "Component" tag
- Off

Example:

```
[Component="Select [IT Service] Where [Defining-
Application] = '37D340E0445B0250'"  Inherited="On" ][/
Component]
```

☛ *For more details on variations, see the* ***MEGA Common Features*** *guide, "Handling Repository Objects".*

HTML Descriptor Structure

Using several groups and texts

HTML descriptors can be structured in the same way as RTF descriptors. A menu tree of several levels is used to edit the components of the main object of the descriptor. The descriptor can be complemented by texts placed before and after the texts describing the objects. It is possible to condition editing of these texts by the existence and the number of objects edited.

Using a single text

The multiple level tree functions in numerous cases. It is, however, sometimes desirable to structure HTML descriptors more simply, since they are included in a set of standard pages and descriptors that make up the site.

To do this, use of a single text that includes references to the object components as well as the editing conditions facilitates maintenance of these descriptors and traceability of results.

Displaying the HTML Descriptor Execution Result

Executing the descriptor independently

The descriptor can be executed independently to check its validity and to view its result. This is, however, only a summary view since the hyperlinks are not visible and there is no associated style sheet.

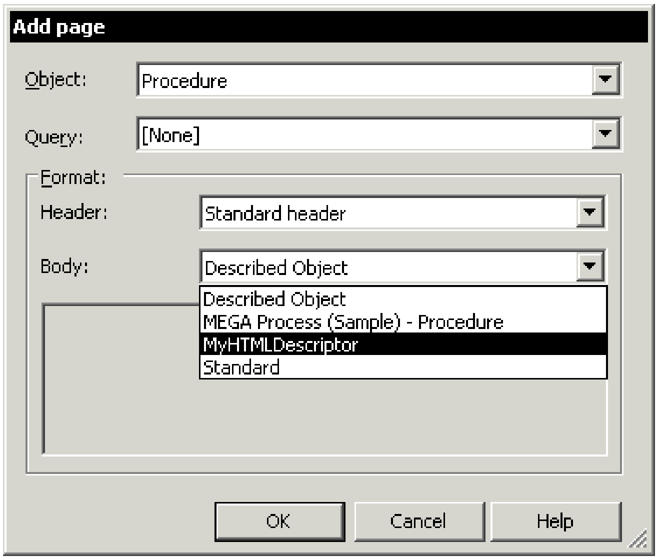
To execute the descriptor:

- In the descriptor editor, select **MEGA > Execute** or click →.

Integrating the HTML descriptor in a Web site

To see the result that will be obtained by users on site generation, it is necessary to include the descriptor in a test site; autonomous generation of a descriptor does not, for example, enable test of the hyperlinks generated, since only the page corresponding to the descriptor has been created.

You can access the HTML descriptor on an object page corresponding to the entry point defined in your descriptor. It will be used instead of a standard formatter.



Object: Procedure

Query: [None]

Format:

Header: Standard header

Body: Described Object

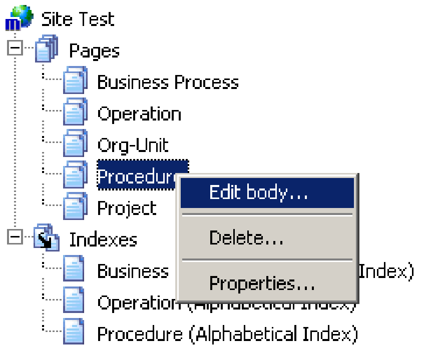
Described Object
MEGA Process (Sample) - Procedure
MyHTMLDescriptor
Standard

Editing HTML Descriptors from a Web Site

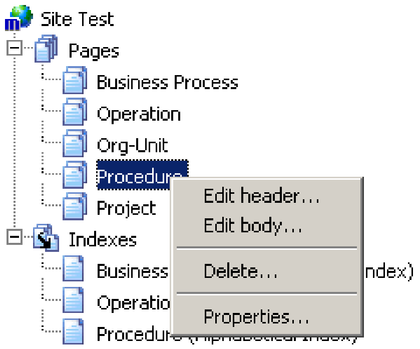
When an HTML descriptor is used in a Web site or a Web site template, from the “Pages” tab of the site or site template, its pop-up menu enables it to be consulted using the descriptor editor.

Editing HTML descriptor header and body

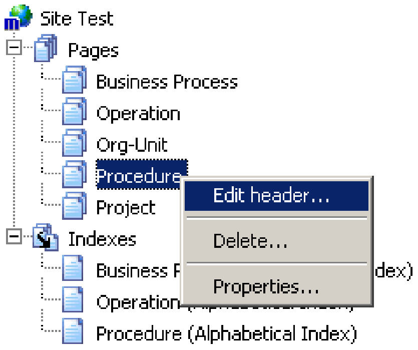
The commands **Edit header** and/or **Edit body** appear depending on whether the page is based on a <Body> and/or <Head> HTML descriptor.



Organizational process page with HTML descriptor <Body> and standard <Head> format.



Organizational process page with HTML descriptors <Body> and <Head>.



Organizational process page with HTML descriptor <Head> and standard <Body> format.

☛ *The general options accessible from the* ***Characteristics*** *tab of the site properties dialog box are not applicable to pages formatted using HTML descriptors.*

Defining a descriptor with complete page generation

You can define an HTML descriptor enabling generation of the complete page, that is including:

- the header, contained by tags <HEAD> and </HEAD>.
- the body, contained by tags <BODY> and </BODY>.

☛ *Opening and closing tags <HTML> </HTML> are not however generated.*

So that the HTML descriptor will take account of the complete page:

1. In the properties dialog box of an HTML descriptor, select the **HTML** tab.
2. In the **HTML page generation format** drop-down list box, select "All the page".

☛ *By default, only the body of the page is generated (value "Only the body").*

SYNTAX OF HTML SOURCES

Elements from the repository in the form of "HTML source" texts are inserted in the form of tags placed between square brackets ("[ ]").

Two forms are possible:

- short form
- extended form

Short form

A short form is obtained by “drag-and-drop” in the descriptor editor, of type [P=”Property”/]

The tag is opened and closed in a single operation.

> ☛ *It is not always possible to perform drag-and-drop. Certain tags (CL) must be entered manually by modifying an existing tag for example.*

Extended form

An extended form, of type [P="Property"]text [Value/][/P]

Among other things, the extended form enables conditioning of the text entered between the beginning and end tags.

Remarks on syntax of HTML sources

As in XML, the character “ / ” is placed at the end in the short form. In extended form where the tags are matched, this character is placed at the start of the final tag.

Double apostrophes ("Property") are optional when the character chain they enclose does not contain any blanks.

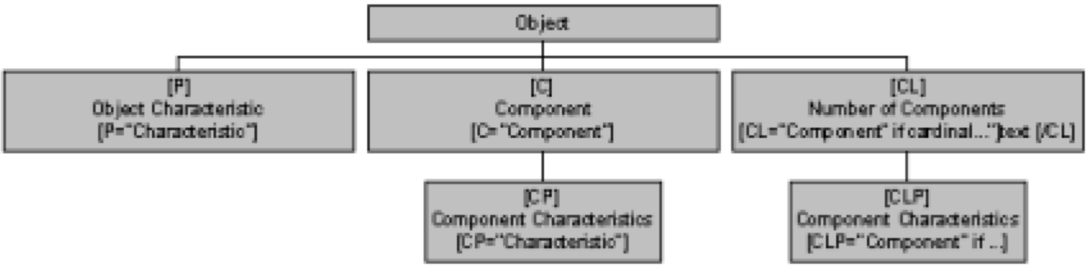
> ☛ *When closing a text, a message warns you if there is an error in the code.*

Tags Specific to HTML Descriptors

On execution of a descriptor, tags allow you to obtain elements from the repository. There are several types. They enable access to:

- Properties of the object.
- Components of the object (that is, of objects that can be reached by a link or query).
- Properties of components.
- The cardinal of components (the number of components).

For each tag, additional parameters enable selection of the type of information or the property to be displayed.



☛ *When you build your descriptor, you can use short form notation. When you save the descriptor, the tags are automatically converted to extended form.*

Property Tag (P)

The [Property] tag enables editing of object characteristics (in particular of information in the **Characteristics** tab of the object properties dialog box).

It can be inserted in a descriptor text in two forms:

- ***[P= "Property"/]***
  This form is obtained by drag-and-drop of a characteristic into the body of the text.
- ***[P="Property"]text [Value/][/P]***
  This form allows conditioning of appearance of a text as a function of the existence of a property.

It should be noted that the name of the object is considered as a characteristic ([P=”Name”/]).

[Value/], between tags P - /P, indicates the position where the value of the property will be inserted on creation of HTML pages.

Example: Applied to the "Enterprise" org-unit, [Property=Name/] returns value "Enterprise" in the generated HTML page.

The [Property] tag can be complemented by the following attributes:

☛ *These attributes are also valid for the Component Properties (CP) tag.*

☛ *To condition editing of properties, see Conditioning HTML Tag Editing.*

Layout (LY)

When several presentations are possible for a characteristic, this attribute enables specification of the presentation adopted. This attribute is used for comments (texts) and for characteristics which have tabulated values.

Comment

[P="Comment" LY="TITLE"/]	First paragraph of comment
[P="Comment" LY="COMPLETE"/]	Complete comment

☛ *The complete comment is used by default.*

[P="Comment" LY="Complete"/] equivalent to [P="Comment"/]

Values contained in table

In the example, Type-xx

[P="Type-xx" LY="External"/]	External value of value
[P="Type-xx" LY="Internal"/]	Internal value of value

☛ *The external value is used by default.*

Bookmark (BMK)

This attribute creates an address in the page. The objective is to be able to call this address via the Link="In" attribute, for example in a diagram containing sensitive zones.

[P="Name" BMK="1"/]	Address (<A NAME="xxx">) in the page.

Link (LNK)

This attribute enables creation of a link:

- either to the object page.
- or within the same page using the Bookmark attribute (see above).

[P="Name" LNK="Out"/]	Link to the object page if it exists
[P="Name" LNK="In"/]	Link to an address defined in the page with the BMK tag

The link is created only if a page or an address is defined for the object. If this is not the case, the tag is ignored. For this reason, when the descriptor is generated in isolation, the links to other pages do not appear in the result.

Menu (MNU)

This attribute indicates that the name of the object must be referenced in the menu frame, with a link to its address.

[P="Name" MNU="n"/]	Link to another page, n representing the link level in the menu part, from 1 to 9

Target

This attribute enables definition of the target window in which the referenced page will open.

Value	The target is displayed ...
_top	In the main frame
_parent	In the main frame
_self	In the current frame (the one in which you clicked)
_blank	In a new window

☛ *You can specify another value. For example, if you specify the value "MyFrame", the page will open in the window called "MyFrame".*

Examples containing the target attribute

```
[Component="Operation"]
[ComponentProperty=Name Target="_blank" Link=Out/]<br>
[/Component]
```

For each operation of the organizational process, if you click on a name, the page concerning the operation will open in a new window.

PictureFormat

The PictureFormat attribute enables recovery of the attribute value image address. The attribute can take values: GIF, BMP or ICO.

```
Example on a risk:
<img src="[Property="Absolute risk" PictureFormat="GIF"/]"/
> [Property="Absolute risk"/]
Displays attribute value image: absolute risk and absolute
risk value
```

If the image does not exist, for example on the name attribute: a transparent image address is returned: empty.gif

☛ *This attribute is also valid with the Component Properties [CP] tag.*

Component Tag (C)

The component tag is inserted in the form: [C= "Link"/] or [C="Link"]text[/C].

"Link" represents the link browsed (for example, "Component") to reach the components, that is the objects connected to the main object. It is also possible to use these tags for queries.

With [C="Link"]text[/C], the text is inserted for each object reached by the link. This enables, for example, the insertion of a specific image before each entry in a list (example [C="Component"]<IMG SRC="CMP.JPG">[/C]). On the other hand, it is recommended that either “Before” or conditioned texts (see later) be used to define a title preceding a list of objects.

This tag is most often inserted with the component properties tag [CP], which enables editing of component properties and in particular of its name.

[C="Link"/]	Properties of objects reached by link
[C="Query"/]	Properties of objects selected

Syntax used:

[C=link" or "query"][CP=Name Link=Out/][/C]

☛ *Use of the LINK attribute is optional if sensitive objects are not required.*

Enter the query in the tag

It is possible to enter the query code directly in the tag [C] in place of the query name. This can be practical if the query is used only in this location.

Example

```
[C="SELECT Operation WHERE [Type Operation] = ’Check’ and
Org-Unit=&name"][CP=Name/]<br>[/C]
```

This returns a list of operations of "Check" type carried out by the org-unit on which you are positioned.

💣 **In this case, the name of the variable used must of necessity be "name".**

☛ *When your query includes double quotation marks, these should be replaced by single quotation marks within a tag (‘Check’ and not "Check").*

A query is saved via its absolute identifier. In event of modification of the query name, descriptors do not need to be modified.

Sorting a list by attribute

The [Component] tag enables to display an object list:

- Presented in Alphabetical Order
- In the link sort if using a MetaAssociationEnd.

We may need to classify objects as a function of an attribute. To do this, the "Sort" attribute is used, specifying if sort order should be:

- ascending ("A")
- descending ("D").

```
[C="MetaAssociationEnd" Sorter="Attribute:A"]
```

Example: sorting project organizational processes by code

[C="Organizational process" Sorter="Organizational Process Code:A"] [CP=name link=out/]<br>[/C]

("A") used alone sorts by internal value by default.

☛ *The internal value is unchanging, while the external value is the value displayed translated in the different languages.*

To sort by external value, you must add "E".

Example: sorting diagrams depending on their external type

[Component= "Select Diagram" Sorter="type :A :E"]

[/Component]

Displaying objects that produce pages

When browsing a list of objects associated with the current object, the "WithPages" attribute enables return of only those objects that produce a page.

WithPages = "Yes"	The object collection contains only those objects that produce a page.
WithPages = "No"	No distinction is made between objects that do or do not produce a page. All objects are returned.

Using an abstract link

To use an abstract link towards a given physical metaclass

- Use the following syntax:

```
[C="<MetaAssociationEnd>" Metaclass="<Metaclass
HexaIdAbs>"]
```

Component Properties (CP)

The Component Properties tag [CP], enables editing of the properties of object components linked to the main object, the link being determined by the Component tag, [C]. It also enables editing of link properties.

This tag should be placed between the component analysis start tag [C="Link"] and the [/C] end tag. (as for the Component tag, it is possible to use a query).

Example:

[C="Link or query"][CP="Name" LNK="Out"][/C]	Link to another page

☛ *To determine attributes available with the CP tag, see Property Tag (P).*

Number of Components (CL and CLP)

Two tags enable processing of the number of components found by traveling a link or using a query, [CL] and [CLP].

In that the processing of tags [CL] and [CLP] is faster than consultation of components by tag [C], it can be advantageous to use these tags for adding text conditioned, for example, by the number of elements found.

Tag [CL] enables testing on the number of components (see Conditioning HTML Tag Editing).

☛ *The [CL] tag cannot be inserted by drag-and-drop. You must enter it manually, by modifying a [C] tag, for example.*

Example:

[CL=Message if cardinal >1]Messages : [/CL]	The text "Messages :" is edited when there is more than one message.

The [CLP] tag allows you to obtain information on the number of components. It is possible to carry out calculations on the result obtained using this tag.

[C="Link or query"] text [CLP="ordinal"/][/C]	Row of object reached

Example:

[CLP="Cardinal-Ordinal+1"]	Objects are presented starting with the last, with indication of their row ("number 3, number 2, number 1 »)

Drawing Tag (Draw)

The drawing tag [DRAW] enables insertion of a diagram (flowchart, etc.).

In order to use it, you should be positioned on "Diagram", either via a group in the descriptor or via a tag [C].

[DRAW/]	Display of the diagram describing the object

This tag can be complemented with Link, Target and SvgBubble attributes.

Link

This attribute creates a link to objects included in the drawing. A hypergraphic map is associated with the drawing.

[DRAW Link="Out"/]	If a page describing the object in the diagram exists, a link is created to this page.
[DRAW Link="In"/]	If a reference for the object exists in the page (BMK attribute), the link is created to this reference.
[DRAW Link="InOut"/]	If a page describing the object in the diagram exists, a link is created to this page. If not, if a reference for the object exists in the page, the link is created to this reference.

In the components of an object, the “Drawing” property of the “Descriptor” element enables rapid insertion of this tag together with its parameters using drag-and-drop.

SvgBubble

The SvgBubble attribute is used when images are generated in SVG format. It enables configuration of the comment appearing in tool tips.

There are three possible values for this attribute:

- "Complete": display complete comment,
- "Title": display comment title, ie. the first line of the comment,
- "None": no comment.

Interlinked Components

On processing of several levels of interlinked components, it can be useful to explicitly identify each of the components so as to be able to access their characteristics.

For this purpose, the ***Id*** attribute enables assignment of an identifier to a component.

The characteristics of this component will subsequently be accessible by specifying its identifier using the attribute ***ParentId***.

☛ *When parentId=root, it is the root that is referenced.*

Example in a text concerning an "Organizational Process":

```
[C=Org-Unit id=XXXX]
[CL="Operation" If="cardinal=0"]
The org-unit [CP=Name parentid=XXXX/] does not carry out
any operation.<br>
[/CL]
[/C]
```

WebObjectCollection (WOC) Tag

The WebObjectCollection (WOC) tag is available on descriptors with a Web site as entry point. It enables recovery of the list of objects producing pages in a Web site.

To define the tag, you can:

- specify nothing: the tag returns all objects of the Web site.
- indicate a MetaClass (for example, Organizational Process): the tag returns all objects of the MetaClass.
- use a query (for example, listing all organizational processes with name beginning with "a").

Example in a text concerning an org-unit:

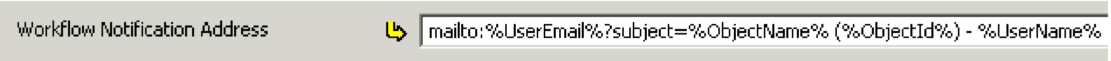
```
[WebObjectCollection="Select Org-Unit"]
  WebObjectCollectionProperty=Name Link="Out"/]
[/WebObjectCollection]
```

You can sort objects in alphabetical order using the"Sorter" attribute, specifying if sorting should be ascending ("A") or descending ("D") :

```
[WebObjectCollection="Procedure" Sorter = "Name:A"]
```

Workflow (WF) Tag

MEGA offers a "Workflow notification address" option. This is available in the user options window, under the **Documentation** > **Web Sites** folder.



Workflow Notification Address: mailto:%UserEmail%?subject=%ObjectName% (%ObjectId%) - %UserName%

The value has variables:

- %UserEmail%: e-mail address of last modifier of the object
- %ObjectName%: name of the object
- %Object%: HexaIdAbs of the object
- %UserName%: name of last modifier of the object
- %UserId%: HexaIdAbs of last modifier of the object

The Workflow (WF) tag generates a hyperlink pointing to the value of the "Workflow notification address" option which has as name the value of _CodeTemplate "Workflow".

When you click this hyperlink, it opens a message to the e-mail address of the last user containing the name and HexaIdAbs of the modified object as well as the name of the last user. The aim is to notify the last user of the object of a problem or to request modification of the object in question.

By default, the link name is "Modify", value that corresponds to the value of _CodeTemplate "Workflow". You can change the hyperlink name using the "Name" attribute:

```
[Workflow Name="My link"/]
```

```
-> generates a hyperlink pointing to the value of the
"Workflow notification address" option which has as name
"My link".
```

With the "Link" attribute, you can modify the value of "Workflow notification address" and arrange that the link returns a Web page containing a form pre-completed with values indicated in the link.

```
[Workflow Link="http://www.mywebsite.en..."/]
```

```
-> generates a hyperlink pointing to "http://
www.mywebsite.en..." which has as name the value of
_CodeTemplate: Workflow
```

You can also use in the link attribute the same variables as in the option: %UserEmail, %ObjectName%...

```
http://mywebsite.fr/
form?email=%UserEmail%&object=%ObjectName%&id=%ObjectId%
```

The value defined on the "Link" attribute overloads the option value

```
[Workflow Name="My link" Link="http://www.mywebsite.en..."/
]
```

```
-> generates a hyperlink pointing to "http://
www.mywebsite.en..." which has as name the value of
_CodeTemplate:
```

Context Tag

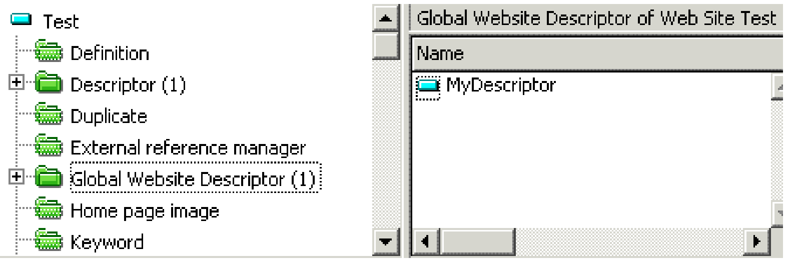
☛ *The Context tag is only used in the case of execution of a descriptor in a Web site.*

[Context=WebSiteHexaIdAbs/]

[Context=WebSiteHexaIdAbs/] returns the hexaIdabs of the Web site that executes the descriptor.

[Context=GlobalWebSiteDescriptor/]

- An example is a descriptor "MyDescriptor" with a Web site as entry point.
- Connect this descriptor to the Web site via the "Global web site descriptor" link.


Test
Definition
Descriptor (1)
Duplicate
External reference manager
Global Website Descriptor (1)
Home page image
Keyword
Global Website Descriptor of Web Site Test
Name
MyDescriptor

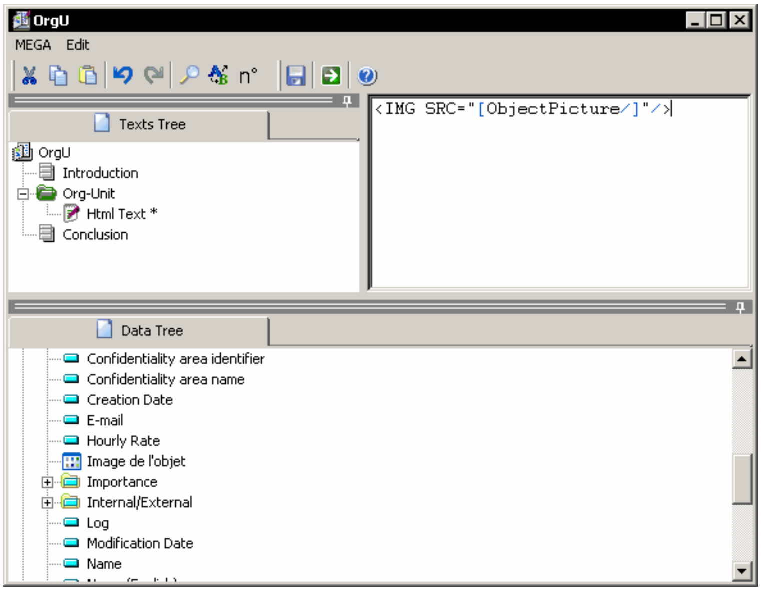
The [Context=GlobalWebSiteDescriptor/] tag executes the above descriptor and inserts the result in the descriptor that calls it.

☛ *In this case, the "Introduction" and "Conclusion" texts of the descriptor are not executed.*

ObjectPicture Tag

TheObjectPicture tag returns the icon of the MetaClass of the object.

You can insert this tag by dragging the object image into the descriptor.



```
<IMG SRC="[ObjectPicture/]"/>
```

On execution, this tag enables copying of the image file and returns the address of the copied file.

If you want the image to appear in the generated page, you must add the IMG tag:

```
<IMG SRC="[ObjectPicture/]"/>
```

This tag can contain the optional PictureFormat attribute that enables choice of image format between: "GIF" (default), "BMP" and "ICO".

Example:

```
[ObjectPicture PictureFormat="ICO"/]
```

In API VBs, the GetObjectPicture function enables the same operation.

It takes the Web generation context as parameter. You can add an optional parameter indicating image format: "GIF", "BMP", "ICO".

Example:

```
stringPictureFileName =
oObject.GetObjectPicture(oWebContext)
-> returns the address of the file in gif format


stringPictureFileName =
oObject.GetObjectPicture(oWebContext, "BMP")
-> returns the address of the file in bmp format
```

Using Settings

As in RTF descriptors, settings enable insertion of data from the repository (for example, the repository Name), and the configuration of each user (for example, Author). These settings are inserted between tags [V] and [/V].

Example

[V="Time"/]	Insertion of time

For more information on creation and modification of settings, see Checking Descriptor Validity.

Menu Bar Configuration

The "Menu" variable enables creation, in the menu frame, of the text indicated with this variable, together with a hyperlink to its location in the page.

[V=MNU]text[/V]	Inserts "Text" in the main frame and in the menu bar. A link is created between the two texts.

Modifying title style

The HEAD attribute can be used to modify the style of the title (from H1 to H6).

[V=MENU Head=3]Text[/V]	Inserts "text" using style H3 with a shift in the menu bar

Inserting a bullet before a title

[V=MNU Bullet="Toto.gif"]Text[/V]	Inserts bullet "Toto.gif" located in the folder "Mega_Std" (has.custom or Hopex.core module), then "Text" in style H1

Inserting HTML code

You can insert HTML code in the menu bar to improve its appearance, without this code appearing in the main frame. You can, for example, offer users the possibility of sending an e-mail to the Web site administrator.

To do this, you must use the "VISIBLE" parameter, which enables you to indicate:

- if tag content should be visible in the menu bar and in the main frame.
- or if the text is only HTML code enabling formatting of the menu bar without a link to the main frame.

[V=MNU Visible=YES] Text [/V]	Inserts "Text" in the main frame and in the menu bar. A link is created between the two texts.
[V=MNU Visible=No] Code HTML [/V]	inserts the HTML code in the menu bar, without a link to the main window.

Example

```
[V=MNU Visible=No]
<a href="mailto:xxx@mega.com">Write to xxx</a>
[/V]
```

Recovering the template of a _Code Template

A _Code Template contains reusable code.

The "CodeTemplate" variable enables recovery of the template or translatable template of _Code Template using the Template attribute, which can take values "Code" or "Translatable".

- [Variable=CodeTemplate Id="HexaIdAbs" Template="Translatable"/] returns the translatable template
- [Variable=CodeTemplate Id="HexaIdAbs" Template="Code"/] -> returns the template
- [Variable=CodeTemplate Id="HexaIdAbs"/] returns the template if it is not empty, and if not returns the translatable template

☛ *The variable Id is the equivalent of the hexaIdabs of a _CodeTemplate.*

Accessing Object, Home and Site Map Pages

You can access object, home and site map pages using the "ObjectPage", "HomePage" and "SiteMapPage" variables respectively.

[Variable=ObjectPage File=293829654227017D/]: recovers the object page that has hexaIdabs "293829654227017D".

[Variable=HomePage/]: recovers the home page

[Variable=SiteMapPage/]: recovers the site map page

Examples

Link to a business process:

```
<has href='[Variable=ObjectPage File=293829654227017D/
]'>Goods and services purchasing</a><br/>
```

Link to home page:

```
<has href='[Variable=HomePage/]'>Home</a><br/>
```

Link to site map:

```
<has href='[Variable=SiteMapPage/]'>Site Map</a><br/>
```

Checking Object Availability

Tag [Variable=IsAvailable Id=XXX] indicates whether the object with hexaidabs XXX is available or not.

Example 1:

```
[Variable=IsAvailable Id=4EA5565E466900BF
If="Value='true'"] Object with hexaidabs "4EA5565E466900BF"
exists<br>[/Variable]

[Variable=IsAvailable Id=4EA5565E466900BF
If="Value='false'"]Object with hexaidabs "4EA5565E466900BF"
does not exist or is unavailable[/Variable]
```

Example 2: List of available MetaClasses

```
[Component="select metaclass" Sorter="name:A"]

[Buffer=metahexid]

[Buffer=metahexid Set][ComponentProperty="_HexaIdAbs"/][/
Buffer]

[Variable=IsAvailable Id=Buffer(metahexid)
If="Value='true'"]

[ComponentProperty=Name/]<br/>

[/Variable]

[/Buffer]

[/Component]
```

Conditioning HTML Tag Editing

In the edition tags it is possible to indicate execution conditions: when the condition is satisfied, the tags are resolved and appear in the final result; if not, they are ignored.

```
[C="Operation" if="last"]</Table><br>[/C]
```

This condition enables closing of a table after the final operation found.

The conditions are introduced in the tag by "if=". They are placed between two double-apostrophes (").

Comparisons are carried out using operators "=", "<>", "<", ">". The negative operator " ! " can be placed before these operators.

The elements that can be tested are:

First	First element of a group. Can be used with tag [C].
Last	Last element of a group. Can be used with tag [C].
Ordinal	Position of the element in a group. Can be used with tag [C].
Cardinal	Number of elements in a group. Can be used with tags [C] and [CLP].
Empty	Indicates that the characteristic is not entered. Can be used with tags [C], [P] and [CP].
Value	Enables testing of the value of a characteristic. Can be used with tags [P] and [CP].
Internal value	Enables testing of the internal characteristic of a tabular characteristic. Can be used with tags [P] and [CP].

Example of use

```
[CL="org-unit"]
  [If="empty"]No org-unit[/If]
  [If="Cardinal=1"]Org-Unit : [/If]
  [If="Cardinal>1"]Org-Units : <br>[/If]
[/CL]
[C=Org-Unit]
  [CP=name/]<br>
[/C]
```

With these conditions, "Org-Unit :" followed by the name of the org-unit appears on one line when there is only one org-unit. When there are several org-units, the text is "Org-Units :", followed by a line break and the names of the org-units. When there are no org-units, the text “No Org-Unit” appears.

Another example

```
[CL="Project" if="Cardinal>0"]<h2>Projects of
organizational process:</H2>[/CL]
```

The text "Projects of organizational process" will appear if there is at least one project linked to the organizational process.

Using the tag [If]

Tag [If] enables creation of repetitions of a condition without having to rewrite the tag n times.

Example

```
[P="Property"]
[If="value=Value1"]Text 1[/If]
[If="Value=Value2"]Text 2[/If]
[/P]
```

Example

```
[C="Link or query"]
[If="First"] Text 1 [CP=name/][/If]
[If="!First and !Last"]Text 2[CP=name/][/If]
[If="Last"]Text 3[CP=name/][/If]
[/C]
```

☛ *The "Else" tag does not exist.*

HTML SOURCE TEXT SETUP

To improve legibility of the generated source it is possible to insert formatting tags in the text:

\n	New line
\c	Inserts a comment in the text
\i	Indentation (shift of 2 characters to the right)
\u	End of indentation (cancellation of shift of 2 characters to the right)
\t	Tabulation (as a function of the indentation defined)
\autoindent	Indicates that future calls to \n are a shortcut to \n\t
\noautoindent	Cancels the shortcut positioned by autoindent

These tags are used only for setup of the generated code.

They do not appear when the result is viewed by the end user on a navigator such as Internet Explorer or Netscape Navigator.

SPECIFIC USES OF HTML DESCRIPTORS

HTML Descriptors and Multilanguage

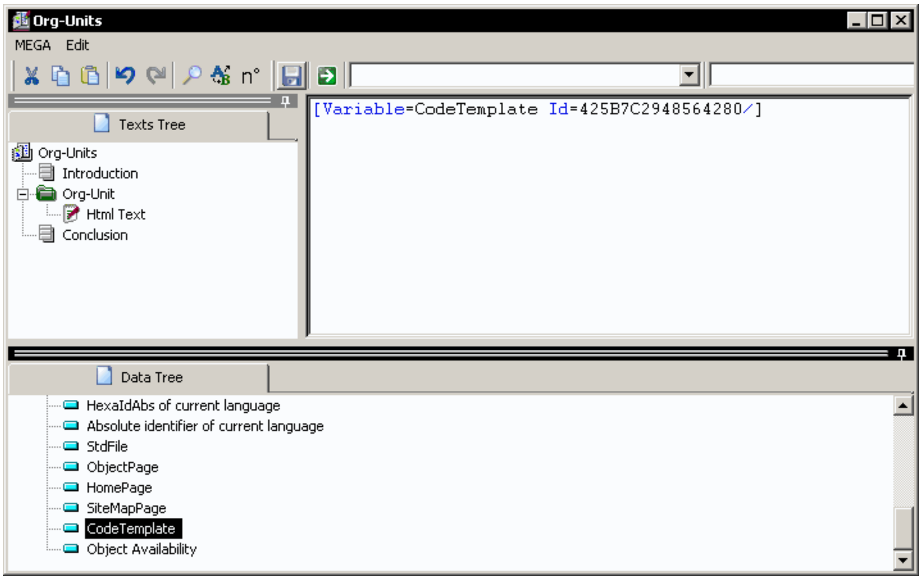
Settings

The "CodeTemplate" setting enables recovery of the root or translatable root of _Code Template.

Syntax

In an HTML and code descriptor framework the following syntax returns the root if it is not empty, otherwise it returns the translatable root:

```
[Variable=CodeTemplate Id="HexaIdAbs"/]
```



```
[Variable=CodeTemplate Id=425B7C2948564280/]
```

Inserting an e-mail Address in a Web Site

You can integrate the e-mail address of a person or service whose name appears in a Web site. To do this:

1. Position on the org-unit or person using tag [CP].
2. Enter the following code:

<A HREF="mailto: [ComponentProperty="E-mail"/]" > [ComponentProperty="E-mail"/] </A>

Creating a Buffer

The tag [Buffer] enables creation of a buffer (or variable) by assigning to it as a value a chain of characters or attribute values.

Declaring the buffer

It is necessary to declare a buffer before using it. A Buffer is declared by its name.

```
[Buffer= "MyBuffer1, MyBuffer2"]
```

☛ *Several buffers can be declared in the same tag.*

Assigning the buffer value

Its value is assigned using the Set attribute.

```
[Buffer="MyBuffer1,MyBuffer2]
[Buffer="MyBuffer1" Set="Text"/]
[Buffer="MyBuffer2" Set][P=Name/][/Buffer]
```

Restoring the buffer value

The buffer value is restored using the Get attribute.

```
[Buffer="MyBuffer" Get/]
```

Testing buffer contents

It is possible to test the contents of a buffer:

```
[If="Buffer(MyBuffer)=0"]text[/If]
```

Recuperating a calculation result

The result of a calculation can be retrieved from several buffer values (when these values are numeric).

```
[Buffer="cpt" set]
[Value Expression="buffer(b1)+buffer(b2)"/]
[/Buffer]
```

Copying an image in the generation folder

The tag [Variable] allows you to copy an image in the site "/Standard" folder. This is useful in the case where the generation folder of the Web site is moved and you wish to keep its related images.

```
[Variable=StdFile File="MyImage.GIF"
source="SourceFolderName" target = "TargetFolderName"/]
```

This allows you to copy the "MyImage.GIF" file (located in the Mega_Std/SourceFolderName folder) in the "/Standard/TargetFolderName" folder.

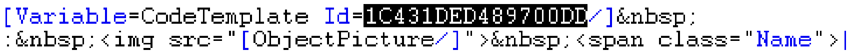
File	File to be copied
Source	Relative address (relative to "Mega_std" of Hopex.core or has.custom module) of the folder in which the file to be copied is located.
target	Relative address (relative to "Standard") of the folder in which the file will be copied.

Accessing _Code Template Properties

In a descriptor, you can display properties of a _Code Template or other object, in order to consult or modify its characteristics.

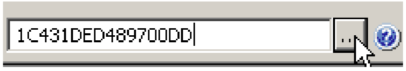
To display _Code Template properties:

1. In the descriptor code, select and copy the _Code Template identifier.



```
[Variable=CodeTemplate Id=1C431DED489700DD/]&nbsp;
:&nbsp;<img src="[ObjectPicture/]">&nbsp;<span class="Name">|
```

2. Paste the identifier in the field at top right of the Edit window.



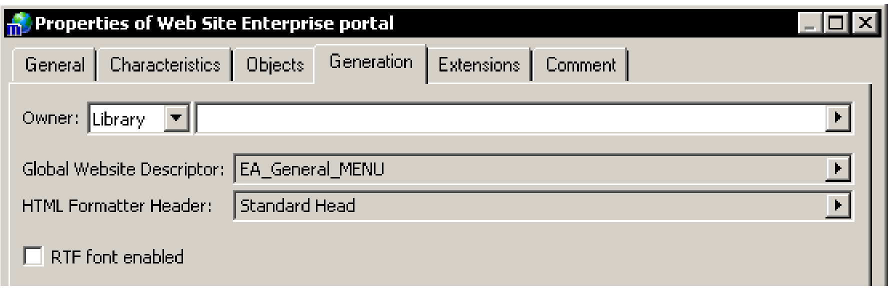
1C431DED489700DD

3. Click button ... at right of the field.
   The _Code Template properties dialog box appears.

Defining the Same Header for All Web Sites

The header formatter is an HTML formatter that can be implemented by:

- A macro
  The formatter is connected to the macro by the "Macro" MetaAssociationEnd.
- A descriptor
  The formatter is connected to the descriptor by the "Head Descriptor" MetaAssociationEnd.



Owner: Library

Global Website Descriptor: EA_General_MENU

HTML Formatter Header: Standard Head

□ RTF font enabled

☛ *This header appears in all pages between tags <head></head>, in dynamic mode (Advisor) or in static mode (desktop generation).*

HTML Descriptors for Dynamic Web Sites

Displaying a MEGA Metatree As a Web Tree

☛ *The Metatree tag is only available in dynamic mode (Advisor Enterprise and Professional Edition).*

[MetaTree=HexaIdAbsDuMetaTree/] displays a MEGA Metatree in the form of a Web tree.

```
Example: [MetaTree=446A8C004B2734F3/]
```

Seven attributes are available:

- Parameter
- ContainerVisible
- ContainerCollapsible
- ContainerCollapsed
- NavigateToObjectPage
- ShowRoot
- ShowTitle

Parameter

Represents the tree entry point: a library for example.

```
Example of use:
[MetaTree=446A8C004B2734F3 Parameter=  " MyParam "]
[/Metatree]
Or
[MetaTree=446A8C004B2734F3]
MyParam
[/Metatree]
```

ContainerVisible

ContainerVisible Attribute Value	Effect
1	The right part is visible.
0	The right part is not visible.

☛ *The default value is 1.*

ContainerCollapsible

ContainerCollapsible Attribute Value	Effect
1	The right part can be open or closed.

ContainerCollapsed

ContainerCollapsed Attribute Value	Effect
1	The right part is closed.
0	The right part is open.

☛ *The default value is 0.*

NavigateToObjectPage

NavigateToObjectPage Attribute Value	Effect
on	When you click a tree element, you navigate towards the object page if it exists. The object page is not displayed in the right part of the tree.
off	When you click a tree element, the object page displays in the right part of the tree (if this part is visible).

☛ *The default is "off".*

ShowRoot

ShowRoot Attribute Value	Effect
on	The tree root has the same name as the repository.
off	The tree root has the same name as the metatree.

☛ *The default is "on".*

ShowTitle

ShowTitle Attribute Value	Effect
on	A frame containing the tree name appears above the tree root.
off	The frame containing the tree name is not displayed.

☛ *The default is "on".*

Adding Favorites Page URL Access

[Variable= FavoritesPage/] enables display of favorites.

☛ *This tag is available only in dynamic mode.*

Code to define URL in menus

```
<li>
<b><a HREF="[Variable=FavoritesPage/]">Favorites</a></b>
</li>
```

You must add an index page to the Web site using formatter "MWA Favorites Page" supplied as standard.

Adding Search Page URL Access

[Variable= SearchPage /] enables display of the search tool.

☛ *This tag is available only in dynamic mode.*

Example code to define URL in menus

```
<li>
<b><a HREF="[Variable=SearchPage/]">Search</a></b>
</li>
```

You must add an index page to the Web site using formatter "MWA Search Index" supplied as standard.

Adding RFC Page URL Access

[Variable= RFCPage /] enables display of the RFC page.

☛ *This tag is available only in dynamic mode.*

Example code to define URL in menus

```
<li>
<b><a HREF="[Variable=RFCPage/]">RFC</a></b>
</li>
```

You must add an index page to the Web site using formatter "MWA RFC Page" supplied as standard.

Adding Logout Page URL Access

[Variable= Logout /] enables the current user to logout in dynamic mode.

☛ *This tag is available only in dynamic mode.*

Example code to define URL in our menus

```
<li class="last">
<b><a HREF="[Variable=Logout/]">logout</a></b>
</li>
```

Adding an Attribute to Property and ComponentProperty Tags

ContextMenu = On	The pop-up menu is displayed on the current object only in dynamic mode.
ContextMenu = Off	The pop-up menu is not displayed on the current object only in dynamic mode.

When we are on an external reference of type other than URL or file (for example, swf):

ExtRefStatus = Complete	We display the complete HTML which includes the SWF.
ContextMenu = Embed	We display only the HTML required to view the SWF and to embed it in the existing code.

ENTERPRISE PORTAL

WHAT IS THE ENTERPRISE PORTAL?

The Enterprise Portal allows to make your static website available as a tile in the HAS Portal.

It enables administrators to secure access to the static website via a proper authentification process (login/password or SSO).

The Enteprise Portal also allows to generate static websites in mono or multi-languages (English, French, Spanish, German, Italian and Portuguese).

Accessing the Enterprise Portal

Installation Prerequisites

The following modules must be imported into **HOPEX**:

- Enterprise Portal
- Static Website Content
- Static Website Macros

Note that the minimum version for each module is 15.3.0+47.

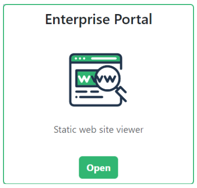
☛ *See "Importing a Module into HOPEX".*

Accessing the Enterprise Portal

To access the Enterprise Portal:

1. Open the HAS main page.
   The Enterprise Portal tile is displayed.

   ☛ *If not, see "Installation Prerequisites".*
2. Click **Open**.
3. Enter your Login and Password and click **Sign in**.



Restricting Access Rights

By default, the Enterprise Portal is accessible to all users.

However, access can be restricted to a specific group of users, ensuring that only those within an authorized group can visualize static websites. Unauthorized users will still have the opportunity to contact an administrator to request access.

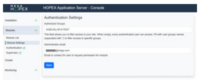
☛ *This feature is available from the version 16.0.2+4.*

To restrict the Enterprise Portal access to a group of users:

1. From the HAS console, click **Modules** > **Module Settings**.
2. Select **Enterprise portal settings**.
   Authentication settings are displayed.
3. In the **Authorized Groups** field, enter the name of the user group(s).

   ☛ *User Groups should be created via a third-party identity provider.*

4. In the **Administrator email** field, enter an email address that unauthorized users can use to request access.



Authentication Settings

Authorized Groups

\\DEVELOP;\\TEST

This field allows you to filter access to your site. When empty, every authenticated user can access. Fill with user groups names (separated with ';') to filter access to specific groups.

Administrator email

[redacted]@mega.com

Email to contact for user to request permission for module

CONFIGURING WEBSITE GENERATION

Configuring General Settings

To set up the display of your static website, you have to edit the website configuration file.

To edit the configuration file:

1. Open the Config.json file in edition mode.

   ☛ *The default path is: C:\ProgramData\MEGA\Hopex Application Server\<name of the HAS instance>\.shadowFiles\Macros\website.static.macros\<version number>\daily_batch\Exe\config.json*

   Make sure you use the "Config.json" file which is inside the "Exe" folder.

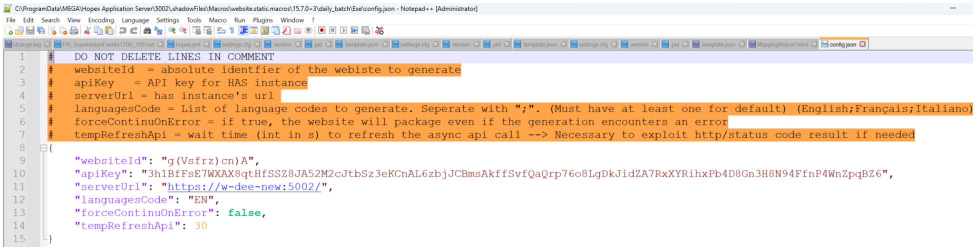
2. Enter/edit the following information if needed:
   - **"websiteID"**: enter the Absolute Identifier of the website to generate.

     ☛ *See "Website Absolute Identifier".*
   - **"apiKey"**: enter the API key.

     ☛ *See "Configuring General Settings".*
   - **"serverUrl"**: enter the HAS instance’s URL.
   - (Optional) **"languagesCode"**: the website is generated in English by default. Edit or add code language(s), separated by ";" to generate the website in the language(s) of your choice.

     ☛ *For example, the following code languages can be entered: FR (French); ES (Spanish); DE (German); IT (Italian); PT (Portuguese).*

     If you set up other languages here, further configuration is required before launching website generation. See "Configuring Optional Languages".
   - (Optional) **"forceContinuOnError"**: the "false" value is set by default. Enter "true" to package the website even if the generation encountered an error.
   - (Optional) **"temRefreshApi"**: "300s" is set by default. Reduce the wait time to refresh the API call if needed.
3. Save and close the file.


C:\ProgramData\MEGA\Hopex Application Server\5002\.shadowFiles\Macros\website.static.macros\15.7.0+3\daily_batch\Exe\config.json - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
change.log
FR_SupervisionEvents1700_100.md
hopex.yml
settings.cfg
version
pid
template.json
settings.cfg
version
pid
template.json
settings.cfg
version
pid
template.json
MappingReport.html
config.json

```
#   DO NOT DELETE LINES IN COMMENT
#   websiteId  = absolute identfier of the webiste to generate
#   apiKey    = API key for HAS instance
#   serverUrl = has instance's url
#   languagesCode = List of language codes to generate. Seperate with ";". (Must have at least one for default) (English;Français;Italiano)
#   forceContinuOnError = if true, the website will package even if the generation encounters an error
#   tempRefreshApi = wait time (int in s) to refresh the async api call --> Necessary to exploit http/status code result if needed
{
    "websiteId": "g(Vsfrz)cn)A",
    "apiKey": "3h1BfFsE7WXAX8qtHfSSZ8JA52M2cJtbSz3eKCnAL6zbjJCBmsAkffSvfQaQrp76o8LgDkJidZA7RxXYRihxPb4D8Gn3H8N94FfnP4WnZpqBZ6",
    "serverUrl": "https://w-dee-new:5002/",
    "languagesCode": "EN",
    "forceContinuOnError": false,
    "tempRefreshApi": 30
}
```

API Key

An API key must be created. It will authorize the user to execute the website generation process.

☛ *See "Managing API Keys".*

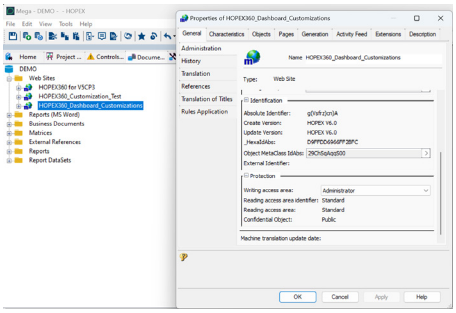
Website Absolute Identifier

The Website Absolute Identifier is essential for specifying which static website is displayed through the Enterprise Portal.

To retrieve the Website Absolute Identifier:

1. Open the HOPEX Windows Front-End
2. In the **Documentation** tab, expand **Web Sites**.

   ☺ *If you cannot see the Documentation tab, click **View** > **Navigation Windows** and select Documentation.*
3. Right click the adequate website to open its **Properties**.
4. In the **General** tab > **Administration** tab, copy the **Absolute Identifier**.



Name HOPEX360_Dashboard_Customizations

Type: Web Site

Absolute Identifier: g(Vsfrz)cn)A

Create Version: HOPEX V6.0

Update Version: HOPEX V6.0

_HexaIdAbs: D9FFDD6966FF2BFC

Object MetaClass IdAbs: 29ChSqAqqS00

External Identifier:

Writing access area: Administrator

Reading access area identifier: Standard

Reading access area: Standard

Confidential Object: Public

Machine translation update date:

Configuring Optional Languages

By default, the website is set to English.

If you want to generate a website in another language, further configuration is required:

- you have to specify the language code(s) in the website configuration file. See "Configuring General Settings".
- you have to modify the Enterprise Portal homepage:
  - for unilingual website generation outside of English
  - for multilingual website generation

☺ *If needed, check that the adequate language(s) in **HOPEX** are activated via **Tools > Options > Installation > Languages**.*

Editing the Enterprise Portal homepage for a unilingual website

By default, the Enterprise Portal homepage opens in English.

To set up the Enterprise Portal homepage in another language:

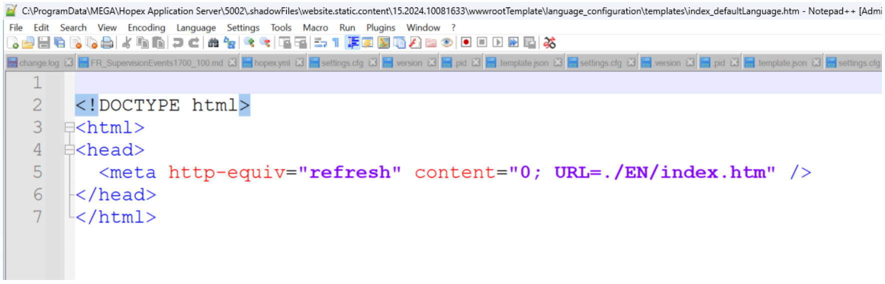
1. Access the Enterprise Portal homepage file in edition mode.

   ☛ *The default path is: C:\ProgramData\MEGA\Hopex Application Server\<name of the HAS instance>\.shadowFiles\website.static.content\<date of the last generation>\wwwrootTemplate\index.htm*

2. Edit the language code.

   ☛ *For example, instead of EN (English), enter another language code such as FR (French); ES (Spanish); DE (German); IT (Italian) or PT (Portuguese).*

3. Save the file.



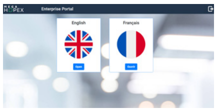
C:\ProgramData\MEGA\Hopex Application Server\5002\.shadowFiles\website.static.content\15.2024.10081633\wwwrootTemplate\language_configuration\templates\index_defaultLanguage.htm - Notepad++ [Admi

```

<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="refresh" content="0; URL=./EN/index.htm" />
</head>
</html>
```

Editing the Enterprise Portal homepage for a multilingual website

For multilingual websites, it is necessary to edit the Enterprise Portal homepage so as to enable the end-user to select the language of their choice via language tiles.

A multi-language template is provided to help you with this task. In this template, language tiles are available by default in English and French, as shown in the image below.



Setting up the Enterprise Portal homepage in several languages

To set up the Entreprise Portal homepage in several languages:

1. Open the "index_severalLanguages.htm" template in edition mode.

   ☛ *The default path is: C:\ProgramData\MEGA\Hopex Application Server\<name of the HAS instance>\.shadowFiles\website.static.content\<date of the last generation>\wwwrootTemplate\language_configuration\templates\index_severalLanguages.htm*

2. (Optional) Add/Edit language tiles.

   ☛ *See "Adding/Editing a language tile".*

   ☺ *This action is not necessary of you generate a bilingual website in English and French.*

3. Copy the content into the "index_htm" Enterprise Portal homepage file.

   ☛ *The default path is: C:\ProgramData\MEGA\Hopex Application Server\<name of the HAS instance>\.shadowFiles\website.static.content\<date of the last generation>\wwwrootTemplate\index.htm*

4. Save the file.

Adding/Editing a language tile

By default, language tiles are available for English and French languages. Therefore, no further configuration is required for these two languages.

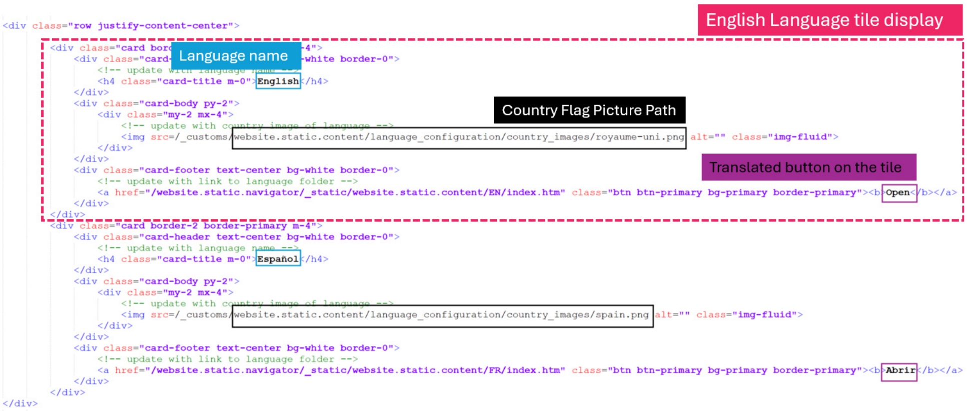
For any other language, you need to set up a new language tile.

To add/edit a language tile:

1. Insert an image of the adequate country flag in png format in the "country_images" folder.

   ☛ *The default path is C:\ProgramData\MEGA\Hopex Application Server\<name of the HAS instance>\.shadowFiles\website.static.content\<date of the last generation>\wwwrootTemplate\language_configuration\country_configuration\country_images.*

2. Open the "index_severalLanguages.htm" template in edit mode
3. Edit the adequate information:
   - Language name
   - Country flag picture
   - Open button on the tile



```
<div class="row justify-content-center">
    <div class="card bor[illegible]-4">
        <div class="card-[illegible]-white border-0">
            <!-- update with language name -->
            <h4 class="card-title m-0">English</h4>
        </div>
        <div class="card-body py-2">
            <div class="my-2 mx-4">
                <!-- update with country image of language -->
                <img src=/_customs/website.static.content/language_configuration/country_images/royaume-uni.png alt="" class="img-fluid">
            </div>
        </div>
        <div class="card-footer text-center bg-white border-0">
            <!-- update with link to language folder -->
            <a href="/website.static.navigator/_static/website.static.content/EN/index.htm" class="btn btn-primary bg-primary border-primary"><b>Open</b></a>
        </div>
    </div>
    <div class="card border-2 border-primary m-4">
        <div class="card-header text-center bg-white border-0">
            <!-- update with language name -->
            <h4 class="card-title m-0">Español</h4>
        </div>
        <div class="card-body py-2">
            <div class="my-2 mx-4">
                <!-- update with country image of language -->
                <img src=/_customs/website.static.content/language_configuration/country_images/spain.png alt="" class="img-fluid">
            </div>
        </div>
        <div class="card-footer text-center bg-white border-0">
            <!-- update with link to language folder -->
            <a href="/website.static.navigator/_static/website.static.content/FR/index.htm" class="btn btn-primary bg-primary border-primary"><b>Abrir</b></a>
        </div>
    </div>
</div>
```

English Language tile display
Language name
Country Flag Picture Path
Translated button on the tile

LAUNCHING WEBSITE GENERATION

Launching Website Generation

To launch website generation:

- Double-click the GenerateAndPackageModule.exe.

  ☛ *The default path is: C:\ProgramData\MEGA\Hopex Application Server\<name of the HAS instance>\.shadowFiles\Macros\website.static.macros\<version number>\daily_batch\Exe\GenerateAndPackageModule.exe*

  The website is generated in batch mode.

Accessing the Trace File

To check the status of website generation, you can open the website generation trace file.

To access the trace file:

- Open the WEBSITEGENERATE-Hopex-[Macro].log

  ☛ *The default path is C:\ProgramData\MEGA\Hopex Application Server\<name of the HAS instance>\Logs>*WEBSITEGENERATE-Hopex-[Macro].log

POST-GENERATION CONFIGURATION

Post-Generation Directory

After website generation, the related files are saved in a directory.

☛ *The default path is C:\ProgramData\MEGA\Hopex Application Server\<name of the HAS instance>.shadowFiles\website.static.content\<version number>\wwwroot.*

☺ *If you have generated a multilingual website, there will be a folder per language.*

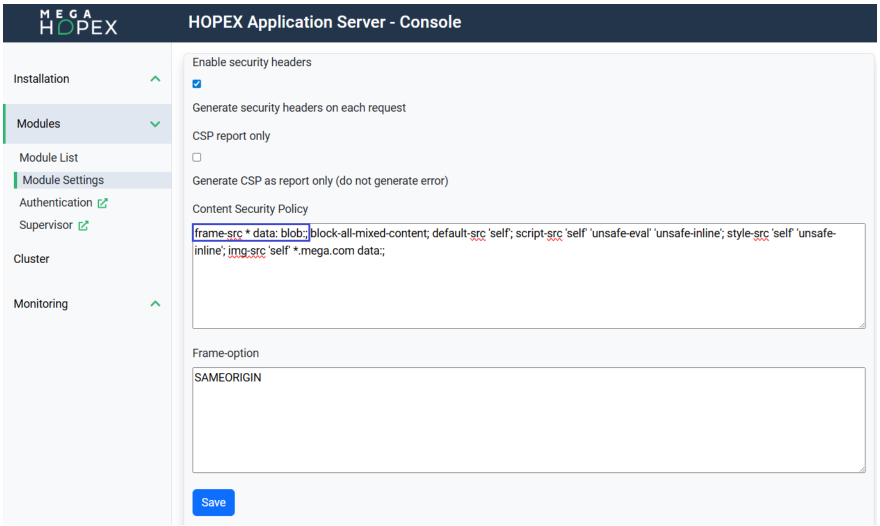
Enabling External References

External references cannot be displayed when the static website is opened through the Enteprise Portal.

Further configuration is required to enable their display.

To enable external references:

1. Open the HAS Console.
2. Navigate to **Modules > Module Settings**.
3. Edit **Http Security Headers**.
4. In the **Content Security Policy** field, enter: *frame-src * data: blob:;*
5. Click **Save**.



Enable security headers

☑

Generate security headers on each request

CSP report only

□

Generate CSP as report only (do not generate error)

Content Security Policy

frame-src * data: blob:; block-all-mixed-content; default-src 'self'; script-src 'self' 'unsafe-eval' 'unsafe-inline'; style-src 'self' 'unsafe-inline'; img-src 'self' *.mega.com data:;

Frame-option

SAMEORIGIN

External references to SharePoint

Additional configuration is required on your SharePoint server to enable the display of external references that redirect to SharePoint.

☛ *See https://learn.microsoft.com/en-us/answers/questions/330765/content-security-policy-settings-for-sharepoint-si.*

HOPEX 360 Web Site Template

1. Overview

1.1. Technical limitations

1.1.1. Web browser Version

This website template is compliant with the following web browsers:

- Firefox 75 and above
- Chrome 81 and above

1.1.2. Resolution

This website is optimised for a 1920 x 1080.

1.1.3. Licensing

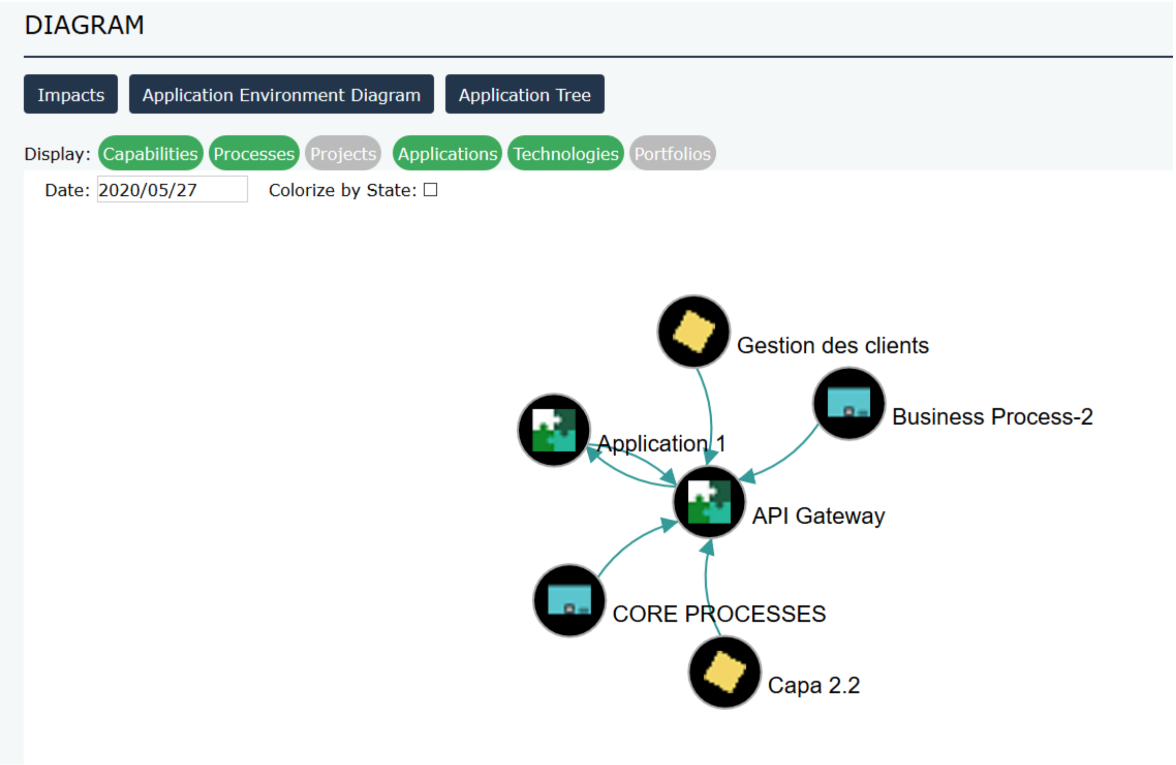
It only considers the scope of the ITPM and ITBM licence.

1.1.4. Deployment

Once generated, the static website needs to be hosted on an IIS webserver to enable all the functionalities.

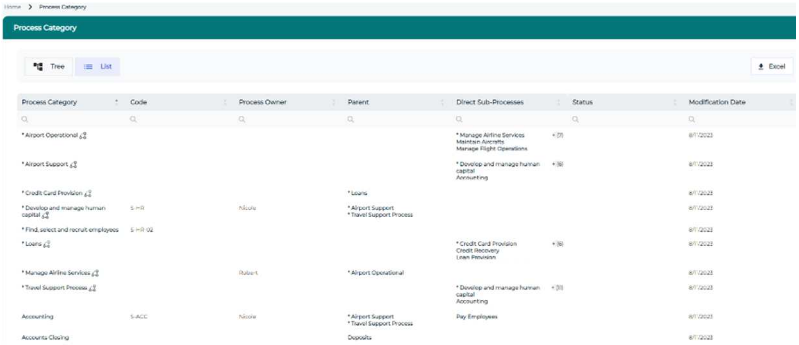
The functionalities that require IIS include but are not limited to:

- Application Impacts diagram



Date: 2020/05/27 Colorize by State: □

- Capabilities and Process tree list



Home > Process Category

Process Category

Process Category	Code	Process Owner	Parent	Direct Sub-Processes	Status	Modification Date
* Airport Operational				* Manage Airline Services Maintain Aircrafts Manage Flight Operations + (7)		8/1/2023
* Airport Support				* Develop and manage human capital Accounting + (6)		8/1/2023
* Credit Card Provision			* Loans			8/1/2023
* Develop and manage human capital	S-HR	Nicole	* Airport Support * Travel Support Process			8/1/2023
* Find, select and recruit employees	S-HR-02					8/1/2023
* Loans				* Credit Card Provision Credit Recovery Loan Provision + (6)		8/1/2023
* Manage Airline Services		Robert	* Airport Operational			8/1/2023
* Travel Support Process				* Develop and manage human capital Accounting + (11)		8/1/2023
Accounting	S-ACC	Nicole	* Airport Support * Travel Support Process	Pay Employees		8/1/2023
Accounts Closing			Deposits			8/1/2023

1.1.5. Languages

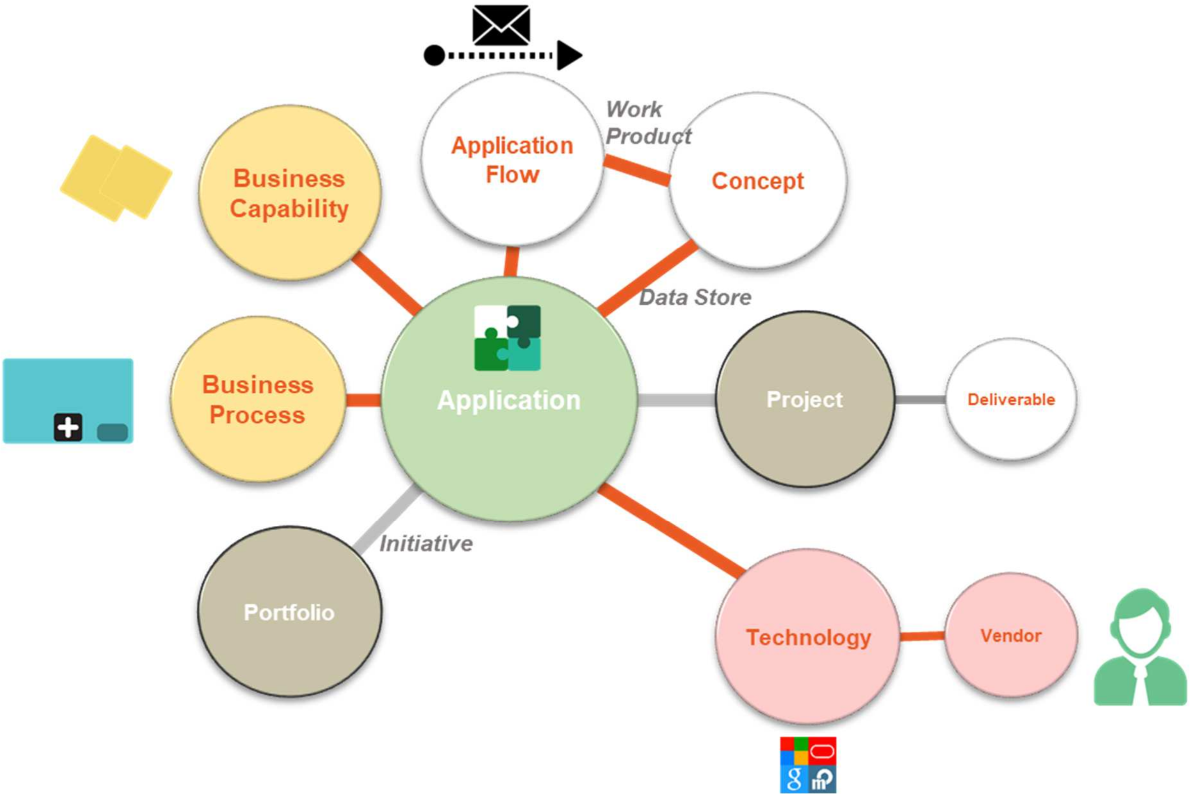
The website structure is available in English, French, Spanish and Portuguese. The translations can be amended using the _Code Templates referenced in the descriptors.

The website is generated in the language set on the user running the web generation.

The translation of the content itself relies on the available languages of HOPEX.

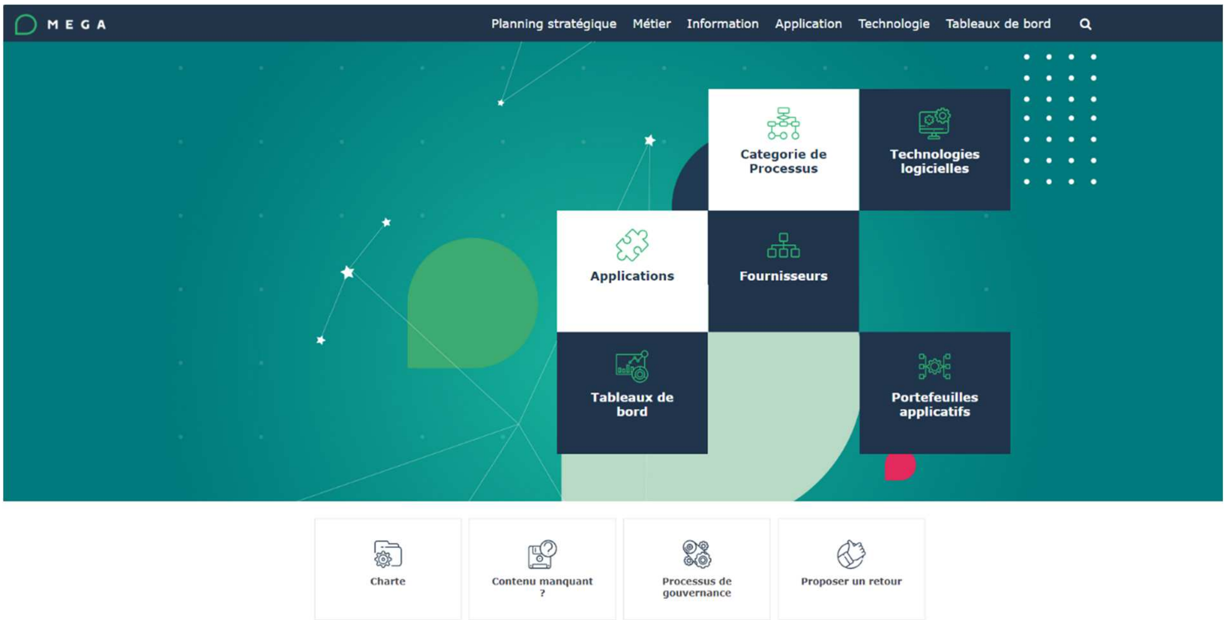
1.2. Data scope

HOPEX360 covers the following data. For more details, refer to the tables at the end of the document.



1.3. Homepage

1.3.1. Overview



1.3.2. Client specific

The 4 tiles at the bottom do not point anywhere. It is up to the consultant and/or the client to customize the links in the Homepage descriptor.

The purpose of these links is to point to:

- **Client specific Charter**: online document
- **Missing content**: email to a dedicated email address
- **Governance Process**: online document
- **Provide Feedback**: email to a dedicated email address

To customize the tiles at the bottom:

1. Open the "HOPEX360_Home" Descriptor.
2. Open the "Bottom Tiles" text.
3. Add a link within the <h6> tag:
   - Client specific charter: line 47
   - Missing content: line 92
   - Governance Process: line 145
   - Provide feedback: line 188

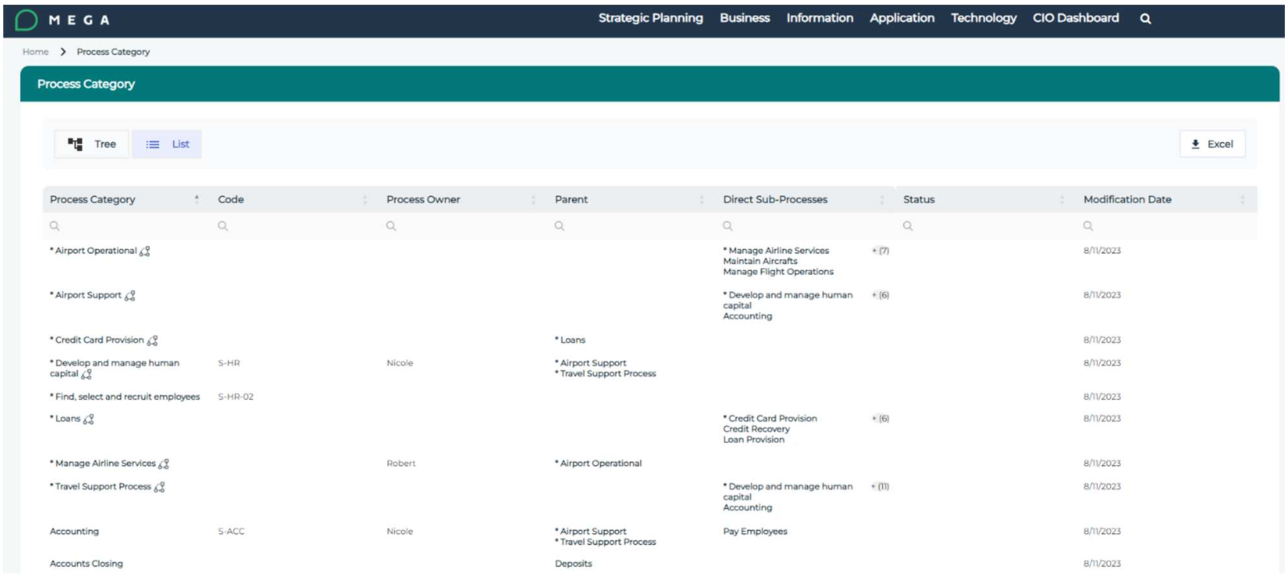
1.4. Site Map

The top menu allows to navigate through the pages.

Here is a breakdown of the navigation:

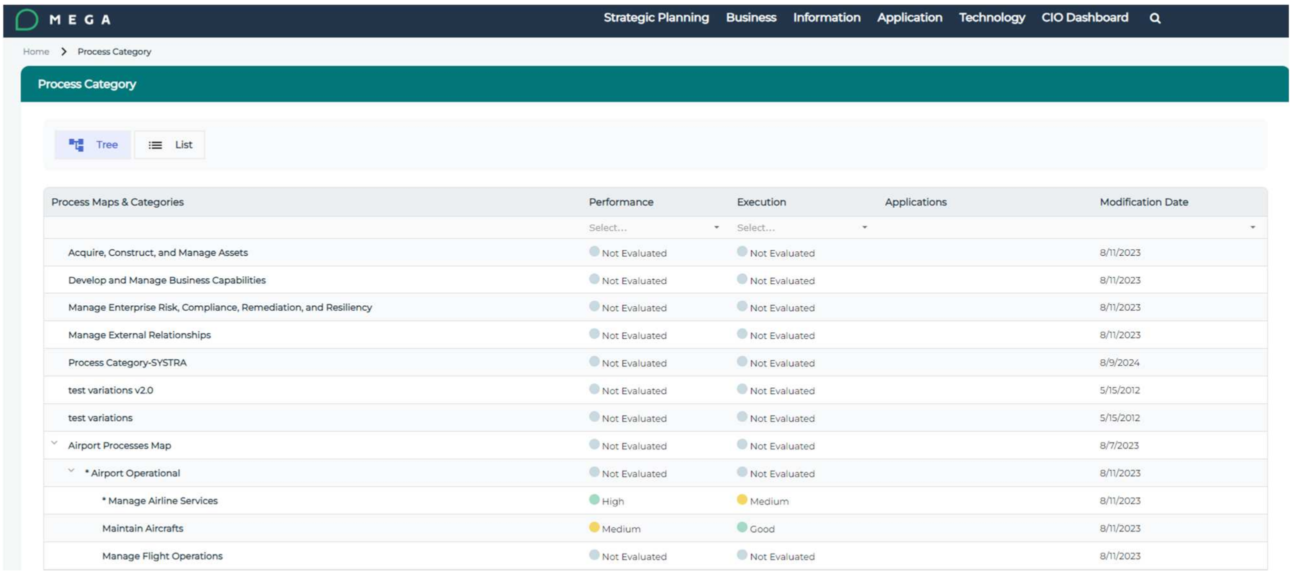
Navigation menu			Associated Object
Strategic Planning	Enterprises	Enterprises list: searchable hierarchical list	Enterprises
Business	Capability	Capabilities Tree list: searchable hierarchical list	Business Capability Maps Business Capabilities
		Capabilities list: alphabetically sorted flat list	
	Process	Processes Tree list: searchable hierarchical list	Process Categories
		Processes list: alphabetically sorted flat list	
Information	Dictionary	Concepts list: alphabetically sorted flat list	Concept
	Business Information Maps	Business Information Map list: searchable hierarchical list	Business Information Maps
Application	Portfolios	App portfolios list: alphabetically sorted flat list	Portfolio typed as Application
	Applications	Applications list: alphabetically sorted flat list	Application
	Projects	Projects list: alphabetically sorted flat list	Project
Technology	Portfolios	Tech portfolios list: alphabetically sorted flat list	Portfolio typed as Technology
	Technologies	Technologies list: alphabetically sorted flat list	Technology
	Vendor	Vendors list: alphabetically sorted flat list	Org Unit typed Vendor
CIO Dashboard		Main dashboard	Specific report page
		Technology Obsolescence	Specific report page
		Strategic Planning	Specific report page
		Rationalization	Specific report page
		Reports list: alphabetically sorted flat list	Specific report page
Search		Pop up to type in key words	Results shown in table

1.5. List index



Process Category	Code	Process Owner	Parent	Direct Sub-Processes	Status	Modification Date
* Airport Operational				* Manage Airline Services Maintain Aircrafts Manage Flight Operations + (7)		8/11/2023
* Airport Support				* Develop and manage human capital Accounting + (6)		8/11/2023
* Credit Card Provision			* Loans			8/11/2023
* Develop and manage human capital	S-HR	Nicole	* Airport Support * Travel Support Process			8/11/2023
* Find, select and recruit employees	S-HR-02					8/11/2023
* Loans				* Credit Card Provision Credit Recovery Loan Provision + (6)		8/11/2023
* Manage Airline Services		Robert	* Airport Operational			8/11/2023
* Travel Support Process				* Develop and manage human capital Accounting + (11)		8/11/2023
Accounting	S-ACC	Nicole	* Airport Support * Travel Support Process	Pay Employees		8/11/2023
Accounts Closing			Deposits			8/11/2023

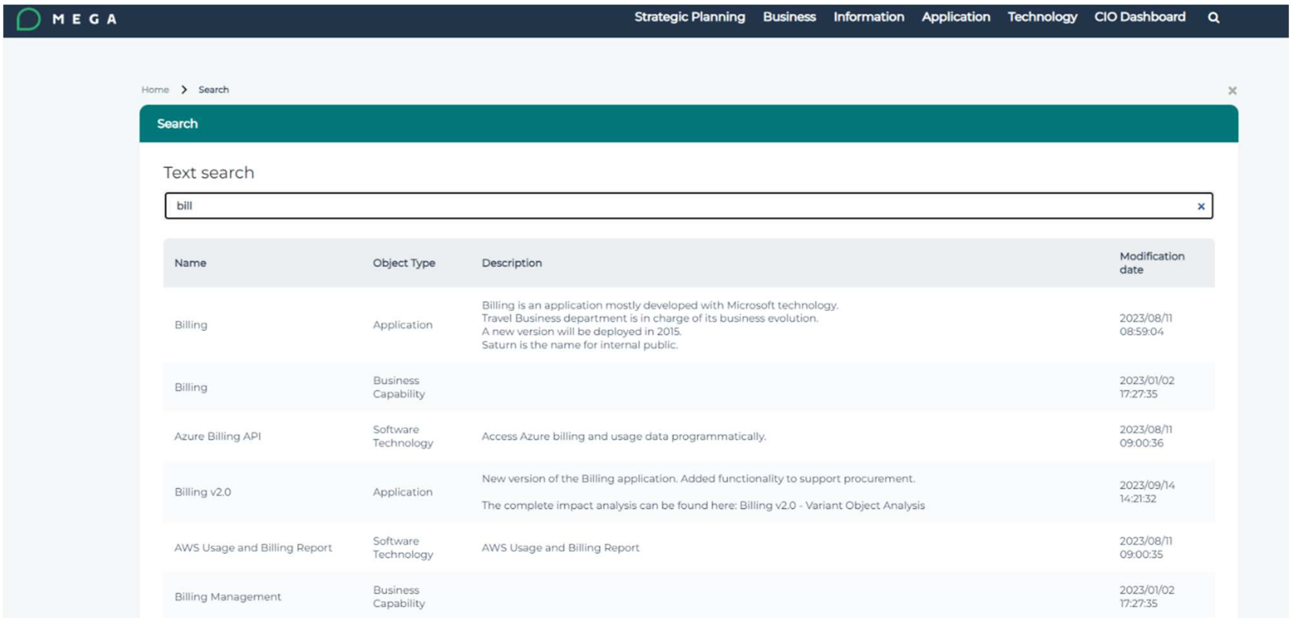
1.6. Tree list index



Process Maps & Categories	Performance	Execution	Applications	Modification Date
	Select...	Select...		
Acquire, Construct, and Manage Assets	● Not Evaluated	● Not Evaluated		8/11/2023
Develop and Manage Business Capabilities	● Not Evaluated	● Not Evaluated		8/11/2023
Manage Enterprise Risk, Compliance, Remediation, and Resiliency	● Not Evaluated	● Not Evaluated		8/11/2023
Manage External Relationships	● Not Evaluated	● Not Evaluated		8/11/2023
Process Category-SYSTRA	● Not Evaluated	● Not Evaluated		8/9/2024
test variations v2.0	● Not Evaluated	● Not Evaluated		5/15/2012
test variations	● Not Evaluated	● Not Evaluated		5/15/2012
Airport Processes Map	● Not Evaluated	● Not Evaluated		8/7/2023
* Airport Operational	● Not Evaluated	● Not Evaluated		8/11/2023
* Manage Airline Services	● High	● Medium		8/11/2023
Maintain Aircrafts	● Medium	● Good		8/11/2023
Manage Flight Operations	● Not Evaluated	● Not Evaluated		8/11/2023

1.7. Search

The search is implemented by the Fusejs library (https://fusejs.io/).



Text search

bill ×

Name	Object Type	Description	Modification date
Billing	Application	Billing is an application mostly developed with Microsoft technology. Travel Business department is in charge of its business evolution. A new version will be deployed in 2015. Saturn is the name for internal public.	2023/08/11 08:59:04
Billing	Business Capability		2023/01/02 17:27:35
Azure Billing API	Software Technology	Access Azure billing and usage data programmatically.	2023/08/11 09:00:36
Billing v2.0	Application	New version of the Billing application. Added functionality to support procurement. The complete impact analysis can be found here: Billing v2.0 - Variant Object Analysis	2023/09/14 14:21:32
AWS Usage and Billing Report	Software Technology	AWS Usage and Billing Report	2023/08/11 09:00:35
Billing Management	Business Capability		2023/01/02 17:27:35

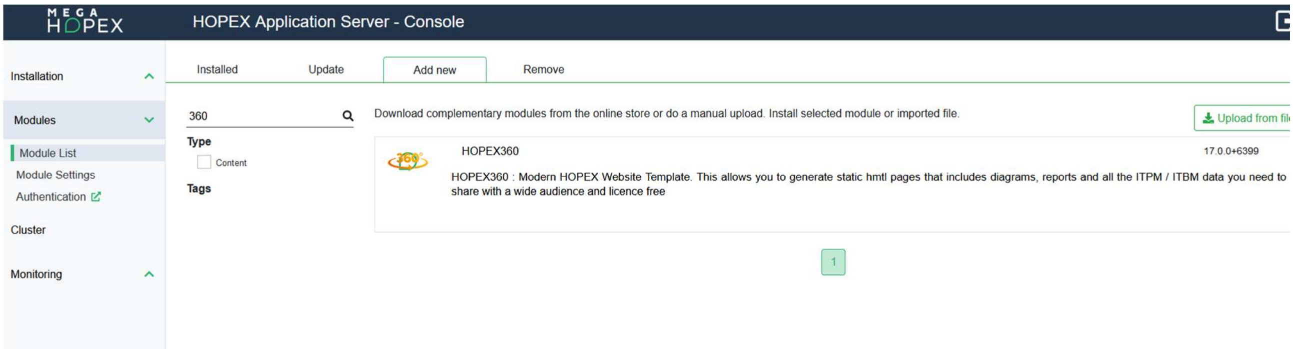
2. Configuration

2.1. Installation

The website template is delivered as an HAS Module.

To install the module:

1. From the **HAS Console**, go in **Modules > Module List > Add new**.
2. Search for "360" and install the module in a version that is relevant to your HOPEX Version.



Download complementary modules from the online store or do a manual upload. Install selected module or imported file.

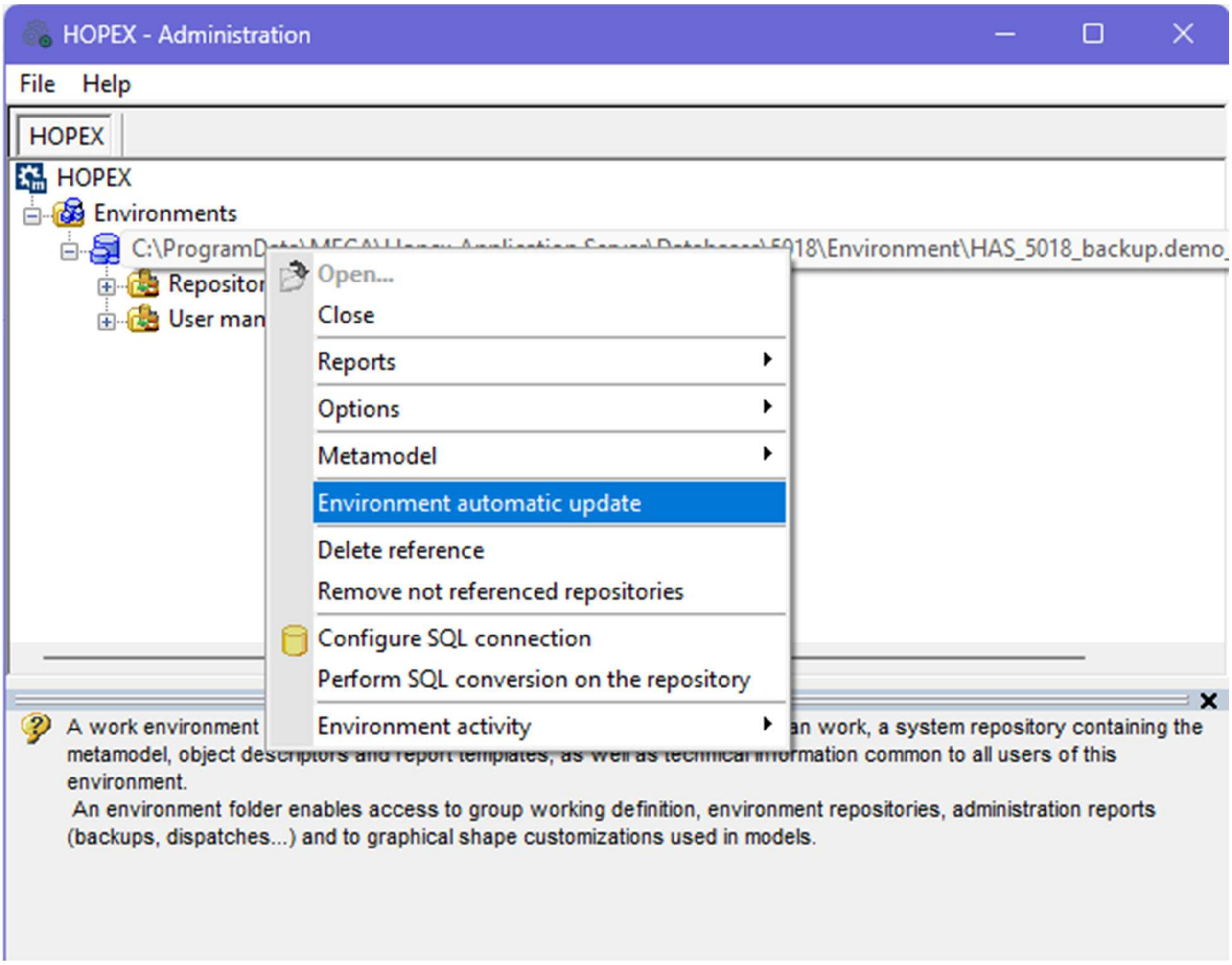
HOPEX360 17.0.0+6399

HOPEX360 : Modern HOPEX Website Template. This allows you to generate static hmtl pages that includes diagrams, reports and all the ITPM / ITBM data you need to share with a wide audience and licence free

BEWARE: after the module is installed from the HAS Console, it is required to run the Easy Update from the Administration.exe

3. Run the Administration.exe located in C:\ProgramData\MEGA\Hopex Application Server\<HAS Instance>\

4. Log in to the environment using the System account.
5. Right-click the environment and launch the **Environment automatic update**.



A work environment an work, a system repository containing the metamodel, object descriptors and report templates, as well as technical information common to all users of this environment.
An environment folder enables access to group working definition, environment repositories, administration reports (backups, dispatches...) and to graphical shape customizations used in models.

This utility installs the HOPEX objects and the relevant HTML resources.

2.2. Required files

Apart from the HOPEX Objects delivered in the HOPEX repository, the following HTML files will be stored in the HOPEX360 module shadow files folder.

- CSS files
- JS files
- Images
- Fonts

2.3. Translation

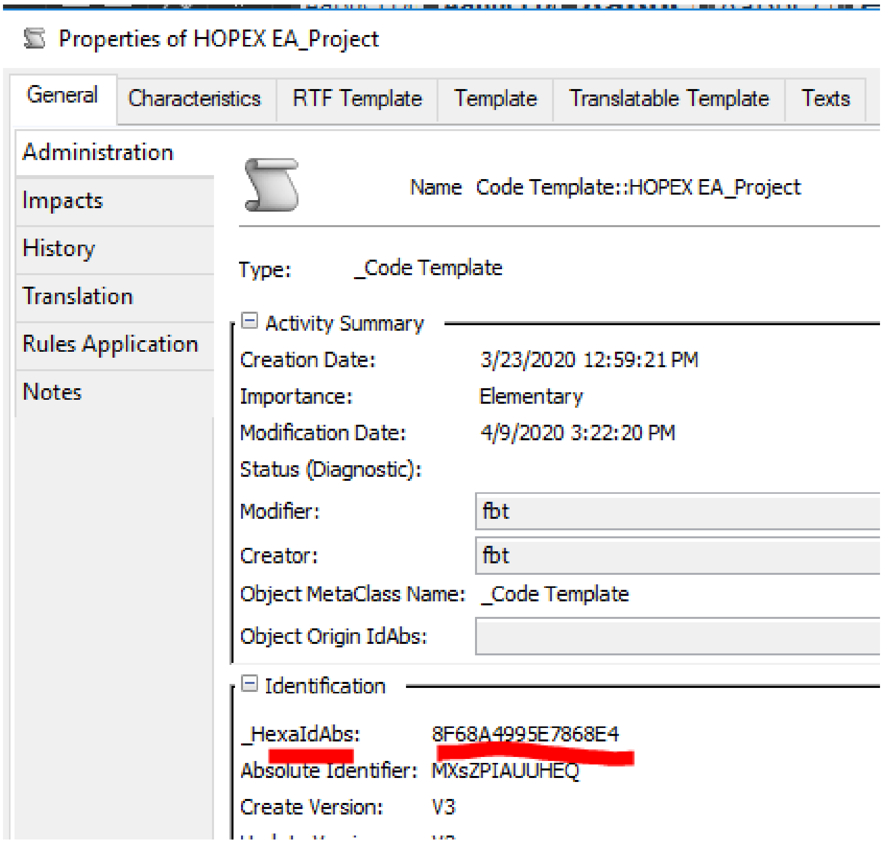
Translations are available on the following items:

- Menu
- Section
- Widget
- Table header

For each of them, a reference is written in the code to the relevant Code Template, as follow:

```
[Variable=CodeTemplate Id=8F68A4995E7868E4/]
```

Where the Id is the HexaIdAbs of the code template, as shown below in the _Code Template property page:



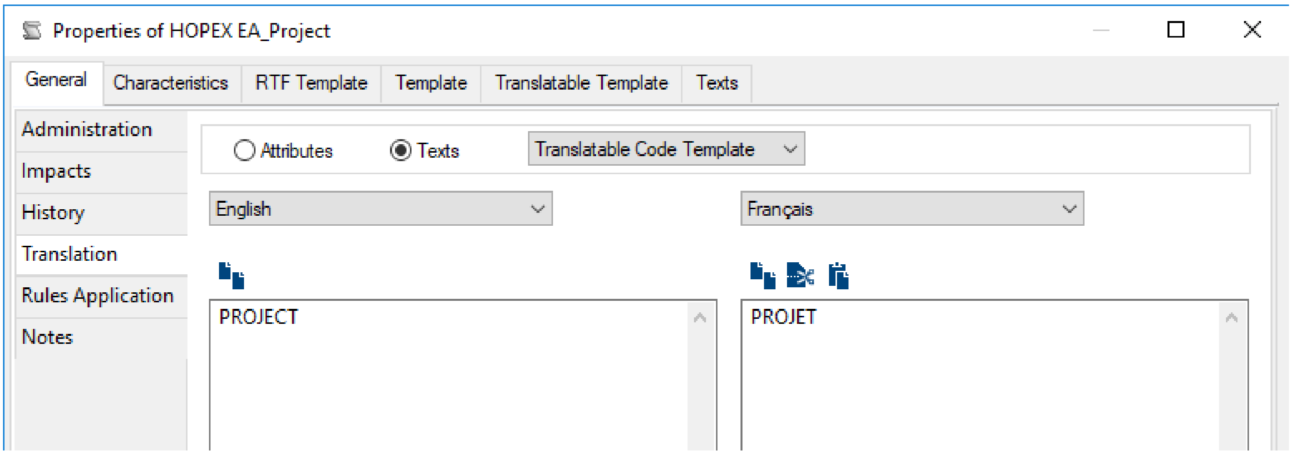
Name Code Template::HOPEX EA_Project

Type: _Code Template

Creation Date:	3/23/2020 12:59:21 PM
Importance:	Elementary
Modification Date:	4/9/2020 3:22:20 PM
Status (Diagnostic):	
Modifier:	fbt
Creator:	fbt
Object MetaClass Name:	_Code Template
Object Origin IdAbs:	

_HexaIdAbs:	8F68A4995E7868E4
Absolute Identifier:	MXsZPIAUUHEQ
Create Version:	V3

The text can be modified in **General > Translation**.


Properties of HOPEX EA_Project

○ Attributes ● Texts Translatable Code Template

English | Français

PROJECT

PROJET

2.4. Post-generation script & JSON files

JSON files are generated as part of the website generation to cover different functionalities. These files are stored in the **\json** folder of the generated website.

Those JSON files contain the key data used in the reporting, via java script functions.

This is implemented through 4 MetaCommand Managers triggering the relevant macros:

- HOPEX EA_Post Generation Script App JSON. This generates one JSON per Application to enable the Application Impact diagram. The files are named as follows:
  - <Application hexaidabs>_integration.json
- HOPEX360_Post generation Script_BoxInBox Structures JSON: this generates 53JSON files:
  - bcm.JSON
  - fm.JSON
  - tfm.JSON
- "HOPEX EA_Post Generation Script_Search Indexing"

  ⚠ For relative link reason, this JSON is stored in **\standard\assets\js** folder.

  - searchindexcontent.json

2.4.1. CIO Dashboard JSON

The 'HOPEX EA_Post Generation Script CIO Dashboard JSON' MetaCommand Manager generates 4 JSON files:

- **Bcm.JSON**

Topic	Key information
Purpose	This is a hierarchical list of Business Capability Maps including: - Name - HexaIdAbs - Children The main purpose is to build the drawing of the Business Capability Map
Pages using the file	The following page uses this data as a filter on the Box in Box: - Business Capability Map

JSON data structure	[
	{
	"name": "Airport Capability Map",
	"id": "72E379AE5ABE8363",
	"children": [
	{
	"name": "Analytics",
	"id": "4395476F59D30ABB",
	"children": []
	}
	]
	}
	]

- **Fm.JSON**

Topic	Key information
Purpose	This is a hierarchical list of Functionalities
Pages using the file	The following page uses this data: - Functionality Map

JSON data structure	[{
	"name": "Flight Reservations Functionality Map",
	"id": "369646D1614B7AF8",
	"children": [
	{
	"name": "Available flight display",
	"id": "3696480A614B7CB6",
	"children": []
	},
	{
	"name": "Flight booking",
	"id": "369650AE614B817B",
	"children": []
	}
	]
	}
	]

- **Tm.JSON**

Topic	Key information
Purpose	This is a hierarchical list of Functionalities
Pages using the file	The following page uses this data: - Technical Functionality Map

JSON data structure	[{
	"name": "Cloud Technical Functionality Map",
	"id": "369646D1614B7AF8",
	"children": [
	{
	"name": "Remote Access",
	"id": "3696480A614GFDB6",
	"children": []
	},
	{
	"name": "UI",
	"id": "369652GKL14B817B",
	"children": []
	}
	]
	}
	]

2.4.2. Tree list JSON

The ‘HOPEX EA_Post Generation Script Process and Capa Trees JSON’ MetaCommand Manager generates 2 JSON files:

- **DatasArrayBCMHierarchy.js**

Topic	Key information
Purpose	The purpose is to provide the full hierarchical structure of the Business Capabilities and their supporting applications.
Pages using the file	The following page uses this JSON data: - Business Capability Tree List
JSON data structure	var products = [{text: "BCM To Be (EN) ", modification_date:"2020/04/14 08:06:47", performance:"", execution:"", link: "726B7F225E84F5C5.htm", id: "726B7F225E84F5C5", parent:"0", applications: [{text:"N/A"} },];

- **DatasArrayProcessHierarchy.js**

Topic	Key information
Purpose	The purpose is to provide the full hierarchical structure of the Process Categories and their supporting applications.
Pages using the file	The following page uses this JSON data: - Process Category Tree List
JSON data structure	var products = [{text: "Process Category-2 (EN) ", modification_date:"2020/04/16 10:48:07", performance:"3", execution:"3", link: "51797E5D5E77826E.htm", id: "51797E5D5E77826E", parent: "9ACC0F275D444F86", applications: [{ text:"API Gateway (EN) ", link: "D37AD7455CF088CA.htm" },] },];

2.4.3. Application page JSON

These are not post generated, but instead, the HOPEX360_Object_Application descriptor generates a JSON file per Application.

Topic	Key information
Purpose	The JSON file enables the Impact force diagram on each Application page
Pages using the file	Each Application Object Page
JSON data structure	{"root" : {"id":"D37AD7485CF08ECB", "name":"service-draws (EN) ", "type_id":"B1EDB2562C14016F", "type_name":"Application", "href":"D37AD7485CF08ECB.htm", "image":"../standard/appl.ico.gif", "states":[ ] }, "sources": {}, "targets": {"FD280B395C926AC9": {"id":"FD280B395C926AC9", "name":"AngularJS 1.7 (EN)", "type_id":"2885299152A85717", "type_name":"Technology", "href":"FD280B395C926AC9.htm",

	"image":"../standard/technology.ico.gif" }, },];

2.4.4. Search

The “HOPEX EA_Post Generation Script_Search Indexing” MetaCommand Manager has a Macro attached which generates a ‘searchindexcontent.json’ file.

Topic	Key information
Purpose	This JSON file is used to reference all the objects included in the website to enable a full text search
Pages using the file	In every page, the menu includes a magnifying glass to toggle the search tool. This search is implemented using the Fuse library (https://fusejs.io/)

JSON data structure	{pages":
	[{
	"title":"",
	"Object type":"",
	"Modification Date":"",
	"text":"",
	"tags":"",
	"url":""
	},
	{
	"title":"<img src='../standard/APPL.ICO.gif'> API Gateway",
	"ObjectType":"Application",
	"ModificationDate":"2020/04/16 10:14:30",
	"text":"",
	"tags":"",
	"url":"D37AD7455CF088CA.htm"
	},
	],};

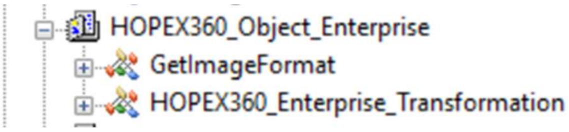
2.5. HTML Report Integration

2.5.1. Overview

To integrate HTML reports in HOPEX360, you must:

- Create a VB Macro.
- Call this VB Macro into a descriptor.

Below is an example of the "HOPEX360_Enterprise_Transformation" macro which is connected to the "HOPEX360_Object_Enterprise" descriptor.


HOPEX360_Object_Enterprise
GetImageFormat
HOPEX360_Enterprise_Transformation

2.5.2. Creating a macro

2.5.2.1. Creating a new macro

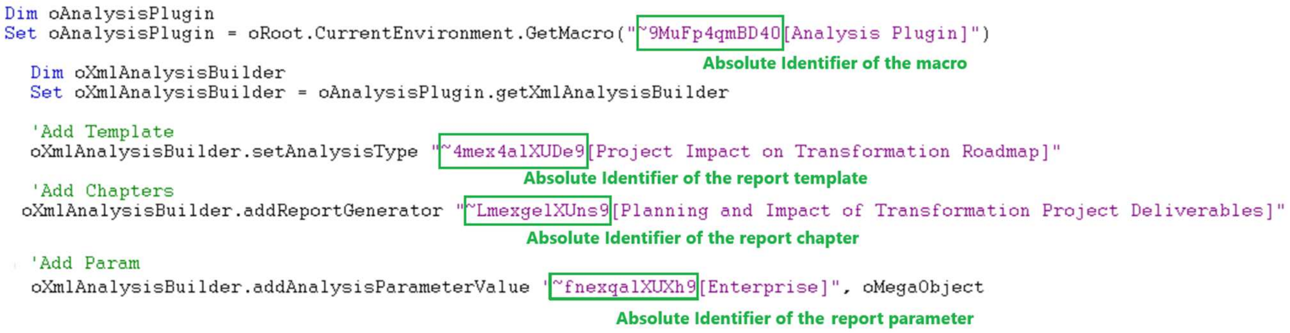
From the **View** menu > **Navigation** Windows > **Utilities** tab:

- Expand **Macros**.
- Right-click **MEGA**.
- Click **New** > **Macro**.
- Click **Create a (VB)Scipt Macro**.
- Rename it.

You can use the code of an existing HOPEX360 macro as a reference by copying and pasting it.

2.5.2.2. Updating the content of a macro

Below is an example of the code of the "HOPEX360_Enterprise_Transformation" macro.



```
Dim oAnalysisPlugin
Set oAnalysisPlugin = oRoot.CurrentEnvironment.GetMacro("~9MuFp4qmBD40[Analysis Plugin]")

   Dim oXmlAnalysisBuilder
   Set oXmlAnalysisBuilder = oAnalysisPlugin.getXmlAnalysisBuilder

   'Add Template
   oXmlAnalysisBuilder.setAnalysisType "~4mex4alXUDe9[Project Impact on Transformation Roadmap]"

   'Add Chapters
  oXmlAnalysisBuilder.addReportGenerator "~LmexgelXUns9[Planning and Impact of Transformation Project Deliverables]"

   'Add Param
   oXmlAnalysisBuilder.addAnalysisParameterValue '~fnexqalXUXh9[Enterprise]", oMegaObject
```

Absolute Identifier of the macro

Absolute Identifier of the report template

Absolute Identifier of the report chapter

Absolute Identifier of the report parameter

To update this code, you must change at least the following elements:

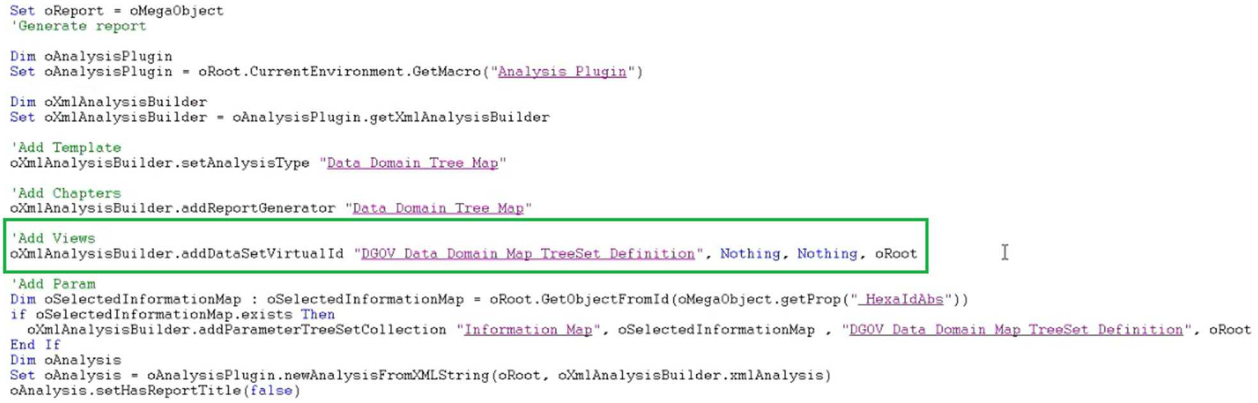
- Absolute Identifier of the Macro

- Absolute Identifier of the Report Template
- Absolute Identifier of the Chapter
- Absolute Identifier of the Report Parameter (or Views, depending on how the report template is built)

Tip: Use the “Show/Hide fields” button {} to display the Absolute Identifiers.

2.5.2.3. Views

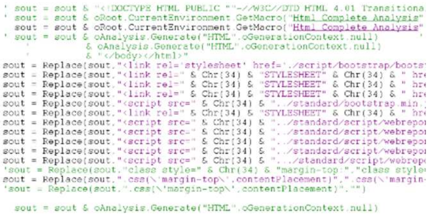
Some reports require to create views (Tree Maps for example). Below is an example of the appropriate code for reference.



```
Set oReport = oMegaObject
'Generate report

Dim oAnalysisPlugin
Set oAnalysisPlugin = oRoot.CurrentEnvironment.GetMacro("Analysis Plugin")

Dim oXmlAnalysisBuilder
Set oXmlAnalysisBuilder = oAnalysisPlugin.getXmlAnalysisBuilder

'Add Template
oXmlAnalysisBuilder.setAnalysisType "Data Domain Tree Map"

'Add Chapters
oXmlAnalysisBuilder.addReportGenerator "Data Domain Tree Map"

'Add Views
oXmlAnalysisBuilder.addDataSetVirtualId "DGOV Data Domain Map TreeSet Definition", Nothing, Nothing, oRoot

'Add Param
Dim oSelectedInformationMap : oSelectedInformationMap = oRoot.GetObjectFromId(oMegaObject.getProp("_HexaIdAbs"))
if oSelectedInformationMap.exists Then
  oXmlAnalysisBuilder.addParameterTreeSetCollection "Information Map", oSelectedInformationMap , "DGOV Data Domain Map TreeSet Definition", oRoot
End If
Dim oAnalysis
Set oAnalysis = oAnalysisPlugin.newAnalysisFromXMLString(oRoot, oXmlAnalysisBuilder.xmlAnalysis)
oAnalysis.setHasReportTitle(false)
```

2.5.2.4. Replacing conflicting code

To ensure compatibility with HOPEX360, it is recommended to replace any conflicting CSS and JavaScript code. Below is an example of the appropriate code for reference (that you can copy from an existing HOPEX360 macro).



```
' sout = sout & "<!DOCTYPE HTML PUBLIC ""-//W3C//DTD HTML 4.01 Transitiona
' sout = sout & oRoot.CurrentEnvironment.GetMacro("Html Complete Analysis"
  sout = sout & oRoot.CurrentEnvironment.GetMacro("Html Complete Analysis"
' sout = sout & oAnalysis.Generate("HTML",oGenerationContext,null)        '
   '            & oAnalysis.Generate("HTML",oGenerationContext,null)
   '            & "</body></html>"
sout = Replace(sout,"<link rel='stylesheet' href='./script/bootstrap/boots
sout = Replace(sout,"<link rel=" & Chr(34) & "STYLESHEET" & Chr(34) & " hre
sout = Replace(sout,"<link rel=" & Chr(34) & "STYLESHEET" & Chr(34) & " hre
sout = Replace(sout,"<link rel=" & Chr(34) & "STYLESHEET" & Chr(34) & " hre
sout = Replace(sout,"<script src=" & Chr(34) & "../standard/bootstrap.min.
sout = Replace(sout,"<link rel=" & Chr(34) & "STYLESHEET" & Chr(34) & " hre
sout = Replace(sout,"<script src=" & Chr(34) & "../standard/script/webrepo
sout = Replace(sout,"<script src=" & Chr(34) & "../standard/script/webrepo
sout = Replace(sout,"<script src=" & Chr(34) & "../standard/script/webrepo
sout = Replace(sout,"<script src=" & Chr(34) & "../standard/script/webrepo
sout = Replace(sout,"<script src=" & Chr(34) & ".../standard/script/webrep
'sout = Replace(sout,"class style=" & Chr(34) & "margin-top:","class style
sout = Replace(sout,".css(\'margin-top\',contentPlacement)",".css(\'margin-
'sout = Replace(sout,".css(\'margin-top\',contentPlacement)","")

  sout = sout & oAnalysis.Generate("HTML",oGenerationContext,null)
```

2.5.2.5. Limitations

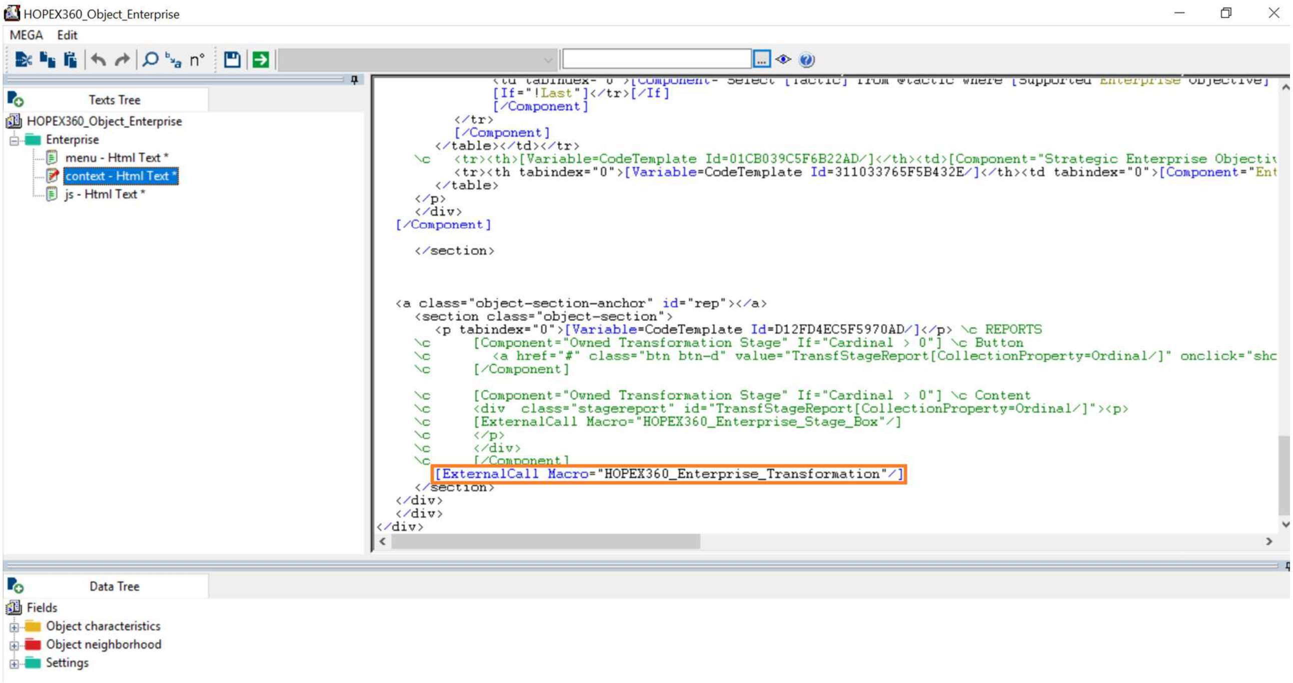
Certain HOPEX reports (e.g. Gant Charts) cannot be migrated to HOPEX360 due to their reliance on interactive features, which limits their portability.

2.5.3. Calling the macro into a descriptor

To call the macro into a descriptor:

- Open the adequate descriptor > Context section.
- Use the ExternalCall Macro and enter the name of the macro.

Below is an example of the “HOPEX360_Object_Enterprise” descriptor.



```
              <td tabindex="0">[Component="Select [Tactic] from @tactic where [Supported Enterprise Objective]
              [If="!Last"]</tr>[/If]
              [/Component]
          </tr>
          [/Component]
        </table></td></tr>
    \c    <tr><th>[Variable=CodeTemplate Id=01CB039C5F6B22AD/]</th><td>[Component="Strategic Enterprise Objecti
          <tr><th tabindex="0">[Variable=CodeTemplate Id=311033765F5B432E/]</th><td tabindex="0">[Component="Ent
        </table>
    </p>
    </div>
  [/Component]

    </section>



  <a class="object-section-anchor" id="rep"></a>
    <section class="object-section">
      <p tabindex="0">[Variable=CodeTemplate Id=D12FD4EC5F5970AD/]</p> \c REPORTS
    \c      [Component="Owned Transformation Stage" If="Cardinal > 0"] \c Button
    \c        <a href="#" class="btn btn-d" value="TransfStageReport[CollectionProperty=Ordinal/]" onclick="shc
    \c      [/Component]

    \c      [Component="Owned Transformation Stage" If="Cardinal > 0"] \c Content
    \c      <div  class="stagereport" id="TransfStageReport[CollectionProperty=Ordinal/]"><p>
    \c      [ExternalCall Macro="HOPEX360_Enterprise_Stage_Box"/]
    \c      </p>
    \c      </div>
    \c      [/Component]
        [ExternalCall Macro="HOPEX360_Enterprise_Transformation"/]
    </section>
  </div>
  </div>
</div>
```

3. CIO Dashboard

3.1. Implementation

Apart from the Application Obsolescence (HOPEX360_CIO_Dashboard_V2_Application_Obsolescence) , the CIO Dashboards are implemented as follows:

- For the configuration
  - A HOPEX descriptor is built using standard HOPEX technology but calls the following item.
  - a JS Macro within HOPEX, this is where the translation and the customization need to take place
- During the generation, there are 2 steps:
  - the descriptor generates the html pages that has a reference to a JSON file.
  - the aforementioned JSON file is generated by the JS Macro and contains the relevant data to display the dashboard.

Here is a list of the key components underneath the dashboards:

Dashboard	HOPEX Descriptor	Generated html page	JS Macro	Generated JSON (located in JSON folder)
Applications	HOPEX360_CIO Dashboard_V2_Applications	C6D1CA84602F44D1.htm	Applications Dashboard	C6D1CCBB602F7983_dashboard.JSON
Technologies	HOPEX360_CIO Dashboard_V2_Technologies	C6D1CAA2602F4568;htm	Technologies Dashboard	C6D1CCF7602F79D9_dashboard.JSON
BCM	HOPEX360_CIO Dashboard_V2_Application_Business Rationalisation_Box-in-Box	C6D1CB10602F460F.htm	Application Business Rationalisation Dashboard	C6D1CD0A602F7A02_dashboard.JSON
FM	HOPEX360_CIO Dashboard_V2_Application_Functional Rationalisation_Box-in-Box	C6D1CB2B602F46B6.htm	Application Functional Rationalisation Dashboard	C6D1CD14602F7A2B_dashboard.JSON
TFM	HOPEX360_CIO Dashboard_V2_Technology_Functional Rationalisation_Box-in-Box	C6D1CB43602F474B.htm	Technology Functional Rationalisation Dashboard	C6D1CD23602F7A54_dashboard.JSON

3.2. Application

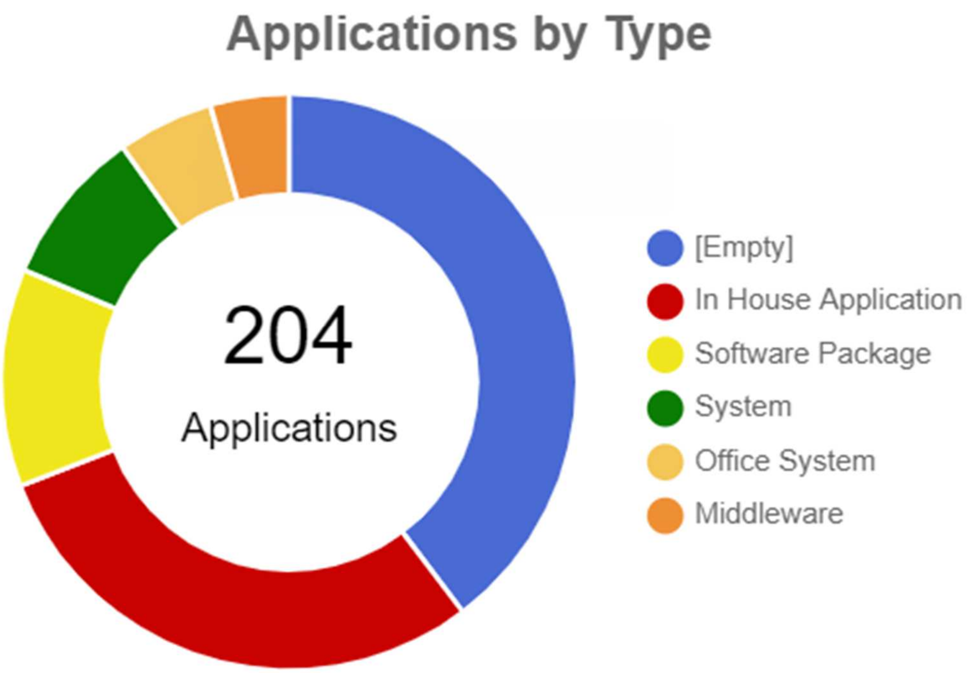
3.2.1. Filters

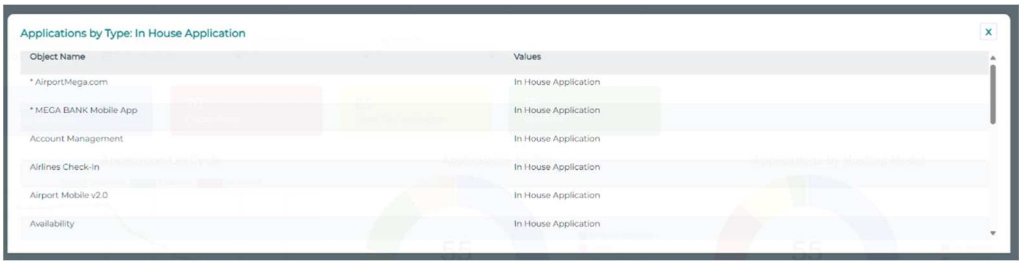
All the widgets shown on this page are filtered by 4 options:

- **Current Date**: this allows to analyse how the dashboard looks like over time
- **Lifecycle State**: this allows to only display the Applications in a particular state (Production, Preparation, Retirement)
- **Support State**: this allows to only display the Applications based upon their support state (General support, Extended support, no longer supported)
- **By Portfolio**: this allows to filter the Applications by Portfolio

3.2.2. Drill down

On click, all the widgets can be drilled down. A detailed table will pop up over the page, as shown below.



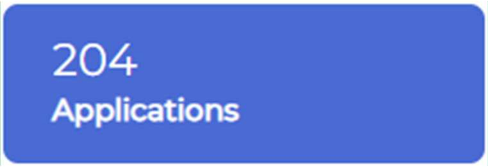


Applications by Type: In House Application

Object Name	Values
* AirportMega.com	In House Application
* MEGA BANK Mobile App	In House Application
Account Management	In House Application
Airlines Check-In	In House Application
Airport Mobile v2.0	In House Application
Availability	In House Application

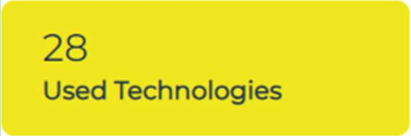
3.2.3. Indicators

- **Applications**

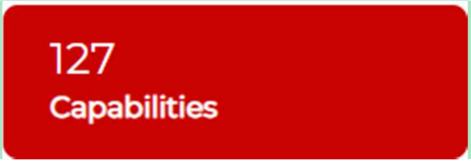
	Key information
Purpose	This widget shows the number of Applications in the selected Portfolio
Data structure	- (optional) Portfolio - Application
Example	 204 Applications

- **Obsolete Applications**

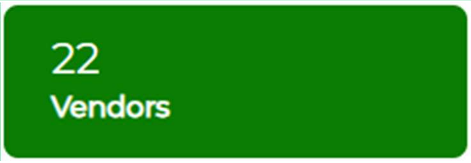
	Key information

Purpose	This widget shows the number of Technologies used by the Applications of the selected Portfolio. This is the list of Technologies used by at least one Application of the Portfolio.
Data structure	- (optional) Portfolio - Application - Technology
Example	 28 Used Technologies

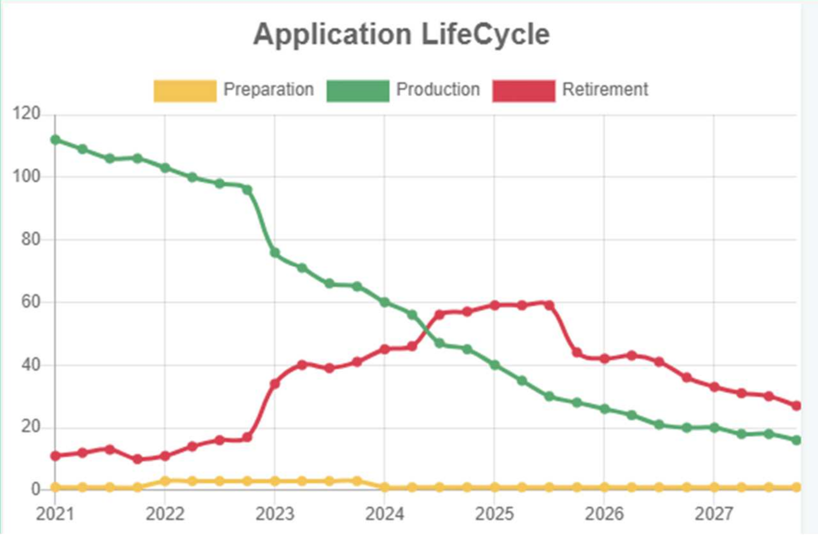
- **Capabilities**

	Key information
Purpose	This widget shows the number of Capabilities supported by the Applications of the selected Portfolio. This is the list of Capabilities linked to at least one Application of the Portfolio.
Data structure	- (Optional) Portfolio - Application - Business Capability
Example	 127 Capabilities

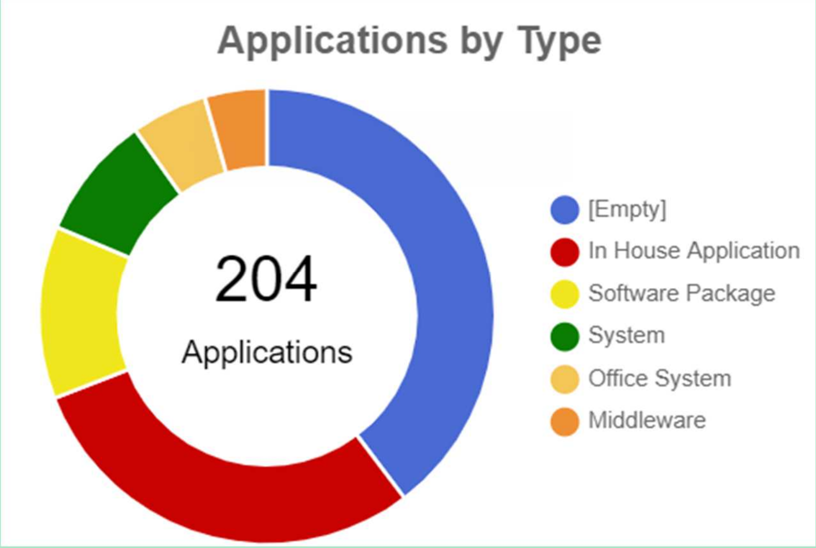
- **Vendors**

	Key information
Purpose	This widget shows the number of Vendors providing Technologies supporting the Applications of the selected Portfolio. This is the list of Org Units typed as Vendor linked to at least one Technology supporting at an Application of the Portfolio.
Data structure	- (Optional) Portfolio - Application - Technology - Vendor
Example	 22 Vendors

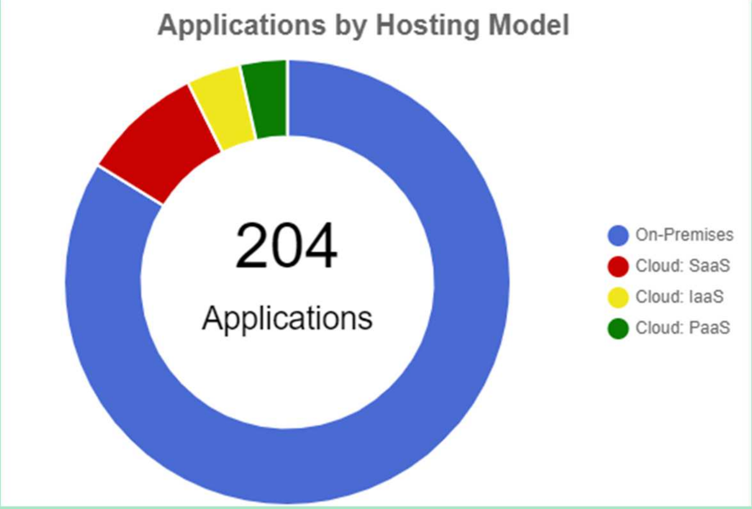
3.2.4. Application LifeCycle

	Key information
Purpose	This shows the number of Applications by state over time for the selected perimeter(s)
Data structure	- (optional) Portfolio - Application - Object Life - Time Periods
Example	

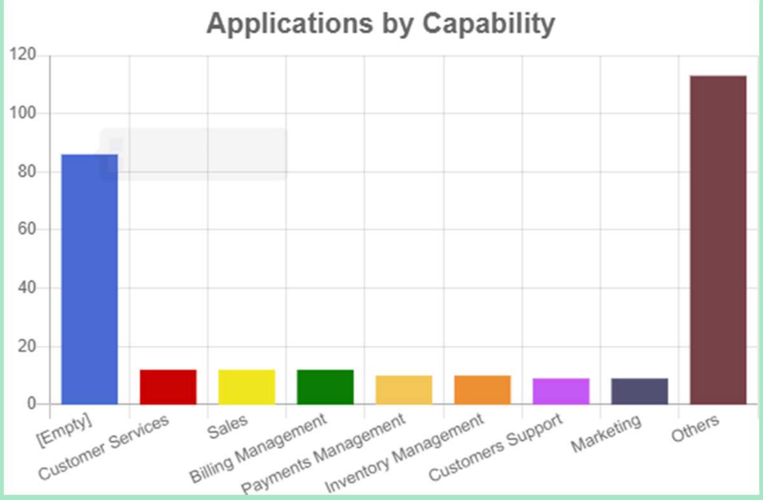
3.2.5. Application By Type

	Key information
Purpose	This shows the number of Applications by Type of Application
Data structure	- (optional) Portfolio - Application - Application Type (System, In House, Software Package..)
Example	

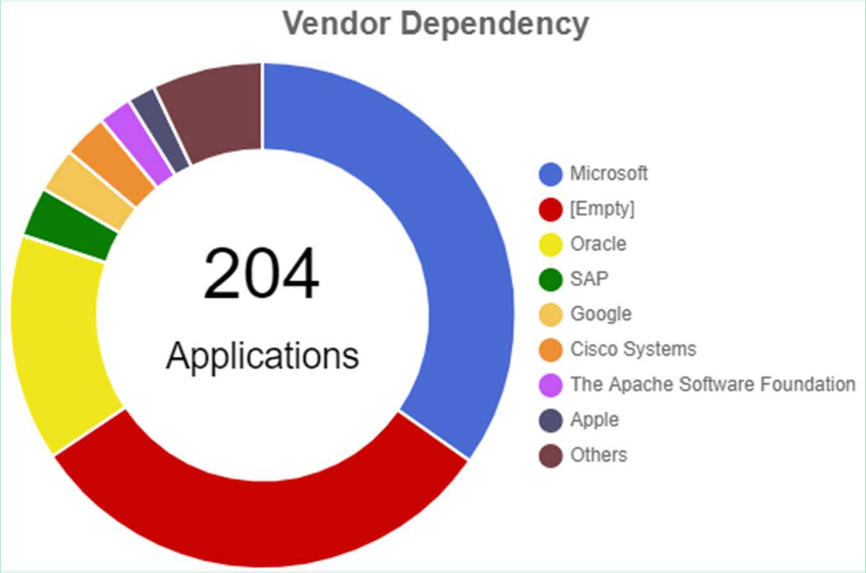
3.2.6. Application By Hosting Model

	Key information
Purpose	This shows the number of Applications by Hosting Model of Application
Data structure	- (optional) Portfolio - Application - Application Hosting Model (On Premises, Saas, Iaas, Paas
Example	
Example	

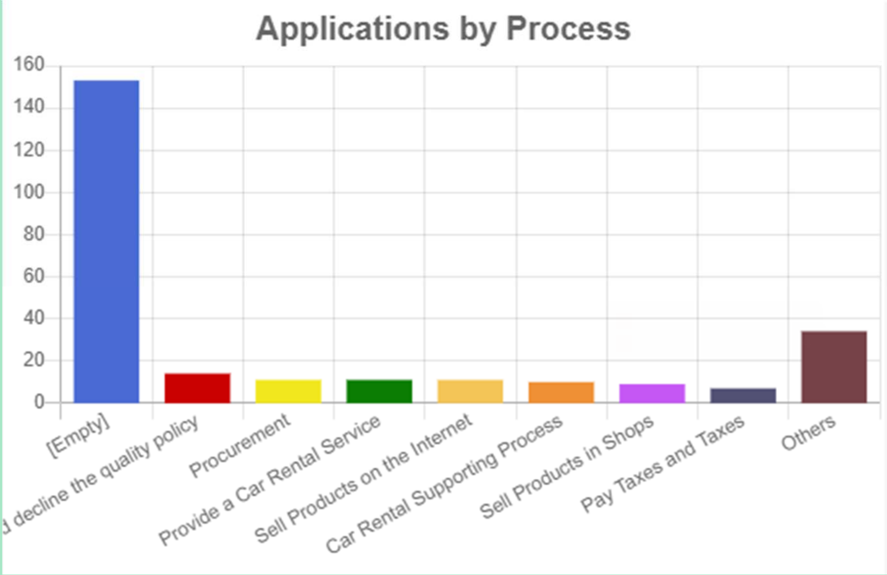
3.2.7. Application By Capability

	Key information
Purpose	This shows the number of Applications by Fulfilled Capabilities
Data structure	- (optional) Portfolio - Application - Capability
Example	

3.2.8. Vendor Dependency

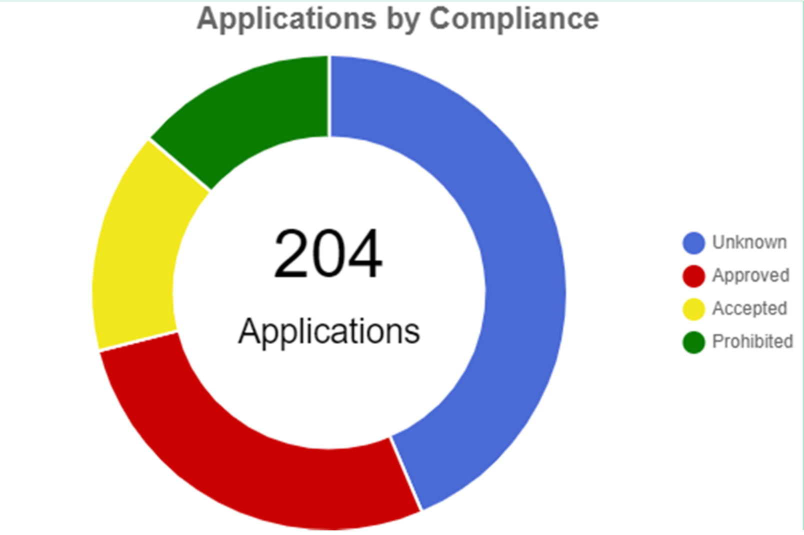
	Key information
Purpose	This shows the number of Applications by Vendor
Data structure	- (optional) Portfolio - Application - Technology to Org-Unit typed Vendor
Example	

3.2.9. Applications by Process

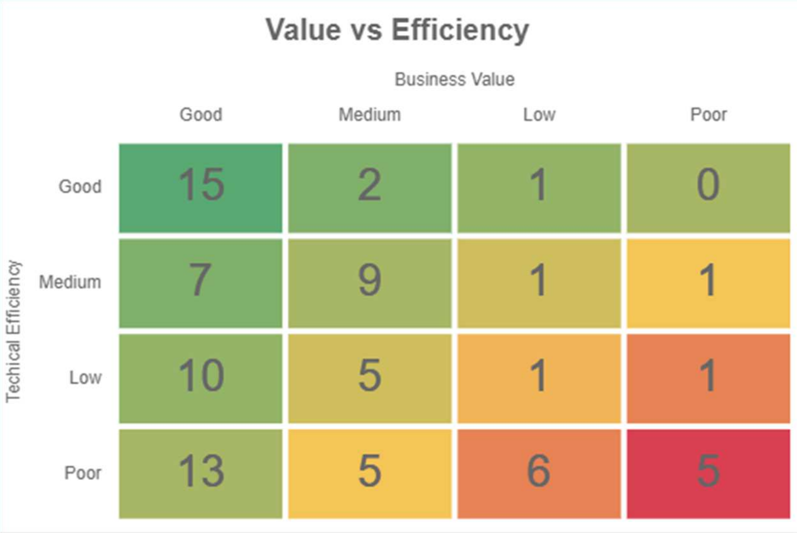
	Key information
Purpose	This shows the number of Applications by Process Categories
Data structure	- (optional) Portfolio - Application - Process Category
Example	

3.2.10. Application Compliance

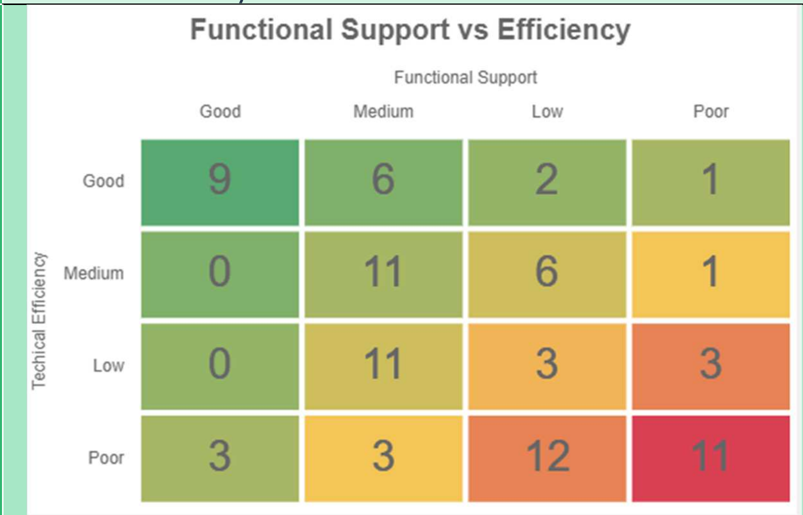
	Key information

Purpose	This shows the number of Technologies supporting Applications of the selected perimeter(s) broken down by Company Standard
Data structure	- (optional) Portfolio - Application - Technology – Company Standard
Example	

3.2.11. Value vs. Efficiency

	Key information
Purpose	This shows a clickable Heatmap of Applications highlighting their Value and Efficiency
Data structure	- (optional) Portfolio - Application - Value - Efficiency
Example	

3.2.12. Functional Support vs. Efficiency

<table>
<tr><th></th><th>Key information</th></tr>
<tr><td><b>Purpose</b></td><td>This shows a clickable Heatmap of Applications highlighting their Functional Support and Efficiency</td></tr>
<tr><td><b>Data structure</b></td><td>- (optional) Portfolio
- Application
- Functional Support
- Efficiency</td></tr>
<tr><td><b>Example</b></td><td>
Functional Support vs Efficiency
<table>
<tr><th></th><th></th><th colspan="4">Functional Support</th></tr>
<tr><th></th><th></th><th>Good</th><th>Medium</th><th>Low</th><th>Poor</th></tr>
<tr><td>Techical Efficiency</td><td>Good</td><td>9</td><td>6</td><td>2</td><td>1</td></tr>
<tr><td></td><td>Medium</td><td>0</td><td>11</td><td>6</td><td>1</td></tr>
<tr><td></td><td>Low</td><td>0</td><td>11</td><td>3</td><td>3</td></tr>
<tr><td></td><td>Poor</td><td>3</td><td>3</td><td>12</td><td>11</td></tr>
</table></td></tr>
</table>

3.3. Technologies

3.3.1. Filters

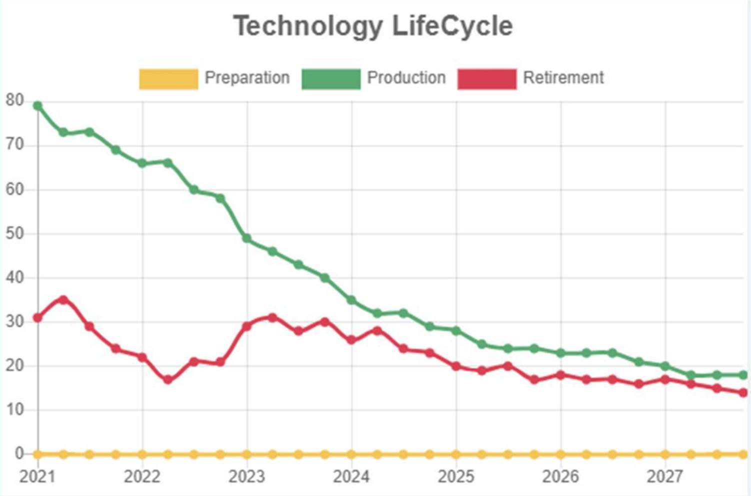
All the widgets shown on this page are filtered by 4 options:

- **Current Date**: this allows to analyse how the dashboard looks like over time
- **Lifecycle State**: this allows to only display the Technologies in a particular state (Production, Preparation, Retirement)
- **Support State**: this allows to only display the Technologies based upon their support state (General support, Extended support, no longer supported)
- **By Portfolio**: this allows to filter the Technologies by Technology Portfolio

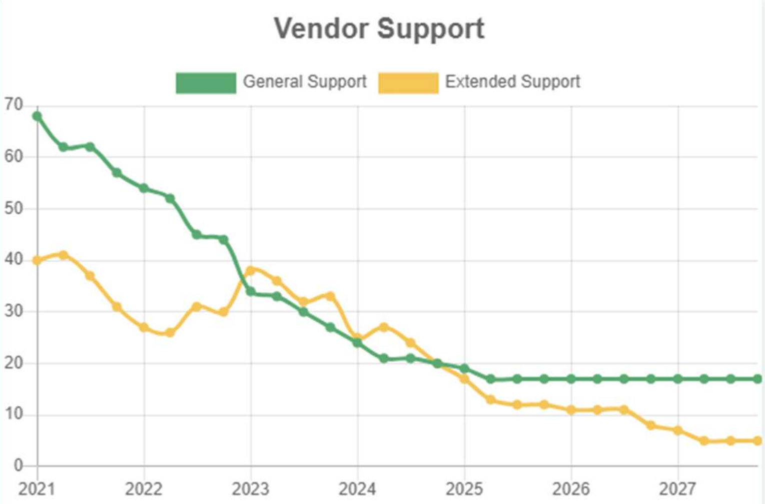
3.3.2. Drill down

On click, all the widgets can be drilled down. A detailed table will be pop up over the page, exactly as the Applications Dashboard.

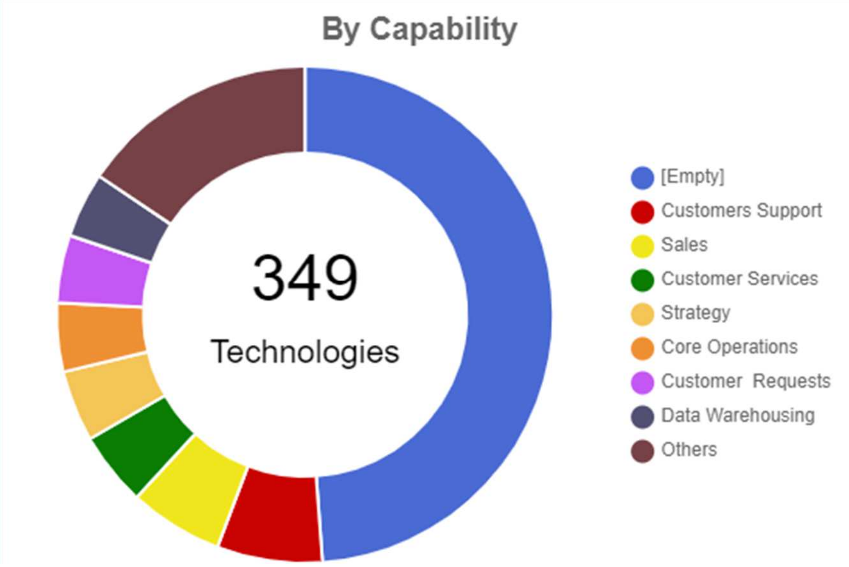
3.3.3. Technology LifeCycle

	Key information
Purpose	This shows the number of Technologies by state over time for the selected perimeter(s)
Data structure	- (optional) Portfolio - Technology - Time Period
Example	

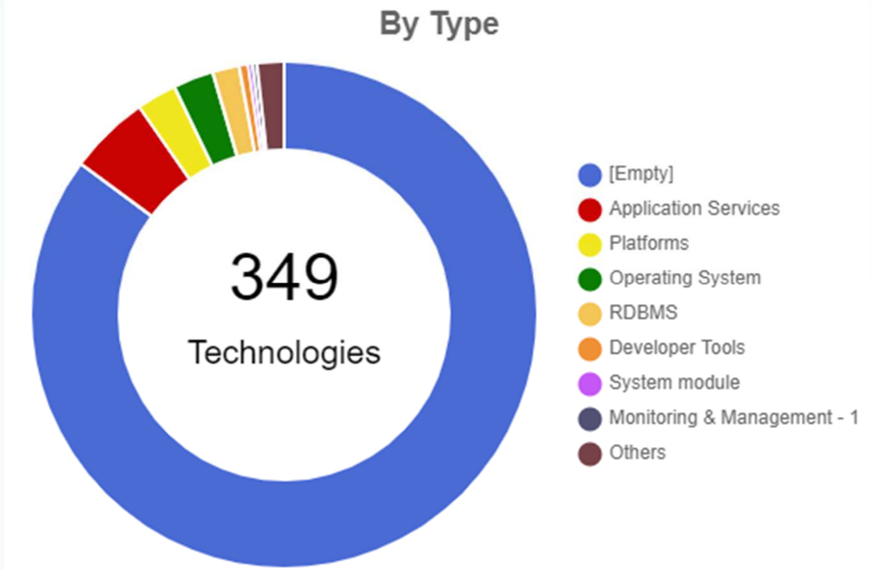
3.3.4. Vendor Support

	Key information
Purpose	This shows the Technologies Support State over time
Data structure	- (optional) Portfolio - Technology - Time Period and Support dates
Example	

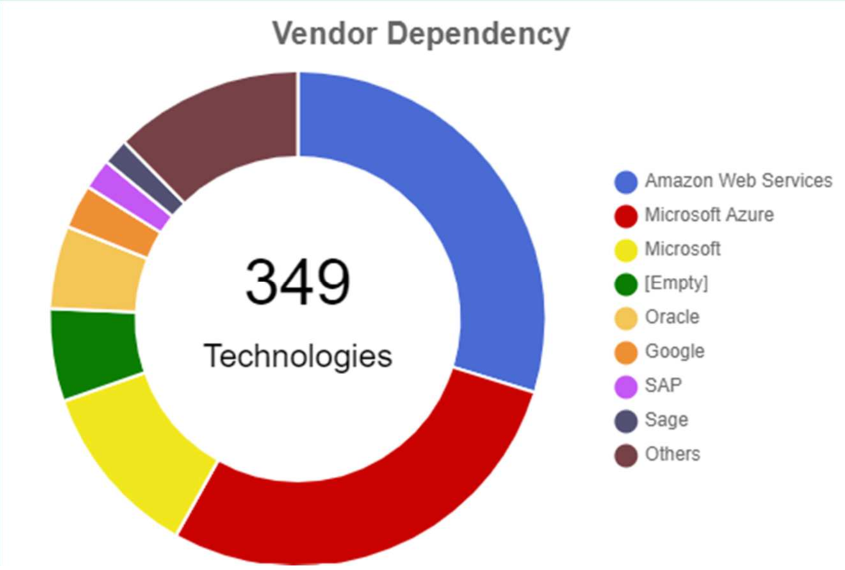
3.3.5. By Capability

	Key information
Purpose	This shows a pie chart of Technologies groups by supported Capabilities
Data structure	- (optional) Portfolio - Technology - Application - Business Capability
Example	

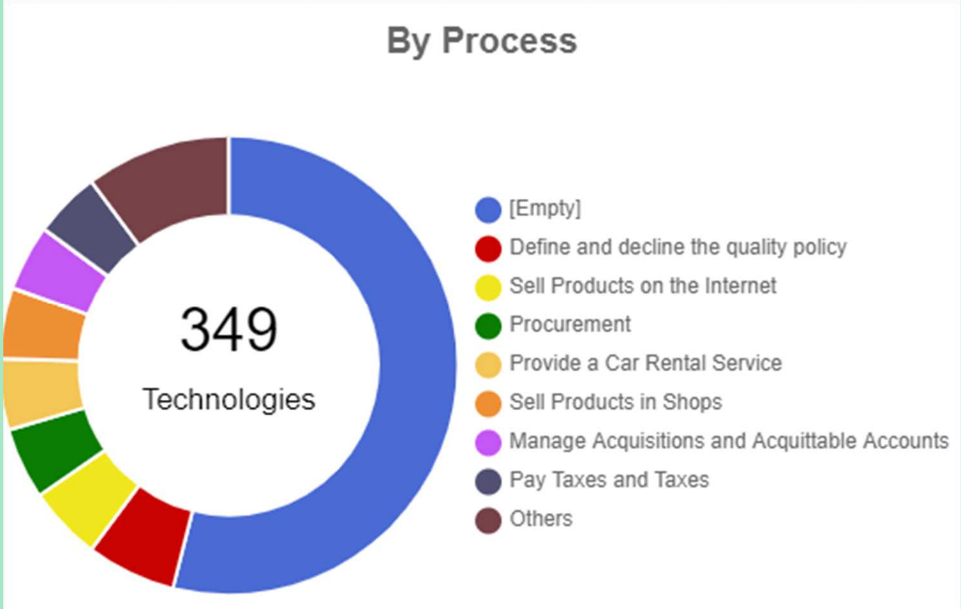
3.3.6. By Type

	Key information
Purpose	This shows a pie chart of Technologies grouped by Type
Data structure	- (optional) Portfolio - Software Technologies - Technology Type
Example	

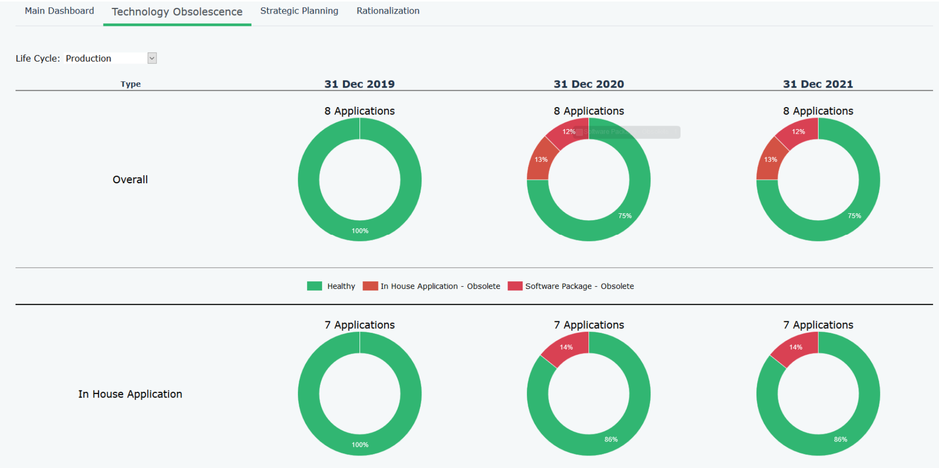
3.3.7. Vendor Dependendy

	Key information
Purpose	This shows the Technologies grouped by Vendor
Data structure	- (optional) Portfolio - Technology - Org Unit typed Vendor
Example	

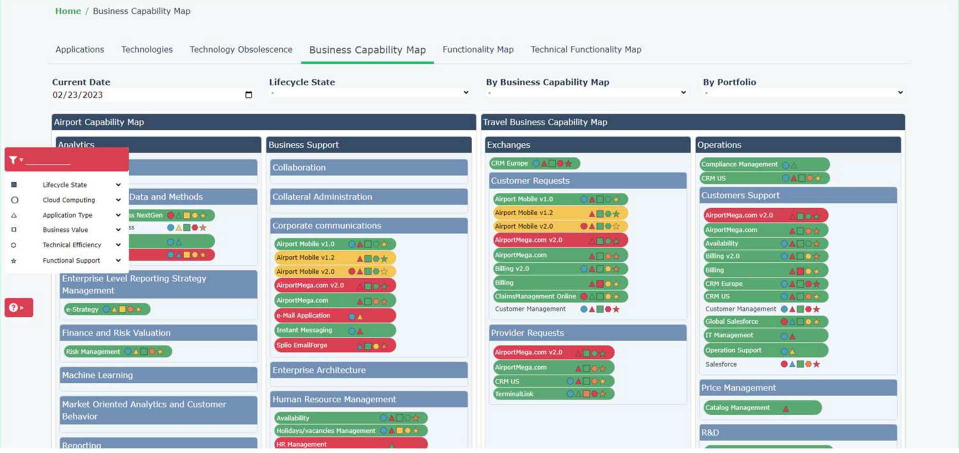
3.3.8. By Process

	Key information
Purpose	This shows a pie chart of Technologies grouped by Process they support via Application
Data structure	- (optional) Portfolio - Process Category - Application - Software Technology
Example	

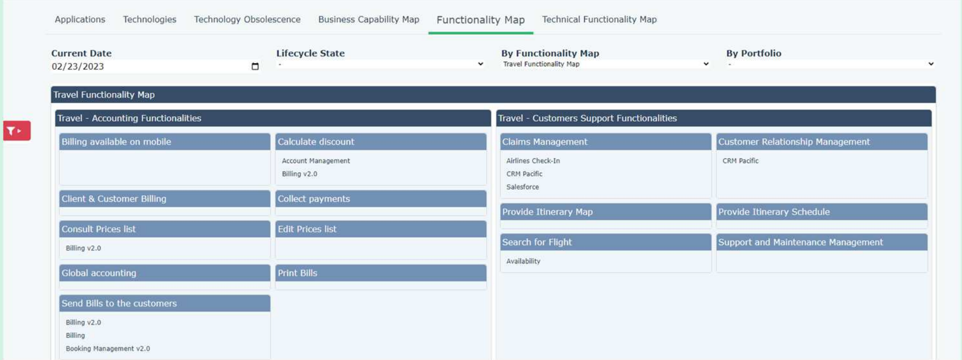
3.4. Technology Obsolescence

	Key information
Purpose	Shows the number of Applications by Technology Obsolescence, which
Filter	Lifecycle status
Source JSON file	None. The data is generated within the Hopex Descriptor
Data structure	- Application – Application Type - Object Lifecyle - Time Period - Technology – End of Support / End of Extended Support Logic: At a given point in time, if a Technology linked to a given Application is no longer supported, then this Application is obsolete
Example	 Main Dashboard Technology Obsolescence Strategic Planning Rationalization Life Cycle: Production Type 31 Dec 2019 31 Dec 2020 31 Dec 2021 Overall 8 Applications 100% 8 Applications 12% 13% 75% 8 Applications 12% 13% 75% 🟩 Healthy 🟥 In House Application - Obsolete 🟥 Software Package - Obsolete In House Application 7 Applications 100% 7 Applications 14% 86% 7 Applications 14% 86%

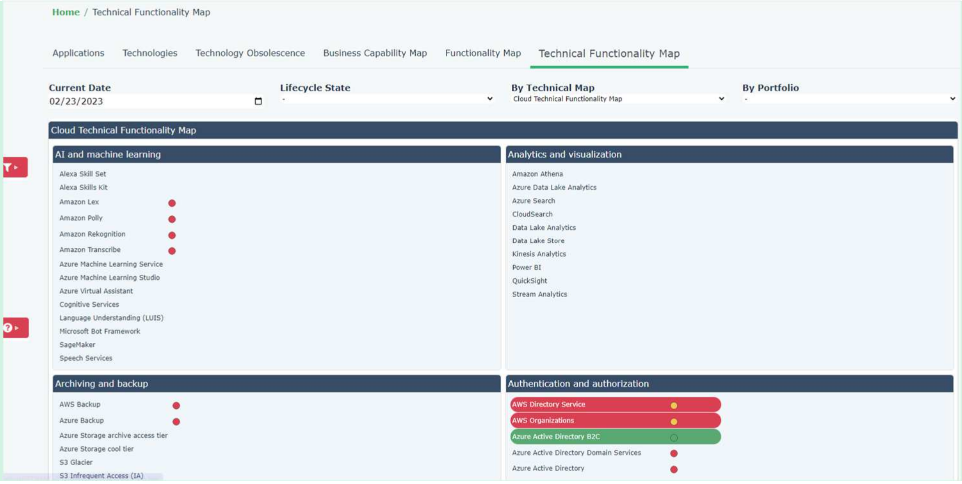
3.5. Business Capability Map

	Key information
Purpose	This is a graphical view of the Business Capability Map hierarchy and their supporting Applications
Filter	4 filters are available: - Current Date - Lifecyle State: to filter Applications - By Business Capability Map: Business Capability Map allows to switch from one to another - By Portfolio: to filter Applications - On the left-hand side, a toolbox allows to colour code the Applications based upon their characteristics
Data structure	- Business Capability Map - Business Capability and their Components to the 4th level
Example	

3.6. Functionality Map

	Key information
Purpose	This is a graphical view of the Functionality Map hierarchy and their supporting Applications.
Filter	4 filters are available: - Current Date - Lifecyle State: to filter Applications. - By Functionality Map: Functionality Map allows to switch from one to another. - By Portfolio: to filter Applications. On the left-hand side, a toolbox allows to colour code the Applications based upon their characteristics
Data structure	- Functionality Map - Functionality and their Components to the 4th level
Example	 Applications Technologies Technology Obsolescence Business Capability Map Functionality Map Technical Functionality Map Current Date 02/23/2023 Lifecycle State - By Functionality Map Travel Functionality Map By Portfolio - Travel Functionality Map Travel - Accounting Functionalities Billing available on mobile Calculate discount: Account Management, Billing v2.0 Client & Customer Billing Collect payments Consult Prices list: Billing v2.0 Edit Prices list Global accounting Print Bills Send Bills to the customers: Billing v2.0, Billing, Booking Management v2.0 Travel - Customers Support Functionalities Claims Management: Airlines Check-In, CRM Pacific, Salesforce Customer Relationship Management: CRM Pacific Provide Itinerary Map Provide Itinerary Schedule Search for Flight: Availability Support and Maintenance Management

3.7. Technical Functionality Map

	Key information
Purpose	This is a graphical view of the Technical Functionality Map hierarchy and their supporting Technologies
Filter	4 filters are available: - Current Date - Lifecyle State: to filter Software Technologies. - By Technical Functionality Map: Technical Functionality Map allows to switch from one to another. - By Portfolio: to filter Software Technologies. On the left-hand side, a toolbox allows to colour code the Software Technologies based upon their characteristics
Data structure	- Technical Functionality Map - Technical Functionality and their Components to the 4th level
Example	 Home / Technical Functionality Map Applications Technologies Technology Obsolescence Business Capability Map Functionality Map Technical Functionality Map Current Date: 02/23/2023 Lifecycle State: - By Technical Map: Cloud Technical Functionality Map By Portfolio: - Cloud Technical Functionality Map AI and machine learning Alexa Skill Set Alexa Skills Kit Amazon Lex ● Amazon Polly ● Amazon Rekognition ● Amazon Transcribe ● Azure Machine Learning Service Azure Machine Learning Studio Azure Virtual Assistant Cognitive Services Language Understanding (LUIS) Microsoft Bot Framework SageMaker Speech Services Analytics and visualization Amazon Athena Azure Data Lake Analytics Azure Search CloudSearch Data Lake Analytics Data Lake Store Kinesis Analytics Power BI QuickSight Stream Analytics Archiving and backup AWS Backup ● Azure Backup ● Azure Storage archive access tier Azure Storage cool tier S3 Glacier S3 Infrequent Access (IA) Authentication and authorization AWS Directory Service ● AWS Organizations ● Azure Active Directory B2C ○ Azure Active Directory Domain Services ● Azure Active Directory ●

4. Indexes Details

4.1. Tree lists

4.1.1. Overview

A Tree List displays a hierarchical structure through a table. The end-user can filter, expand / fold and search.

Only two lists use this rendering:

- Business Capability Maps / Business Capability
- Process Category

4.1.2. Structure

These pages, which implement jQuery, use the DevExpress JS component.For more information, see: https://js.devexpress.com/Demos/WidgetsGallery/Demo/TreeList/Overview/jQuery/Light/.

The data is generated from a HOPEX post generation script and is stored under the **<website folder>/json/** folder.

4.1.3. Capabilities Tree list

- **Rendering**

The Capabilities Tree list shows the following information:

Column	Key information
Business Capability	Business Capabilities hierarchy starting from the Business Capability Maps, through the Business Capability Composition
Performance	Performance assessment of relevant Business Capability
Execution	Execution assessment of relevant Business Capability
Supporting Apps	List of applications linked to the Business Capability as Realisers
Modification Date	Last modification date of the Business Capability

- **Customisation**

This page uses the DevEpress JS library. For more information, see: https://js.devexpress.com/Demos/WidgetsGallery/Demo/TreeList/Overview/jQuery/Light/.

This page relies on two components:

- The HOPEX Descriptor : HOPEX EA_Index Business Capa Tree List. This contains the HTML and the Javascript part
- A post generation script "HOPEX EA_Post Generation Script Process and Capa Trees JSON.Macro". This generates a JSON file containing the data: ../json/DatasArrayBCMHierarchy.js

The structure of this file is as follows:

```
var products = [
{
text: "Business Capability Name",
modification_date:"2020/04/14 08:06:35",
performance:"",
execution:"",
link: "F034F0705CF0502C.htm",
id: "F034F0705CF0502C",
parent:"F03JKSLMP5CF0502C ",
applications: [
{text:"Application Name"}
]
},
]
```

4.1.4. Processes Tree list

- **Rendering**

The Processes Tree list shows the following information

Column	Key information
Process Category	Process Category hierarchy, through the Owner Process Category link
Performance	Performance assessment of relevant Process Category
Execution	Execution assessment of relevant Process Category
Supporting Apps	List of applications linked to the Process Category
Modification Date	Last modification date of the Process Category

- **Customisation**

This page uses the DevEpress JS library. For more details : https://js.devexpress.com/Demos/WidgetsGallery/Demo/TreeList/Overview/jQuery/Light/

This page relies on two components:

- The HOPEX Descriptor : HOPEX EA_Index Process Category Tree List. This contains the HTML and thee Javascript part
- A post generation script "HOPEX EA_Post Generation Script Process and Capa Trees JSON.Macro". This generates a JSON file containing the data: ../json/DatasArrayBCMHierarchy.js

The structure of this file is as follows:

```
var products = [
{
text: "Process Category Name",
modification_date:"2020/04/14 08:06:35",
performance:"",
execution:"",
link: "F034F0705CF0502C.htm",
id: "F034F0705CF0502C",
parent:"F03JKSLMP5CF0502C ",
applications: [
{text:"Application Name"}
]
},
]
```

4.2. Lists

4.2.1. Overview

These are flat list of items, showing key attributes or associations.

4.2.2. Structure

The data is being generated from the relevant HOPEX Descriptor.

The rendering is managed by a single HOPEX Descriptor for all lists (HOPEX EA_General_DataTable_JS) and includes:

- Filtering
- Excel export
- Pagination

More information on the “Datatable” java script library: https://datatables.net/

4.2.3. Enterprises list

Column	Key information
Business Capability	Name of the Enterprise
Comment	Comment of the Enterprise
Transformation phase	Transformation Stage defined against the Enterprise

4.2.4. Capabilities list

Column	Key information
Business Capability	Name of the Business Capability
Comment	Comment of the Business Capability
Parent	Parent Business Capability
Components	List of the Business Capability Components

Supporting Applications	List of the supporting Applications (realizer)

4.2.5. Processes list

Column	Key information
Process Category	Name of the Process Category
Code	Comment of the Process Category
Owner	Person defined as the Process Category Owner
Parent	Parent Business Capability
Direct Sub-Process	List of the Business Capability Components
Status	Current workflow status of the Process Category
Release	Modification date

4.2.6. Concepts list

Column	Key information
Name	Name of the Concept
Subject Area	Business Dictionary containing the Concept
Comment	Comment of the Concept
Storage Implementation	Business Information Realizing the Concept
Implementation	Embedding Business Work Product with their sending and receiving Applications

4.2.7. Business Information Map list

Column	Key information
Name	Name of the Business Information Map
Comment	Comment of the Business Information Map

4.2.8. Application portfolios list

Column	Key information
Portfolio	Name of the Portfolio typed as Application Portfolio
Parent Portfolio	Parent Portfolio
Sub Portfolio	List of the Sub Portfolios
Applications	List of the Applications listed in the Portfolio

4.2.9. Applications list

Column	Key information
Application	Name of the Application

Code	Code of the Application
Version	Version of the Application
Current State	Current workflow status of the Application
Business Capability	List of Business Capabilities supported by the Application
Technology	List of Technologies supporting the Application

4.2.10. Projects list

Column	**Key information**
Project	Name of the Project
Project Domain	Project Domain the Project belongs to
Project Deliverable	List of the Project Deliverable : Application and Project Impact

4.2.11. Technology Portfolios list

Column	**Key information**
Portfolio	Name of the Portfolio typed as Technology Portfolio
Parent Portfolio	Parent Portfolio
Sub Portfolio	List of the Sub Portfolios
Technology	List of Technologies listed in the Portfolio

4.2.12. Technologies list

Column	**Key information**
Technology	Name of the Technology
Vendor	Org Unit typed as Vendor providing the Technology
Technology Compliance	Company Standard of the Technology
End of Support	End of the Support of the Technology
End of Extended Support	End of the Extended Support of the Technology

4.2.13. Vendors list

Column	**Key information**
Vendor	Name of the Org unit typed as Vendor
Software Technology	List of the Technologies provided by the Vendor

5. Object pages details

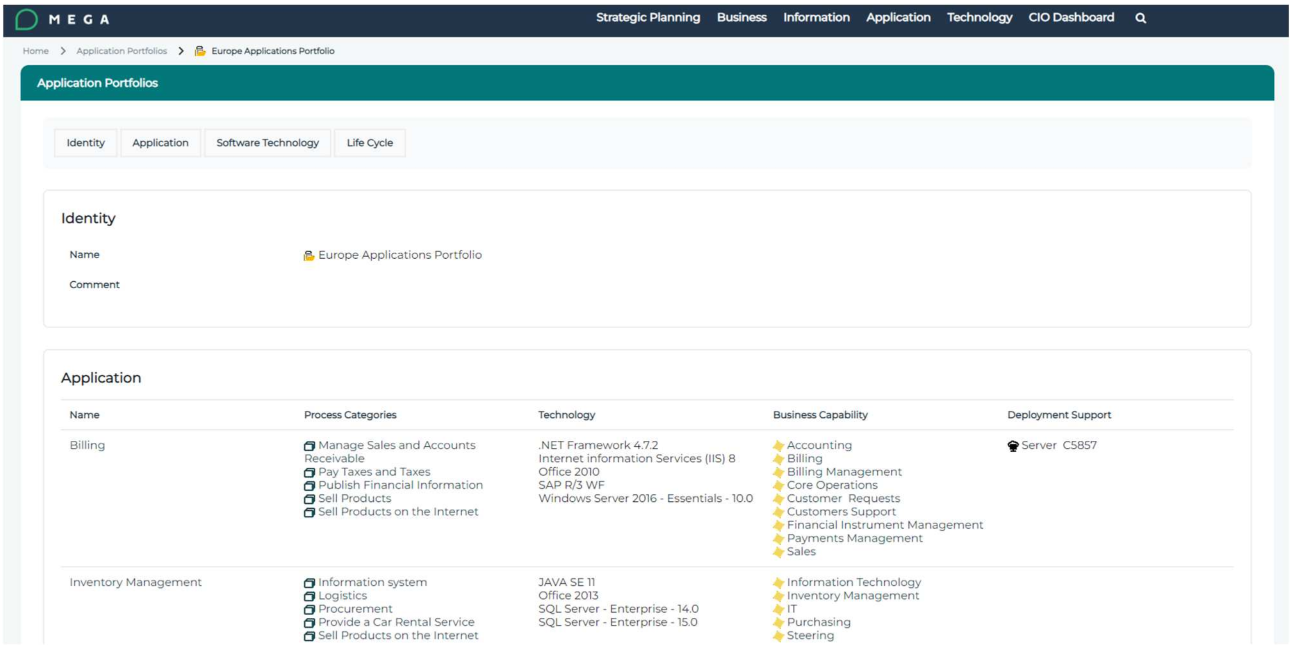
5.1. Overview

An Object page is broken down into:

- A breadcrumb on top, showing the hierarchical structure
- A sections menu, allowing quick access to relevant data:

  The sections:

  - Identity: this section displays key attributes
  - Specific sections showing key links around a given object



Identity

Name	Europe Applications Portfolio
Comment	

Application

Name	Process Categories	Technology	Business Capability	Deployment Support
Billing	Manage Sales and Accounts Receivable Pay Taxes and Taxes Publish Financial Information Sell Products Sell Products on the Internet	.NET Framework 4.7.2 Internet information Services (IIS) 8 Office 2010 SAP R/3 WF Windows Server 2016 - Essentials - 10.0	Accounting Billing Billing Management Core Operations Customer Requests Customers Support Financial Instrument Management Payments Management Sales	Server C5857
Inventory Management	Information system Logistics Procurement Provide a Car Rental Service Sell Products on the Internet	JAVA SE 11 Office 2013 SQL Server - Enterprise - 14.0 SQL Server - Enterprise - 15.0	Information Technology Inventory Management IT Purchasing Steering	

5.2. Business Information Map

Section	Key information
Identity	Name Vision Mission Table showing the Theme Goal and Strategy Business Capability Map
Diagram	Enterprise Diagram
Transformation Stage	One sub tab per Transformation Stage showing - Name - Comment - Goal / Objective and Tactic - Projects
Reports	- TreeMap - Box in Box - Chord

5.3. Business Capability Map

Section	Key information
Identity	Name Comment
Diagram	Describing diagram

5.4. Business Capability

Section	Key information
Identity	Name Release Date Validation State Comment
Diagram	Describing diagram
Owner Business Capability / Map	Parent BCM or Parent Business Capability
Functionality	Functionality attached to the Business Capability
Application System	Application system supporting the Business Capability
Application	Application supporting the Business Capability

5.5. Process Category

Section	Key information
Identity	Name Code Parent (link to the Parent Process Category) Comment Status Release Date
Diagram	Describing diagram
Owner	Person linked to Process Category
Sub-process	Component of the Process Category
Application	Application supporting the Process Category

5.6. Process

Section	Key information
Identity	Name Parent (link to the Parent Process) Comment Status Modification Date
Diagram	Describing diagram

5.7. Concept

Section	Key information

Identity	Name Comment
Store	Application store recording the Concept
Application Flow	Application flow carrying the Concept through Business Work Product

5.8. Business Information Map

Section	**Key information**
Identity	Name Comment
Reports	- TreeMap - Box in Box - Chord

5.9. Application portfolio

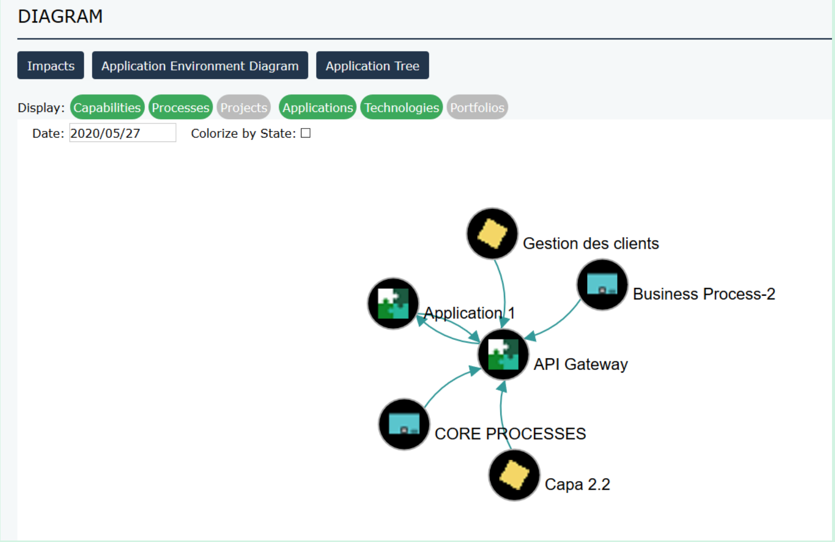
Section	**Key information**
Identity	Name Comment
Applications	Application grouped into the Portfolio
Lifecycle	Gantt chart of all Application lifecycles of the considered portfolio

5.10. Technology portfolio

Section	**Key information**
Identity	Name Comment
Technologies	Technologies grouped into the Portfolio
Lifecycle	Gantt chart of all Technologies lifecycles of the considered portfolio

5.11. Application

Section	**Key information**
Identity	Name Comment
Diagram	This section shows: - First the Impact Force diagram, which is auto generated. This relies on: o The JSON generated against each App o The use of a webserver such as IIS

	 DIAGRAM Impacts Application Environment Diagram Application Tree Display: Capabilities Processes Projects Applications Technologies Portfolios Gestion des clients Business Process-2 Application 1 API Gateway CORE PROCESSES Capa 2.2 Date: 2020/05/27 Colorize by State: □ - A tab per Diagram attached to the App
Owner	Responsibility Assignments and the relevant users
Business Capability	Business Capabilities supported by the Application
Process Category	Process Category supported by the Application
Project	Projects impacting the Application
Version	Versions of the Application, which means Applications linked via the Variations mechanism
Interface	List of sent and received application flows
Data Store	Data stores touched by the Application
Functionality	Functionalities offered by the application
Technologies	Technologies grouped into the Portfolio
Lifecycle	Gantt chart of all Technologies lifecycles of the considered portfolio. Standard HOPEX Report
Technology conflict	Gantt chart highlighting the conflicts between the lifecycles of the Application and its supporting Technologies

5.12. Project

Section	**Key information**
Identity	Name Comment
Deliverables	List of all deliverables provided by the Project and their Impact

5.13. Technology

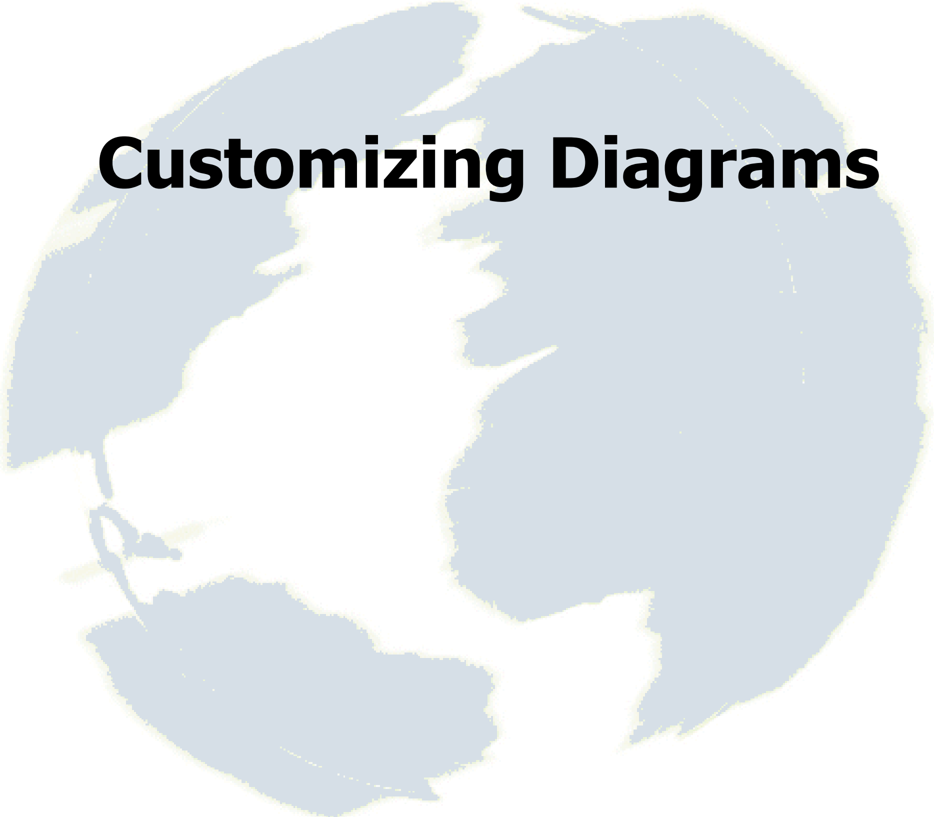
Section	**Key information**
Identity	Name Code Company Standard Vendor (link to the Org Unit typed Vendor) Type (link to the _Type) Comment
Owner	Person linked to the Technology

Lifecycle	Lifecycle of the Technology
Using Application	Applications supported by the Technology

5.14. Vendor

Section	**Key information**
Identity	Name Email address Comment
Technologies	List of all technologies provided by the Vendor

Customizing Diagrams



CREATING AND EDITING SHAPES

With the **Shapes Editor** tool you can create and modify shapes that can be used in diagrams.

> **The Shapes Editor is available in HOPEX Windows Front-End only.**

A shape is composed of drawing elements and is saved in an individual file. This file can be treated like any other file.

Shapes are used by reference: if you make changes to a given shape, the shape is changed automatically in all diagrams that use it. In the diagrams, each object type is represented by a shape, so each object type has the same shape in all diagrams. This ensures consistency of presentation.

With the graphical features of **HOPEX** you can customize the default shapes. You can also use these features during diagramming to improve legibility.

An example would be to highlight an org-unit by surrounding it with a border. Similarly, you can insert text at certain places to explain an important point, without this comment being assigned to a particular element.

☛ *For more details on graphical shapes handling, see the **HOPEX Common Features** guide.*

The following functionalities are common to all products in **HOPEX**:

- ✓ Shapes Editor and Shapes
- ✓ Shapes Used in Diagrams
- ✓ Positioning Objects in a Shape

Shapes Editor and Shapes

The **Shapes Editor** tool enables you to create and modify shapes that can be used in diagrams.

> ☛ *The **Shapes Editor** is available in **HOPEX Windows Front-End** only.*

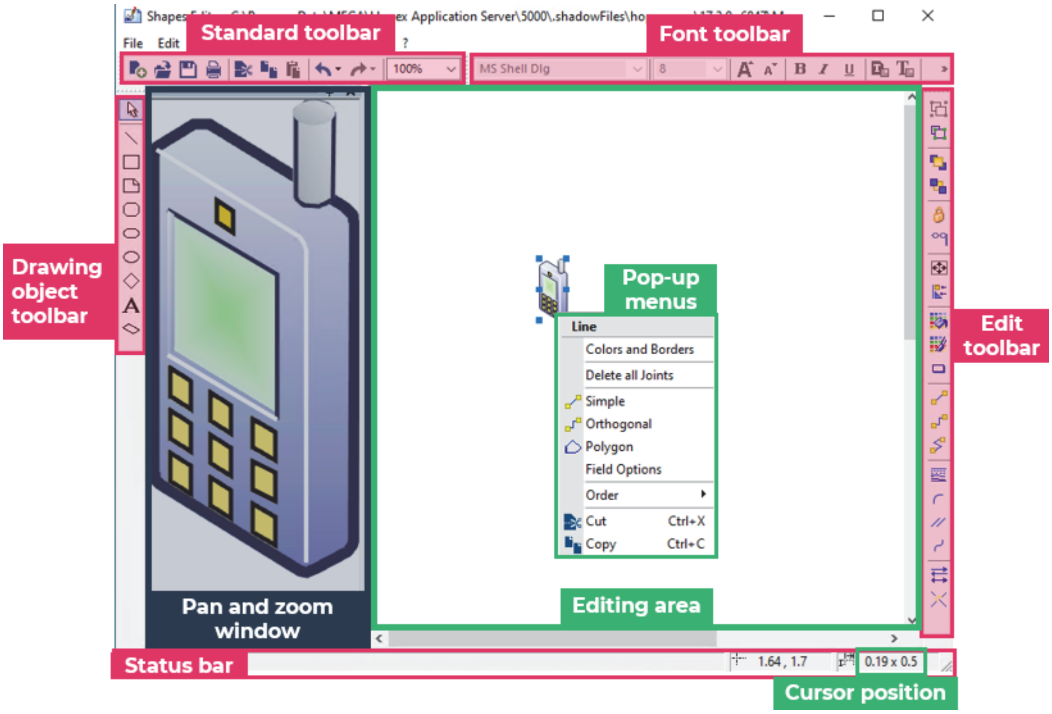
See:

- Opening the Shapes Editor
- Editing Shapes
- Exchanging files with other software

Opening the Shapes Editor

To open the **Shapes Editor**:

- In **HOPEX** menu bar, select **Tools > Shape Editor**.
  The **Shapes Editor** opens.



The **Shapes Editor** includes:

- an editing area, to edit shapes

  ☛ *To edit shapes, see Editing Shapes.*

- a pan and zoom window, which can be docked, fixed, enlarged, or hidden
- toolbars, which can be displayed or hidden

  ☛ *To display/hide a toolbar or status bar, see Managing Shapes Editor toolbars.*

- a status bar, which displays the horizontal and vertical position of the cursor in the editing area.

Managing Shapes Editor toolbars

Toolbars include buttons that enable to directly execute the menu commands.

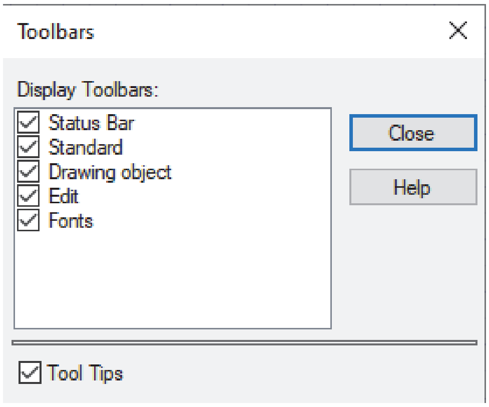
You can:

- display/hide each toolbar
- move a toolbar to another side
- hide tooltips

To display/hide toolbars and tooltips:

1. In the **Shapes Editor** menu bar, select **View > Toolbar**.

   ☛ *Alternatively, you can right-click a toolbar.*



Toolbars

Display Toolbars:

☑ Status Bar
☑ Standard
☑ Drawing object
☑ Edit
☑ Fonts

Close

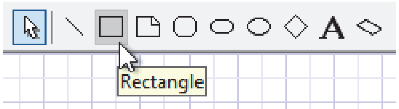
Help

☑ Tool Tips

2. Clear/Select the toolbars you want to hide/display.

To hide/display tooltips:

1. In the **Shapes Editor** menu bar, select **View > Toolbar**.
2. Clear/Select **Tool Tips** to hide/display the button name when you hover the cursor over the button.



To move a toolbar:

1. Click the toolbar start

2. You can:
   - drag and drop the toolbar to another toolbar area
   - move the toolbar in the same toolbar area

Editing Shapes

You can:

- create new shapes
- modify the standard shapes provided by **HOPEX**

To modify an existing shape:

1. In the **Shapes Editor** menu bar, select **File > Open**.
2. Select the shape.

To create a shape:

1. In the **Shapes Editor** menu bar, select **File > New**.
2. Define your page setup and click **OK**.

To modify drawing objects:

➧ Right-click the drawing object you want to modify.

> ☛ *Note that the commands in this menu are also available in the* ***Format*** *and* ***Drawing*** *menus.*

> ☛ *For more details on graphical shapes handling, see the* ***HOPEX Common Features*** *guide.*

Additional remarks on shapes

You can:

- run several instances of the shape editor at the same time, which enable you to access several shapes at the same time.
- call one shape from inside another.

Exchanging files with other software

You can transfer diagram drawings from one function to another by using **Cut**, **Copy** and **Paste**.

Exporting drawings

To export diagram drawings:

➧ In the **Shapes Editor** menu bar, select **File > Save As**.

Importing drawings

You can import drawings from applications operating under Windows, such as Paint or Photoshop.

Customized shapes should be saved in .mgs format (**HOPEX** proprietary format). The shape can then be loaded.

After import, you can only enlarge or reduce the diagram size. Distortion may occur when modifying the size of a diagram dot-for-dot (bitmap).

> ☛ *To modify certain non-vectorial elements contained in .MGS (proprietary format) shapes, it may be necessary to cut and paste to the graphic editor in which they were produced.*

See:

- Shapes Used in Diagrams
- Positioning Objects in a Shape

SHAPES USED IN DIAGRAMS

For shapes that correspond to objects in the repository, you can define formatting for the object name.

To define object name formatting:

1. Create a text field and enter the value "&Name&" for the name displayed in this diagram.
2. Place it in the foreground.
3. Indicate the format for the text in the **Graphical Options** dialog box (**Format > Colors and Borders** menu).

To associate parts of the shape with the object name:

1. Select the shape parts and the object name (hold [Ctrl] key).
2. In the **Edit** toolbar, click **Group** .
   In this case, the size of the selected shape parts is proportional to the name size and not to the overall object size.

> ☛ *Certain shapes display additional fields which depend on the type of object they represent.*

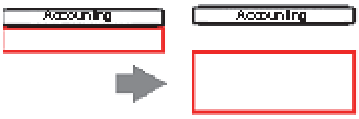
Tips on Using Shapes

Avoiding split shapes

To prevent drawing objects in a shape from drifting apart when you increase their size:

- Align the part that is not proportional to the name with the top left border.

When you increase the height of a shape, the space between the top of the drawing and the non-proportional objects increases with the overall size and the shape may drift apart.



Initial size

It is recommended that you check that the size specified for the shape will allow you to manipulate the shape without systematically distorting it.

Optimal distortion

Use a grid when defining shapes to ensure that any resizing is proportional. To work in finer detail, you can enlarge the shape overall, modify it and then reduce it overall.

Aligning links

As links are drawn from the center of shapes, their alignment on the grid is simplified when their size is a multiple of double the grid.

Optimizing performance

The display and printing speeds depend on the contents of the shapes:

- Shapes containing bitmaps and metafiles have low printing speeds.
- Thick lines take longer to display and greatly increase the size of the print file.
- Grouped elements slow down display of shapes. They should be reserved for the part that must be proportional to the name.
- Circles and rounded rectangles take longer to display than shapes with angles.

Reinitializing Shapes

You can reassign the graphic characteristics, specified in shape folder, to drawing objects and links .

To reassign the repository graphic characteristics:

- In the diagram menu, select **Drawing > Reinitialize Shapes**.

Example:

You have modified a shape font in the diagram. These modifications are lost when you perform a **Reinitialize Shapes**.

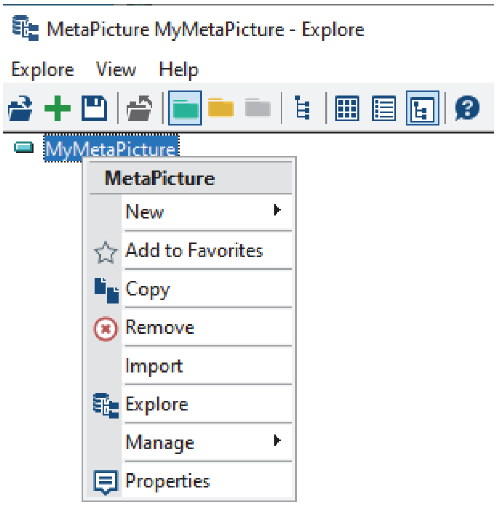
Modifying an Image Display Size

When needed you can make an image bigger than an icon size in a diagram shape.

Prerequisite: the image is a .jpg file.

To make an image bigger:

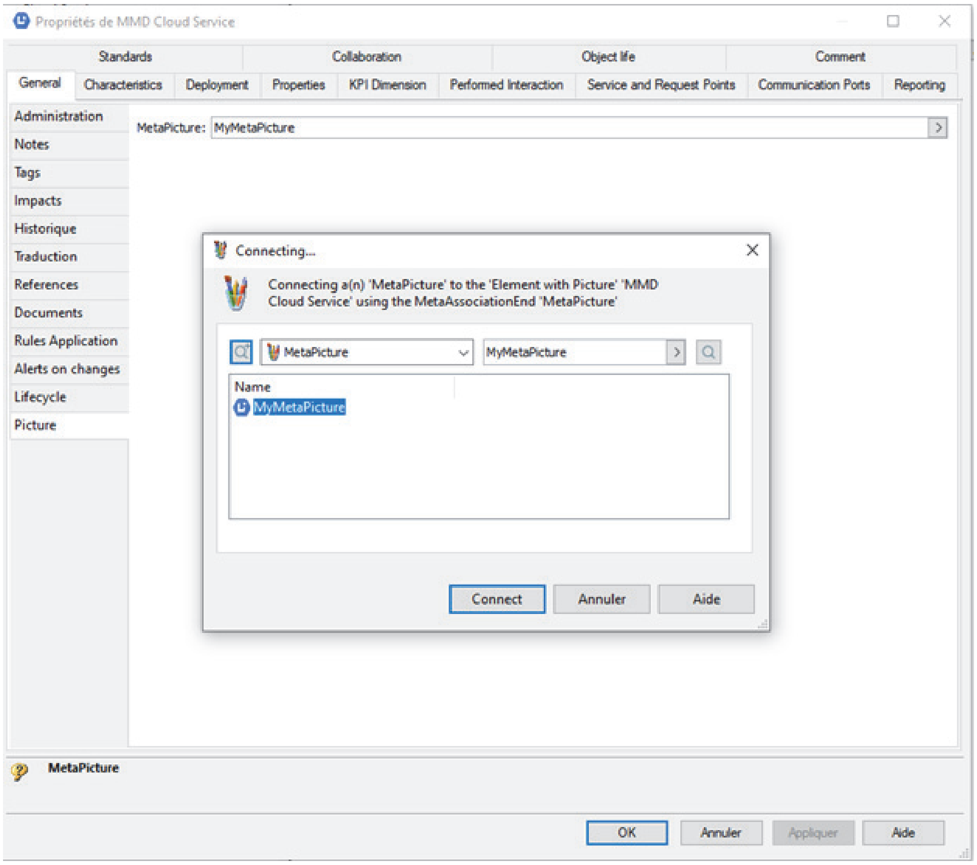
1. Import the image in your **MetaPicture**:
   - From the **Explorer**, right-click the **MetaPicture** and select **Import**.



   - In the dialog box, click **Browse** ... and select your image (*.jpg format).

2. Connect the MetaPicture to the diagram element concerned:
   - In the element properties, display **General > Picture**.
   - In the **MetaPicture** field, click the arrow and connect your MetaPicture.

     `E.g.: Cloud Service` **`DiagramTypeField`**



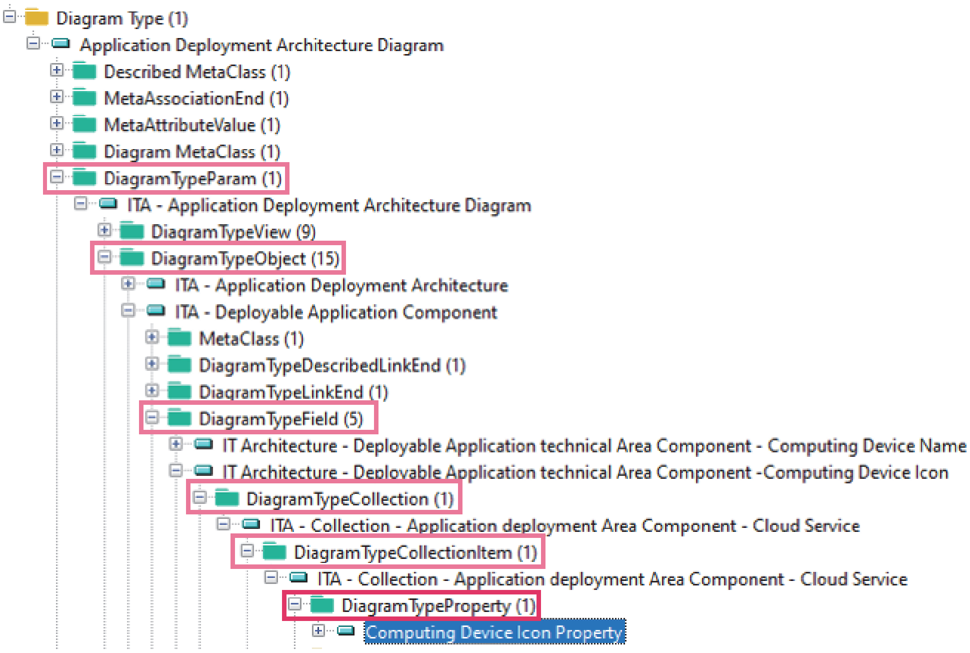
MetaPicture: MyMetaPicture

Connecting a(n) 'MetaPicture' to the 'Element with Picture' 'MMD Cloud Service' using the MetaAssociationEnd 'MetaPicture'

MetaPicture | MyMetaPicture

Name: MyMetaPicture

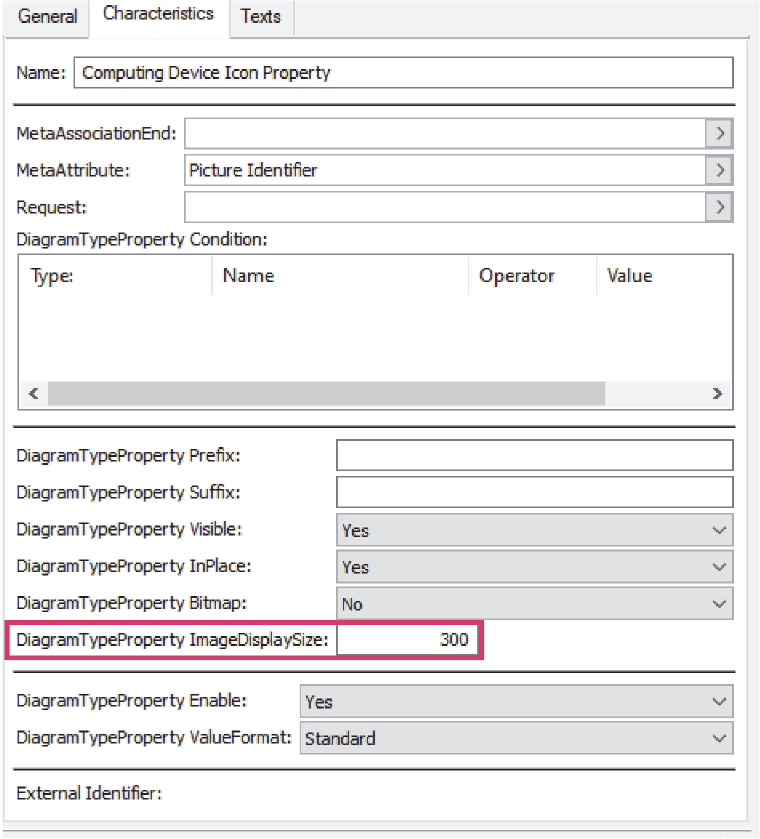
3. Set the image size:
   - Access the **DiagramTypeProperty** property corresponding to the image.



- Diagram Type (1)
  - Application Deployment Architecture Diagram
    - Described MetaClass (1)
    - MetaAssociationEnd (1)
    - MetaAttributeValue (1)
    - Diagram MetaClass (1)
    - DiagramTypeParam (1)
      - ITA - Application Deployment Architecture Diagram
        - DiagramTypeView (9)
        - DiagramTypeObject (15)
          - ITA - Application Deployment Architecture
          - ITA - Deployable Application Component
            - MetaClass (1)
            - DiagramTypeDescribedLinkEnd (1)
            - DiagramTypeLinkEnd (1)
            - DiagramTypeField (5)
              - IT Architecture - Deployable Application technical Area Component - Computing Device Name
              - IT Architecture - Deployable Application technical Area Component -Computing Device Icon
                - DiagramTypeCollection (1)
                  - ITA - Collection - Application deployment Area Component - Cloud Service
                    - DiagramTypeCollectionItem (1)
                      - ITA - Collection - Application deployment Area Component - Cloud Service
                        - DiagramTypeProperty (1)
                          - Computing Device Icon Property

   - In its **DiagramTypeProperty ImageDisplaySize** field, define the image size.

     `E.g.: 300`



Name: Computing Device Icon Property

MetaAssociationEnd:

MetaAttribute: Picture Identifier

Request:

DiagramTypeProperty Condition:

Type:	Name	Operator	Value

DiagramTypeProperty Prefix:

DiagramTypeProperty Suffix:

DiagramTypeProperty Visible: Yes

DiagramTypeProperty InPlace: Yes

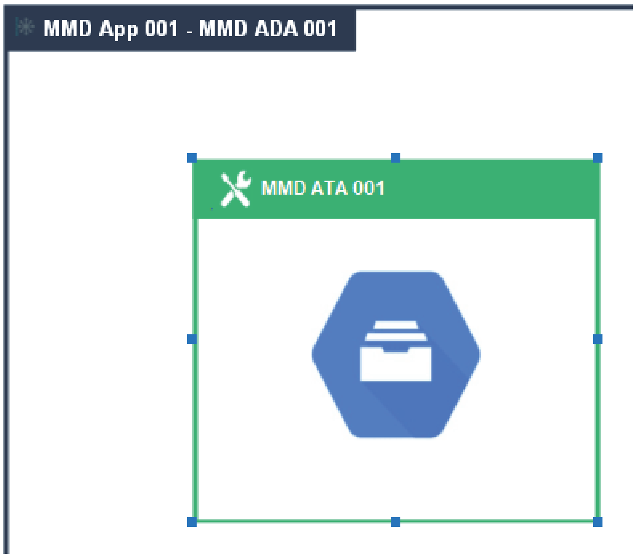
DiagramTypeProperty Bitmap: No

DiagramTypeProperty ImageDisplaySize: 300

DiagramTypeProperty Enable: Yes

DiagramTypeProperty ValueFormat: Standard

External Identifier:



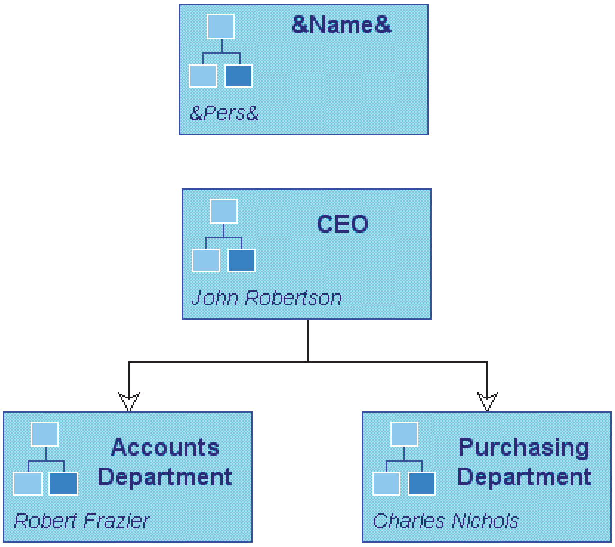
Positioning Objects in a Shape

When default deformation of a shape is not suitable, you can program its automatic deformation using a shape programming language.

This language sets size and position of fields and drawing objects when a shape is deformed. The **Field Positioning Wizard** automatically generates the code of this language.

Method shape: shape representing methodological concept instances in diagrams (org-unit shape, operation shape).

Calculated shape: particular method shape object of which content depends on the instance represented. The real content is dynamically calculated when the diagram is loaded.



Org-unit shape with org-unit and person names

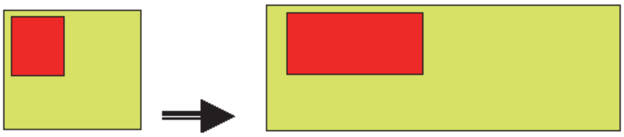
This text is identified by a name (&Name&, &pers&) derived from diagram configuration.

See:

- Identifying Elementary Objects
- Modifying Code Generation Specifications
- Code Generation
- Tips on Deformation Code Generation

Identifying Elementary Objects

By default, objects undergo deformation proportionally related to the shape.



Before running the **Field Positioning Wizard**, objects that should not undergo proportional deformation (particular position, fixed size, etc.) should be named.

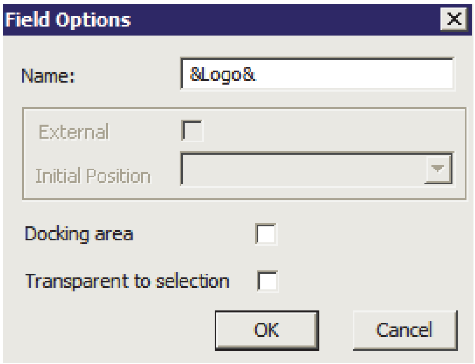
Each field name should be unique.

To name a shape object:

1. In the **Shapes Editor** menu bar, select **File > Open**.

   ☛ *To open the **Shapes Editor** see Opening the Shapes Editor.*

2. Select a shape.
3. In the **Shapes Editor Editing area**, right-click the shape and select **Field Options**.
   The **Field Options** dialog box appears.



Name: &Logo&

External □

Initial Position

Docking area □

Transparent to selection □

4. In the **Name** field, enter a name for the shape. The name should be prefixed and suffixed by &.

```
Example: &Logo&
```

☛ *Calculated fields are self-named. The text they contain can be used directly as the object identifier (for example &Name&).*

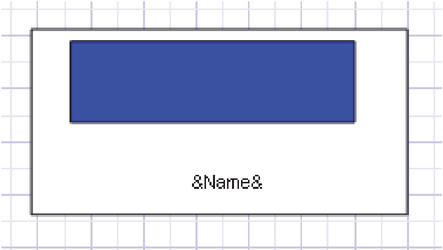
Modifying Code Generation Specifications

When your shape has been drawn and the elements named, you can complete specifications for the corresponding code generation.

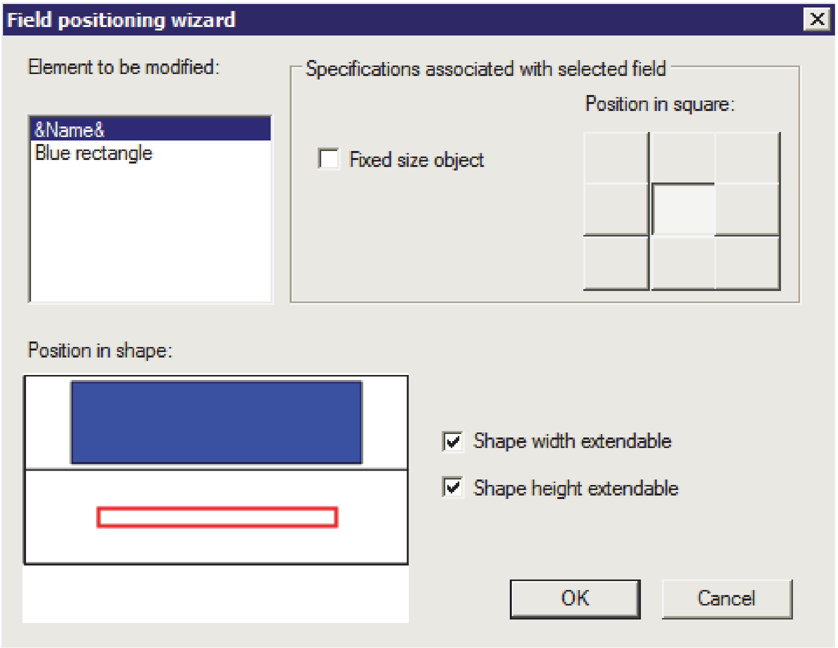
To run the shape code generation wizard:

1. In the **Shapes Editor** menu bar, select **Edit > Field Positioning > Field Positioning Assistant**.
   The **Field positioning wizard** appears.
   All named fields appear in the dialog box. The drawn shape is automatically squared and the different elements of the shape are distributed in the squares.

   For example, for a shape including a blue rectangle and a field &Name&:

   You obtain the following dialog box:

   Field positioning wizard

   Element to be modified:

   &Name&
   Blue rectangle

   Specifications associated with selected field

   Position in square:

   ☐ Fixed size object

   Position in shape:

   ☑ Shape width extendable

   ☑ Shape height extendable

   OK

   Cancel

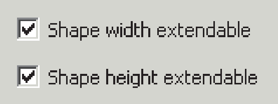
   In the **Element to be modified** pane, the selected element is highlighted in red in the **Position in shape** pane.
   You must correctly position the objects in the **Position in shape** pane.
2. To modify the specifications of selected fields and objects, see:
   - Shape height and width
   - Positioning of the object in its square
   - Object deformation

Shape height and width

By default, height and width of the overall shape are extendable. This means that in the diagram you can stretch the shape so that it exceeds the size of its component objects.

To set height and/or width:

- Clear the corresponding box(es).

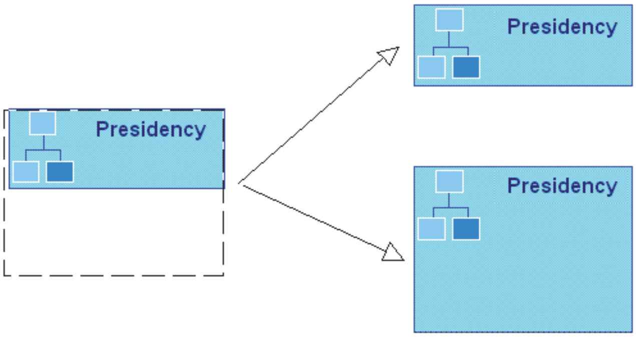


☑ Shape width extendable

☑ Shape height extendable

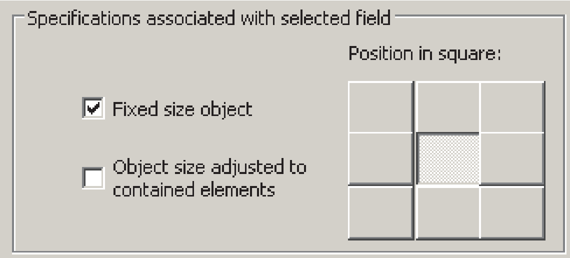
In the following example, the shape is resized to double its height. When the **Shape height extendable** option is:

- ***cleared***: the result is as displayed at ***top*** on right
- ***selected***: the result is as displayed at ***bottom*** on right



Positioning of the object in its square

By default, elements are placed in the center of the square.



Specifications associated with selected field

Position in square:

☑ Fixed size object

□ Object size adjusted to contained elements

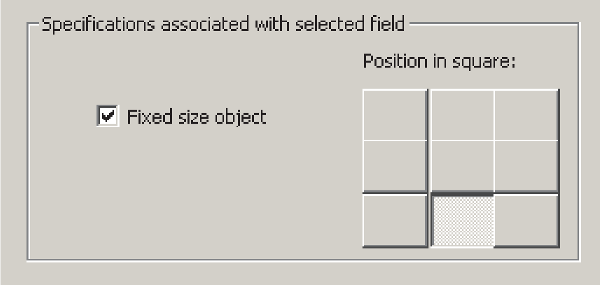
Element placed in center

Elements can be framed vertically and horizontally.

Use of this will be shown in later examples.

Object deformation

By default, all elements are subject to deformation proportional to that of the shape. It is however possible to set size for certain objects of the shape.



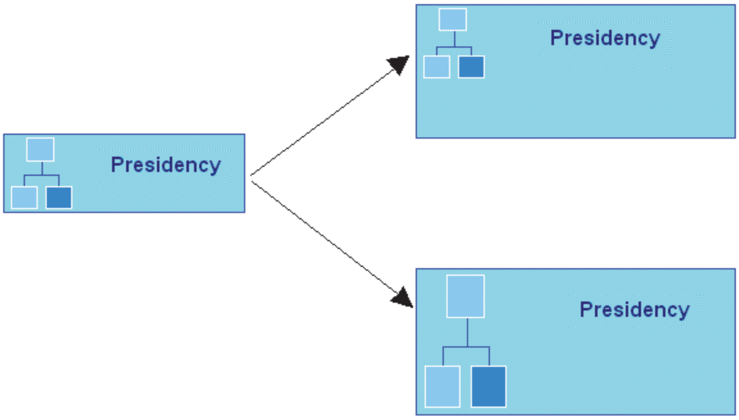
Specifications associated with selected field

☑ Fixed size object

☛ *For calculated fields, the **Fixed size object** option is not taken into account.*

In the following example, height and width of the shape are increased. When the **Fixed size object** option is:

- ***selected***, he result is as displayed at ***top*** on right.
- ***cleared***, he result is as displayed at ***bottom*** on right.



Code Generation

To validate specifications of the shape:

- In the **Field positioning wizard**, click **OK**.
  The deformation code is automatically generated.

Tips on Deformation Code Generation

Name the fields

Only named fields can intervene in the generated code. It is therefore essential that the different elements of the shape be carefully named. For more details, see Identifying Elementary Objects.

Draw accurately

The generator interprets the shape drawn in the shape editor. It considers that two objects are aligned when at least 80% of their width is shared.

MEGA Shapes Programming

A shape is a drawing stored in a file with extension ".MGS". Format of this file is the property of MEGA International. Certain shapes are used to represent methodological concepts used in HOPEX diagrams. Others are for decorative purposes only.

Shapes provided by HOPEX

HOPEX provides several sets of shapes used to represent the objects in diagrams:

- pictures (icons)
- pictures.7220
- pictures.7600 (corresponding to HOPEX V1R3 pictures)
- pictures.9000 (corresponding to HOPEX V4 pictures)
- pictures.15000 (corresponding to HOPEX V5 pictures)
- pictures.17000 (corresponding to HOPEX Aquila pictures)

Each set of shapes is sorted by categories:

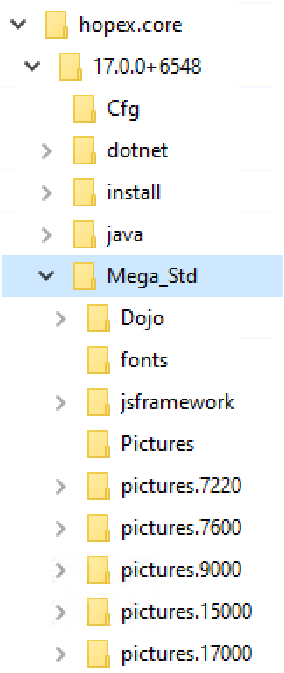
- Art
- Background
- Method

The shapes provided by **HOPEX** are stored in:

> **<Hopex Application Server>\ <name of the HAS instance>\.shadowFiles\hopex.core\**<module version>**Mega_Std**

```
E.g.: C:\ProgramData\MEGA\Hopex Application
Server\5000\.shadowFiles\hopex.core\17.0.0+6548\Mega_Std
```

The structure is as follows:



You must not modify or replace these shapes since they will be deleted and reinstalled from one version of HOPEX to another.

Customizing the shapes

To customize the shapes provided by **HOPEX** you must first install **HOPEX Application Server Customization** module.

➔ *To install **HOPEX Application Server Customization** module, see Importing a Module into HOPEX documentation.*

Your customized shapes (.MGS) must be stored in the following folder:

<Hopex Application Server>\ <name of the HAS instance>\.shadowFiles\has.custom\<module version>**hopex.core\Mega_Std**

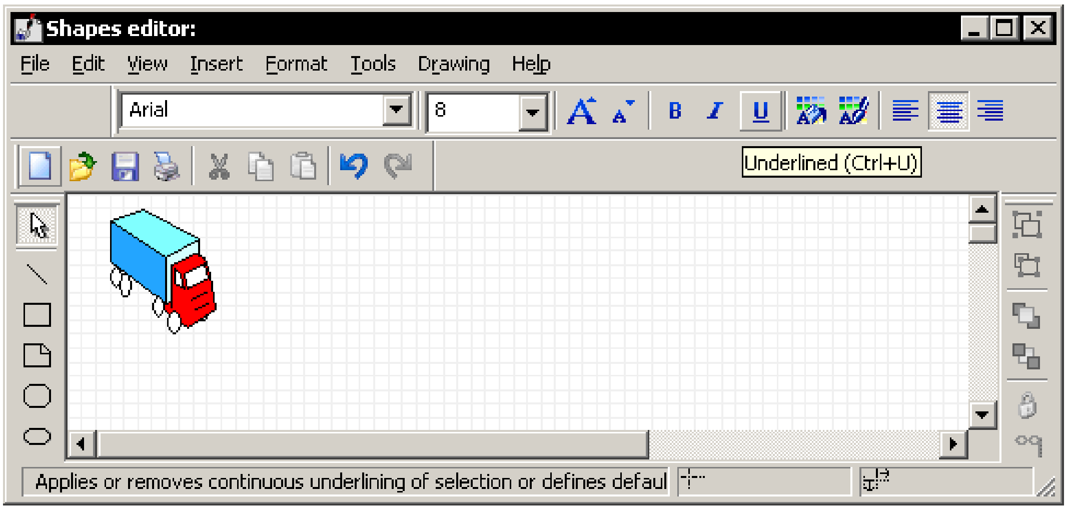
```
E.g.: C:\ProgramData\MEGA\Hopex Application
Server\5000\.shadowFiles\has.custom\15.2.0+13\hopex.core\Mega_Std
```

Your shapes overload the shapes provided by **HOPEX**.

For detailed information regarding customization, see *HAS Custom - Managing customization lifecycle* module documentation.

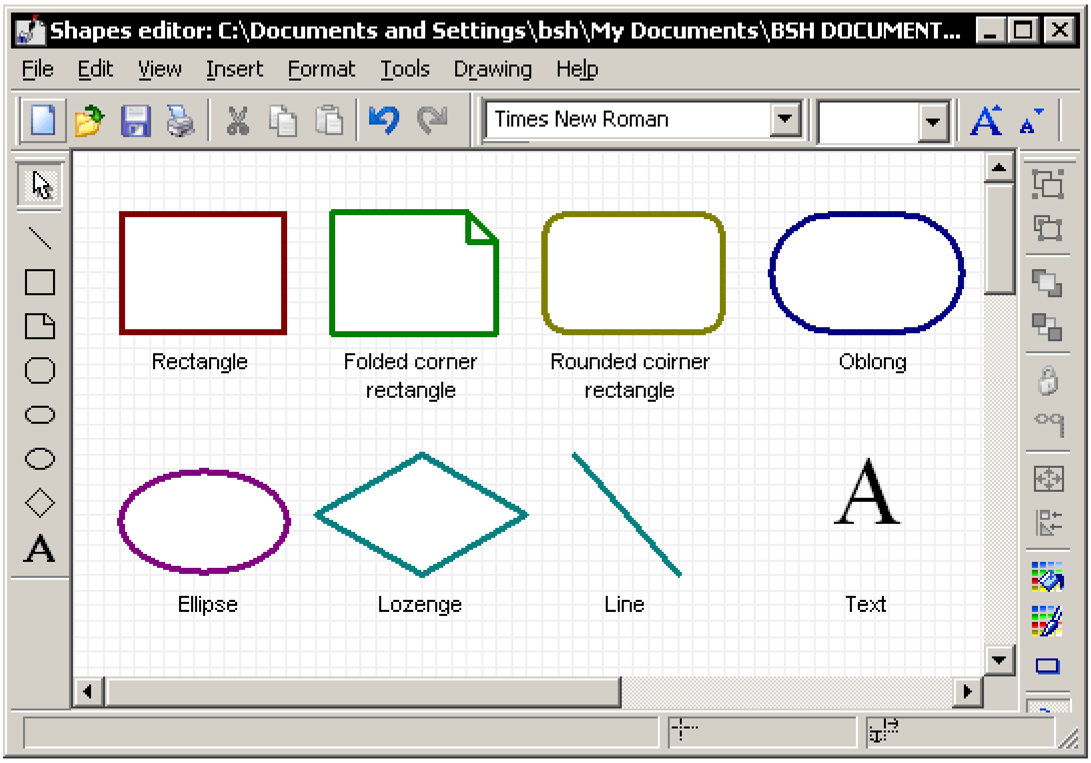
SHAPE EDITING

The "Shape Editor" enables loading and editing of shapes:



Applies or removes continuous underlining of selection or defines defaul

To draw a shape, 8 basic drawing object types are available: line, rectangle, folded corner rectangle, rounded corner rectangle, oblong, ellipse, lozenge, and text, together with images (bmp, wmf, jpeg,..).


Shapes editor: C:\Documents and Settings\bsh\My Documents\BSH DOCUMENT...
File Edit View Insert Format Tools Drawing Help
Times New Roman
Rectangle
Folded corner rectangle
Rounded coirner rectangle
Oblong
A
Ellipse
Lozenge
Line
Text

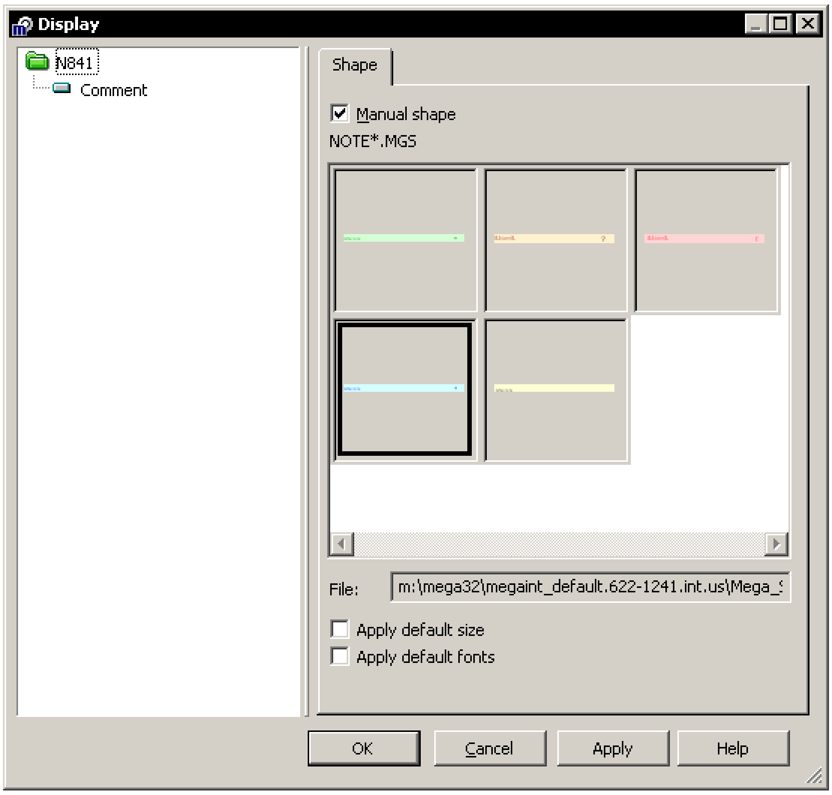
METHOD SHAPES

Method shapes are used to represent methodological concept instances in HOPEX diagrams.

Shape Families

At diagram content definition, each methodological concept is attributed a file name "mask" (eg. *thing**.mgs) enabling grouping under the same prefix of shapes available in this diagram for this concept.

For example, shapes respecting "NOTE*.MGS" format are available for the "Note" concept in diagrams:



☑ Manual shape

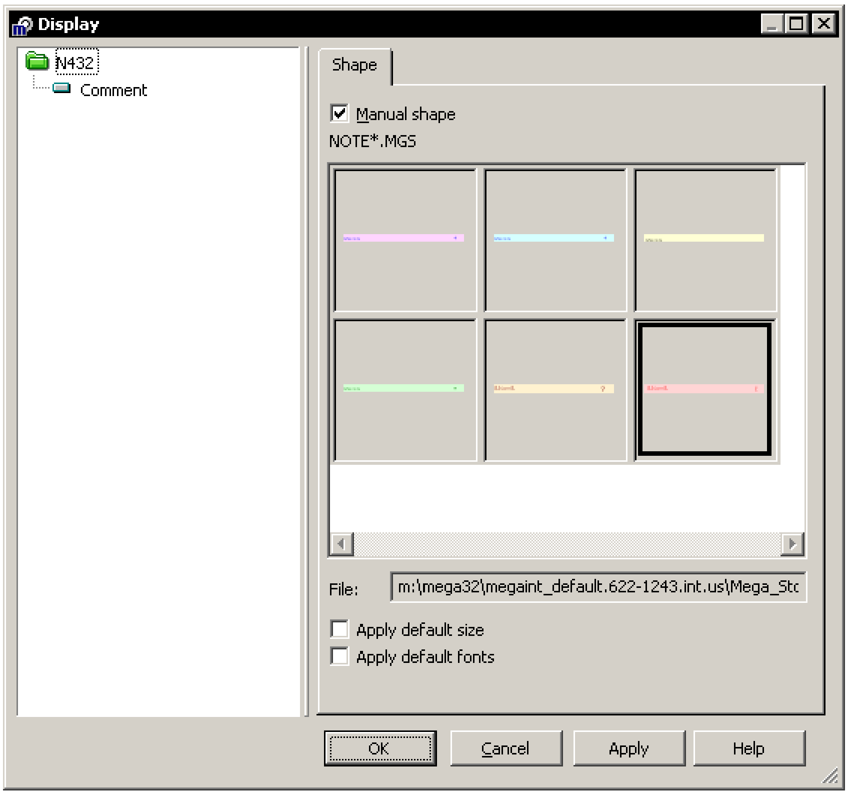
NOTE*.MGS

File: m:\mega32\megaint_default.622-1241.int.us\Mega_

□ Apply default size

□ Apply default fonts

As a result, any new shape available in the custom Mega_Std folder (C:\...\HOPEX Application Server\<HAS instance name>\.shadowFiles\has.custom\<Custom module version>\hopex.core\Mega_Std) with the same shape prefix automatically becomes available for this concept:



☑ Manual shape

NOTE*.MGS

File: m:\mega32\megaint_default.622-1243.int.us\Mega_Stc

☐ Apply default size

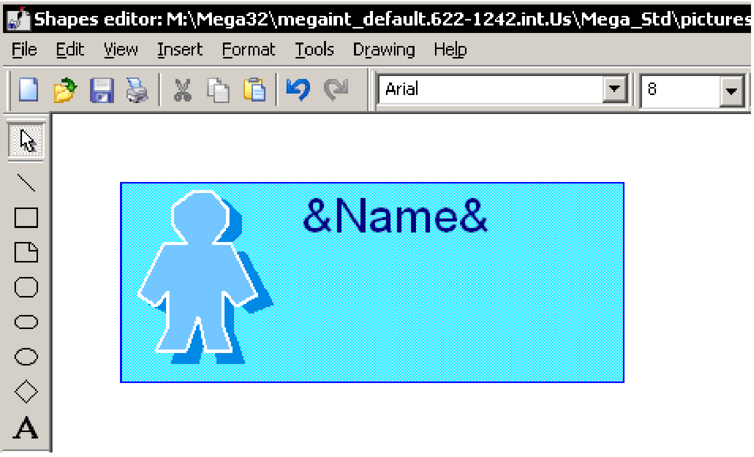
☐ Apply default fonts

At diagram content definition, shapes to be used by default are also specified.

Calculated Fields

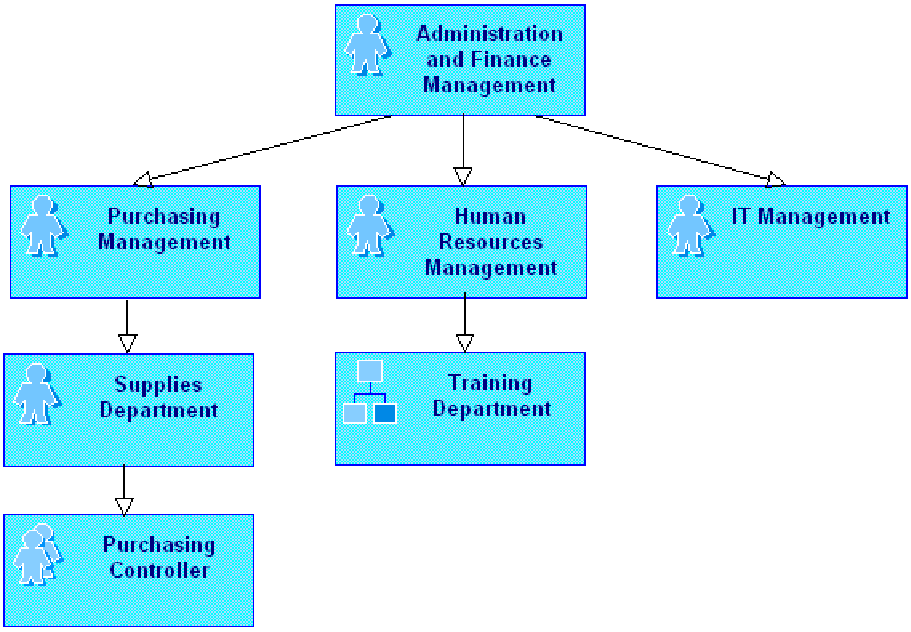
In addition to the 8 basic drawing object types, methodological shapes contain particular objects of which content depends on the instance represented. These are "calculated fields".

They are represented in shapes by text objects prefixed and suffixed "&":

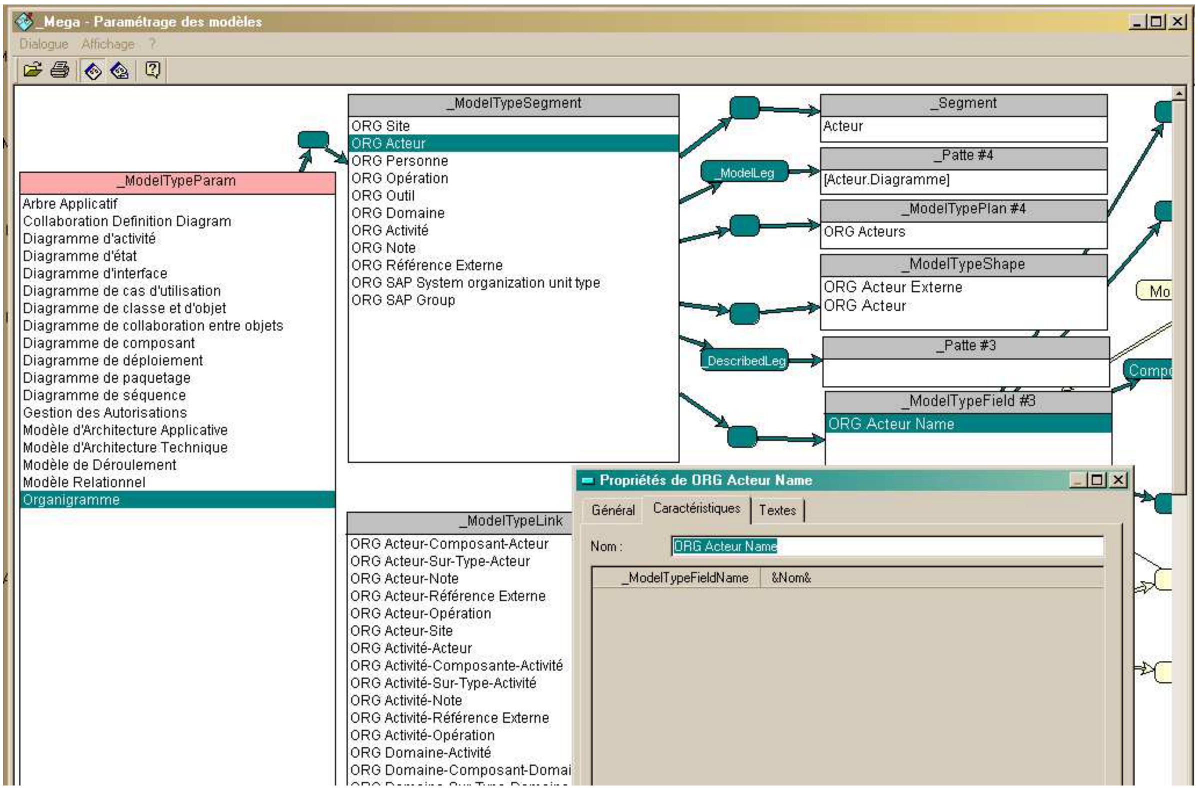

Shapes editor: M:\Mega32\megaint_default.622-1242.int.Us\Mega_Std\pictures

&Name&

The real content is dynamically calculated when the diagram is loaded.



The field calculation mode is specified in diagram content configuration.



_ModelTypeParam
Arbre Applicatif
Collaboration Definition Diagram
Diagramme d'activité
Diagramme d'état
Diagramme d'interface
Diagramme de cas d'utilisation
Diagramme de classe et d'objet
Diagramme de collaboration entre objets
Diagramme de composant
Diagramme de déploiement
Diagramme de paquetage
Diagramme de séquence
Gestion des Autorisations
Modèle d'Architecture Applicative
Modèle d'Architecture Technique
Modèle de Déroulement
Modèle Relationnel
Organigramme

_ModelTypeSegment
ORG Site
ORG Acteur
ORG Personne
ORG Opération
ORG Outil
ORG Domaine
ORG Activité
ORG Note
ORG Référence Externe
ORG SAP System organization unit type
ORG SAP Group

_Segment
Acteur

_Patte #4
[Acteur.Diagramme]

_ModelTypePlan #4
ORG Acteurs

_ModelTypeShape
ORG Acteur Externe
ORG Acteur

_Patte #3

_ModelTypeField #3
ORG Acteur Name

_ModelTypeLink
ORG Acteur-Composant-Acteur
ORG Acteur-Sur-Type-Acteur
ORG Acteur-Note
ORG Acteur-Référence Externe
ORG Acteur-Opération
ORG Acteur-Site
ORG Activité-Acteur
ORG Activité-Composante-Activité
ORG Activité-Sur-Type-Activité
ORG Activité-Note
ORG Activité-Référence Externe
ORG Activité-Opération
ORG Domaine-Activité
ORG Domaine-Composant-Domai

Propriétés de ORG Acteur Name

Général | Caractéristiques | Textes

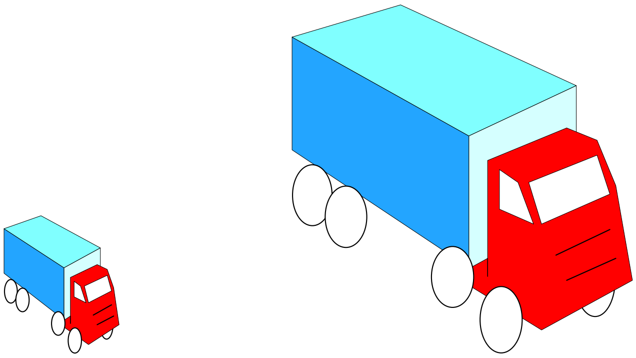
Nom : ORG Acteur Name

_ModelTypeFieldName	&Nom&

DEFAULT DEFORMATION

Proportional Deformation

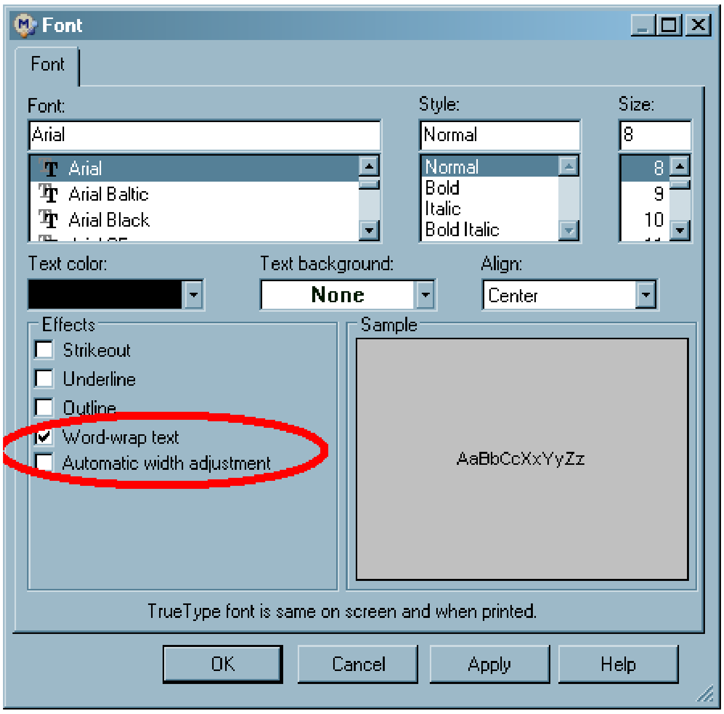
By default, shape deformation conforms to proportional logic. The same reduction or enlargement factor is applied to each object:



An option allows aspect ratio locking, forcing a shape height and width to be modified by the same factor.

Text Fields Deformation

The behavior of text fields can be tweaked by two options.



Font

Font: Arial

Style: Normal

Size: 8

Text color:

Text background: None

Align: Center

Effects

☐ Strikeout

☐ Underline

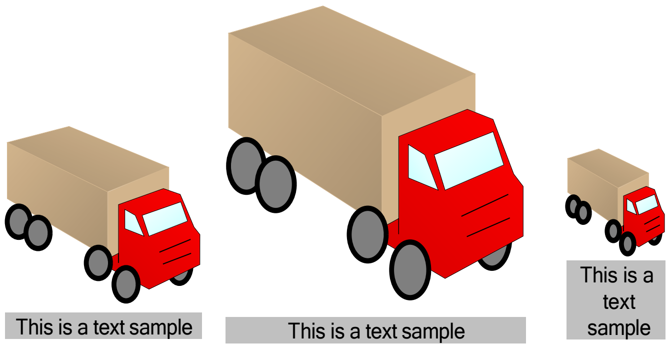
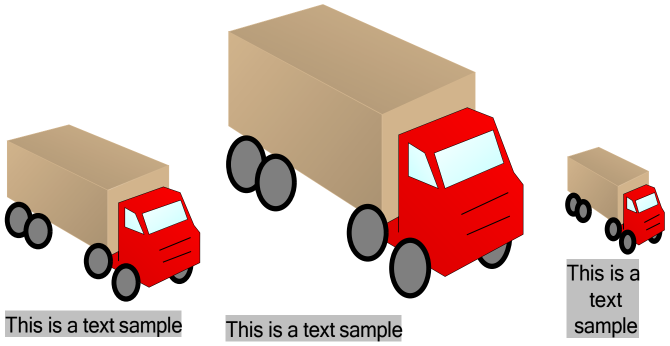
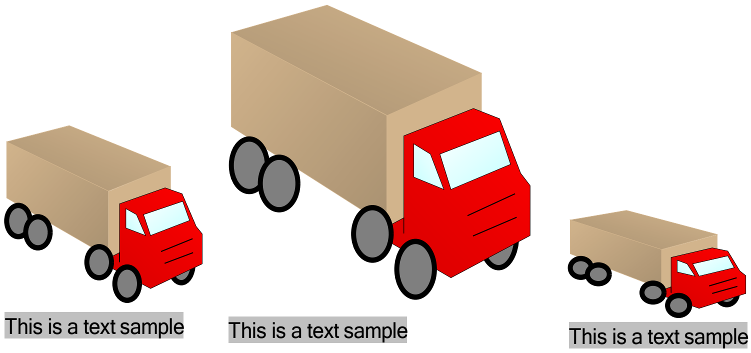
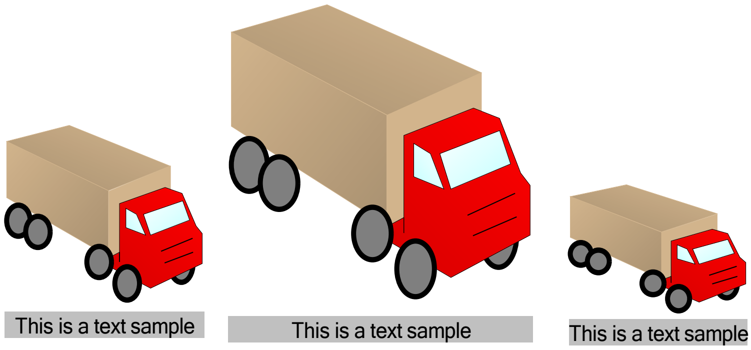
☐ Outline

☑ Word-wrap text

☐ Automatic width adjustment

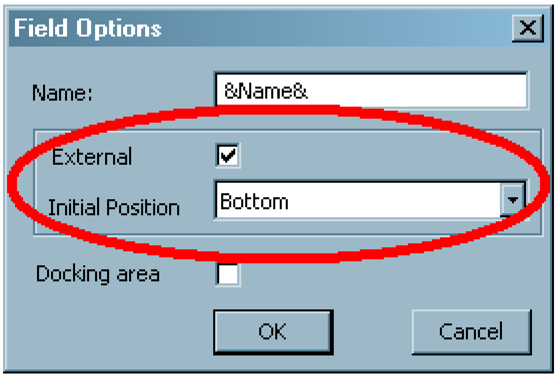
Sample

TrueType font is same on screen and when printed.

Word-wrap text	**Automatic width adjustment**	
Yes	No	 This is a text sample This is a text sample This is a text sample
The field width is deformed proportionally while the height is calculated to allow display of the complete text		
Yes	Yes	 This is a text sample This is a text sample This is a text sample
The field height is calculated to allow display of the complete text. The field width is then calculated to fit the text width.		
No	Yes	 This is a text sample This is a text sample This is a text sample
The text is always displayed on one line. The field width is calculated to fit the displayed text.		
No	No	 This is a text sample This is a text sample This is a text sample
The text is always displayed on one line. The field width is deformed proportionally but cannot be smaller than the displayed text.		

External Text Fields

Text fields can be defined to be external to the shape. External fields are ignored during a shape deformation and appear outside the shape frame. The user can modify the position and the width of such fields directly in the diagram.



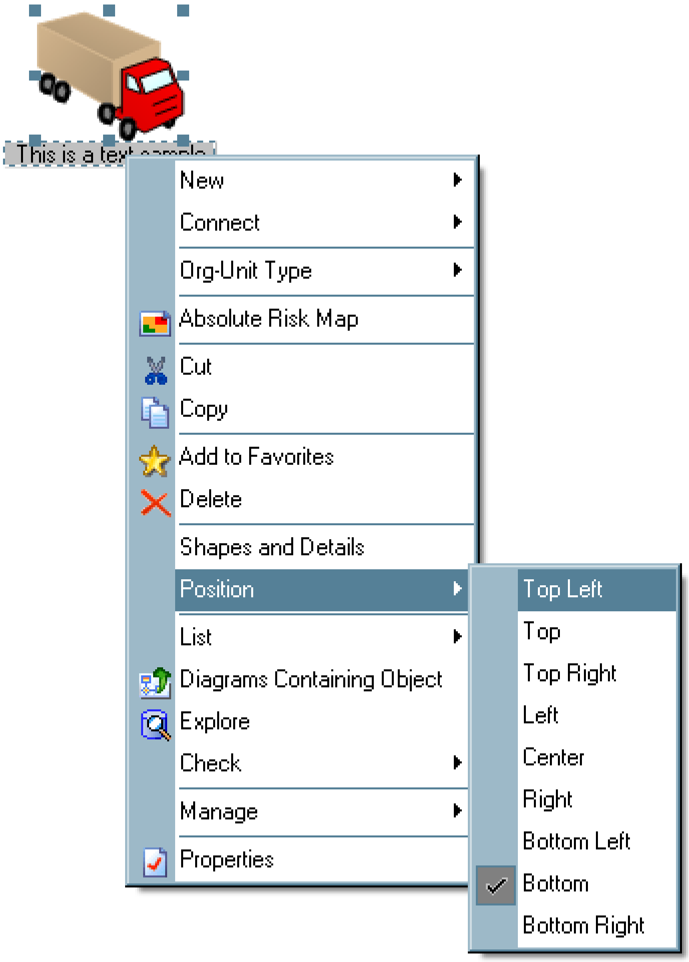
Field Options

Name: &Name&

External ☑

Initial Position: Bottom

Docking area □

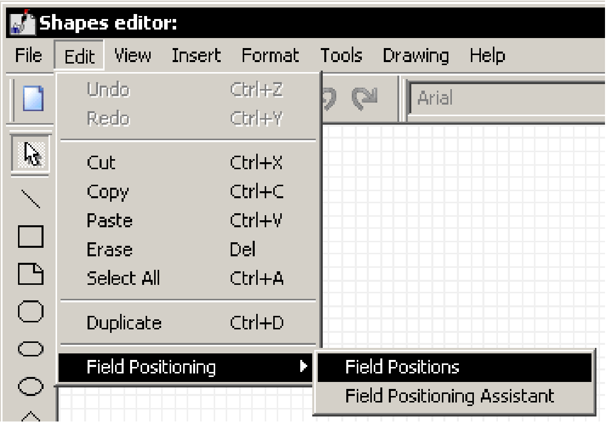


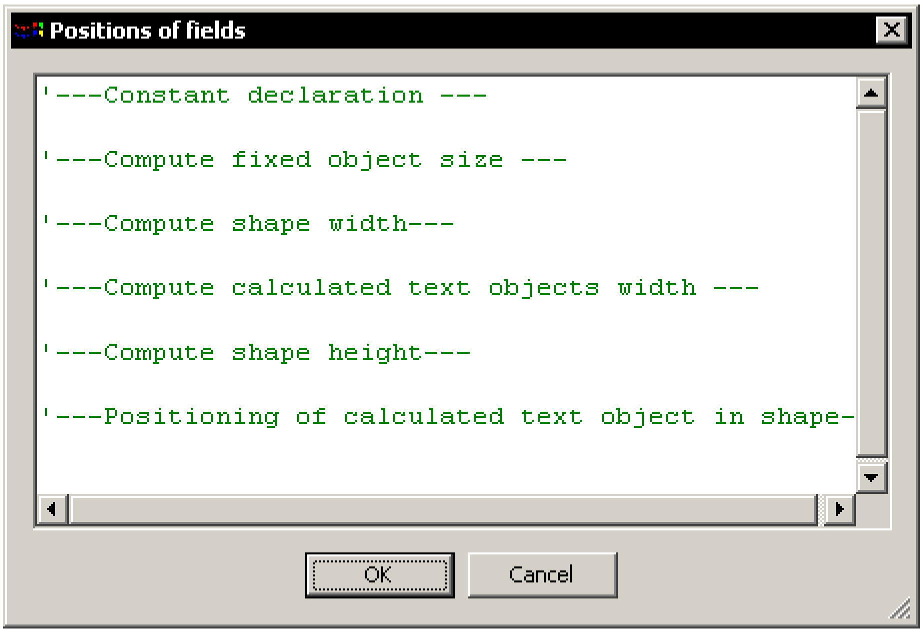
PROGRAMMED DEFORMATION

When default deformation is not suitable, it is possible to program deformation using shape programming language.

Deformation Code Editor

In the shape editor, the "Field positioning" option in the "Edit" menu gives access to a shape deformation program entry space:



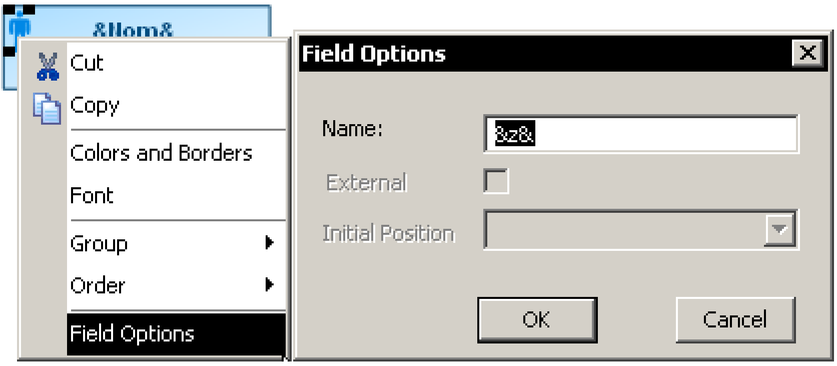


```
'---Constant declaration ---

'---Compute fixed object size ---

'---Compute shape width---

'---Compute calculated text objects width ---

'---Compute shape height---

'---Positioning of calculated text object in shape-
```

Object Identification

Basic Object Identification

In order to be referenced at programming, each shape object must first be identified by a name. This naming can be via the "Field Options" command in the pop-up menu of each object. The name should be prefixed and suffixed by "&".



Name: &z&

□ External

Initial Position

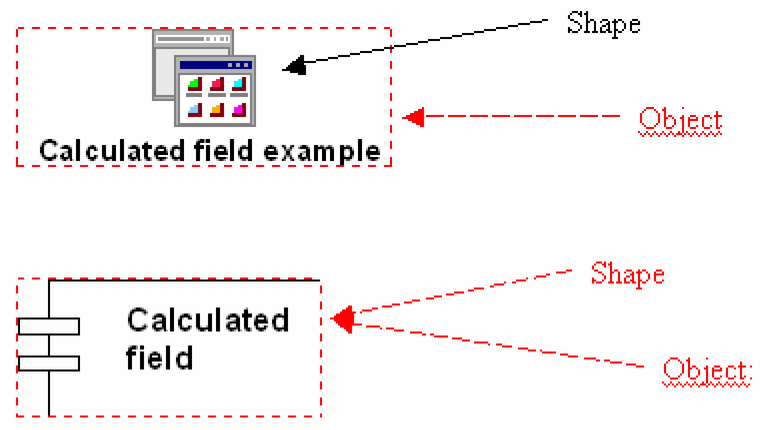
Identifying Calculated Fields

Calculated fields are "self-named". The text they contain can be directly used as the object identifier. (Example: &Name&).

Virtual Object Identification

The language is enhanced by two virtual objects for general use:

- "Shape" object: Rectangle containing graphical elements of the shape except for calculated fields.
- "Object" object: Rectangle containing all graphical elements of the shape including calculated fields.

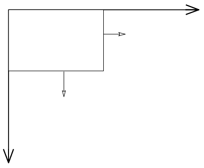


Use of the "Object" object should be limited to cases where we wish to place calculated fields outside the Shape object.

Language Syntax

The value unit is 1/360 inch (14/360 inch ≈ 1mm).

The coordinates system point of origin is the top left of the shape.



Size

width: width
height: height

shape.default.width: Original width (in the MGS) of the shape without calculated fields

shape.default.height: Original height (in the MGS) of the shape without calculated fields

Position

top: top

bottom: bottom

left: left

right: right

hcenter: horizontal median

vcenter: vertical median

Operators

= , + , - , *

max(val1,val2,...,val10)   Operand number is limited to 10.

Functions

RelX(*n)* and RelY(*n*), where n is a value, enable definition of constants proportional to shape width and height.

For example, with CNST = RelX(10),

CNST is 10 if shape width is equal to its initial width.

CNST is 15 if shape width is equal to 1.5 times its initial width.

Comment

' at start of line

To access these properties, syntax is as follows:

- For calculated fields: *object name without &s . property* (eg. Att.width)
- For the "Shape" object: Shape.*property* (eg. Shape.hCenter)
- For the "Object" object: Object.*property* (eg. Object.top)
- For other named objects: Shape.*object name with &s*.property (eg. Shape.&Logo&.bottom)

Programming

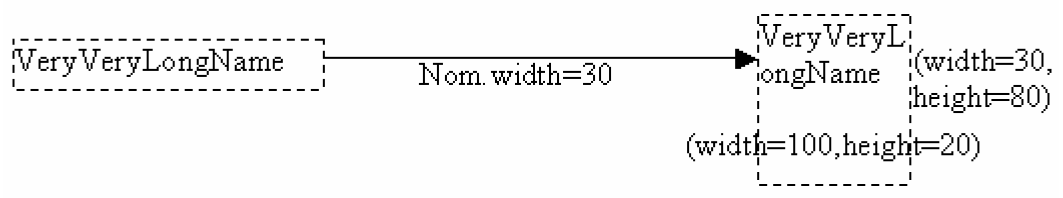
The shape deformation program is applied each time the user resizes an object based on this shape, or when one of its calculated fields changes value.

Program lines are executed singly and sequentially.

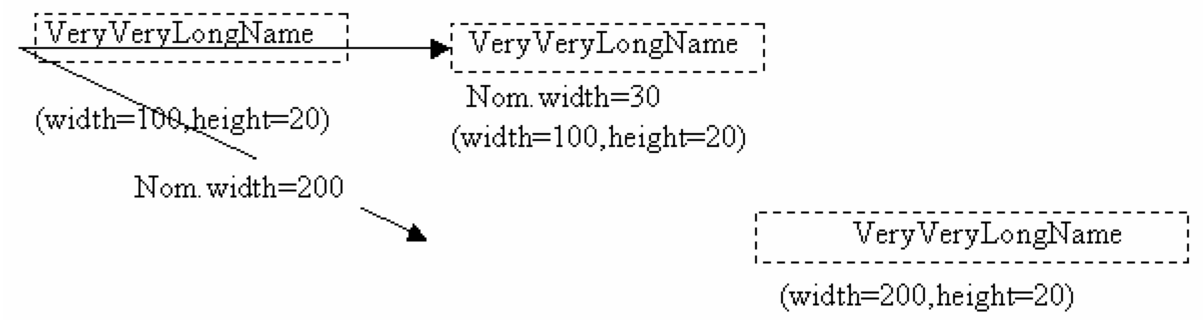
Line order is therefore of major importance, and any inversion of these lines can cause quite unexpected deformation behavior.

Calculated Field Programming

If "Word-wrap text" is active, any modification of field width by code will result in immediate recalculation of its height:



If "Word-wrap text" is inactive, the calculated field should be sufficiently wide to display the text on a single line. If in programming we attempt to assign too narrow a width, an adequate minimum width is authoritatively reassigned.

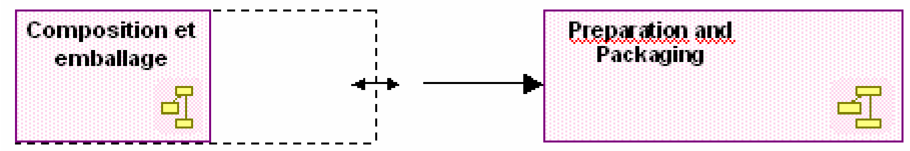


Modification of field height is never taken into account. It is always automatically recalculated as a function of field width.

General Programming Recommendations

Deformation code is called in 2 cases:

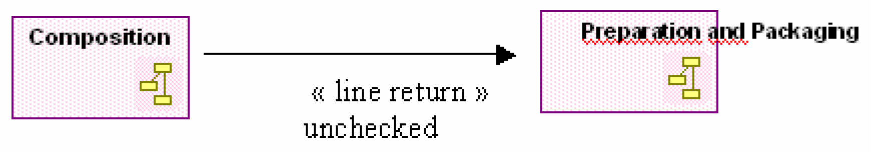
When we change size of a concept based on this shape:



Before code application

Here the code should handle text resizing.

The content of a calculated field is changed:



Before code application

Here the code should handle shape resizing.

To limit side effects due to incorrect programming, the following steps are recommended:

- Fix the size of fixed size objects.
- Fix the final width of the shape as a function of the width of fixed size objects, calculated fields and the width required for the shape (shape width before code application, see example below). *(NB1)*
- Fix the final width of the calculated fields as a function of the final width of the shape. *(NB2)*
- Fix the final height of the shape as a function of the height of fixed size objects, calculated fields and the height required for the shape. *(NB2)*
- Fix the positions of objects within the shape

NB1: Before deformation code execution, the width of calculated fields with word-wrap active is 1mm (authorized minimum), and the width of calculated fields with word-wrap inactive is equal to the effective value necessary for visibility of content. **The width of all calculated fields of the shape must therefore be defined.**

NB2: Since shape height depends on height of calculated fields, and height of calculated fields automatically depends on their width, it is essential to fix the width of calculated fields before fixing shape height.

Default Code

A shape without a code will not have the same deformation behavior as a shape with a code performing no function (no modification of shape objects properties). A default code is applied to shapes without code.

For each calculated field, the following instruction block is inserted by default in the code.

Field refers to the calculated field.

The following constants are calculated compared with the initial state of the shape (the state that can be seen in the shape editor)

FieldWidthInit: width of calculated field

FieldTopInit: vertical position of calculated field

FieldLeftInit: horizontal position of calculated field

```
BLOCK:
    Shape.width = max( Shape.width , Field.width )
    Field.width   = RelX ( FieldWidthInit )
    Shape.width = max( Shape.width , Field.width )
    Field.top       = Shape.top + RelY ( FieldTopInit )
    Field.top       = Shape.top + RelY ( FieldTopInit )
```

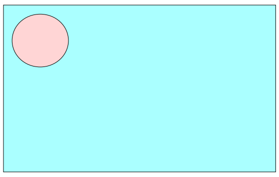
This code enables the calculated field to maintain the same relative position within the shape.

In programming a shape, position and size of all calculated fields of the shape must be managed.

EXAMPLES

Shape With Fixed Size Object

The aim is to retain the size and position of circle &Circle&, whatever the size of the shape.



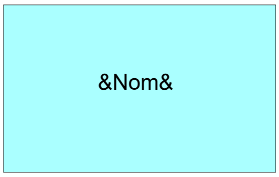
To fix dimensions of the fixed size object:

```
Shape.&Circle&.width=140

Shape.&Circle&.height=140
```

To freeze its position must be added:

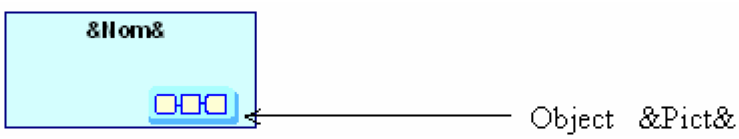
```
Shape.&Circle&.top = Shape.top + 20

Shape.&Circle&.left = Shape.left + 20
```

Shape With Calculated Field



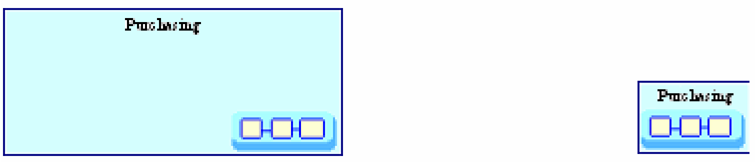
```
'---Constant---
CX_SPACE=10
CY_SPACE=10

'---Shape width--- (at minimum the width required to display text)
Shape.width=Max(Shape.width, Nom.width +2*CX_SPACE)

'---Fields width--- (field size is readapted to shape size)
Nom.width=Shape.width-2*CX_SPACE

'---Shape width--- (at minimum the height required to display text)
Shape.height=Max(Shape.height, Nom.height +2*CY_SPACE)

'---Objects Position---
Nom.hCenter = Shape.hCenter
Nom.vCenter = Shape.vCenter
```

Shape With Pictogram



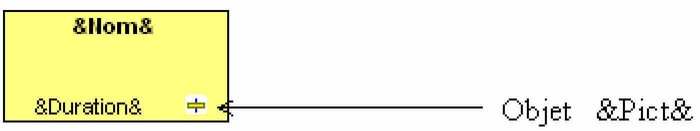
The pictogram should remain at bottom right and at fixed size.

```
'---Constant---
CX_SPACE = 10
CY_SPACE = 10
```



```
'---Fixed size object---
Shape.&Pict&.width  = 2
Shape.&Pict&.height = 1

'--- Shape width ---
Shape.width = Max ( Shape.width, Max( Nom.width, Shape.&Pict&.width) +
2*CX_SPACE)

'--- Fields width ---
Nom.width = Shape.width - 2 * CX_SPACE

'--- Shape height ---
Shape.height = Max ( Shape.height, Nom.height + Shape.&Pict&.height + 3 *
CY_SPACE)

'---Objects Position---
Nom.left =  Shape.left + CX_SPACE
Nom.top =  Shape.top + CY_SPACE

Shape.&Pict&.right  = Shape.right - CX_SPACE
Shape.&Pict&.bottom = Shape.bottom - CY_SPACE
```

Shape With Calculated Field and Pictogram Juxtaposed

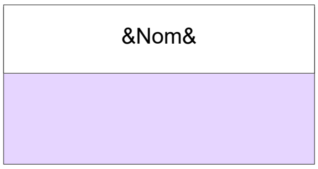


The pictogram should remain at bottom right and at fixed size.

The calculated field &Duration& and the pictogram should not overlap.

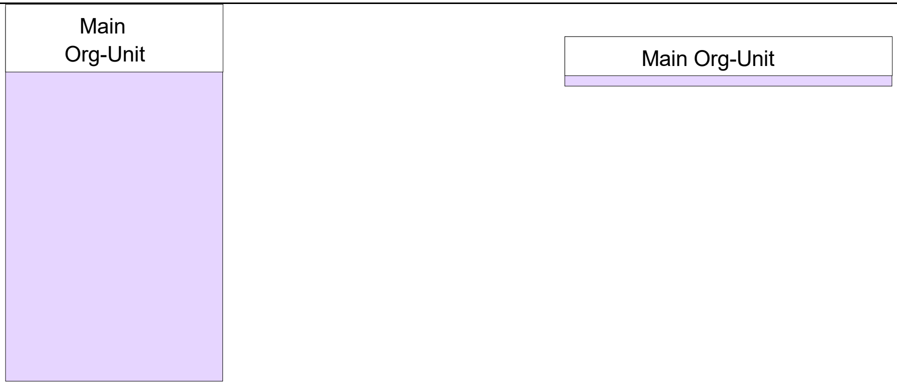
```
'---Constant---
CX_SPACE = 10
CY_SPACE = 10

'---Graphics size---
Shape.&Pict&.width  = 70
Shape.&Pict&.height = 35

'--- Shape width ---
Shape.width = max  ( shape.width, Nom.width + 2*CX_SPACE ,
                                  Duration.width + Shape.&Pict&.width +
3*CX_SPACE)

'--- Fields width ---
Nom.width   = Shape.width - 2 * CX_SPACE
Duration.width = Shape.width - Shape.&Pict&.width - 3*CX_SPACE

'--- Shape height ---
Shape.height =max(Shape.Height , Nom.Height+shape.&Pict&.height
+3*CY_SPACE, Nom.Height + Duration.height + 3*CY_SPACE )

'---Objects Position---
Nom.left    =  Shape.left + CX_SPACE
Nom.top     =  Shape.top + CY_SPACE

Shape.&Pict&.right  = Shape.right - CX_SPACE
Shape.&Pict&.bottom = Shape.bottom - CY_SPACE

Duration.left = Shape.left + CX_SPACE
Duration.bottom = Shape.bottom - CY_SPACE
```

Column Shape



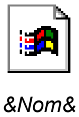
The white rectangle is named &TitleRect&, it should frame the name and retain its fixed size when the shape is elongated downwards. (shape type used to represent org-units in flowcharts)

```
'---Constant---
CX_SPACE=10
CY_SPACE=10

'--- Shape width ---
Shape.width=Max(Shape.width,Nom.width+2*CX_SPACE)

'--- Fields width ---
Nom.width=Shape.width-2*CX_SPACE

'--- Shape height ---
Shape.height=Max(Shape.height,Nom.height+4*CY_SPACE)

'--- Graphics Size---
Shape.&TitleRect&.height=Nom.height+2*CY_SPACE
Shape.&TitleRect&.width=Shape.width

'---Objects Position---
Shape.&TitleRect&.left=Shape.left
Shape.&TitleRect&.top=Shape.top
Nom.hCenter=Shape.&TitleRect&.hCenter
Nom.vCenter=Shape.&TitleRect&.vCenter
```



Shape With Title Below

In this example, the calculated field is outside the shape. This is a use case of the "Object" virtual object.



```
'---Constant---
CY_SPACE = 10

'---Shape size---
Shape.width = Shape.default.width
Shape.height = Shape.default.height

'---Object width---
Object.width = Max(Nom.width, Object.width, Shape.width)

'--- Fields width ---
Nom.width = Object.width

'---Objects Position---
```

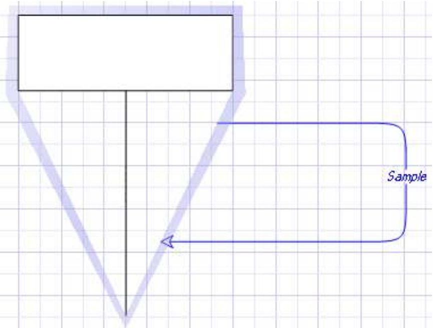
```
Nom.left = Object.left
Nom.top = Shape.bottom + CY_SPACE
Shape.top = Object.top
Shape.hCenter = Object.hCenter
```

Here we must also place the shape object within the Object object.

Note: Object is recalculated after shape code application (smallest rectangle containing calculated fields and Shape).

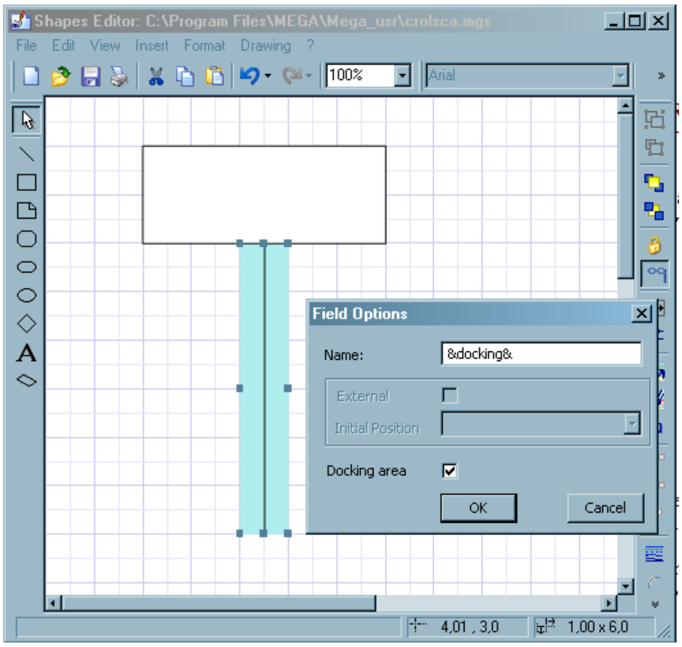
Docking Area

HOPEX automatically computes a convex hull for each shape. This area is used to determine the intersections between the shapes and links connected to it. However, for some shapes, this area is not adequate.



To improve the behavior of the shape regarding links, it is possible to define a custom docking area. With such an area defined, links will start and end only on the edge of this area.

To define a custom docking area, draw a basic graphic object and check the appropriate option in the "field options" dialog box.


Shapes Editor: C:\Program Files\MEGA\Mega_usr\crolsca.mgs
File Edit View Insert Format Drawing ?
100%
Arial
4,01 , 3,0
1,00 x 6,0

Field Options

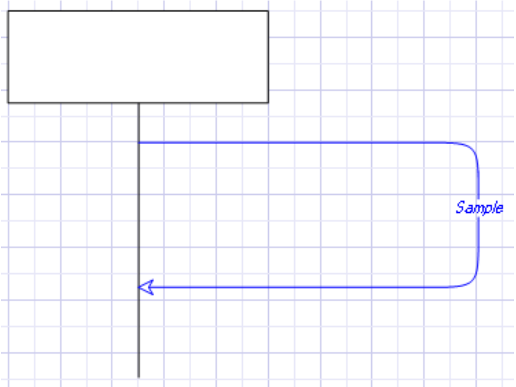
Name: &docking&

External □

Initial Position

Docking area ☑

OK Cancel



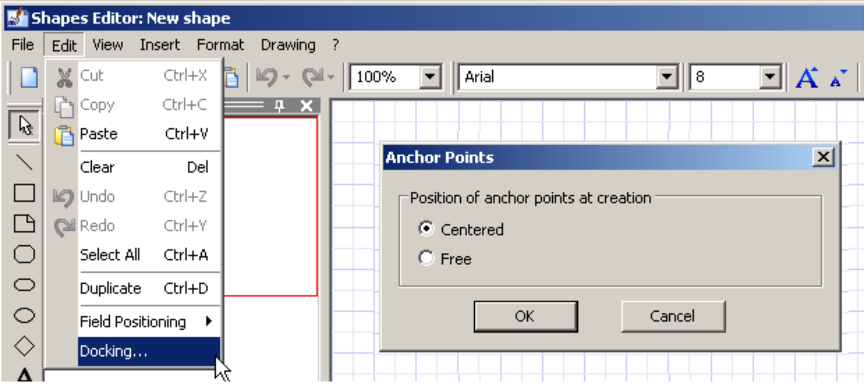
To further improve the behavior of the shape, this field can be positioned like any other field with the shape programming language. To further improve the look of the shape, this field can be hidden in the shape editor.

However, only one docking area is allowed, and it must be convex.

Position of Anchor Point at Creation

When connecting two objects in the diagram editor with a link, there are two different behaviors which will position the anchor point on the shape.

The behavior to be used with a shape can be set in the shape editor.



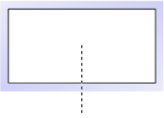
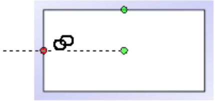
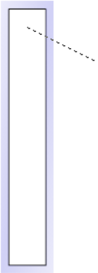
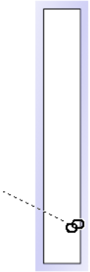
Shapes Editor: New shape

Anchor Points

Position of anchor points at creation

● Centered

○ Free

	From	To
Centered The anchor point is centered in the shape		 Targeting the green dots will change the style (orthogonal/straight) and orientation of the link
Free The anchor point is positioned where the mouse is clicked or released The default line style is used		

CONFIGURING DIAGRAMS

Once the required MetaModel extensions are defined, you need to display these in diagrams.

Studio allows you to carry out multiple tasks such as creation or modification of diagram types, addition of MetaClasses and MetaAssociations available in the diagram type, definition of default display of MetaClasses, object shapes in a diagram type, association of a diagram type with one or several MetaClasses.

This chapter covers the following points:

- ✓ Diagram Type Creation
- ✓ Configuring a Diagram Type
- ✓ Configuring Tabular Entry mode

DIAGRAM TYPE CREATION

Accessing Diagram Types

To access diagram types:

- From the **HOPEX** menu bar, select **View > Navigation Windows > MetaStudio**.
  The diagram types are grouped in the **Diagram Types** folder.

Sorting Diagram Types

Diagram types are sorted by category. These categories are presented as sub-folders of **Dagram Types** folder.

To create a folder of diagram types:

1. In the **MetaStudio** navigation window, right-click the **Diagram Types** folder and select **New > Folder of Diagram Types**.
   The **Creation of Folder of Diagram Types** window appears.
2. Enter the folder **Name**.
3. Click **OK**.
4. Create a diagram type from this sub-forder or drag and drop a diagram type in this folder.

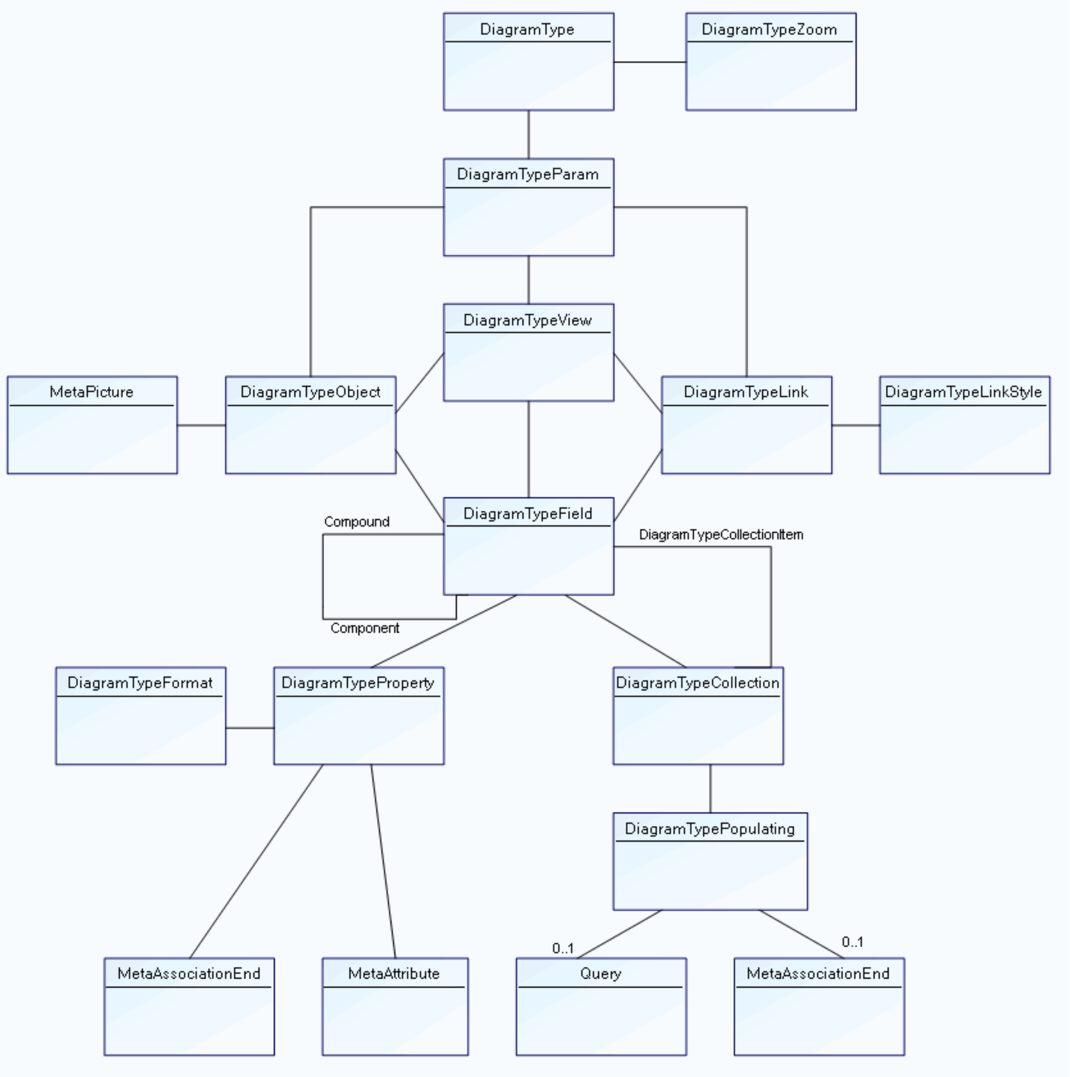
Creating a Diagram Type

To create a diagram type:

1. In the **MetaStudio** navigation window, right-click the **Diagram Types** folder (or sub-folder) and select **New > Diagram Type**.
   The **Creation of Diagram Type** window appears.
2. Enter the diagram type **Name**.
3. Click **OK**.

Diagram Type Concepts

Diagram type configuration uses concepts presented in the following metamodel:



A ***DiagramType*** can describe a specific number of concepts.

A ***DiagramTypeZoom*** enables a MetaClass to be described by a diagram type.

E.g.: a *Process* can be described by a *Process Diagram*).

Configurations of a diagram type are carried by a ***DiagramTypeParam***.

Diagram type configuration is based on the following concepts:

- ***DiagramTypeObject***: object types (MetaClasses) that can be used in a diagram type.
- ***DiagramTypeView***: views available in a diagram type. A view groups one or several object types that can be used in a given diagram type.
- ***DiagramTypeLink***: links that can be displayed in a diagram type.
- ***DiagramTypeField***: fields that can appear in an object shape or link.

These concepts enable definition of graphical representation of objects, links, views, and fields (in graphical representation of objects and links).

Concept Descriptions:

Diagram Concept	Definition
DiagramType	Type of diagram
DiagramTypeCollection	Configuration of an attribute repeated n times in a diagram configuration
DiagramTypeField	Representation of a field displaying a list of properties or a basic property
DiagramTypeFormat	Defines a format or string that can change according to a specific condition
DiagramTypeLink:	Representation of a MetaAssociation in a diagram type
DiagramTypeLinkStyle	Defines a link style that can change according to a specific condition
DiagramTypeParam	Configuration of a diagram type
DiagramTypePath	Representation of a path (eg. MetaAssociation) in a diagram type
DiagramTypePathPart	Configuration of a DiagramTypePath element
DiagramTypeView	Configuration of a view and its content
DiagramTypeProperty	Configuration of a basic attribute in a diagram configuration
DiagramTypeObject	Representation of an object type in a diagram type
DiagramTypePopulating	Defines the method (MetaAssociationEnd, Query) of populating a collection specified by DiagramTypeCollection
DiagramTypeZoom	A DiagramTypeZoom enables a MetaClass to be described by a diagram type.

CONFIGURING A DIAGRAM TYPE

See:

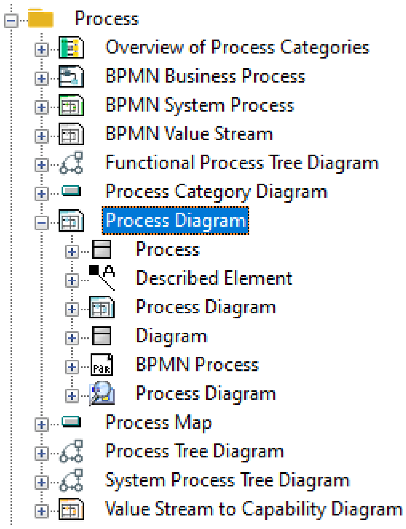
- Diagram Type Configuration Example: Process Diagram
- Creating and Backing Up a Diagram Configuration
- Using a New MetaClass in a Diagram
- Adding a Zoom to a Diagram Type
- Modifying a Diagram Type
- Modifying the distance between the source object and the created one
- Modifying the Suggested Objects
- Adding the Creation Wizard when Adding an Object
- Configuring MetaAssociations

Diagram Type Configuration Example: Process Diagram

Here is an illustration of a diagram type configuration: **Process Diagram**.

To access the Process Diagram configuration:

1. In the **MetaStudio** navigation window, expand **Diagram Types > Process > Process Diagram**.
   The **BPMN Process** DiagramTypeParam corresponds to the object whose icon contains PAR.

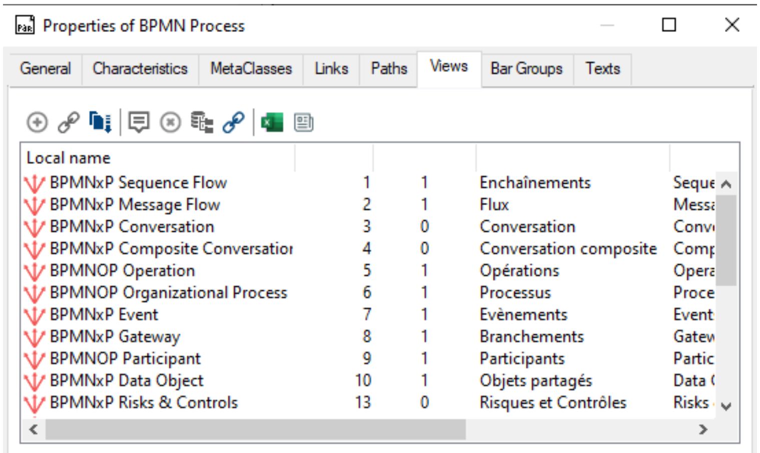


Process
- Overview of Process Categories
- BPMN Business Process
- BPMN System Process
- BPMN Value Stream
- Functional Process Tree Diagram
- Process Category Diagram
- Process Diagram
  - Process
  - Described Element
  - Process Diagram
  - Diagram
  - BPMN Process
  - Process Diagram
- Process Map
- Process Tree Diagram
- System Process Tree Diagram
- Value Stream to Capability Diagram

2. To view the flowchart configuration elements, expand the **BPMN Process** DiagramTypeParam folder. It includes:
   - ***DiagramTypeViews***: views.
   - ***DiagramTypeObjects***: MetaClasses that can be used in process diagrams. A DiagramTypeView can contain one or several DiagramTypeObjects.
   - ***DiagramTypeLinks***: enables description of representation of a MetaAssociation in a diagram.

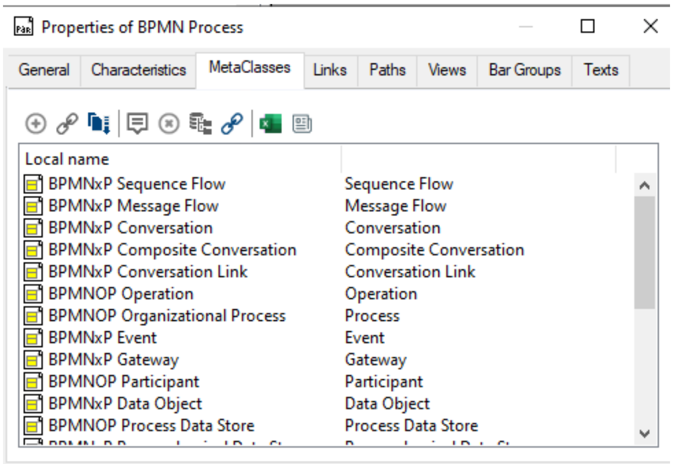
From the DiagramTypeParam Property pages, you can access:

- DiagramTypeViews from the **Views** tab



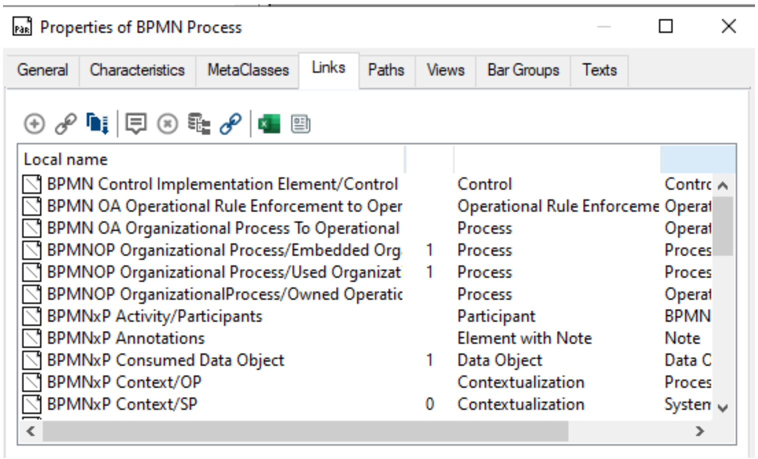
Local name				
BPMNxP Sequence Flow	1	1	Enchaînements	Seque
BPMNxP Message Flow	2	1	Flux	Messa
BPMNxP Conversation	3	0	Conversation	Conv
BPMNxP Composite Conversatior	4	0	Conversation composite	Comp
BPMNOP Operation	5	1	Opérations	Opera
BPMNOP Organizational Process	6	1	Processus	Proce
BPMNxP Event	7	1	Evènements	Event
BPMNxP Gateway	8	1	Branchements	Gatew
BPMNOP Participant	9	1	Participants	Partic
BPMNxP Data Object	10	1	Objets partagés	Data (
BPMNxP Risks & Controls	13	0	Risques et Contrôles	Risks

- DiagramTypeObjects from the **MetaClasses** tab



Local name	
BPMNxP Sequence Flow	Sequence Flow
BPMNxP Message Flow	Message Flow
BPMNxP Conversation	Conversation
BPMNxP Composite Conversation	Composite Conversation
BPMNxP Conversation Link	Conversation Link
BPMNOP Operation	Operation
BPMNOP Organizational Process	Process
BPMNxP Event	Event
BPMNxP Gateway	Gateway
BPMNOP Participant	Participant
BPMNxP Data Object	Data Object
BPMNOP Process Data Store	Process Data Store

- DiagramTypeLinks from the **Links** tab



Local name			
BPMN Control Implementation Element/Control		Control	Contrc
BPMN OA Operational Rule Enforcement to Oper		Operational Rule Enforceme	Operat
BPMN OA Organizational Process To Operational		Process	Operat
BPMNOP Organizational Process/Embedded Org	1	Process	Proces
BPMNOP Organizational Process/Used Organizat	1	Process	Proces
BPMNOP OrganizationalProcess/Owned Operatic		Process	Operat
BPMNxP Activity/Participants		Participant	BPMN
BPMNxP Annotations		Element with Note	Note
BPMNxP Consumed Data Object	1	Data Object	Data C
BPMNxP Context/OP		Contextualization	Proces
BPMNxP Context/SP	0	Contextualization	Systen

From these tabs you can create objects and consult object properties.

Configuring the display in diagram (DiagramTypeParam)

The diagram display configuration is performed on the DiagramTypeParam. You can set for example:

- **DiagramTypeParam LineStyle**: to define the line style
- **DiagramTypeParam RoundedCorner**: to define the corner style (rounded or not)
- **DiagramTypeParam LanePool**: to use swimlane pools
  Pools enable regular vertical or horizontal presentation of objects.

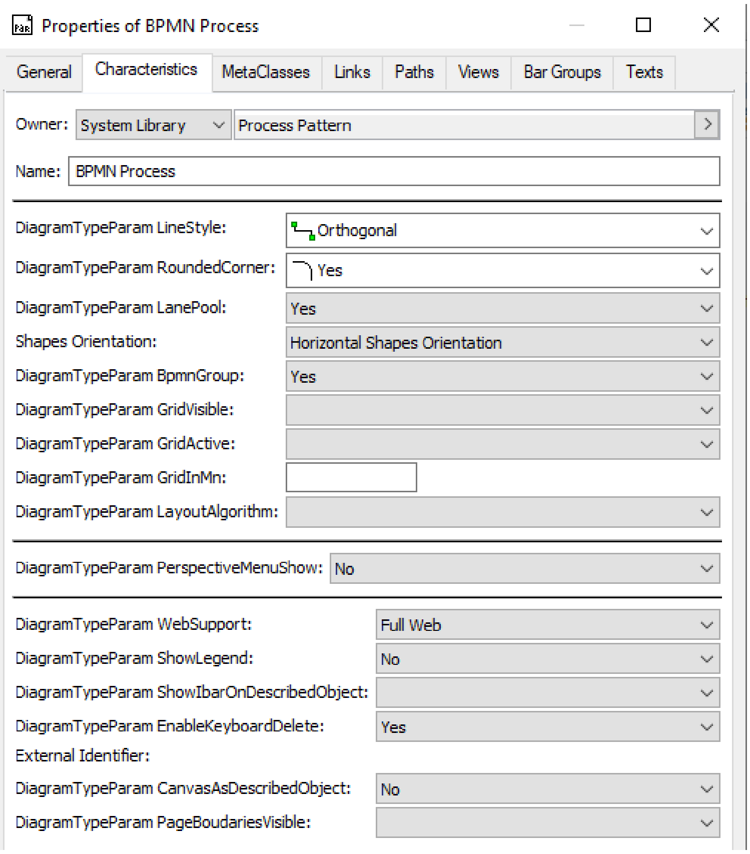
  To be able to place an object in a pool, the Lane attribute should be configured on DiagramTypeObject.
- **DiagramTypeParam GridVisible**: to define the grid visibility

  ☛ *This setting overrides behavior specified in user options (**Tools > Diagrams > Display > Diagram editor grid**).*
- **DiagramTypeParam GridActive**: to activate the grid

  ☛ *This setting overrides behavior specified by user in the diagram (**Align object on grid**)*
- **DiagramTypeParam ShowLegend**: to show the legend
- **DiagramTypeParam ShowIbarOnDescribedObject**: to show the selected object arrows even if the object is the described object. These arrows give access to inserting toolbars to connect objects.
- **DiagramTypeParam pageBondariesVisible**: to display the page boundaries by default

To access general properties related to display in the diagram:

1. Access the **DiagramTypeParam** properties and display its **Characteristics** tab.



Owner: System Library | Process Pattern

Name: BPMN Process

DiagramTypeParam LineStyle: Orthogonal

DiagramTypeParam RoundedCorner: Yes

DiagramTypeParam LanePool: Yes

Shapes Orientation: Horizontal Shapes Orientation

DiagramTypeParam BpmnGroup: Yes

DiagramTypeParam GridVisible:

DiagramTypeParam GridActive:

DiagramTypeParam GridInMn:

DiagramTypeParam LayoutAlgorithm:

DiagramTypeParam PerspectiveMenuShow: No

DiagramTypeParam WebSupport: Full Web

DiagramTypeParam ShowLegend: No

DiagramTypeParam ShowIbarOnDescribedObject:

DiagramTypeParam EnableKeyboardDelete: Yes

External Identifier:

DiagramTypeParam CanvasAsDescribedObject: No

DiagramTypeParam PageBoudariesVisible:

2. Configure the required **DiagramTypeParam** parameters

Creating and Backing Up a Diagram Configuration

Configuration of diagrams is not ensured by **MEGA**, unlike standard customizations (metamodel extensions, shape customizations). It is an advanced configuration that requires:

- validation of configurations by compiling the metamodel in **HOPEX Administration**
- configuration backup
- repeat of the previous steps after each **HOPEX** version update. Diagram configurations are reinitialized at environment updates

Given the complexity of the diagram configuration procedure, configurations should be carried out in an environment other than the production environment.

The simplest method of backing up a diagram configuration is to export it.

```
Example: if you have modified organizational chart
configuration, you should export the "Organizational Chart"
DiagramTypeParam.
```

To apply your configuration to a Diagram Type after updating the environment:

1. In **HOPEX Administration**, connect to the environment concerned with a user having rights to modify **HOPEX** data.
2. Import the exported file containing your customized configurations.
3. Connect the DiagramTypeParam to the corresponding DiagramType.
4. Compile the metamodel.
5. Copy the image files used by your configuration in the Mega_std folder of the **HOPEX** installation.

To view in your working sessionthe result of changes in configuration of diagrams and navigation trees:

1. In the **HOPEX** Start page, click **MetaStudio Console**.
2. In the dialog box that opens, click **Refresh Context**.

You can get the same result with the script editor: enter the ***CurrentEnvironment.refreshcontext*** command and click **Execute**.

Using a New MetaClass in a Diagram

If you consider it necessary to create a MetaClass, you will certainly need to use it in one or several diagrams.

To use a new MetaClass in a diagram:

1. Connect the new MetaClass to the Diagram MetaClass.
2. Create a diagram type view and associate the new MetaClass with this view.
   Alternatively, you can connect the new MetaClass to an existing view.

   ☛ *A view groups one or several object types that can be used in a given diagram type.*
3. Assign an image (shape) to the new MetaClass.
   This shape represents occurrences of the MetaClass in this diagram type.

   The object shape contains a field used to display an attribute (usually the object name). This field can also be configured.
4. Create and configure links that can be displayed in the diagram type concerned.

Creating a view in a diagram type

Before creating a view, in the metamodel diagram connect your new MetaClass to the Diagram MetaClass.

☛ *For more details, see The Metamodel Diagram.*

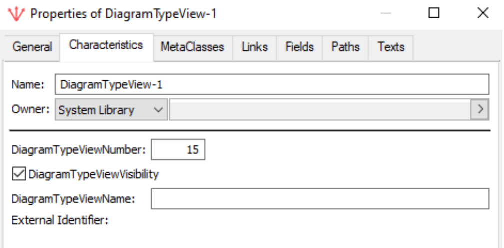
To create a view in a diagram type:

1. In the **MetaStudio** navigation window, expand the **Diagram Types** folder.
2. Expand the folder of the diagram type concerned.

3. Right-click the ***DiagramTypeParam*** concerned and select **Properties**.
4. In the **Views** tab, click **New**.
   A view is created.

In the **Characteristics** tab of the view Properties:

1. In the **DiagramTypeViewNumber** field enter a value (between 1 and 30 inclusive).
2. Select **DiagramTypeViewVisibility** if you want this view to be visible by default in the diagram.



Name: DiagramTypeView-1
Owner: System Library
DiagramTypeViewNumber: 15
☑ DiagramTypeViewVisibility
DiagramTypeViewName:
External Identifier:

Adding a MetaClass to a view

A MetaClass that can be used in a diagram (DiagramTypeObject) must be associated with a view (DiagramTypeView).

To add an object type (DiagramTypeObject) to a view:

1. Open the view property pages.
2. In the **MetaClasses** tab, click **New**.
   A DiagramTypeObject is created. It appears in the list.
3. Open the new DiagramTypeObject property pages.
4. In **Characteristics** tab, **MetaClass** field, click the arrow and select **Connect MetaClass**.
5. From the **Query** tool select the MetaClass that interests you and click **OK**.
   The new DiagramTypeObject is now connected to the MetaClass you selected.

> All DiagramTypeObjects must be connected to the Diagram MetaClass by the DiagramTypeLinkEnd MetaAssociationEnd.

Defining a shape for the new MetaClass

To define the shape that will be used for the new MetaClass:

1. In the DiagramTypeObject Property pages, select the **Characteristics** tab.
2. In the **DiagramTypeShape** box, enter the file mask to be used to graphically represent the object.
3. In the **MetaPicture** tab, click **Connect** to connect the image that interests you.

Defining a field in the new shape

The new shape you have created contains at least one field - that which enables display of the object name.

To define this field:

1. In the DiagramTypeObject Property pages, select the **Fields** tab.
2. Click **New** to create a new field (DiagramTypeField).
3. Open the DiagramTypeField Property pages and select the **Characteristics** tab.
4. In the **DiagramTypeField Name** box, specify the **&name&** value representing the name of the variable defined in the shape supplied by **HOPEX**.
5. In the **Properties** tab, click **New** to create a new DiagramTypeProperty. Give it the same name as the MetaPicture.
6. In the DiagramTypeProperty Properties window, **Characteristics** tab, select the MetaAttribute to be displayed (in this case, **Name**).
7. Click **OK** to close all dialog boxes.

Field limit

Do not add more than 12 **DiagramTypeFields** in a **DiagramTypeObject**.

If you need more than 12 **DiagramTypeFields**, create as many **DiagramTypeProperties** as needed and group them in less than 12 **DiagramTypeFields**.

Adding a Zoom to a Diagram Type

A ***DiagramTypeZoom*** enables a MetaClass to be described by a diagram type.

If you want a given diagram type to describe a new MetaClass, you must:

- connect to the diagram type a new DiagramTypeZoom
- connect the desired MetaClass to the DiagramTypeZoom
- connect to the DiagramType the MetaAssociationEnd "Described <MetaClass>" ("Description.<MetaClass>")

To add a zoom to a diagram type:

1. In the **MetaStudio** navigation window, right-click the diagram type and select **New > DiagramTypeZoom**.
2. Enter the name of the DiagramTypeZoom and click **OK**.
3. Right-click the DiagramTypeZoom and select **Properties**.
4. Display the empty collections and right-click **MetaClass > Connect**.
5. Connect the desired MetaClass.
6. In the **MetaStudio** navigation window, right-click the diagram type and select **Connect > MetaAssociationEnd**.
7. Connect the "Described <MetaClass>" MetaAssociation. Connect the desired <MetaClass> using the query tool.

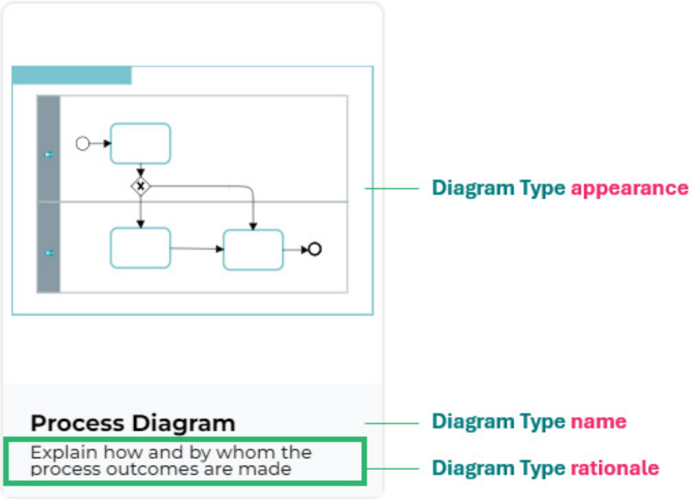
The new MetaClass can now be described by the diagram type concerned.

Modifying a Diagram Type

Modifying the Diagram Type tile displayed at creation

You can configure the Diagram Type information displayed at diagram creation:

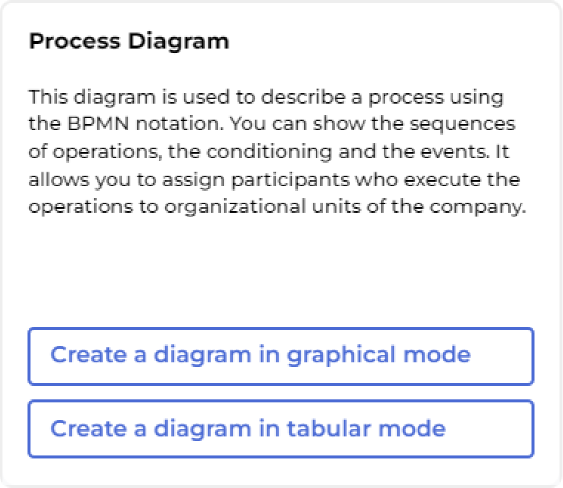
- the Diagram Type ***name***
- the Diagram Type ***rationale***



Process Diagram

Explain how and by whom the process outcomes are made

- the Diagram Type ***description***
  Hovering the cursor over the tile displays its ***description***.



Process Diagram

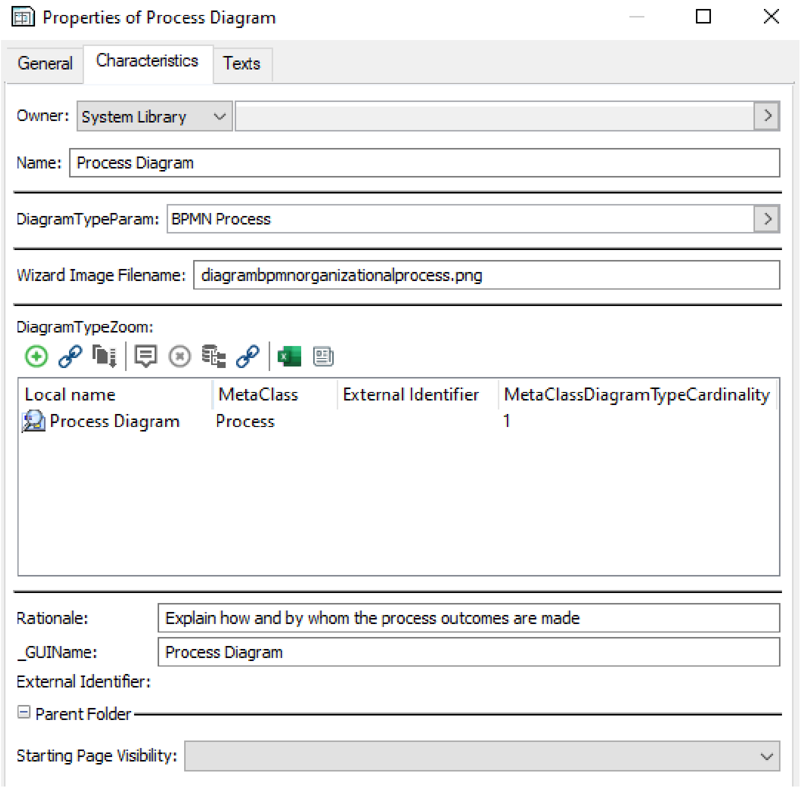
This diagram is used to describe a process using the BPMN notation. You can show the sequences of operations, the conditioning and the events. It allows you to assign participants who execute the operations to organizational units of the company.

Create a diagram in graphical mode
Create a diagram in tabular mode

To modify a Diagram Type display:

1. Access the Diagram Type properties.

   ☛ *See Accessing Diagram Types.*



Properties of Process Diagram

Owner: System Library

Name: Process Diagram

DiagramTypeParam: BPMN Process

Wizard Image Filename: diagrambpmnorganizationalprocess.png

DiagramTypeZoom:

Local name	MetaClass	External Identifier	MetaClassDiagramTypeCardinality
Process Diagram	Process		1

Rationale: Explain how and by whom the process outcomes are made

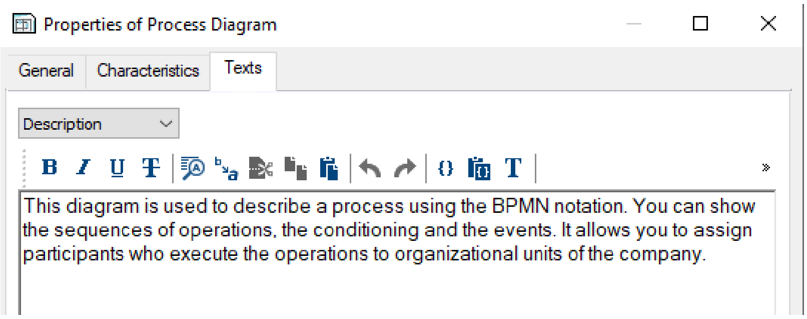
_GUIName: Process Diagram

External Identifier:

⊟ Parent Folder

Starting Page Visibility:

2. In **Characteristics** tab, to modify:
   - the Diagram Type name displayed: modify its **_GUIName**
   - the Diagram Type rationale: modify its **Rationale**
3. In **Texts** tab, to modify:
   - the Diagram Type description: modify its **Description**

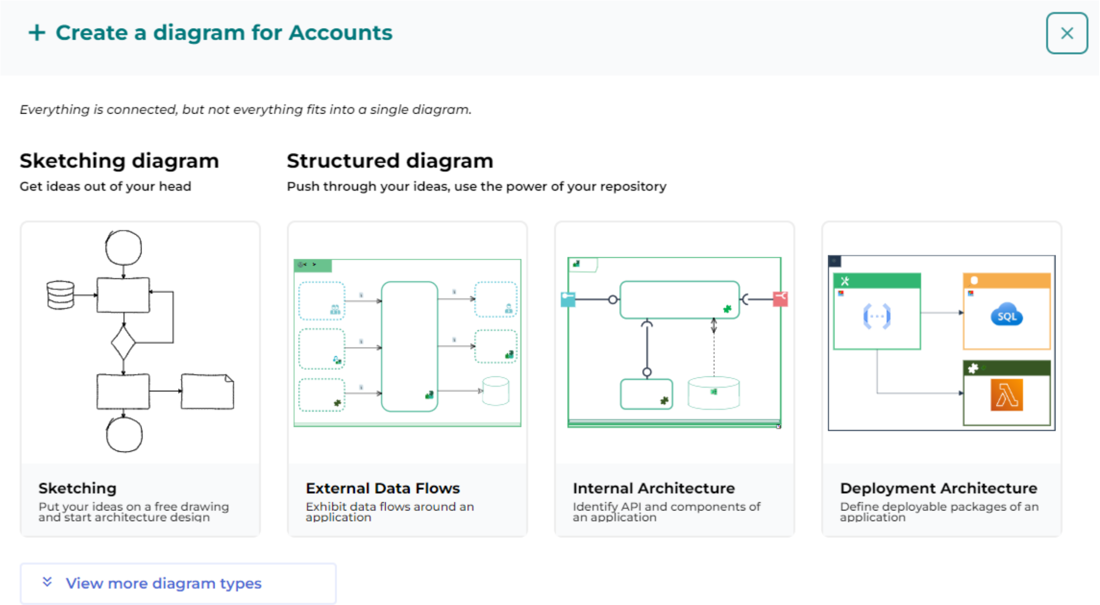


Properties of Process Diagram

This diagram is used to describe a process using the BPMN notation. You can show the sequences of operations, the conditioning and the events. It allows you to assign participants who execute the operations to organizational units of the company.

Modifying the Diagram Types displayed by default at creation

At diagram creation, a maximum of three Diagram Types are displayed by default. More Diagram Types are also available with the **View more diagram types**.



+ Create a diagram for Accounts

×

Everything is connected, but not everything fits into a single diagram.

Sketching diagram

Get ideas out of your head

Structured diagram

Push through your ideas, use the power of your repository

Sketching
Put your ideas on a free drawing and start architecture design

External Data Flows
Exhibit data flows around an application

Internal Architecture
Identify API and components of an application

Deployment Architecture
Define deployable packages of an application

View more diagram types

Each profile is associated with a **Covered Domain** (methodological domain), whose **DiagramType Presentations** define for each MetaClass its Diagram Type display rule:

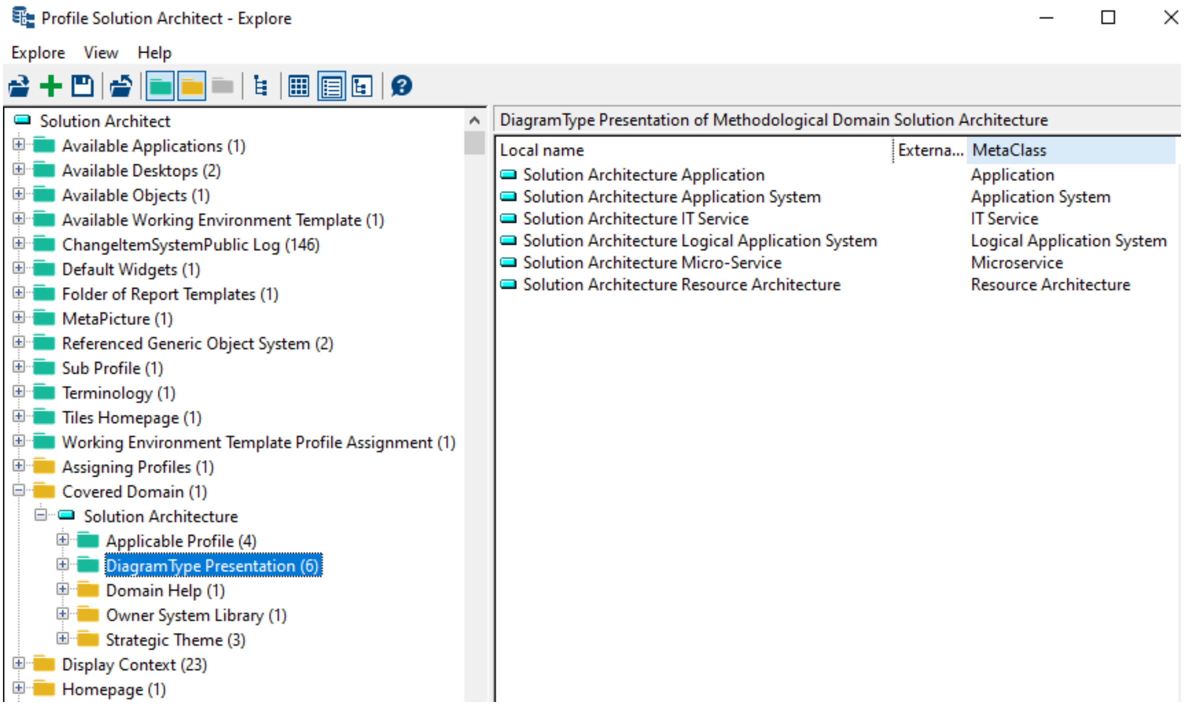
- the Diagram Types displayed by default
  These Diagram Types are the first three main Diagram Types (**DiagramType IsMain** = "Yes")
- the other available Diagram Types
  These Diagram Types are not main Diagram Types (**DiagramType IsMain** = "No")
- the Diagram Type display order

To modify the Diagram Type display rule:

1. Explore the **Profile** concerned and expand **Covered Domain > DiagramType Presentation**.
   Each DiagramType Presentation defines for a MetaClass its Diagram Type presentation rule.

   E.g.: the **Solution Architect** profile is associated with the **Solution Architect** Covered Domain.

   The **Solution Architect** Covered Domain includes six **DiagramType Presentations.** Each of them defines the Diagram Types proposed at creation for its related MetaClass.



Profile Solution Architect - Explore

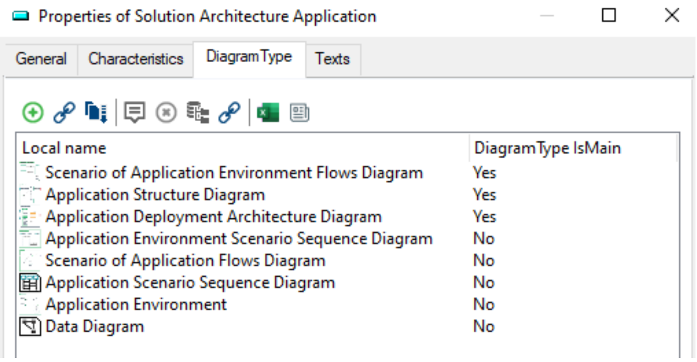
- Solution Architect
  - Available Applications (1)
  - Available Desktops (2)
  - Available Objects (1)
  - Available Working Environment Template (1)
  - ChangeItemSystemPublic Log (146)
  - Default Widgets (1)
  - Folder of Report Templates (1)
  - MetaPicture (1)
  - Referenced Generic Object System (2)
  - Sub Profile (1)
  - Terminology (1)
  - Tiles Homepage (1)
  - Working Environment Template Profile Assignment (1)
  - Assigning Profiles (1)
  - Covered Domain (1)
    - Solution Architecture
      - Applicable Profile (4)
      - DiagramType Presentation (6)
      - Domain Help (1)
      - Owner System Library (1)
      - Strategic Theme (3)
  - Display Context (23)
  - Homepage (1)

DiagramType Presentation of Methodological Domain Solution Architecture

Local name	Externa...	MetaClass
Solution Architecture Application		Application
Solution Architecture Application System		Application System
Solution Architecture IT Service		IT Service
Solution Architecture Logical Application System		Logical Application System
Solution Architecture Micro-Service		Microservice
Solution Architecture Resource Architecture		Resource Architecture

2. Access the DiagramType Presentation properties.

   E.g.: the **Solution Architecture Application** DiagramType Presentation details the eight **Diagram Types** available at creation for the **Application** MetaClass. The first three "Main Diagran Types" are proposed by default at creation.



Local name	DiagramType IsMain
Scenario of Application Environment Flows Diagram	Yes
Application Structure Diagram	Yes
Application Deployment Architecture Diagram	Yes
Application Environment Scenario Sequence Diagram	No
Scenario of Application Flows Diagram	No
Application Scenario Sequence Diagram	No
Application Environment	No
Data Diagram	No

3. To modify the Diagram Type display, in its corresponding **DiagramTypeMain** field, select:
   - "Yes": the Diagram Type is a available as default Diagram Type
   - "No": the Diagram Type is available
4. To modify the Diagram Type display order:
   - Click **Reorganize** .
   - Drag and drop the Diagram Types so as to get another display order.
   - Click **OK**.

Modifying a Diagram Type Multiplicity

At creation, by default, the Diagram Type has no multiplicity restriction.

The end-user can use several times the same Diagram Type to describe an object. For data consistency, you may want to restrict this multiplicity, to one for example.

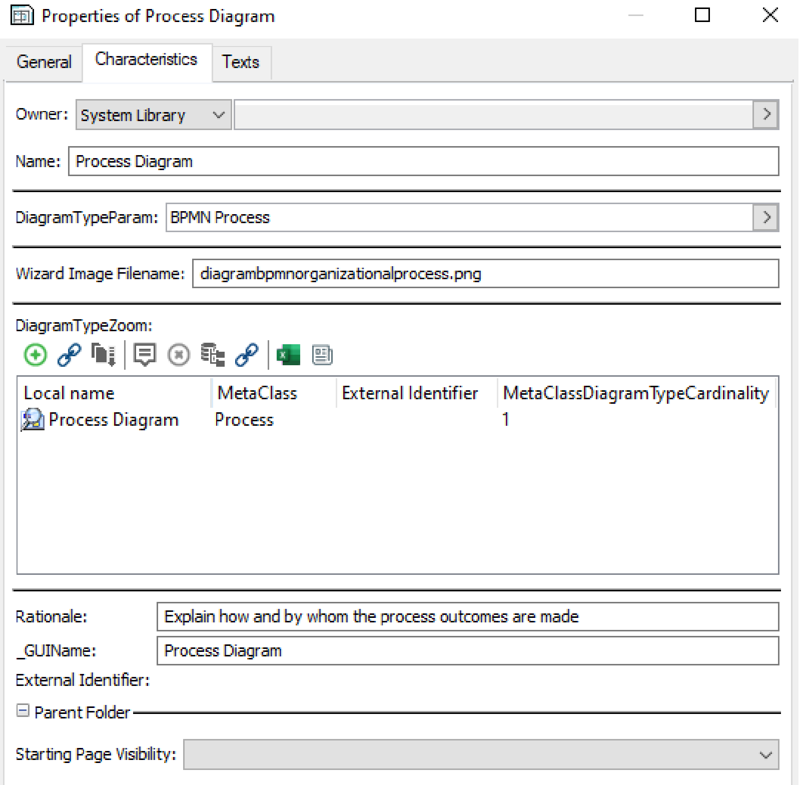
To configure a Diagram Type multiplicity:

1. Access the Diagram Type properties.

   ☛ *See Accessing Diagram Types.*

2. In the **DiagramTypeZoom** section, define the **MetaClassDiagramTypeCardinality** value.

   E.g.: "1" to restrict the creation to one.



Owner: System Library

Name: Process Diagram

DiagramTypeParam: BPMN Process

Wizard Image Filename: diagrambpmnorganizationalprocess.png

DiagramTypeZoom:

Local name	MetaClass	External Identifier	MetaClassDiagramTypeCardinality
Process Diagram	Process		1

Rationale: Explain how and by whom the process outcomes are made

_GUIName: Process Diagram

External Identifier:

⊟ Parent Folder

Starting Page Visibility:

Modifying the distance between the source object and the created one

By default, when an object is added from another object inserting toolbar, it is created at 120 px from the source object.

To modify the default distance between the source and created objects:

1. Access the **DiagramTypeLink** or **DiagramTypePath** properties.
2. In its **Texts > _Settings**, enter:

```
[General]
AutoCreationSpaceH=100
AutoCreationSpaceV=200
```

With:

AutoCreationSpaceH: horizontal space from left to right between both objects.

AutoCreationSpaceV: vertical space from top to bottom between both objects.

Modifying the Suggested Objects

You can:

- remove a suggested object from the object inserting toolbar
- add suggested objects
- add suggested objects for usage object
- add candidate objects

These configurations are performed in the **DiagramTypeObject** properties.

Removing an object suggested in the object inserting toolbar

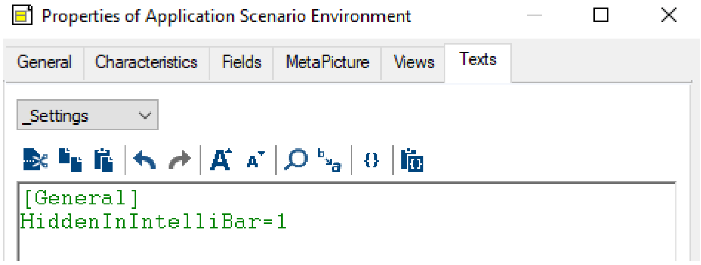
In a diagram, you can add an object from another object inserting toolbar. As some object types are not often used, or used only once in a diagram, they do not need to be added in the suggested objects of the object inserting toobar.

You can remove a suggested object from the object inserting toolbar.

To remove an object suggested in the object inserting toolbar:

1. Access the **DiagramTypeObject** properties.
2. In its **Texts > _Settings** enter:

```
[General]
HiddenInIntellibar=1
```



```
[General]
HiddenInIntelliBar=1
```

The object is no longer suggested in the object inserting toolbar.

Adding suggested objects

You can suggest the objects to add in a diagram.

As these suggested objects are computed before opening the **Add <Object>** window, queries must be performant.

To add suggested objects:

1. Access the **DiagramTypeObject** properties.

2. In its **Texts > _Settings** enter:

```
[General]
QuerySuggested=~xxxx[query name]
```

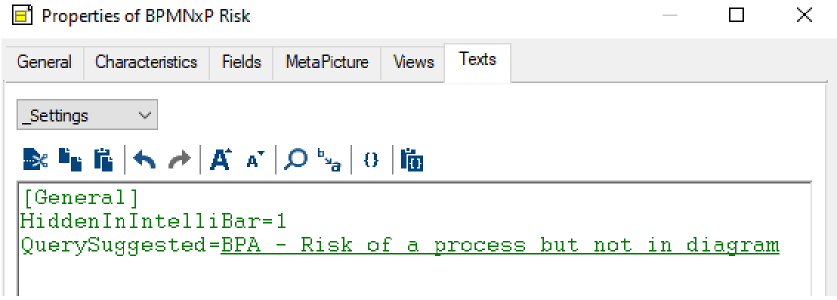
with the following available parameters:

- Described Object: "&DescribedObject"
- Diagram: "&DiagramObject"
- Object From: "&FromObject"
- Object Target: "&ToObject"

Example for the "BPMNxP Risk" DiagramTypeObject:

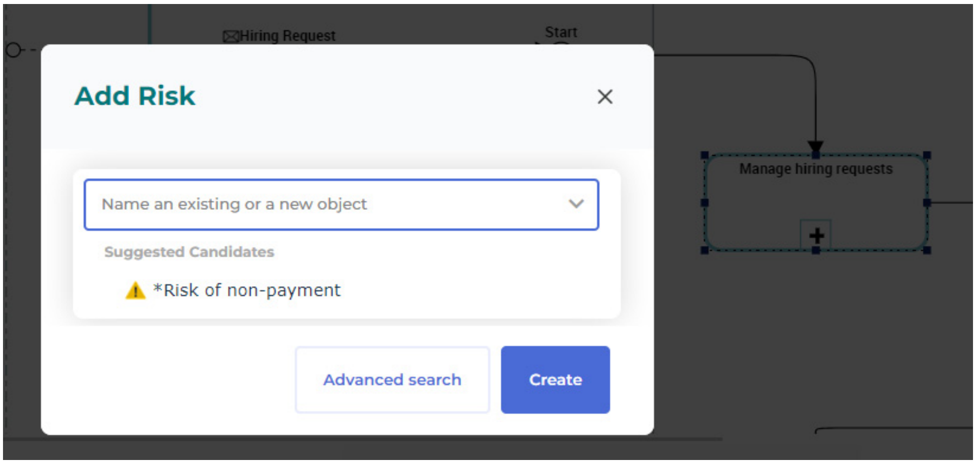
```
[General]
QuerySuggested=~u5)ZVKfGarP0[BPA - Risk of a process but not in diagram]
```

This query suggests to add the risks already linked to the process but not displayed in the diagram.



```
[General]
HiddenInIntelliBar=1
QuerySuggested=BPA - Risk of a process but not in diagram
```

Here the "Manage hiring requests" process is linked to the "*Risk of non-payment", but this risk is not displayed in the diagram. Thanks to the query it is suggested in the **Suggested Candidates** of the **Add Risk**.



Adding suggested objects for usage object

You can suggest the objects to add in a diagram.

These suggested objects are computed before opening the **Add <Object>** window. Suggested queries must be performant.

To add suggested objects for usage object:

1. Access the **DiagramTypeObject** properties.
2. In its **Texts > _Settings** enter for example:

```
[General]
UsedObjectQuerySuggested=~xxxx[query name]
```

with the following available parameters:

- Described Object: "&DescribedObject"
- Diagram: "&DiagramObject"
- Object From: "&FromObject"
- Object Target: "&ToObject"

```
Example for the "BPMNBP Participant" DiagramTypeObject:
[General]
UsedObjectQuerySuggested=~27)Z1ygGaHE2[BPA - children org
units of the org unit responsible for the described process]
```

with the following query code:

```
Select [Org-Unit] Into @root Where [Assigned
Participant].[BPMN Element]:[Process] = &"DescribedObject"
Select [Org-Unit] Where [Aggregation of] Deeply in @root
```

Adding candidate objects

You can define candidate objects to add in a diagram. This is not as useful as a suggested object.

The query is computed when clicking the drop-down list. Candidate queries must be performant.

To add candidate objects:

1. Access the **DiagramTypeObject** properties.
2. In its **Texts > _Settings** enter for example:

```
[General]
QueryCandidate=~xxxx[query name]
```

with the following available parameters:

- Described Object: "&DescribedObject"
- Diagram: "&DiagramObject"
- Object From: "&FromObject"
- Object Target: "&ToObject"

Adding the Creation Wizard when Adding an Object

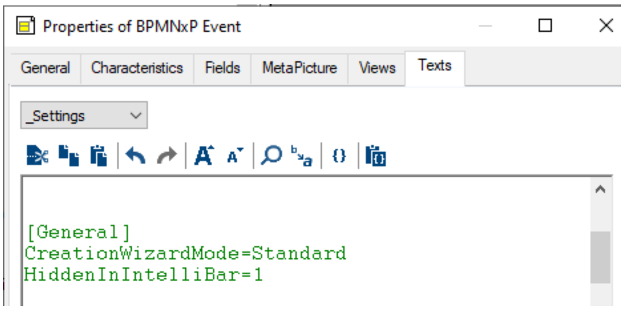
By default when you add a new object in a diagram, its creation wizard is not launched. You can configure the object so that its wizard creation is launched.

To add the wizard creation when adding an object to a diagram:

1. Access the DiagraTypeObject properties.
2. In its **Texts > _Settings**, enter:

   ```
   [General]
   CreationWizardMode=Standard
   ```

   For example for the BPMN Event DiagramTypeObject:



```
[General]
CreationWizardMode=Standard
HiddenInIntelliBar=1
```

Configuring MetaAssociations

Creating a MetaAssociation

A DiagramTypeLink enables representation of a MetaAssociation (link) in a diagram. You cannot draw a link between two objects in a diagram if the corresponding MetaAssociation does not already exist in the metamodel.

To create a DiagramtypeLink that could be displayed in the diagram type that interests you:

1. In the **DiagramTypeParam** property pages, select the **Links** tab.
2. Click **New** to create a DiagramTypeLink.
3. Access the **DiagramTypeLink** property pages > **Characteristics** tab.
4. in the the **MetaAssociationEnd** field click the arrow and select **Connect MetaAssociationEnd**.
5. Use the **Connecting** wizard to **Connect** the MetaAssociationEnd that interests you.
6. In the **DiagramTypeLink** property pages, select **Unidirectional** if you want the MetaAssociation to have only one valid direction.

Configuring a link display

You can customize the display of links in a diagram using attributes accessible in the **MetaAssociation** Property pages (DiagramTypeLink).

To access configuration of links in a diagram:

1. In the **MetaStudio** navigation window, access the **DiagramTypeParam** Property pages on which the link concerned depends, and select its **Links** tab.
2. Access the property pages of the **Link** concerned > **Characteristics** tab.
3. You can configure the following display characteristics:
   - **DiagramTypeLink LineStyle**: link line style
   - **DiagramTypeLink LineBeginStyle**: default arrow style on first link end.
   - **DiagramTypeLink LineEndStyle**: default arrow style on second link end.
   - **DiagramTypeLink LineDouble**: specifies default link style as double line.
   - **DiagramTypeLink RoundedCorner**: specifies default line style as rounded angles.
   - **DiagramTypeLink PenColor**: default line color.
   - **DiagramTypeLink PenStyle**: default line style (solid, dotted, etc.).
   - **DiagramTypeLink PenSize**: default line width.
   - **DiagramTypeLink BrushColor**: for double lines and/or arrows with fill, specifies default fill color.
   - **DiagramTypeLink InsideConnection**: specifies if the repository link must be represented and managed in any special way.
   - **DiagramTypeLink ReverseInReorganization**: for automatic reorganization function, enables definition of hierarchical direction of link.

CONFIGURING TABULAR ENTRY MODE

Tabular entry mode, available with **HOPEX Web Front-End**, is used to define the content of a diagram through a table.

The characteristics of the objects described in the diagram, or the links between the objects, are defined in the table columns.

The diagrams that can be updated in table entry mode as standard are as follows:

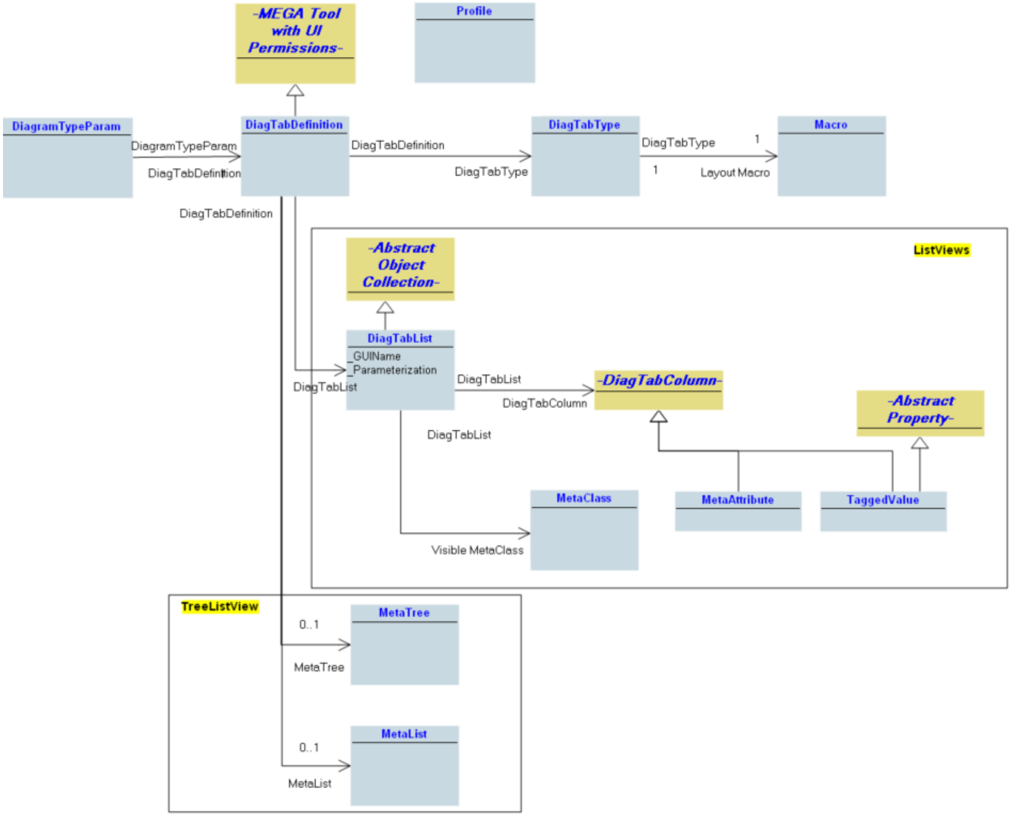
- Business process diagrams
- Functional process diagrams
- Process diagrams
- Application process diagrams
- Organizational charts
- Customer journey mapping.

You can configure tabular entry mode for any existing Diagram Type.

This section explains how to create your own customizations to edit a diagram with tabular entry mode.

Tabular Entry Mode Concepts

Tabular entry diagram configuration uses concepts presented in the following MetaModel.



Configurations of a diagram type are carried by a ***DiagramTypeParam*** associated with multiple ***DiagTabDefinitions***.

DiagTabDefinitions apply to specific profiles. The ***DiagTabDefinition*** description is based on the following concepts:

- ***DiagTabType***: used for diagrams whose presentation is calculated by a macro
- ***DiagTabList***: used to describe the fields of a tabular entry diagram
- ***MetaTree*** and ***MetaList***: used to describe the fields of a tree tabular entry diagram

To specify that a diagram can be edited with tabular entry mode, proceed as follows:

1. Create a ***DiagTabDefinition*** and link it to a diagram type
2. Connect the dedicated profiles, see Defining the Authorized Profiles
3. Create the presentation macro, see Using the Layout Macros

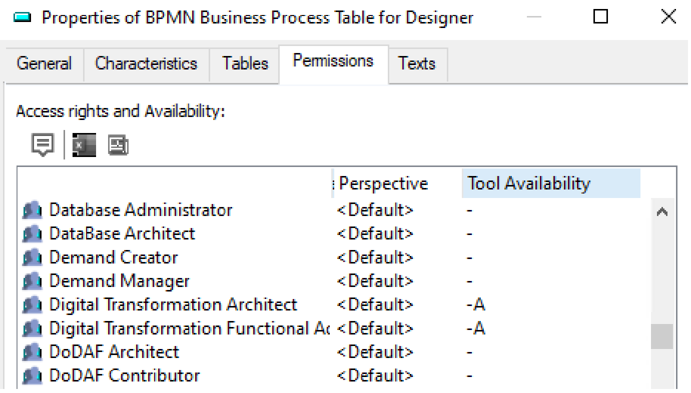
4. Depending on the type of diagram, specify the fields corresponding to:
   - a tabular entry diagram, see Specifying a Tabular Entry Structure;
   - a tree tabular entry diagram, see Specifying a tree entry structure.

Defining the Authorized Profiles

The profiles authorized to use the tabular entry mode for a diagram is defined on the ***DiagTabDefinition***.

To associate a profile with a ***DiagTabDefinition***:

1. Access the ***DiagTabDefinition*** properties.
2. In the **Permissions** page, select the profile you want to associate with the diagram and in the **Tool Availability** column, set "Available".



Access rights and Availability:

	Perspective	Tool Availability
Database Administrator	<Default>	-
DataBase Architect	<Default>	-
Demand Creator	<Default>	-
Demand Manager	<Default>	-
Digital Transformation Architect	<Default>	-A
Digital Transformation Functional A	<Default>	-A
DoDAF Architect	<Default>	-
DoDAF Contributor	<Default>	-

Using the Layout Macros

The layout macros are used to compute the diagram presentation each time it is refreshed, for example, to compute a horizontal or a vertical representation of a diagram.

A layout macro is written in JAVA using the *diagram plug-in* method.

*A diagram plug-in macro must be implemented in a **HOPEX Macro** repository object. It defines a number of functions which are called in response to events occurring in the diagram.*

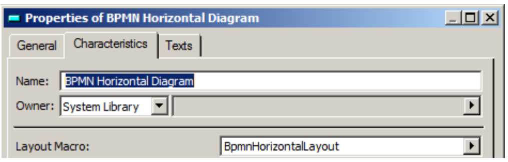
*For more details on the diagram plug-in method, see the "Writing a diagram plug-in" chapter in the **HOPEX Studio** guide.*

Those macros are linked to a **DiagTabDefinition** through **DiagTabType** objects.

To associate a layout macro with a ***DiagTabType***:

1. Open the **Characteristics** property page of the ***DiagTabType*** that interests you.

2. In the **Layout Macro**, select the macro you want to associate to the ***DiagTabType***.

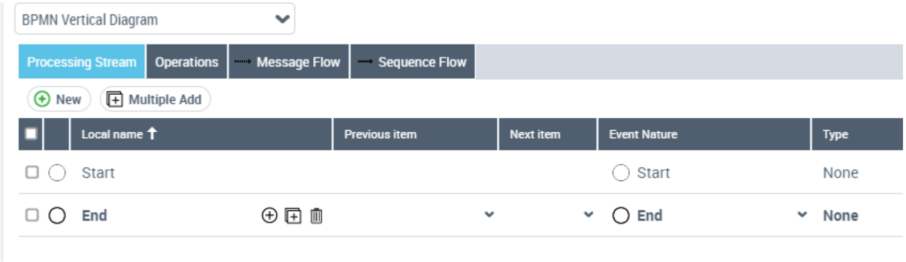


Name: BPMN Horizontal Diagram
Owner: System Library
Layout Macro: BpmnHorizontalLayout

Specifying a Tabular Entry Structure

To specify the table used to describe the component parts of a tabular entry diagram, specify:

- the order of tabs associated with the ***Metaclasses*** represented,
- the content of columns for a given ***Metaclass*** and display order,
- the display of a tab that provides an overview of all the objects in the diagram (***MetaAttributes***, ***MetassociationEnd***, ***Path***, for example).



□		Local name ↑	Previous item	Next item	Event Nature	Type
□	○	Start			○ Start	None
□	○	End			○ End	None

Example of the Process Tabular Diagram

The ***DiagTabList*** description is based on the following concepts:

- ***Visible MetaClass***: groups the concrete ***Metaclasses*** described in the tabs, see DiagTabList Properties.
- ***DiagTabColumn***: used to describe the columns associated with the ***Metaclass*** and proposed in the corresponding tab. See DiagTabColumn Parameterization.

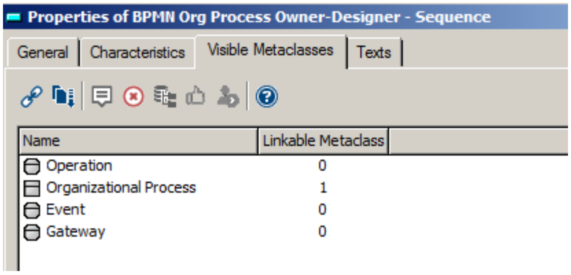
  A column can be represented by:
  - a ***MetaAttribute*** to describe the MetaAttribute corresponding to the column,
  - a ***MetaAssociationEnd*** to describe the MetaAssociationEnd corresponding to the column,
  - or a ***TaggedValue*** to specify the name of the column.

See :

- DiagTabList Properties
- DiagTabColumn Parameterization

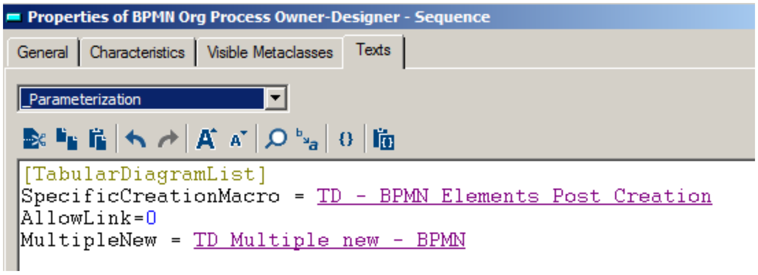
DiagTabList Properties

The list of ***Metaclasses*** described in tabs is defined in the **Visible MetaClasses** property page of the ***DiagTabList*** object.



Name	Linkable Metaclass
Operation	0
Organizational Process	1
Event	0
Gateway	0

You can adjust your list with the buttons and tools of interest by specifying a **[TabularDiagramList]** in the **_Parameterization** section of the **Texts** property page of the ***DiagTabList*** object.



```
[TabularDiagramList]
SpecificCreationMacro = TD - BPMN Elements Post Creation
AllowLink=0
MultipleNew = TD Multiple new - BPMN
```

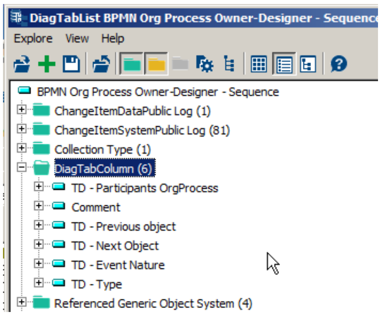
The following table shows the parameterization and the corresponding functionalities provided for ***DiagTabList*** objects.

Parameterization	Functionality
SpecificCreationMacro	Takes a macro as argument. The Macro implements a specific creation for he ***New*** button in the list view.
MultipleNew	Takes a MetaWizard as an argument. The MetaWizard is called by the ***Multiple Add*** button, see Using Multiple Add.
AllowAdd	Takes “0” or “1” to hide or show the ***New*** button.
AllowDelete	Takes “0” or “1” to hide or show the ***Delete*** button.
AllowLink	Takes “0” or “1” to hide or show the ***Connect*** button.
AllowUnLink	Takes “0” or “1” to hide or show the ***Disconnect*** button.

DiagTabList Parameterization

DiagTabColumn Parameterization

DiagTabColumns are used to customize the columns of the tabular diagram table.



BPMN Org Process Owner-Designer - Sequence
- ChangeItemDataPublic Log (1)
- ChangeItemSystemPublic Log (81)
- Collection Type (1)
- DiagTabColumn (6)
  - TD - Participants OrgProcess
  - Comment
  - TD - Previous object
  - TD - Next Object
  - TD - Event Nature
  - TD - Type
- Referenced Generic Object System (4)

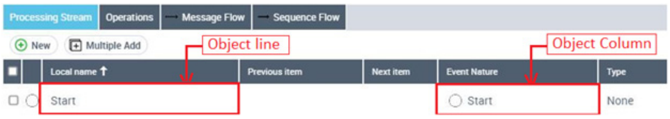
*List of **DiagTabColumns** conected to a **DiagTabList***

The columns may be associated with:

- a ***MetaAttribute***; “Comment” for example, in this case, the ***MetaAttribute*** must be connected to the corresponding ***DiagTabList***,
- or a ***MetaAssociationEnd*** or a ***Path***; in this case you have to create a dedicated ***TaggedValue*** and specify a set of parameterizations to describe the way it works:
  - If the link between the row object and the column object is direct, see Direct links;
  - If an intermediate object exists between the row object and the column object, see Indirect link;
  - To customize your column presentation and tools, see Specific cases.

Direct links

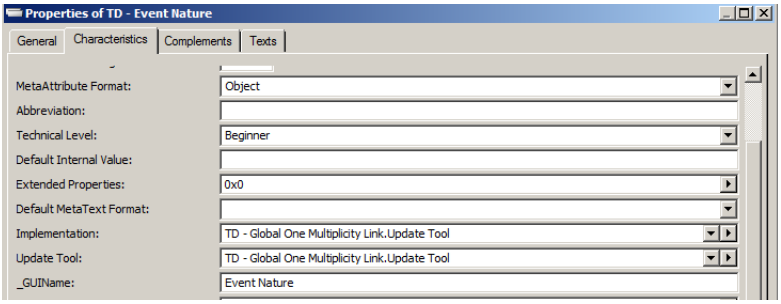
For example, the “TD - Event Nature” ***DiagTabColumn*** represents the link between an event and its nature.



□		Local name ↑	Previous item	Next item	Event Nature	Type
□	○	Start			○ Start	None

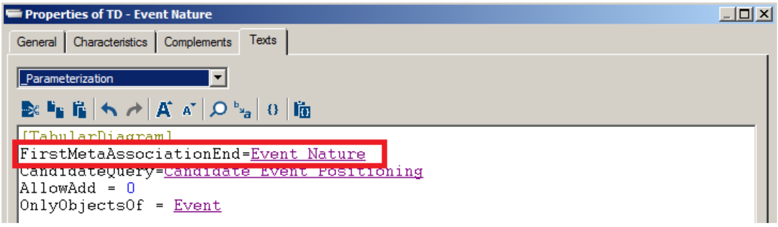
To specify the parameters of a direct link between the row object and the column object:

1. Open the **Characteristics** property page of the ***TaggedValue*** object and, in the **Implementation** and **UpdateTool** fields, connect one of the following macros:
   - ***~3Von8cy1NniM[TD - Global One Multiplicity Link.Update Tool]*** if link multiplicity is equal to "1",
   - ***~mBKk9WGUTbQP[TD - Direct Relation UpdateTool]***, if link multiplicity is different from"1".



MetaAttribute Format: Object
Abbreviation:
Technical Level: Beginner
Default Internal Value:
Extended Properties: 0x0
Default MetaText Format:
Implementation: TD - Global One Multiplicity Link.Update Tool
Update Tool: TD - Global One Multiplicity Link.Update Tool
_GUIName: Event Nature

2. Open the **Texts** property page of the ***TaggedValue*** object and, in the **_Parameterization** section, use the **[TabularDiagram]** command to specify the link used for column calculation.



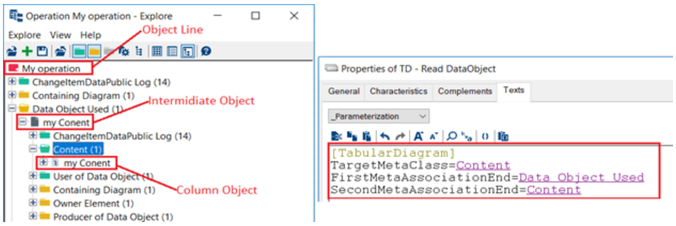
```
[TabularDiagram]
FirstMetaAssociationEnd=Event Nature
CandidateQuery=Candidate Event Positioning
AllowAdd = 0
OnlyObjectsOf = Event
```

Indirect link

To specify the parameters of a link with an intermediate object between the row object and the object column:

1. Open the ***Characteristics*** property page of the ***TaggedValue*** object, in the **Implementation** and **UpdateTool** fields, connect the ***~mBKk9WGUTbQP[TD - Direct Relation UpdateTool]*** macro;

2. Open the **Texts** property page of the ***TaggedValue*** object, in the **_Parameterizations** section, use the **[TabularDiagram]** command to specify:
   - the ***FirstMetaAssociationEnd*** that refers to the link between the row object and the intermediate object,
   - the ***SecondMetaAssociationEnd*** that refers to the link relating the intermediate object and column object,
   - the ***TargetMetaClass*** to specify the type of the column object.



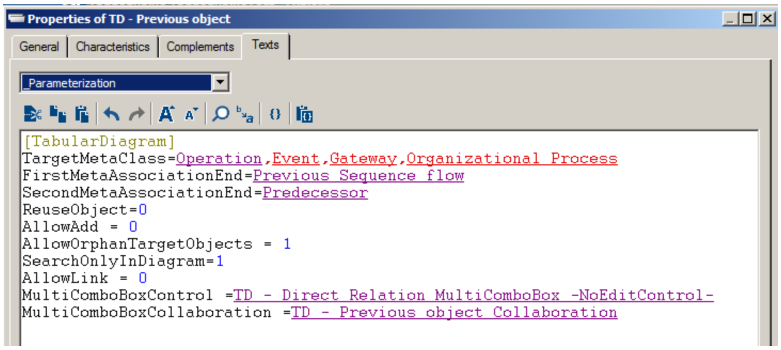
```
[TabularDiagram]
TargetMetaClass=Content
FirstMetaAssociationEnd=Data Object Used
SecondMetaAssociationEnd=Content
```

Specific cases

If you want to manage cases different from those previously shown, you need to create your own macro for ***Implementation*** and ***UpdateTool***.

To customize your columns presentation and tools with a set of key elements:

1. In the ***Texts*** property page of the ***TaggedValue*** object, use the **[TabularDiagram]** command in the **_Parameterization** section.



```
[TabularDiagram]
TargetMetaClass=Operation,Event,Gateway,Organizational Process
FirstMetaAssociationEnd=Previous Sequence flow
SecondMetaAssociationEnd=Predecessor
ReuseObject=0
AllowAdd = 0
AllowOrphanTargetObjects = 1
SearchOnlyInDiagram=1
AllowLink = 0
MultiComboBoxControl =TD - Direct Relation MultiComboBox -NoEditControl-
MultiComboBoxCollaboration =TD - Previous object Collaboration
```

The table below shows the parameterization and the corresponding functionalities provided for ***DiagTabColumn*** objects.

Parameterization	**Functionality**
TargetMetaClass	Takes a set of ***MetaClasses*** separated with "," as an argument. Used to specify the type(s) of column objects.
FirstMetaAssociationEnd	Takes a ***MetaAssociationEnd*** as an argument. Used to specify the link between: - the row object and column object when the link is direct - the row object and the intermediate object when the link is indirect.
SecondMetaAssociationEnd	Takes a ***MetaAssociationEnd*** as an argument. Used to specify the link between: - the row object and column object when the link is direct, or - the row object and the intermediate object when the link is indirect.
AllowAdd	Takes "0" or "1". Used to specify if the user can create a new element in the column.
AllowLink	Takes "0" or "1". Used to specify if the user can create a new element in the column.
MetaTest	Takes a ***MetaTest*** as an argument. Used to exclude a column according to a condition.
CandidateQuery	Takes a ***MetaTest*** as an argument. Used to exclude a column according to a condition.
OnlyObjectsOf	Takes a ***MetaClass*** as an argument. Indicates that the column will be available only to this specified ***MetaClass***. For other types it is be disabled.
ReuseObject	Takes "0" or "1". Used when the link is indirect. Enables to specify whether the intermediate object is reused or whether we create a new one on any Create or Connect operation.
SearchOnlyInDiagram	Takes "0" or "1". Used to limit a ***DropDownList*** to elements that already exist in the diagram. For example the **Previous Item** and **Next Item** object cannot suggest objects which are not in the diagram.

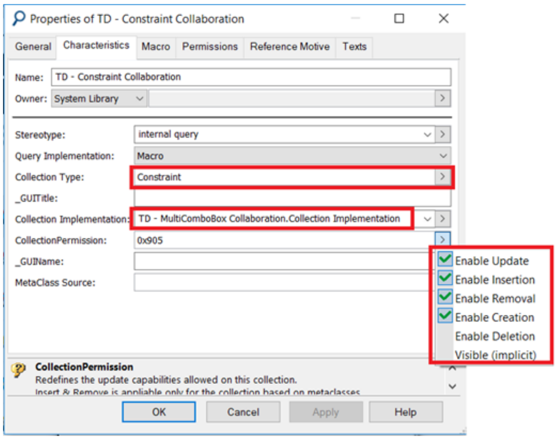
Parameterization	**Functionality**
NewParticipants	Takes “Internal” or “External”. Used to specify the type of participant to create or connect but it is useful when the column groups **Participants**.
MultiComboBoxControl	Takes a macro as an argument. Used to specify the multicombobox control in the column. The ***~b8KkWhBUTPLP[TD - Direct Relation MultiComboBox]*** specific macro is used in all existing tabular diagrams. This macro takes into account all the key elements mentioned earlier. However you can specify your own macro.
MultiComboBoxCollaboration	Takes a ***Query*** as an argument. It is mandatory when you have a multicombobox control. There are some configurations you will need bring to your query, see MultiComboBoxCollaboration Parameters.

DiagTabColumn Parameterization

MultiComboBoxCollaboration Parameters

To specify a multicombobox controlled by a query in a column:

1. Connect the macro ***~fLJGeQ4YTfjH[TD - MultiComboBox Collaboration.Collection Implementation]*** in the **Collection Implementation** field,
2. Set the **Collection Permission**.
3. Define the ***~hQxd6kwcTTW9[TD - Global Collaboration UIManager]*** macro as ***UIManager*** for the query if you have creation permission.



Name: TD - Constraint Collaboration
Owner: System Library
Stereotype: internal query
Query Implementation: Macro
Collection Type: Constraint
_GUITitle:
Collection Implementation: TD - MultiComboBox Collaboration.Collection Implementation
CollectionPermission: 0x905
_GUIName:
MetaClass Source:

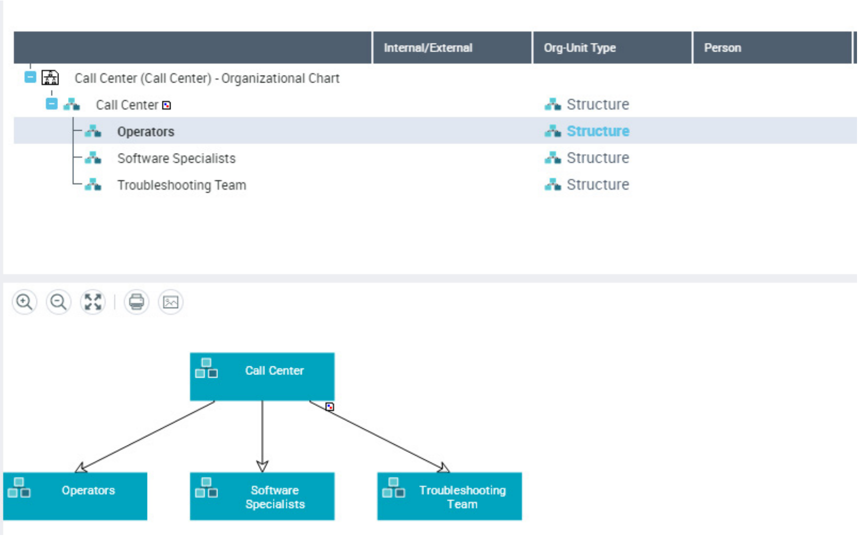
☑ Enable Update
☑ Enable Insertion
☑ Enable Removal
☑ Enable Creation
□ Enable Deletion
□ Visible (implicit)

CollectionPermission
Redefines the update capabilities allowed on this collection.
Insert & Remove is appliable only for the collection based on metaclasses

Example of MultiComboBoxCollaboration parameterization

Specifying a tree entry structure

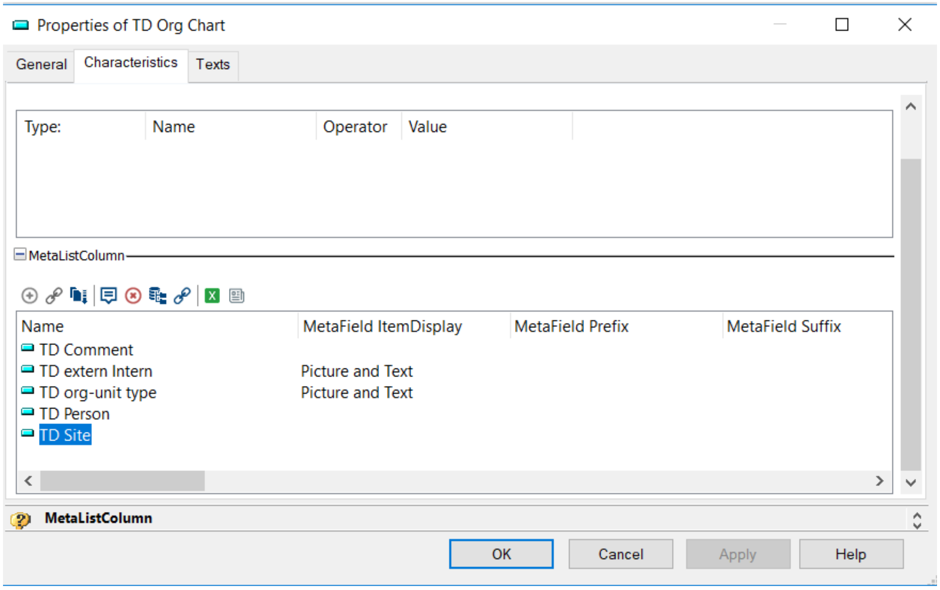
A tree entry structure is presented for an *Organizational Chart*, for example. A tree table is used to specify the diagram components.



	Internal/External	Org-Unit Type	Person
Call Center (Call Center) - Organizational Chart			
Call Center		Structure	
Operators		Structure	
Software Specialists		Structure	
Troubleshooting Team		Structure	

For that type of diagram, you have to specify:

- the structure of the tree represented,
- the content of columns for a given MetaClass and display order.
- ***MetaTree*** and ***MetaList***: used to describe the fields of a tree table entry diagram.
  - The structure of the tree is represented by a ***MetaTree*** connected the ***DiagTabDefinition.***
  - The content of the columns is represented by a ***MetaList*** connected the ***DiagTabDefinition.***
- The columns of the list , which are specified by **MetaField** in the MetaList **Characteristics** property page.



Type:	Name	Operator	Value

Name	MetaField ItemDisplay	MetaField Prefix	MetaField Suffix
TD Comment			
TD extern Intern	Picture and Text		
TD org-unit type	Picture and Text		
TD Person			
TD Site			

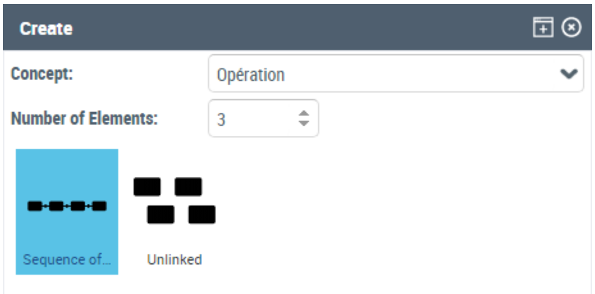
MetaListColumn

Specifying specific behaviors

Using Multiple Add

The **Multiple Creation** button is used to quickly create several objects, linked together, in the same diagram.

```
For example, you can simultaneously create a number of
operations in a process diagram.
```



Create

Concept: Opération

Number of Elements: 3

Some ***MetaWizards*** are provided. Each one is dedicated to a specific type of diagram.

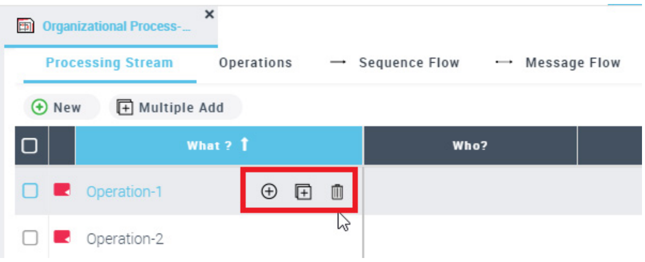
The table below shows the correspondence between diagram type and ***MetaWizards***.

Parameterization	**MetaWizard**
BPMN Process diagrams (Business process diagram, process diagram, value stream diagram, system process diagram)	***~TkV7PZ5DOHC1[TD Multiple new - BPMN]***.
Organizational Chart diagram	***~5(q3D01DO5GV[TD Multiple new - Org Chart]***
Customer Journey diagram	1- for the **Phase** creation: ***~sk4OPR69On41[TD Multiple new - Phases Creation CJ]*** 2- for the **Step** creation: ***~ijASx3SBOnU9[TD Multiple new - Step creation CJ]***

Using a floating toolbar

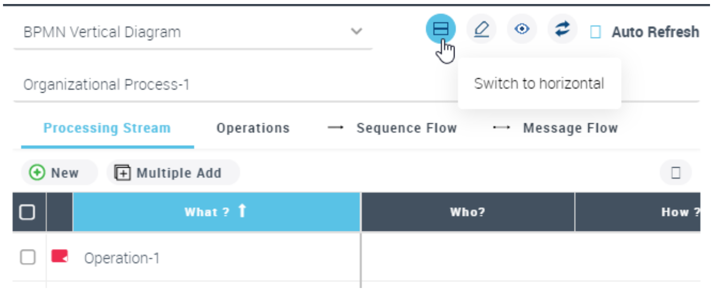
The floating toolbar enables to perform actions from the object row.

For example, you can **Add**, **Multiple Add** or **Remove** an object.



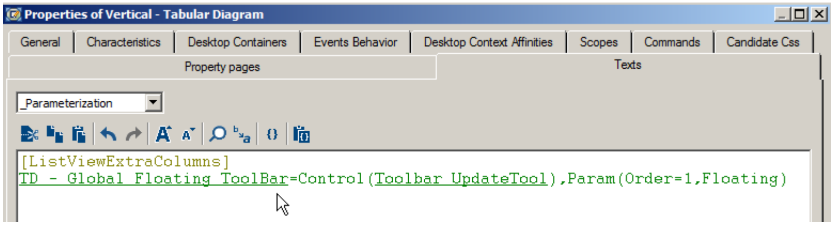
The floating toolbar is available, for example, on vertical and horizontal dektops associated with the tabular entry diagram of a process.

For example, ***~XlbPk1AaO90A[Vertical - Tabular Diagram]*** and ***~NGLfkPwBM1JL[Horizontal - Tabular Diagram]*** desktops.



To specify you want to use the floating toolbar in a tabular entry diagram desktop:

1. Access the ***Desktop*** object properties.
2. In **Texts** > **_Parameterization**, use the **[ListviewExtraColumns]** and the ***TD - Global Floating ToolBar*** generic macro to specify the command below:

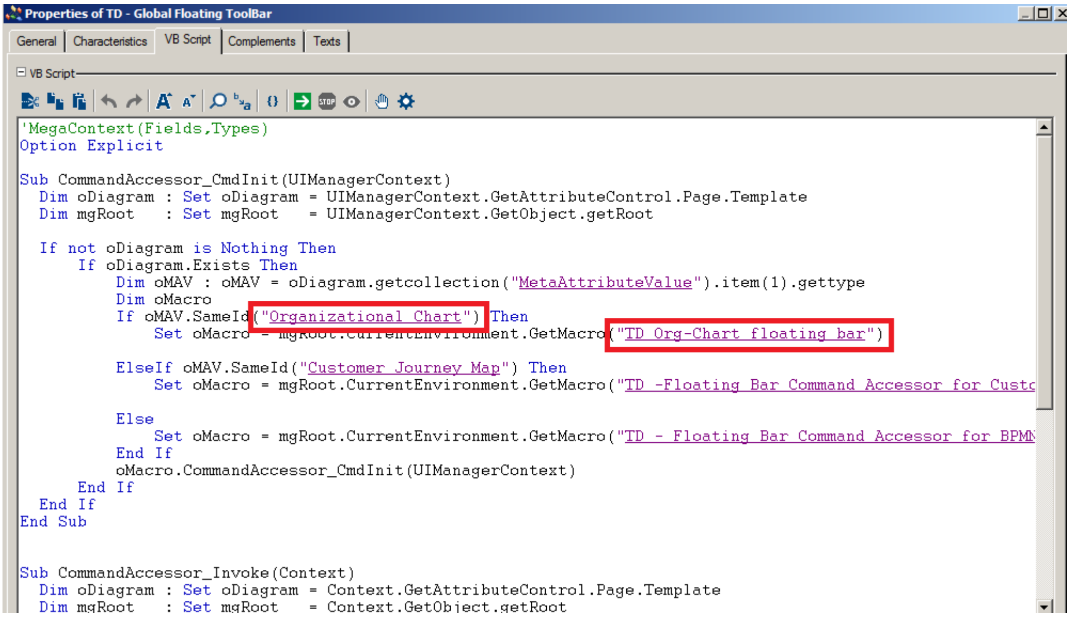


```
[ListViewExtraColumns]
TD - Global Floating ToolBar=Control(Toolbar UpdateTool),Param(Order=1,Floating)
```

To specify that you want to use the floating toolbar in a tabular entry diagram associated with a ***DiagramType***:

1. Use the ***MetaAttributValue*** associated with the ***DiagramType*** to connect the macro specifying the button behavior of your floating toolbar.

   In the example below, the toolbar behavior of an **Organizational Chart** tabular diagram is described in ***~fdZMjcNKOT4G[TD Org-Chart floating bar]*** macro.



```
'MegaContext(Fields,Types)
Option Explicit

Sub CommandAccessor_CmdInit(UIManagerContext)
  Dim oDiagram : Set oDiagram = UIManagerContext.GetAttributeControl.Page.Template
  Dim mgRoot   : Set mgRoot   = UIManagerContext.GetObject.getRoot

  If not oDiagram is Nothing Then
      If oDiagram.Exists Then
          Dim oMAV : oMAV = oDiagram.getcollection("MetaAttributeValue").item(1).gettype
          Dim oMacro
          If oMAV.SameId("Organizational Chart") Then
              Set oMacro = mgRoot.CurrentEnvironment.GetMacro("TD Org-Chart floating bar")

          ElseIf oMAV.SameId("Customer Journey Map") Then
              Set oMacro = mgRoot.CurrentEnvironment.GetMacro("TD -Floating Bar Command Accessor for Custo

          Else
              Set oMacro = mgRoot.CurrentEnvironment.GetMacro("TD - Floating Bar Command Accessor for BPMN
          End If
          oMacro.CommandAccessor_CmdInit(UIManagerContext)
      End If
  End If
End Sub


Sub CommandAccessor_Invoke(Context)
  Dim oDiagram : Set oDiagram = Context.GetAttributeControl.Page.Template
  Dim mgRoot   : Set mgRoot   = Context.GetObject.getRoot
```

2. Specify the macro to be used in the ***CommandAccessor_CmdInit*** subroutine and also in the ***CommandAccessor_Invoke*** subroutine.

Customizing the Metamodel

INTRODUCTION TO STUDIO

Each organization is unique, with its own culture and methods. Ready-to-run software often integrates in the organization to only a limited extent. An activity as strictly regulated as enterprise architecture modeling gains in flexibility when the software tool can adapt to methodological recommendations, to demands of the domain and to graphical preferences. **Studio** allows you to implement and manage customizations and **HOPEX** metamodel extensions as well as to adapt the tool to your own particular use.

Studio aims at simplifying product customizations using an ergonomic graphical interface. A **Studio** navigator allows you to precisely configure the multiple aspects of the **HOPEX** environment: navigation, object properties pages, diagram types, modeling rules and the graphical interface.

Customizations or extensions that can be made to the metamodel can impact various aspects of **HOPEX**:

- the creation of new concepts or new attributes. The metamodel diagram allows you to easily create MetaClasses and to create MetaAssociations between the different MetaClasses.
- the exploitation and use of the newly-created concepts. This means being able to show them in diagrams, use them in navigation trees, and being able to customize their properties pages.

This documentation covers the following topics:

- ✓ ***Managing the Metamodel***: presents the metamodel diagram, creation of new MetaAttributes, MetaClasses and MetaAssociations. It also presents other tools contributing to metamodel management.
- ✓ ***Query Syntax***: describes syntax used by ERQL (Entity-Relationship Query Language).
- ✓ ***Creating Consistency Checks***: describes how to apply rules to objects so as to check their consistency.
- ✓ ***Making a concept variable***: describes how to make a concept variable, i.e. to allow variation of the concept instances. That means new instances can be created from the previous one inheriting some of the existing connections. Furthermore, inherited objects can be removed and or replaced.
- ✓ ***Customizing Perimeters***: enables building a set of objects and links from a root object.

For other aspects of **Studio** as customization of properties pages, refer to the corresponding technical articles.

MANAGING THE METAMODEL

The following points are covered here:

INTRODUCTION TO METAMODEL MANAGEMENT

The Object Management Group (OMG) establishes and maintains computer industry specifications and promotes the theory and practice of object technology for interoperable enterprise specifications.

Within the OMG, **HOPEX** works in several domains. In particular, **HOPEX** performs a reviewer role in the specification of Meta Object Facility (MOF).

The objective of this standard is to ensure interoperability of different modeling tools by common definition of the concepts used.

OMG metamodeling architecture comprises four layers.

An example of use of this four layer architecture:

M3 MetaMetaModel	Basic objects	MetaClass	MetaAssociation	MetaClass
M2 Metamodel	Metamodel objects	Org-Unit	An Org-Unit sends a Message	Message
M1 Model	HOPEX user objects	Client	The Order is sent by the Customer	Order
M0 Objects	End user objects	Mr Smith	Mr Smith issues Order No. COM1727	COM1727

The metamodel enables definition of semantics of models that will be created.

They are located in level 2 of metamodel architecture defined by the OMG.

The **HOPEX** metamodel can be modified or extended to suit specific requirements. To do this, certain precautions must be taken to assure maintenance of its extensions over time.

Metamodel Extensions and Modifications

☛ *The metamodel defines the language for expressing a model. It defines the structure used to store the data managed in a repository The metamodel contains all the MetaClasses used to model a system, as well as their MetaAttributes and the MetaAssociations available between these MetaClasses. The metamodel is saved in the environment system repository.*

You can create metamodel extensions to manage new object types. Repositories that exchange data (export, import, etc.) must have the same metamodel, otherwise certain data will be rejected or inaccessible.

The metamodel diagram enables modification of repository structure (create object types, links, characteristics, modify object and link typing).

This modification type should be used with extreme care.

You can create metamodel extensions using *command files*, in text format (ASCII).

To take command files into account in the system repository:

1. Open **HOPEX Administration**.
2. In the navigation tree, right-click the desired environment and select **Open**.
3. Expand the **Repositories** folder.
4. Right-click **SystemDb** and select **Object Management > Import**.

Whether you have created your extensions using a metamodel diagram or a command file, you must validate them by a translation of the environment metamodel, using the command **Metamodel > Translate and Compile** in the environment pop-up menu.

Warning Regarding Metamodel Modification

You can modify the structure (or *metamodel*) of environment repositories, for example when you want to add a MetaAssociation between two MetaClasses.

Note that these modifications should be carried out with caution because they will have to be maintained by the administrator. The metamodel is saved in the system repository of the environment: all repositories of an environment have the same metamodel.

💣 **Do not create extensions to the metamodel without careful consideration. If you create metamodel extensions, name your extensions (MetaClass, MetaAssociation, MetaAttribute or text) with a specific prefix that identifies your company, so as to avoid**

any conflict with a new HOPEX concept when updating your HOPEX version.

☛ *Data exchanges between repositories with different metamodels can result in rejects.*

Metamodel extensions are managed differently from modifications to the standard metamodel.

Creating metamodel extensions

After you create new MetaClasses, MetaAssociations or MetaAttributes, back up these metamodel extensions with their absolute identifiers (export as MGR file). You will need them when creating a new environment.

Upgrading a site or environment does not impact extensions you have added to the metamodel.

Managing modifications perfomed on HOPEX standard metamodel

In case you had to modify HOPEX metamodel (or technical data), you must keep your update logfile (MGL file) so as to import your updates after each HOPEX migration (HF/CP or major version).

As HOPEX data is MEGA proprietary data, when upgrading a site or an environment to a new version, the standard installed concepts will be automatically reassigned the standard values.

You must save the MGR type files that you extracted after having made modifications, in order to reapply these after upgrading your environment.

You should apply your modifications to the environment (and only those changes, not an extraction of the complete metamodel) in order to have both the upgraded standard version and your own modifications.

Metamodel extensions backup

A simple way of backing up your metamodel extensions is to work in a private workspace using the metamodel diagram.

You can then make an initial verification in this private workspace before dispatching. Usually, you must translate and compile the metamodel so that your extensions will be correctly taken into account.

Once satisfied with operation of your extensions, you can export the logfile of your private workspace before you dispatch it. In this way you will obtain a command file that you can then reimport into another environment when using your extensions or at a change of version.

Transferring metamodel extensions

The absolute identifiers of metamodel concepts are indispensable if **HOPEX** is to run properly. Metamodel extensions (creation of MetaClass, MetaAssociation, MetaAttribute) should be processed as follows:

- Carry out workspace logfile export if you have been working in a private workspace, or a metamodel extensions logical backup in other cases.
- Save the generated file, which you will use to transfer the extensions into other environments.

Compiling the environment

After you create or import extensions into an environment, it must be compiled. This is done in **HOPEX Administration**, by requesting translation of the metamodel in the current language.

Restrictions

- Before exchanging data (e.g.: Export, Import) between repositories that are not in the same environment, make sure that these repositories have the same metamodel.

  ☛ *Repositories within the same environment share the same metamodel.*

- Extensions to the metamodel (new MetaAssociations and MetaClasses) cannot be graphically handled in standard diagrams. Diagram configuration needs to be customized to take into account such extensions (contact HOPEX Professional Services for diagram customization).

Implementation conditions

So that modifications to the metamodel are visible to users:

- the metamodel must be compiled
- a new workspace must be opened. If the user has a private workspace open, it must be dispatched or updated so that the user has access to metamodel modifications.

Precautions Concerning Metamodel Extensions

Certain precautions must be taken when customizing the metamodel:

- Always work in a test environment. Remember that modification or deletion of a metamodel concept (MetaClass/MetaAttribute) will affect all data linked to this concept.
- Create a "Super User" with administration rights and rights to modify **HOPEX** data.

Concepts

Metamodel definition commands concern:

- MetaClasses (or object types)
- links possible between these MetaClasses (called MetaAssociations)
- the two MetaAssociationEnds of each of these MetaAssociations
- MetaAttributes (characteristics).
  Object comments or texts are assimilated in MetaAttributes.

Concept	Syntax
MetaClass (or object type)	metaclass
MetaAssociation (or link)	MetaAssociation
MetaAssociationEnd	MetaAssociationEnd
MetaAssociationType (or link type)	MetaAssociationType
MetaAttribute (or characteristic)	MetaAttribute
MetaAttributeGroup (or group of MetaAttributes)	MetaAttributeGroup
MetaAttributeValue (or value of MetaAttribute)	MetaAttributeValue

Concept names

The name of a concept is unique, ie. the name for a MetaClass, a MetaAssociation, a MetaAttribute or a text must be used only once, including all languages.

Names of MetaClasses with standard naming rule contain 255 characters maximum.

☛ *For information on namespaces, see "Managing Namespaces.*

Name of a concept must begin with a letter (A to Z). The following characters are authorized for use in concept names:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

a b c d e f g h i j k l m n o p q r s t u v w x y z

à á â ã ä å ç é è ë ê ì í î ï ñ ò ó ô ö ù ú û ü ÿ ý

0 1 2 3 4 5 6 7 8 9

* ( ) / = + % ? $ _ & €

, ; - :

☛ *Hiragana, Katakana and Kanji characters are accepted for object names in the Japanese language.*

You cannot use "& , ; - : " as the first character. There can be spaces in the name, except as first and last character.

☛ *Unlike the other characters, the ampersand (&) is not controlled as the first character:*

- *it can be used in an object name.*
- *it should not be used in the name of a MetaClass, MetaAssociation, MetaAssociationEnd or MetaAttribute.*

Technical concepts other than repository core concepts, (MetaClass, MetaAssociation and MetaAttribute) have names beginning with an underscore (_).

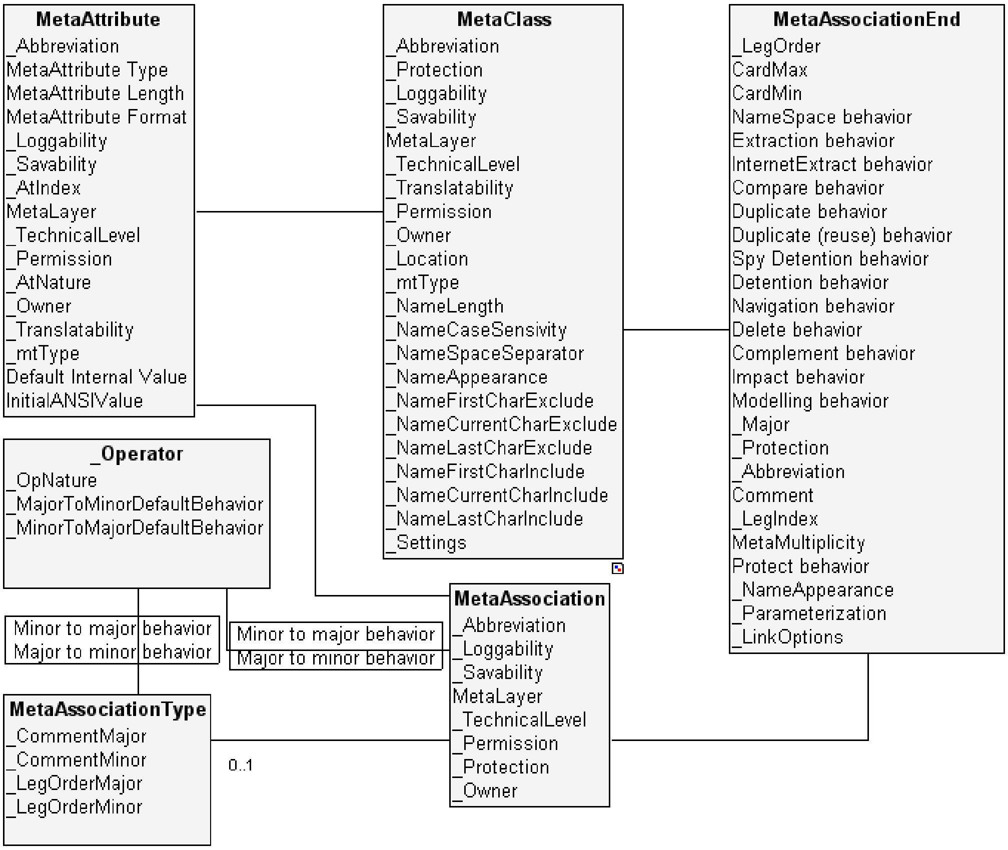
💣 **It is impossible to define extensions with names beginning with characters "_" or "$".**

💣 **It is not recommended to restrict the list of characters as this may generate issues when executing queries or descriptors (see "Restricting the names of object types).**

As you cannot use a name already in use, you must attach a prefix to your extension names. This avoids conflicts that might occur later when the standard metamodel is upgraded.

For example, if you create a metamodel extension called "Table" for **HOPEX Business Process Analysis** and later install **HOPEX Database Builder**, a conflict will occur.

Metamodel diagram



All of these MetaClasses have the following attributes:

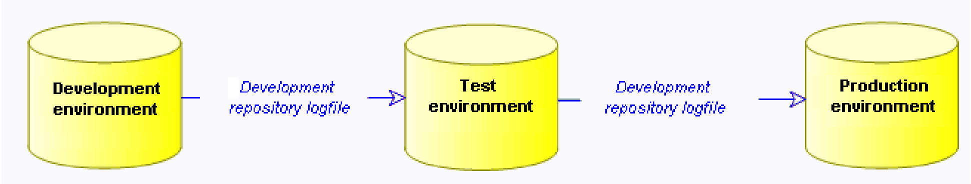
- Name
- Absolute Identifier
- Authorization
- Creation Date
- Modification Date
- Creator, Modifier
- Creator Name
- Modifier Name
- Authorization Level
- Creation Version
- Modification Version
- Comment and History

Creating Metamodel Extensions: Method

How to Create Metamodel Extensions

When you create a metamodel extension, it is recommended that you first create a development environment, create the metamodel extension in this environment, then import the extension in a test environment.

Once extensions have been validated in the test environment, you can import them into the production environment.



To create a metamodel extension:

1. Create a development environment and a test environment and initialize the system repository logfiles.

   ☛ *Select a user with rights to modify **HOPEX** data.*

2. Create the metamodel extensions in the development environment.

   ☛ *Remember to prefix or suffix the extensions created.*

3. Export the system repository logfile.
4. Import this logfile in the test environment - with a user with rights to modify **HOPEX** data - to verify the import result.
5. If there are no rejects, import the logfile again, saving updates.
6. Translate and compile the metamodel in the test environment and verify that extensions function correctly.
7. Import the logfile containing the extensions in the production environment.

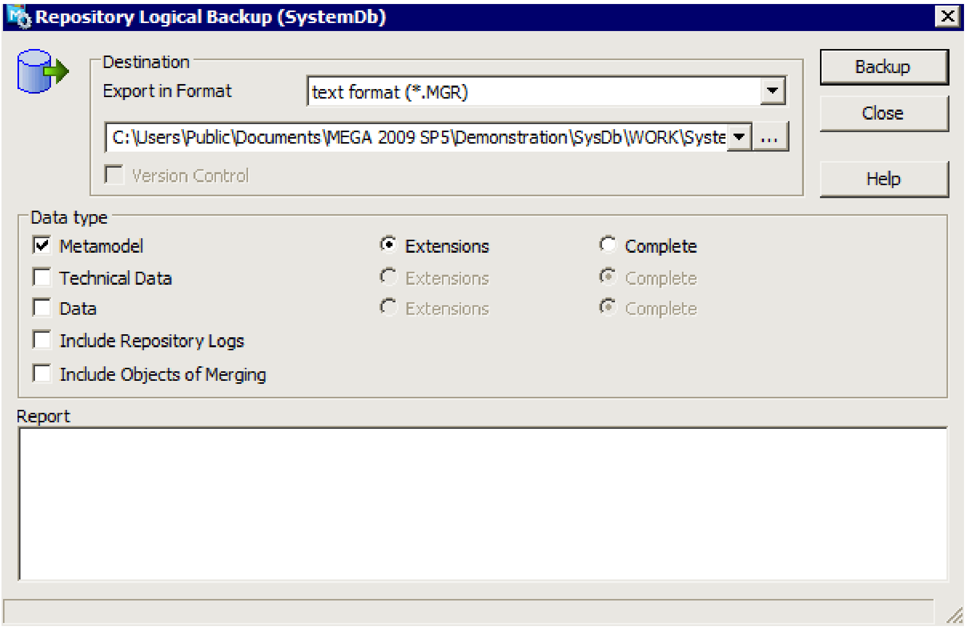
When the extensions have been validated, you can import the logfile with your extensions in the production environment, following the procedure described above.

Extensions Backup

To back up the metamodel extensions you have created:

1. Run **HOPEX Administration** and connect to the desired environment.
2. Expand the **Repositories** folder and right-click **SystemDb** and select **Logical Backup**.

3. In the **Repository Logical Backup (SystemDb)**, check only the boxes **Metamodel** and **Extensions**.



Repository Logical Backup (SystemDb)

Destination

Export in Format: text format (*.MGR)

C:\Users\Public\Documents\MEGA 2009 SP5\Demonstration\SysDb\WORK\Syste

□ Version Control

Data type

☑ Metamodel	● Extensions	○ Complete
□ Technical Data	○ Extensions	● Complete
□ Data	○ Extensions	● Complete
□ Include Repository Logs		
□ Include Objects of Merging		

Report

4. Click **Backup**.
The generated file contains all extensions created in the environment.

THE METAMODEL DIAGRAM

Creating a Metamodel Diagram

> ☛ *The metamodel diagram is available with the* ***Studio*** *technical module.*

To create a metamodel diagram:

1. In the **MetaStudio** navigation window, right-click the **Metamodel** folder and select **New > Metamodel**.
2. Enter the metamodel **Name** and click **OK**.
3. Right-click the metamodel you have just created and select **New > Diagram**.
   The window that opens proposes selection of diagram type.
4. Select **Metamodel Diagram** and click **Create**.
   The Metamodel diagram opens. It describes a metamodel instance.

   In it you can place MetaAssociations and MetaClasses.

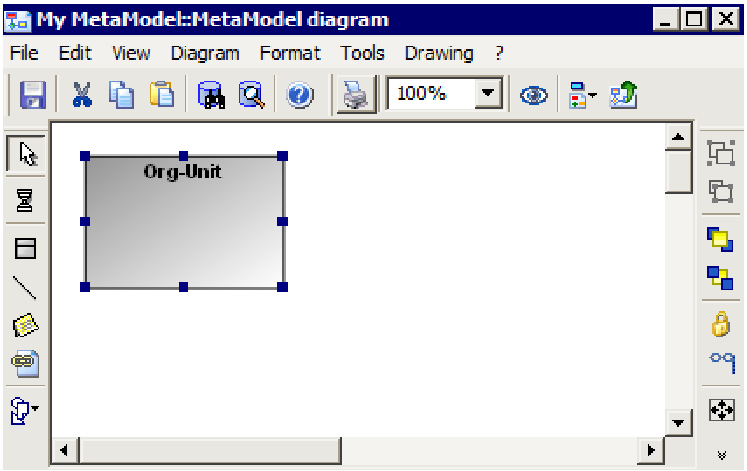
   > ☛ *A MetaClass or a MetaAssociation placed in the diagram is automatically connected to the described metamodel, see "Adding a MetaClass to a Metamodel Diagram and "Adding a MetaAssociation.*

Adding a MetaClass to a Metamodel Diagram

To add an existing MetaClass to a metamodel diagram:

1. In the metamodel diagram insert toolbar, click **MetaClass** ⊟ then click in the diagram.
   The **Add Concrete MetaClass** dialog box appears.
2. In the **Name** field, click the arrow and select **List**.
3. In the list, select the required MetaClass.
4. Click **OK**.
   The MetaClass name is displayed in the **Add Concrete MetaClass** dialog box.

5. Click **Connect**.
The MetaClass is added to the metamodel diagram.


My MetaModel::MetaModel diagram
File Edit View Diagram Format Tools Drawing ?
100%
Org-Unit

You can access the attributes of the MetaClass by opening its Properties window.

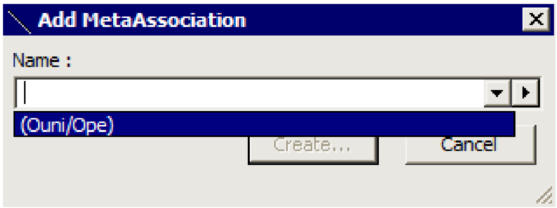
> ☛ *In the same way you can place several MetaClasses in the metamodel diagram.*

Adding a MetaAssociation

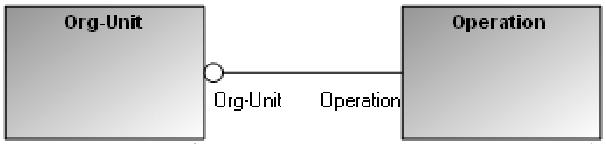
To add a MetaAssociation between two MetaClasses in the diagram:

1. In the metamodel diagram insert toolbar, click **MetaAssociation** ╲, then draw a link from the first MetaClass to the second.
2. In the **Add MetaAssociation** dialog box, click the arrow to select the MetaAssociation that interests you.

> ☛ *The vertical arrow only appears if the MetaAssociation has already been created. To create a MetaAssociation, click the horizontal arrow and select **New**.*



3. Click **Connect**.
   The selected MetaAssociation appears in the diagram.



You can access the attributes of the MetaClass by opening its Properties window.

> ☛ *The major MetaAssociationEnd is indicated by a black diamond (composition), white diamond (aggregation) or a slash (default). See "Creating a MetaAssociation and "Specifying MetaAssociation Behavior for more details.*

METACLASSES

The following points are covered here:

- "Creating a MetaClass (Object Type)
- "Typing a MetaClass

Creating a MetaClass (Object Type)

To create a MetaClass in the metamodel diagram:

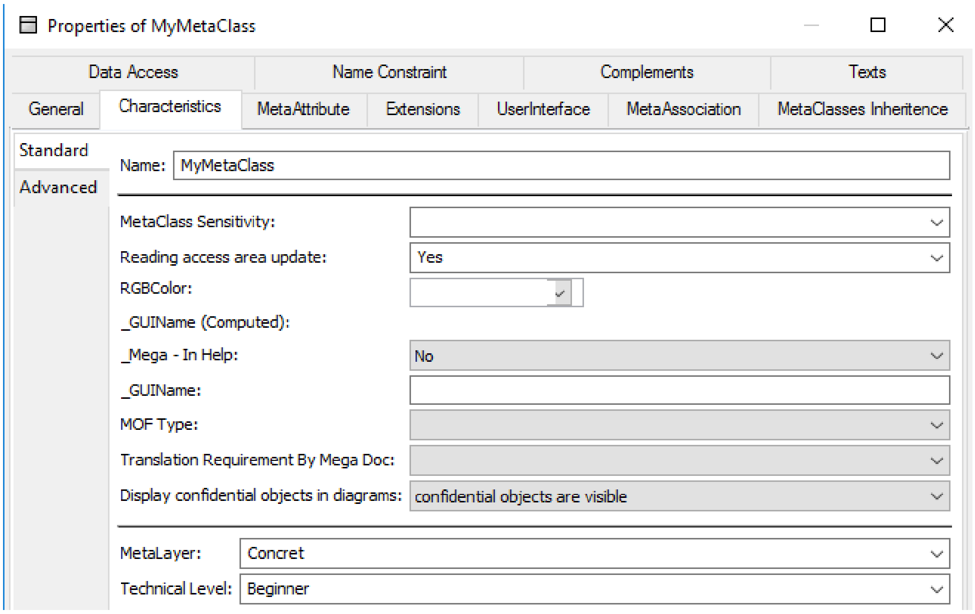
1. In the metamodel diagram insert toolbar, click **Concrete** ⊟, then click in the diagram.
2. Enter the **Name** of the MetaClass.

   ☛ *For information regarding naming, see "Concept names.*
3. Click **Create**.
   The MetaClass is created.

   ☛ *At creation some characteristics are automatically created (e.g. : "Absolute identifier", "Creator Name", "Creation Date").*

To access the MetaClass properties:

❱ Right-click the MetaClass and select **Properties**.
The MetaClass Properties window opens.



Name: MyMetaClass

MetaClass Sensitivity:

Reading access area update: Yes

RGBColor:

_GUIName (Computed):

_Mega - In Help: No

_GUIName:

MOF Type:

Translation Requirement By Mega Doc:

Display confidential objects in diagrams: confidential objects are visible

MetaLayer: Concret

Technical Level: Beginner

Restricting the names of object types

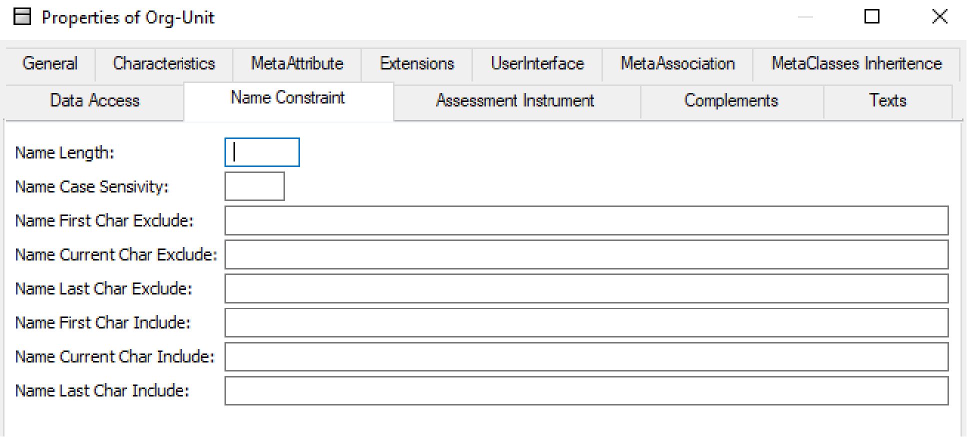
Characters that can be used in object names are subject to the same restrictions as concept names.

☛ *See "Concept names.*

By default, names are in upper/lower case, and are limited to 255 characters.

The names are stored the way they are entered. However, the system verifies that once the name is converted into uppercase and stripped of any accents, it is not already in use. For instance, for objects of the same type, objects "SMith" and "Smith" are considered to be identical.

This configuration is carried out in the MetaClass Properties window > **Name Constraint** tab.



Name Length:
Name Case Sensivity:
Name First Char Exclude:
Name Current Char Exclude:
Name Last Char Exclude:
Name First Char Include:
Name Current Char Include:
Name Last Char Include:

For each MetaClass, you can specify:

- Maximum name length: **Name Length**

  ☛ *_NameLength must be equal to or greater than 16.*
- Systematic name conversion: **Name Case Sensitivity**
  - to uppercase: U (Upper)
  - to lowercase: L (Lower)
  - no conversion: I (Ignore)
- Illegal characters:
  - for first character: **Name First Char Exclude**
  - for characters 2 to n-1: **Name Current Char Exclude**
  - for character n: **Name Last Char Exclude**

For each MetaClass, you can add to the list of authorized characters:

- for first character: **Name First Char Include**
- for characters 2 to n-1: **Name Current Char Include**
- for character n: **Name Last Char Include**

☛ *It is recommended that you do not add to authorized characters for MetaClasses other than technical. Correct operation of query and execution of descriptors is not guaranteed for objects with added characters.*

Typing a MetaClass

The ***MetaAssociation*** between ***MetaClass*** and ***_Operator*** is used to modify standard behavior of tools (e.g.: export, protection) for the specified MetaClasses.

This MetaAssociation carries two attributes:

- _ScanInit
  If _ScanInit has value "S", all objects of this MetaClass must be systematically taken into account by the tool.

  This is the case for Sort, Tag, and Language objects when exporting **HOPEX** objects.

  ```
  Default value: no value
  ```
- _ScanInit
  If _ScanType has value "D", when an object of this type is extracted, the minor objects linked to it are also extracted, as well as their descendants.

  This is the case for Information, Rule, Relationship, MetaAssociationEnd and Entity.

  ```
  Default value: no value
  ```

  ☛ *For more details, see **HOPEX Administration-Supervisor** guide.*

METAASSOCIATIONS

The following points are covered here:

- "Creating a MetaAssociation
- "Reversing Major/Minor Orientation
- "Modifying Object Protection
- "Imposing MetaAssociation Uniqueness
- "Specifying MetaAssociation Behavior
- "Processing MetaAssociationTypes

Creating a MetaAssociation

Creating a MetaAssociation (link) implements:

- a major MetaClass via a major MetaAssociationEnd
- a minor MetaClass via a minor MetaAssociationEnd

To create a MetaAssociation, see "Adding a MetaAssociation.

Semantic rule

The *major* MetaClass is that which is modified at deletion of its MetaAssociation with the minor MetaClass.

The *minor* MetaClass is that which retains its semantic value at deletion of its MetaAssociation with the major MetaClass.

☛ *For more details on major and minor objects, see the* ***HOPEX Administration - Supervisor*** *guide.*

MetaAssociation orientation

When creating a MetaAssociation in the metamodel diagram, it is the direction in which the MetaAssociation was drawn that determines the major MetaAssociationEnd and the minor MetaAssociationEnd.

The major MetaAssociationEnd is on the origin side of the MetaAssociation, the minor MetaAssociationEnd is on the arrival MetaClass.

When a MetaAssociation is created, the "Order" attribute is assigned to it automatically.

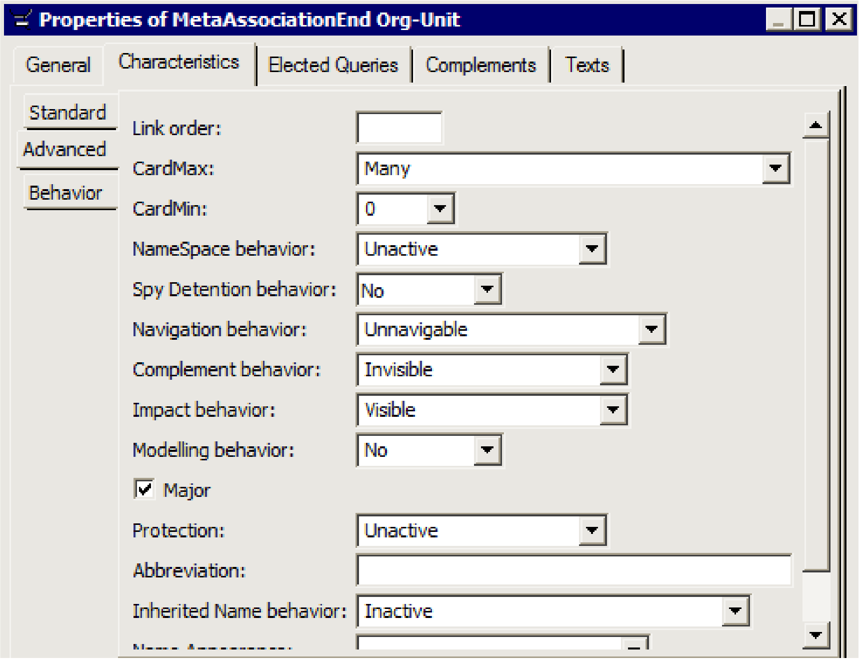
☛ *The major MetaAssociationEnd is indicated by a black diamond (composition), white diamond (aggregation) or a slash (default). See "Specifying MetaAssociation Behavior for further information.*

Reversing Major/Minor Orientation

After creation of a MetaAssociation, major/minor orientation can be reversed in the metamodel diagram by opening the Properties window of each MetaAssociationEnd.

To modify MetaAssociation orientation:

1. In the Properties window of the major MetaAssociationEnd, select the **Characteristics** tab, then the **Advanced** subtab.
2. Clear the **Major** check box.



Link order:

CardMax: Many

CardMin: 0

NameSpace behavior: Unactive

Spy Detention behavior: No

Navigation behavior: Unnavigable

Complement behavior: Invisible

Impact behavior: Visible

Modelling behavior: No

☑ Major

Protection: Unactive

Abbreviation:

Inherited Name behavior: Inactive

3. Open the properties of the minor MetaAssociation End and select the **Major** tab.

Modifying Object Protection

For MetaClasses and MetaAssociationEnds, the ***Protection*** attribute allows you to activate ("A") or deactivate ("U") checks related to user authorization level. By default, for all major MetaClasses and MetaAssociationEnds, this attribute is active.

If no value is specified for ***Protection***, a Major/Minor orientation value is deduced:

- If the MetaAssociationEnd is major, ***Protection*** is "Active" (Value "A")
- If the MetaAssociationEnd is minor, ***Protection*** is "Inactive" (Value "U")

☛ *This function is available with the **HOPEX Power Supervisor** technical module only.*

For MetaClasses and MetaAssociationEnds for which you wish to deactivate this check, assign the value "U" to the ***Protection*** attribute, and there will be no check at updating.

To modify MetaClass and MetaAssociationEnd protection:

1. In the Properties window of a MetaClass or MetaAssociationEnd, select the **Characteristics** tab, then the **Advanced** subtab.
2. For the **Protection** attribute, select value **Inactive**.

Imposing MetaAssociation Uniqueness

This prevents connecting several objects to the same object.

```
Example: To ensure that an org-unit is connected to a single
tag, you must specify on the MetaAssociationEnd between
Org-Unit and Tag that the MetaMultiplicity attribute is
0..1 or 1.
```

Uniqueness is applied particularly on sub-type and certain composition links.

*For compatibility with versions earlier than Service Pack 1 of **HOPEX** version 6.1, it is also possible to specify the _CardMax attribute between MetaClass and MetaAssociationEnd. This characteristic takes value "U" to impose uniqueness . Other possible values for this characteristic are "1", which means that the uniqueness is only for documentary purposes, and "N", which indicates there is no restriction.*

Only the maximum MetaMultiplicity is taken into account to impose uniqueness of a MetaAssociation. The minimum MetaMultiplicity of a MetaAssociationEnd is purely for documentary purposes. It does not give an obligatory MetaAssociation.

Specifying MetaAssociation Behavior

The MetaAssociationType, associated with its orientation, enables determination of the operation to be executed on each of its MetaAssociationEnds for the following functions: extraction, protection, object deletion, query isolated objects, comparison and impact analysis in the explorer. Main operators are the following:

_Operator	Used for
Extract	Object extraction
Delete (Propagate)	Object deletion
Protect	Object protection
Isolate	Query isolated objects
Compare	Comparison of repositories
Duplicate	Object duplication

Operators act differently depending on major/minor orientation of links and the MetaAssociationTypes associated with the MetaAssociations .

The behavior of an operator related to a MetaAssociation is specified in the "Operator Name Behavior" attributes calculated on each MetaAssociationEnd. This attribute is visible:

- in the **Behavior** tab of the properties of a MetaAssociation.

A MetaAssociation can have only one type. To modify the MetaAssociationType, you must disconnect it from its current type and then connect it to the new type.

This can be done in the metamodel diagram by opening the Properties window of each MetaAssociation.

You can directly modify the behavior of a link for a given operator in the **Behavior** tab of the MetaAssociation:

☛ *Link type processing during extraction:*

The extraction processes the values in the following order:

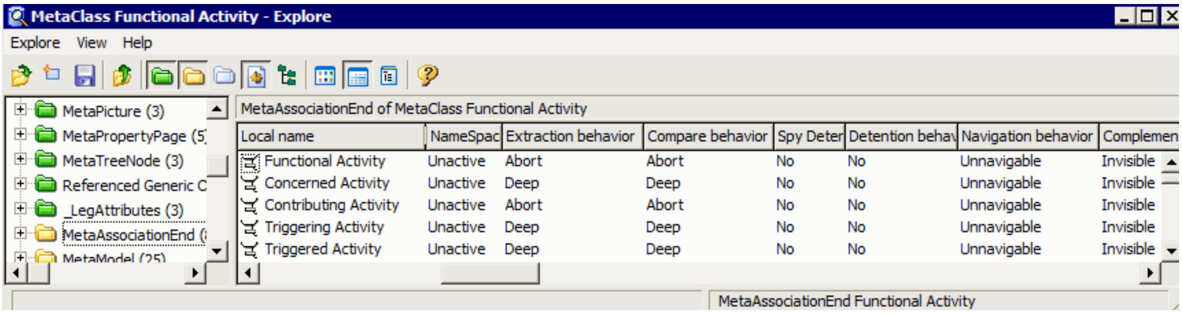
- *The value of the* "Minor to major behavior" *or* "Major to minor behavior" *characteristic (depending on whether the MetaAssociationEnd followed is major or minor) for the link between the link and the operator, if it exists.*
- *The value of the* "Minor to major behavior" *or* "Major to minor behavior" *characteristic of the link between the link type and the operator.*
- *The default value A (Abort) for* "Minor to major behavior" *and D (Deep) for*"Major to minor behavior" *if none of the previous links are specified.*

If the determined value is S (standard), the value of the _Scantype characteristic for the link between the object type (segment) and the operator is also used, if it exists.

This is the case for Entity, Relationship, MetaAssociationEnd , Rule, and Information, where the value is D (Deep).

This means the Entity, etc. is extracted plus the dependencies.

The behavior of an operator related to a link is specified in the calculated attributes on each MetaAssociationEnd.



Local name	NameSpac	Extraction behavior	Compare behavior	Spy Deter	Detention behav	Navigation behavior	Complemen
Functional Activity	Unactive	Abort	Abort	No	No	Unnavigable	Invisible
Concerned Activity	Unactive	Deep	Deep	No	No	Unnavigable	Invisible
Contributing Activity	Unactive	Abort	Abort	No	No	Unnavigable	Invisible
Triggering Activity	Unactive	Deep	Deep	No	No	Unnavigable	Invisible
Triggered Activity	Unactive	Deep	Deep	No	No	Unnavigable	Invisible

MetaAssociationTypes

The "Composition" and "Association" MetaAssociationTypes can handle most current situations.

☛ *Confirm in* ***Tools > Options****, select* ***Repository > Metamodel****, that the option* ***MetaModel****,* ***Definition of path of MetaAssociation*** *is set to "Standard Mode".*

Composition

This MetaAssociationType indicates a strong dependency between the component and the assembled object. In particular, it indicates that the component has no meaning without this object (example: operations of a procedure).

The component and all its contents are included during extraction, duplication, and protection operations on the assembled object. It should be deleted when the assembled object is deleted.

Association

This MetaAssociation describes the major object, but the two objects are able to exist independently of each other. An example would be procedure and application. The minor object is included in an extraction, but is not copied, protected, or deleted with the major object.

MetaAssociation Types Kept to Maintain Compatibility

To access these MetaAssociationTypes:

- Set option to : **Tools > Options > Repository > Metamodel**, **Definition of path of MetaAssociation**, "Compatibility Mode".

MetaAssociationTypes around a diagram

The MetaAssociation representing graphic content of a diagram is generally of composition type. For extraction of diagrams however, two additional MetaAssociationTypes are used:

Modeling

This MetaAssociationType indicates that one object is modeled by another (Description-Procedure MetaAssociations, etc.). This means that when a diagram is extracted, the object it describes will also be extracted.

Citation

This MetaAssociationType indicates a reference in a diagram to an object which is not intrinsically part of the diagram. For example, processes in a Flowchart need to be extracted with the objects describing them (Messages), while Org-Units should be extracted without their contents (such as their Organizational Chart). The MetaAssociation connecting these org-units to the diagram is citation type.

Other MetaAssociationTypes

alias

This MetaAssociationType is an alias of a generic link and inherits its behavior.

In this case, behavior need only be defined on the generic MetaAssociation. This being so, certain MetaAssociationTypes are no longer used. This is the case for example of "Documentation" and "Annotation" MetaAssociationTypes.

Aggregation

The dependency between the objects concerned is also strong in this case, but the component can exist without the aggregated object, for example: org-units of a flowchart, which can exist in an organizational chart even after the flowchart no longer exists.

These links are processed as for a composition link, except for deletion (the component should not be deleted since it may be used elsewhere) and copying (the component is linked to the copy without itself being copied).

Definition

The link is indispensable in understanding the defined object, for example: synchronization to message. Adding specifications to Message has no effect on synchronization. However, deleting the Message makes Synchronization null and void.

The MetaAssociation to the minor object is processed as for a description link (see below). The major object should be deleted when the minor object is deleted.

Sequence

This MetaAssociationType represents time sequencing of two independent phenomena. Modification or deletion of one of the objects has no impact on the other. Example: Next operation.

Flow

These MetaAssociations express exchange of a flow or message between two objects. Example: Message exchanged between two org-units.

Classification

These MetaAssociations enable classification of objects using tags or other criteria.

Disappearance of the classification has no effect on the objects concerned.

Check

These MetaAssociations express a check to be carried out on an object. Example: MetaAssociation between constraint and operation.

Transformation

These MetaAssociations express transformation of an object from one type to another. Example: A class produces a table in a database.

Sub-Typing

These MetaAssociations express that one of the objects is a particular case of the other. The sub-type is the major object since it inherits all modifications to the super-type.

Extraction of the sub-type also necessitates extraction of the super-type.

Bivalent UML

These MetaAssociations express a strong relationship between two objects. Neither one has any meaning without the other. Example: link between role and association. The two objects are always extracted together.

Hierarchy

These MetaAssociations express a strong dependency of one object on another. Example: MetaAssociation between package and class or between database and table. Deletion of the major object also deletes the minor.

Variance

This MetaAssociationType enables connection of a variant to the object of which it is a variant.

Processing MetaAssociationTypes

Key:

- ***-***: the link is not processed.
- ***D***: downward processing, includes the linked object, then follows the associations from that object. The objects at the end of the links may be included depending on the link type.
- ***S***: standard processing, the object at the end of the link is included.
- ***L***: only the link to the object is included (not the object itself)

Link type	Example	Major Minor	Extraction Compar.	Protect	Dupli	Delete	Isolate
COMPOSITION	Procedure/Operation; Application/Tool; Project/Diagram, Tool, Function, Document ; ...	Aggregation-of Component	- D	- D	D	- -	- S
AGGREGATION	Operation/Rule, Timer ; Information/Entity ; Diagram/Entity, Relationship, MetaAssociationEnd, Constraint, Rule, Message, Operation, Condition, Record, Set, Sequencing, ... ; Record/Attribute; Identifier; Record-Key; Index; Diagram/Attribute (View-of-Diagram); ...	Aggregation-of Parameter	- D	- D	L	- -	- S
DESCRIPTION	Operation/Service; Org-Unit/Operation; Project/Application, Procedure; Installed-Diagram; Site/Org-Unit; Record/Language; ...	Described Description	- S	- -	L	- -	- S
DEFINITION	Logical Group/Entity; MetaAssociationEnd/Entity; Record-Owner, Record-Member; Synchronization/Message ...	Defined Defining	- S	- -	L	- -	- S
CLASSIFICATION	Tag	Characterized Used	- S	- -	L	- -	- S
FLOW	Source-... ; Target-...	SrcTarget Source Target	- -	- -		- -	- -

Link type	Example	Major Minor	Extraction Compar.	Protect	Dupli	Delete	Isolate
TRANSFORMATION	Relationship/Record; Organizational-Operation; ...	Derived Origin	- S	- -	L	- -	- S
SEQUENCE	Next; Timer/Message; ...	Next Previous	- -	- -		- -	- -
MODELING	Descriptor-.. ; Program/Tool; Analysis-Function	Modeled Diagram	S D	- D	L D	- -	- -
CITATION	Diagram/Procedure, Org-Unit, Site, Application, Tool, Information, Function, State, ...	Citing Cited	- S	- S	L	- -	- S
SUB-TYPING	Sub-Type; Equivalence, ...	Sub-Type Super-Type	- D	- -	L	- -	- -
BIVALENT	Output, MetaAssociationEnd/Relationship	Component Component	D D	D D	D D	- -	- -S
HIERARCHY	Database/Table; Table/Column, Key, Index		- D	- D	D	- D	- S
CHECK	Constraint/Entity, Information, Relationship	Checking Checked	D -	- -	L	- -	- -

METAATTRIBUTES

MetaAttribute Characteristics

MetaAttribute Type

Ab	MetaAttribute Type	Description
X	String	Alphanumeric or numeric if not used to compute data with a defined MetaAttribute Length value (1-1024)
D	DateTime	Date (available until year 2038) Format: "YYYY/MM/DD HH:MM:SS" In windows where dates can be entered, the format used is the one defined in the Windows configuration.
W	DateTime64	Date Format: "YYYY/MM/DD HH:MM:SS"
U	AbsoluteDateTime64	Date, which does not include the time Format: "YYYY/MM/DD"
A	VarChar	ASCII text (for large text like comments)
B	VarBinary	Binary text (reserved)
1	Boolean	Boolean (0 or 1)
S	Short	Integer (0-65535)
L	Long	Integer (0 - 4294967295)
Q	Binary	Binary (reserved) with a defined MetaAttribute Length
H	Double	Integer (0 - 18446744073709551616)
F	Float	Floating number
C	Currency	

☛ *See "MetaAttribute Definition Rules.*

MetaAttribute Format

MetaAttribute Format enables indication that external values are stored in a table (value T).

Ab	MetaAttribute Format	Description
S	Standard	for a string (default value)
F	Enumeration	or string type with predefined list of values
T	Enumeration(opened)	for string type with list of values open to user input
D	Duration	for a date
P	Percent	for a percentage
E	Double	for a number
O	Object	for an object
Z	Signed Number	for a signed number
C	Currency	for a currency
B	RGBcolor	for a color
8	UTF-8	for a unicode string

MetaText Format

By default, each new object type is connected to the "Comment" characteristic.

The MetaText Format specifies the text format:

- CONDITION
- XML
- XML not completed
- CSV
- ANSI
- RTF
- HTML
- JSON
- FIELD

MetaAttribute Definition Rules

When you create or modify a MetaAttribute, you must comply with the following rules:

- Once dispatched a MetaAttribute type cannot be modified

  ☛ *Exception for some MetaAttributes, see "Modifying a MetaAttribute Type" for details.*

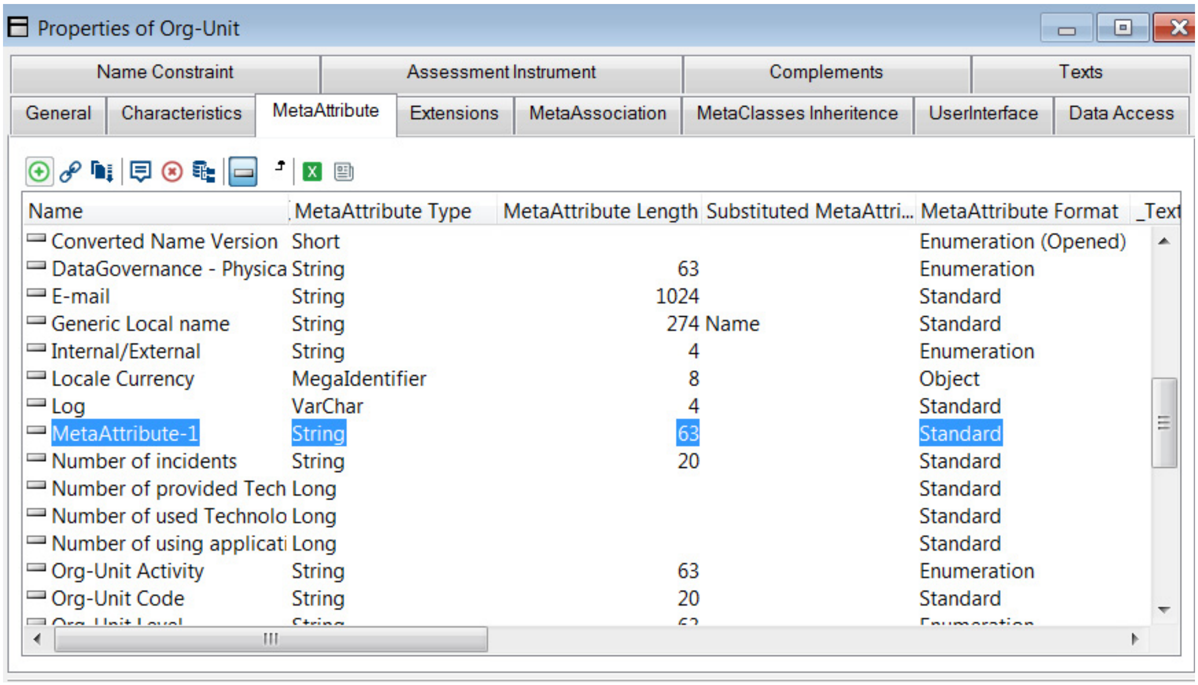
- The following MetaAttributes must have a defined MetaAtributeLength:
  - String ("X")
  - Binary ("Q")
- Once dispatched you cannot modified a MetaAttributeLength value with a lower value than the source one.

Creating a MetaAttribute

To create a MetaAttribute:

1. Open the Properties window of the MetaClass or MetaAssociationEnd.

2. In the **MetaAttribute** tab, click **New** ⊕ .
   The new MetaAttribute is created and appears in the list.



Name	MetaAttribute Type	MetaAttribute Length	Substituted MetaAttri...	MetaAttribute Format	_Text
Converted Name Version	Short			Enumeration (Opened)	
DataGovernance - Physica	String	63		Enumeration	
E-mail	String	1024		Standard	
Generic Local name	String	274	Name	Standard	
Internal/External	String	4		Enumeration	
Locale Currency	MegaIdentifier	8		Object	
Log	VarChar	4		Standard	
MetaAttribute-1	String	63		Standard	
Number of incidents	String	20		Standard	
Number of provided Tech	Long			Standard	
Number of used Technolo	Long			Standard	
Number of using applicati	Long			Standard	
Org-Unit Activity	String	63		Enumeration	
Org-Unit Code	String	20		Standard	
Org-Unit Level	String	63		Enumeration	

☛ *When a MetaClass is created, it is automatically assigned the "Name" and "Comment" attributes.*

☛ *The "Absolute identifier", "Authorization", "Creator Name", "Creation Date", "Modifier Name", and "Modification Date" attributes are also available automatically when an object type is created.*

☛ *The names of the attributes that can be used in queries are limited to 32 characters.*

3. In the **MetaAttribute Type** field, select the attribute type.

   ☛ *By default the **MetaAttribute Type** value is "String" with a **MetaAttribute Length** value set to "63".*

   ☛ *See "MetaAttribute Type.*

   💣 **Once dispatched, you cannot change the MetaAttribute Type value (see exceptions: "Modifying a MetaAttribute Type").**

4. In the **MetaAttribute Format** field, select the attribute format.

   ☛ *See "MetaAttribute Format.*

5. (Mandatory for "X"-type or "Q"-type MetaAttribute) In the **MetaAttributeLength** field, enter the attribute length.

   💣 **Once dispatched, take care not to modify the length with a value lower than the initial value.**

Modifying a MetaAttribute Type

If needed, you can change MetaAttribute types according to the following rules:

- ***for dates***:
  - DateTime to DateTime64 or AbsoluteDateTime64
  - AbsoluteDate64 to DateTime64
  - DateTime64 to AbsoluteDate64
- ***for numerical*** values:

  💣 **In that case you need to align data with the new type: perform the Environment automatic update on the environment (else on the repository: RDBMS Administration > Align DataType and Indexes with MetaModel definition).**

  - Boolean to Short, Long, HexaLong, IdAbs, or DoubleFloat
  - Short to Long, HexaLong, IdAbs, or DoubleFloat
  - Long to HexaLong, IdAbs, or DoubleFloat
  - HexaLong to IdAbs or DoubleFloat
  - IdAbs to DoubleFloat
- ***Strings*** values
  - numeric to alphanumeric
  - alphanumeric to numeric

Defining a tabulated characteristic

☛ *Tabulated characteristic values are saved with the technical data.*

A tabulated characteristic is a characteristic associated with a series of tabulated values, such as Information-Class, Org-Unit-Type, and Message-Type.

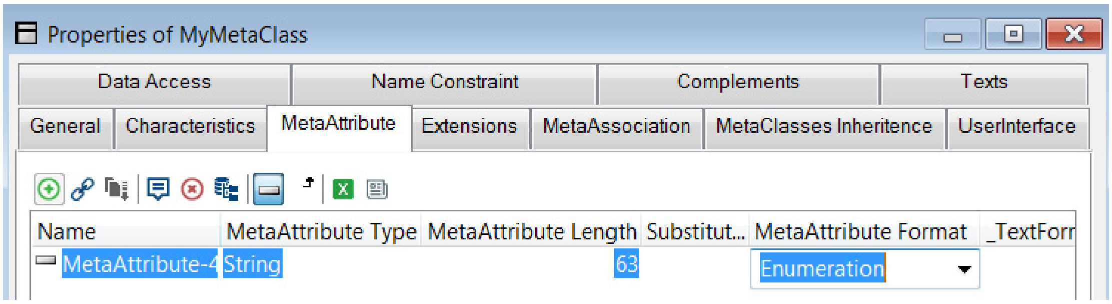
You can change a standard characteristic (S) to a tabulated characteristic after creation by setting **MetaAttribute Format** to "Enumeration" (F) or "Enumeration (opened)" (T).

To define a tabulated characteristic :

1. Access the Properties of the MetaClass concerned.
2. Select the **MetaAttribute** tab.
3. In the MetaAttribute row, in the **MetaAttribute Format** column, select "Enumeration" or "Enumeration opened" value.

   ☛ *To reverse this characteristic, set the* ***MetaAttribute Format*** *back to "S".*

   ☛ *The tabulated values associated with the tabulated characteristics are included in the technical data. They can be modified using* *specialized data entry accessible in the **Characteristics** tab of the MetaAttribute properties.*



Name	MetaAttribute Type	MetaAttribute Length	Substitut...	MetaAttribute Format	_TextForr
MetaAttribute-4	String	63		Enumeration	

When you create a tabulated value, the Administration application automatically generates a name for this value. It should not be modified.

You must indicate the internal value actually stored in the attribute, and the external value to be listed during data entry, in the current language.

☛ *The attribute update version must be at least "16643" for it to be modifiable using this data entry.*

Specialized data entry is also accessible by selecting **Metamodel > Tabulated Values** in the pop-up menu of the environment in the Administration application.

☛ *For more information on tabulated values data entry, see "Editing MetaAttribute Tabulated Values.*

MetaAttribute with unique index

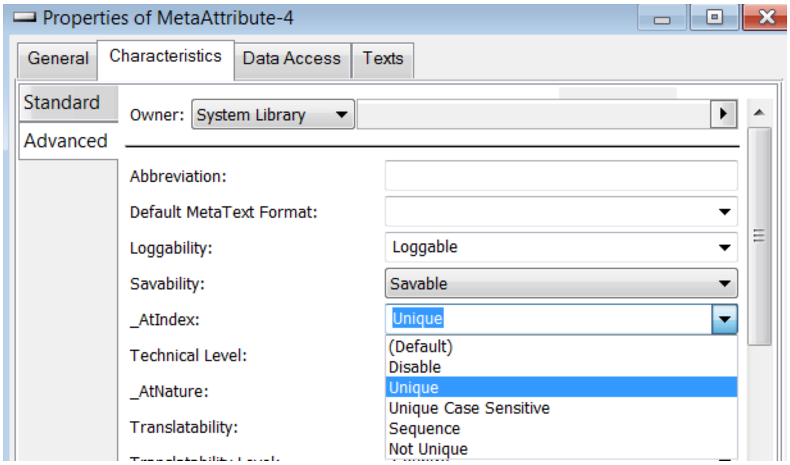
A unique index prevents the same value being assigned to two different attributes.

The **_AtIndex** attribute is accessible in the Properties window of a MetaAttribute in the **Advanced** subtab of the **Characteristics** tab.

To attribute a unique index to a MetaAttribute:

1. In the Properties window of the MetaAttribute, select the **Characteristics** tab, then the **Advanced** subtab.

2. In the **_AtIndex** field, select the "Unique**"** value.



Owner: System Library

Abbreviation:

Default MetaText Format:

Loggability: Loggable

Savability: Savable

_AtIndex: Unique

Technical Level:

_AtNature:

Translatability:

☛ *Attribute uniqueness is only effective in repositories created after uniqueness is declared. It is necessary to reorganize existing repositories.*

_AtIndex	Comment
0	No index
U	Unique index
S	Unique and case sensitive index
N	Non-unique index
Q	Sequence index

Editing MetaAttribute Tabulated Values

A tabulated value of a characteristic is the set of predefined values that this characteristic can take.

For example the values of the "Flow-Type" characteristic of messages.

You can modify these values in the properties of the MetaAttribute.

To modify tabulated values of a MetaAttribute:

1. In HOPEX, access the MetaAttribute properties.
2. In the **Characteristics** tab, **Standard** sub tab, the **MetaAttributeValues for enumerated MetaAttribute only** section lists the MetaAttribute available values.

3. To modify a value, select it and enter its new value and corresponding internal value (the value stored in the repository).

   ☛ *Do not modify the **Name** field. Modify the **English** field which contains the external value of the attribute.*

4. To add a new value, click **Add** ⊕.

Connecting Attributes to a MetaClass

An attribute can be connected to one or several diagrams. The "Name" attribute is connected to all MetaClasses.

To connect an attribute to a MetaClass:

1. In the MetaClass properties, select the **MetaAttribute** tab.
2. Click **Connect** and select the MetaAttribute you want to connect.

   ☛ *You can select several MetaAttributes.*

3. Click **Connect**.
   The selected MetaAttributes appear in the list of MetaAttributes of the MetaClass.

Connecting MetaAttributes to MetaAssociations

A MetaAssociation cannot be connected to a text type attribute. An attribute can be connected to one or several MetaAssociations. The "Order" attribute is connected to all MetaAssociations.

☛ *A "Text" type attribute on a link is only accessible with certain tools such as the explorer, and not by others (diagrams, etc.).*

To connect an attribute to a MetaAssociation:

1. In the Properties window of a MetaAssociation, select the **MetaAttribute** tab.
2. Click **Connect** and select the MetaAttribute.

   ☛ *You can select several MetaAttributes.*

3. Click **Connect**.
   The selected MetaAttributes appear in the list of MetaAttributes of the MetaAssociation.

Customizing MetaAttribute Standard Behavior

Thanks to MetaAttributes extended properties you can customize some MetaAttribute standard behavior.

For example, you can:

- exclude MetaAttribute values from a Reporting Datamart
- prevent MetaAttributes from being annotated

Excluding MetaAttribute values from a reporting Datamart

In some cases, you might not need to export some specific MetaAttribute values in your Reporting Datamart.

To exclude MetaAttribute values from a Reporting Datamart:

1. In HOPEX, access the MetaAttribute properties.
2. Click the **Characteristics** tab, then **Advanced** subtab.
3. Click the **Extended Properties** field arrow and select **Exclude from Reporting Datamart**.

Making MetaAttributes not available for annotation

In some cases, you might want to prevent a MetaAttribute from being annotated.

To make a MetaAttribute not annotable for annotation:

1. In HOPEX, access the MetaAttribute properties.
2. Click the **Characteristics** tab, then **Advanced** subtab.
3. Click the **Extended Properties** field arrow and select **Not annotable**.

Abbreviations

Definition of abbreviations for the names of MetaClasses, attributes and text is optional. As standard, the metamodel is delivered with attribute abbreviations.

Standard Attributes

All MetaClasses have standard attributes:

- Name:
  - limited to 63 characters for MetaClasses without namespace.
  - limited to 255 characters for MetaClasses with namespace, of which 159 characters are for the local name.
- Internal identifier (absolute identifier calculated from the object creation date)
- Creation date
- Creator name (absolute identifier of the user who created the object)
- Last modification date
- Last modifier name (absolute identifier of the user who last modified the object)
- Comment
- Log

CALCULATED METAATTRIBUTES

Creating a macro to calculate a MetaAttribute

A calculated MetaAttribute is a virtual MetaAttribute that computes its value from a calculation macro.

The calculation macro:

- is implemented by a function

  ```
  E.g.: GetAttributeValue(ByVal Object, ByVal AttributeID,
  ByRef Value)
  ```

- may include:
  - constant values
  - values retrieved from other MetaAttributes
  - values retrieved from other calculated MetaAttributes

Calculated MetaAttribute values are not stored, but constantly recomputed, even if the value has not changed.

You can add MetaTriggers to calculated MetaAttributes so as to store their values in the repositories.

> ☛ *See "Optimizing HOPEX Usage Saving Calculated MetaAttribute Values.*

Creating the calculation macro

To calculate a MetaAttribute using a macro:

1. Open the explorer on the new MetaAttribute (or new parameter).
2. Right-click the MetaAttribute (or parameter) and select **New > Macro**
   The macro creation wizard appears.
3. Select **Create a (VB)Script Macro**.
4. Click **Next**.
5. (Optional) Modify the default **Name** ("AttributeName".Macro) of your macro.

   > 🕮 *A macro is an object containing a VB Script code sequence interpreted at execution.*

6. Click **Finish**.
7. Configure the calculation macro.

Configuring the calculation macro

To configure the calculation macro:

1. Edit the "AttributeName".Macro macro and note that in particular the VB Script contains the following functions:

   ☛ *If they are not present, standard implementation is selected.*

- ***GetAttributeValue(ByVal Object, ByVal AttributeID, ByRef Value)***
  Defines attribute access mode. The parameters are:
  - Object: corresponds to the object of which attribute value is requested.
  - AttributeID: absolute identifier of the attribute (or taggedValue).
  - Value: the function returns the attribute value for this object.
- ***SetAttributeValue(ByVal Object, ByVal AttributeID, ByVal Value)***
  Defines attribute save mode. The parameters are:
  - Object: corresponds to the object of which the attribute value must be updated.
  - AttributeID: absolute identifier of the attribute (or taggedValue).
  - Value: the function saves the attribute value for this object.

  ☛ *The attribute nature (**_AtNature**) should be **Virtual**.*

For both of these functions, attribute change mode is a character string.

Conversion must be carried out to change text format to the internal format of the attribute.

Example:

```
Sub GetAttributeValue (ByVal object, ByVal AttID, Value)
   ' internal value reading in integer format.
   numValue = CInt(objet.GetProp(AttID, "Physical"))
      if numValue < 20 then
         Value = "Young"
      elseif numValue < 35 then
         Value = "Youthful"
      elseif numValue < 55 then
         Value = "Mature"
      else
         Value = "Elderly"
   end if
End  Sub
```

You can directly implement read-only and read/write access in the attribute format (without passing via standard conversion).

In this case, you must implement the following two functions, of which prototypes are similar to those above:

- ***GetExtendedAttributeValue(ByVal Format as LONG, ByVal Object, ByVal AttributeID, ByRef Value)***
- ***SetExtendedAttributeValue(ByVal Format as LONG, ByVal Object, ByVal AttributeID, ByVal Value)***

The difference is in the additional parameter: Format. The possible values are:

- 0 internal: value in internal format (binary, integer,...)
- 1 external: value in external format, but before display processing (certain objects have external form that is textual with the addition of index identifiers, as for class attributes or association roles).
- 3 Display: value in external format used in Web sites or Word documents (expurgated when identifiers in external format occur).

If one of the two extended functions is implemented, call by GetProp with "Physical" format on the same attribute is prohibited since it would lead to an infinite recursion.

Optimizing HOPEX Usage Saving Calculated MetaAttribute Values

For performance issue, you might want to add a MetaTrigger to a calculated MetaAttribute so as to manage its refresh. With this customization the calculated MetaAttribute value is not recalculated each time the MetaAttribute is displayed, but only when the MetaTrigger is triggered.

MetaTrigger types

MetaTrigger types are the following:

- ***On Demand***
  The value is calculated in the following cases:
  - if it has never been calculated
  - at generation of reports including the object
  - on user specific requests in the User Interface

    ☛ *See "Optimizing HOPEX Usage Saving Calculated MetaAttribute Values.*
  - with a scheduled job

    ☛ *See "Scheduling On Demand MetaAttribute update.*
- ***On Update***, ***On Update with internal Set*** and ***On Notify***

  ☛ ***On Update*** *and* ***On Notify*** *MetaTrigger types are restricted to MEGA internal use only.*

Adding a MetaTrigger to a calculated MetaAttribute

To manage the calculated MetaAttribute refresh, you need to add a MetaTrigger to the MetaAttribute and activate it.

To add a MetaTrigger to a calculated MetaAttribute:

1. In HOPEX, right-click the MetaAttribute and select **Explore**.
2. Right-click the **MetaTrigger** folder (in the **Impact Analysis** 📁 folders) and select **New**.

   ☛ *You need to display the* ***Empty Collections*** 📁 *folders.*

   ☛ *If the MetaTrigger is already created, click* ***Connect****.*
3. Enter a **Name** and click **OK**.

4. Access the MetaTrigger **Properties**, **Characteristics** tab:
   - in the **MetaTrigger Type** field, select its type.

     Example: On Demand.
   - in the **MetaTrigger Status** field, select "Active".

Limitations

You cannot use **On Demand** MetaTriggers if the MetaAttribute value can be set by the user and the MetaAttribute calculation is based on the MetaAttribute value presence to be triggered or not.

Use case: Updating a calculated MetaAttribute On Demand

Context

"Cost per user" is a calculated MetaAttribute of the Application MetaClass. For performance issue, you do not need to display its updated value at list display.

For example: in **HOPEX IT Portfolio Management** (with Application Portfolio Manager profile), in the **Home page**, when the user click the **Applications** tile.

Objective

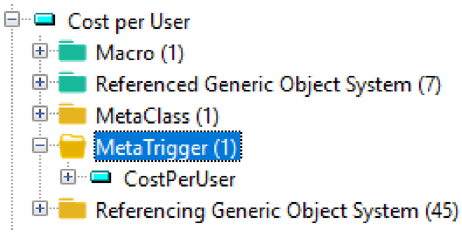
The "Cost per User" values of applications will be refreshed only on user demand.

Configuring the calculated MetaAttribute

To add an On Demand MetaTrigger to the "Cost per user" calculated MetaAttribute:

1. In HOPEX, explore the "Application" **MetaClass**.
2. In the **MetaAttribute** folder, expand the "Cost per user" MetaAttribute.
3. Display the **Empty Collections** folders.
4. In the **Impact Analysis** folders, right-click its **MetaTrigger** folder and select **New**.
5. In the **Name** field enter its name.

   E.g.: "CostPerUser"
6. Click **OK**.
   The MetaTrigger is added to the "Cost per User" calculated MetaAttribute.



Cost per User
- Macro (1)
- Referenced Generic Object System (7)
- MetaClass (1)
- MetaTrigger (1)
  - CostPerUser
- Referencing Generic Object System (45)

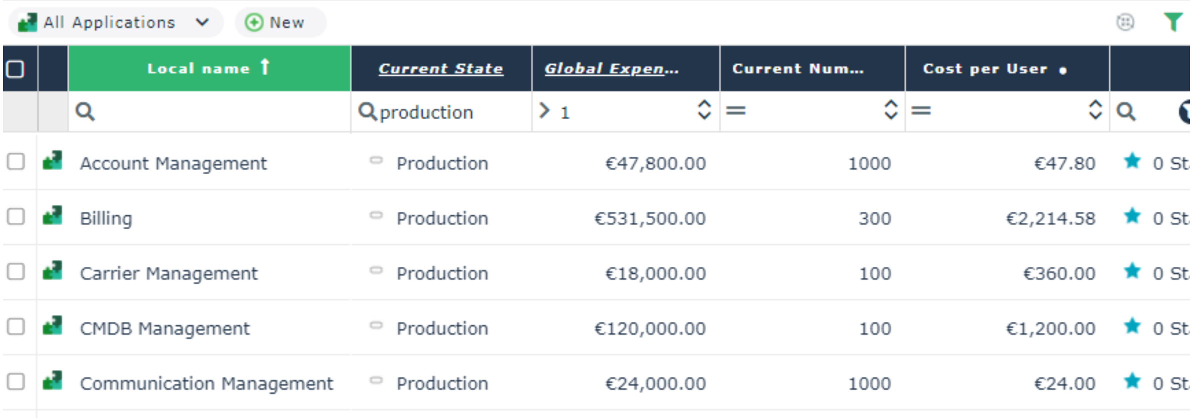
7. Right-click the "CostPerUser" MetaTrigger and select **Properties**.
8. In the **Characteristics** tab:
   - in the **MetaTrigger Type** field, select "On Demand".
   - in the **MetaTrigger Status** field, select "Active".

9. Access a list of applications.

   For example, connect to HOPEX IT Portfolio Management (with Application Portfolio Manager profile).

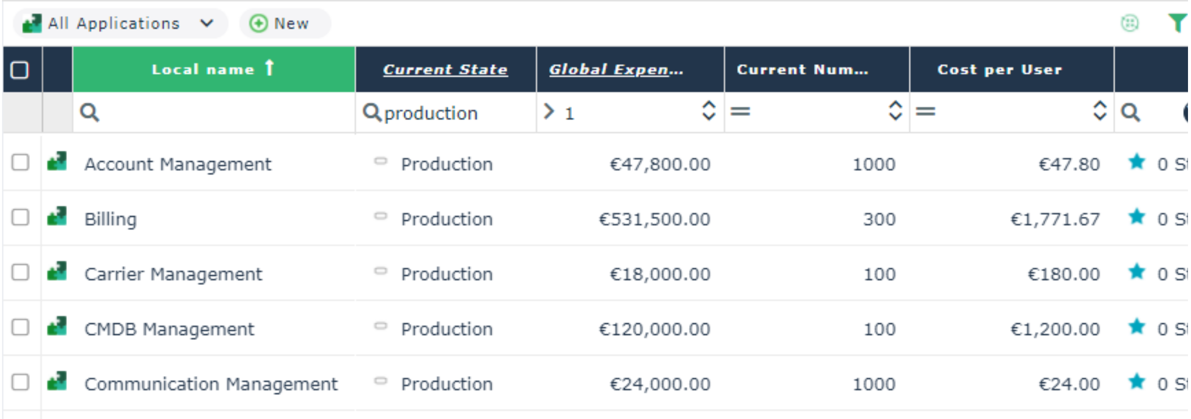
   In the **Cost per User** header, a ▣ indicates that this MetaAttribute is updated on Demand and this column values might not be updated.

   In this case the "Cost per user" values of "Billing" and "Carrier Management" Applications are not up-to-date.



□		Local name ↑	Current State	Global Expen...	Current Num...	Cost per User ●	
			production	> 1	=	=	
□		Account Management	Production	€47,800.00	1000	€47.80	★ 0 St
□		Billing	Production	€531,500.00	300	€2,214.58	★ 0 St
□		Carrier Management	Production	€18,000.00	100	€360.00	★ 0 St
□		CMDB Management	Production	€120,000.00	100	€1,200.00	★ 0 St
□		Communication Management	Production	€24,000.00	1000	€24.00	★ 0 St

10. Click **Synchronize column** ⊞ to refresh all the "Cost per user" values.



□		Local name ↑	Current State	Global Expen...	Current Num...	Cost per User	
			production	> 1	=	=	
□		Account Management	Production	€47,800.00	1000	€47.80	★ 0 S
□		Billing	Production	€531,500.00	300	€1,771.67	★ 0 S
□		Carrier Management	Production	€18,000.00	100	€180.00	★ 0 S
□		CMDB Management	Production	€120,000.00	100	€1,200.00	★ 0 S
□		Communication Management	Production	€24,000.00	1000	€24.00	★ 0 S

All the calculated MetaAttribute values are recalculated and updated.

In this case the "Cost per user" values of "Billing" and "Carrier Management" Applications are now up-to-date.

If in the list, other calculated MetaAttribute values are also configured On Demand they are also updated.

The ⊞ turns to green to indicate that the columns are all synchronized.

Scheduling On Demand MetaAttribute update

By default, to optimize performances, the "Computation of OnDemand MetaAttributes" is triggered every night at 04:00 (Server Time Zone) on each HOPEX repository to perform night recalculation of all of the **OnDemand** MetaAtttibutes.

You may want to customize or create other Triggers on OnDemand MetaAttributes.

Definition

System Trigger Definition

The "Computation of OnDemand and expired OnUpdate MetaAttributes" System Trigger Definition implements the "UpdateOnDemandAttributesMacro" macro.

Execution context

You can execute the trigger on:

- *: all of the objects of all the MetaClasses that have at least one "OnDemand" MetaAttribute are recalculated.

  ```
  Exemple: *
  ```
- <IdAbs>: all of the objects whose MetaClass IdAbs is <IdAbs> are recalculated.

  ```
  Example: MrUiM9B5iyM0
  ```

  ```
  In that case the trigger is executed on all the calculated
  MetaAttributes of the Application MetaClass.
  ```
- <IdAbs1>;<IdAbs2>;...<IdAbsn>: all of the objects whose MetaClass IdAbs is <IdAbs1>, <IdAbs2>.. , or <IdAbsn> are recalculated.

  ☛ *The separating character is: ";".*

  ```
  Example: MrUiM9B5iyM0; FN8AHc2gITnL
  ```

  ```
  In that case the trigger is executed on all the calculated
  MetaAttributes of the Application and Software Technology
  MetaClasses.
  ```

  💣 **Attention: do not add any return (break line) to the character string.**

Defining the System Trigger execution frequency

Before setting up a System Trigger, you must define its execution frequency.

Execution frequencies and involved MetaClasses are defined according to your repository data amount and your HOPEX product usage.

To define a System Trigger frequency:

1. In **HOPEX Administration** (Administration.exe) access the repository management concerned:
   - Connect to the environment concerned.
   - Expend the **Repositories** folder.
   - Expend the repository concerned.
2. Right-click **Scheduler** and select **Manage triggers**.
3. In the **User Triggers** click **New** +.
4. Select the "Computation of OnDemand and expired OnUpdate MetaAttributes" System Trigger Definition.
5. Click **Next**.
6. Enter a **Name** for your Trigger.

   ```
   E.g.: "OnDemand MA"
   ```
7. In the **Job Context** pane enter:

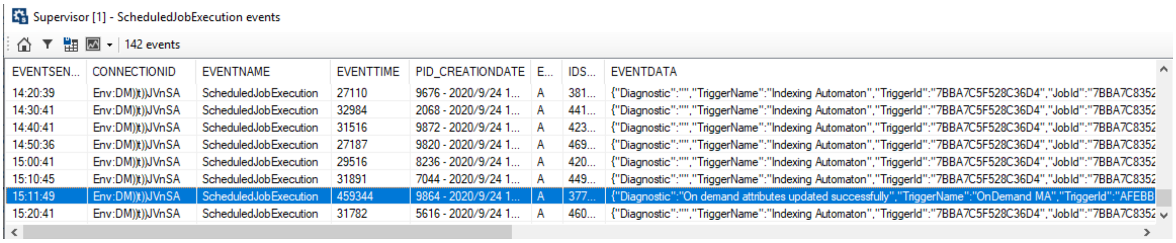
   ```
   *
   ```

   In that case all the objects of all of the MetaClasses that have at least one "OnDemand" MetaAttribute are taken into account in the recalculation.
8. Click **Finish**.
   The trigger is created and added to the list.
9. Activate the System Trigger: in the **User Triggers** list, right-click the trigger and select **Activate**.
10. Execute the Trigger: in the **User Triggers** list, right-click the trigger and select **Execute**.
11. Click **OK**.
12. Once the System Trigger executed, check its execution time:
    - Access the **HOPEX Server Supervisor** tool.

      ☛ *See "Starting HOPEX Server Supervisor.*
    - In the **Events** tab, double-click **ScheduledJobExecution** event row.
    - In the list, check the **Eventime** value of your trigger.

      ☛ *For example, in the **EVENTDATA** column, look for "TriggerName":"OnDemand MA".*

    ```
    E.g.: 459344 ms (i.e.:7’39")
    ```



Supervisor [1] - ScheduledJobExecution events

142 events

EVENTSEN...	CONNECTIONID	EVENTNAME	EVENTTIME	PID_CREATIONDATE	E...	IDS...	EVENTDATA
14:20:39	Env:DM))t))JVnSA	ScheduledJobExecution	27110	9676 - 2020/9/24 1...	A	381...	{"Diagnostic":"","TriggerName":"Indexing Automaton","TriggerId":"7BBA7C5F528C36D4","JobId":"7BBA7C8352
14:30:41	Env:DM))t))JVnSA	ScheduledJobExecution	32984	2068 - 2020/9/24 1...	A	441...	{"Diagnostic":"","TriggerName":"Indexing Automaton","TriggerId":"7BBA7C5F528C36D4","JobId":"7BBA7C8352
14:40:41	Env:DM))t))JVnSA	ScheduledJobExecution	31516	9872 - 2020/9/24 1...	A	423...	{"Diagnostic":"","TriggerName":"Indexing Automaton","TriggerId":"7BBA7C5F528C36D4","JobId":"7BBA7C8352
14:50:36	Env:DM))t))JVnSA	ScheduledJobExecution	27187	9820 - 2020/9/24 1...	A	469...	{"Diagnostic":"","TriggerName":"Indexing Automaton","TriggerId":"7BBA7C5F528C36D4","JobId":"7BBA7C8352
15:00:41	Env:DM))t))JVnSA	ScheduledJobExecution	29516	8236 - 2020/9/24 1...	A	420...	{"Diagnostic":"","TriggerName":"Indexing Automaton","TriggerId":"7BBA7C5F528C36D4","JobId":"7BBA7C8352
15:10:45	Env:DM))t))JVnSA	ScheduledJobExecution	31891	7044 - 2020/9/24 1...	A	449...	{"Diagnostic":"","TriggerName":"Indexing Automaton","TriggerId":"7BBA7C5F528C36D4","JobId":"7BBA7C8352
15:11:49	Env:DM))t))JVnSA	ScheduledJobExecution	459344	9864 - 2020/9/24 1...	A	377...	{"Diagnostic":"On demand attributes updated successfully","TriggerName":"OnDemand MA","TriggerId":"AFEBB
15:20:41	Env:DM))t))JVnSA	ScheduledJobExecution	31782	5616 - 2020/9/24 1...	A	460...	{"Diagnostic":"","TriggerName":"Indexing Automaton","TriggerId":"7BBA7C5F528C36D4","JobId":"7BBA7C8352

13. According to this processing time, you can configure the trigger scheduling:
    - every night on all of the MetaClasses.
    - every night on part of the MetaClasses and at the week-end on all of the MetaClasses

      ☛ *You need to create a trigger for each specific case.*
    - several time a day on the MetaClasses that are the most used in HOPEX Solutions and at the week-end on all of the MetaClasses.

      ☛ *You need to create a trigger for each specific case.*

    To configure each trigger scheduling, see Configuring the scheduling of a trigger.

    ☺ *If needed, contact your MEGA representative to help you setting up the most suitable policy according to your HOPEX usage.*

Abstract Metamodel

Basic Concepts

The abstract metamodel of the **HOPEX** platform offers the possibility of managing the notion of inheritance for MetaClasses and MetaAssociations. This enables a significant reduction in the number of MetaAssociations.

Available from version **MEGA 2009**.

MetaClass types are:

- ***Concrete MetaClasses***: MetaClasses for which instances exist
- ***Abstract MetaClasse***s: generic MetaClasses used only to describe inheritance.

Abstract metamodel

The abstract metamodel on which the **HOPEX** platform is based is described using metamodel diagrams. Graphical rules provide a view of basic concepts.

Abstract MetaClass

An abstract MetaClass is a MetaClass that does not have a concrete occurrence. It enables definition of common attributes to MetaClasses that inherit these.

When a concrete MetaClass inherits an abstract MetaClass, it inherits:

- MetaAttributes

  Example: MetaAttributes of the"BPMN Activity" MetaClass are: "Predicate", "Loop", "Ad hoc", "Multiple", etc.

- MetaAssociations,

  Example: the MetaAssociation inherited from the "Element with Note" MetaClass is: (Note Element/note) enabling linking of a Note object with an "Element with Note".

- Properties pages (MetaPropertyPage).
- Menu commands (MetaCommand).

Generic MetaAssociations

A generic MetaAssociation is a MetaAssociation common to a set of concrete MetaClasses. It is defined between an abstract and another MetaClass. All MetaClasses inheriting the abstract MetaClass inherit this "Generic" MetaAssociation by default.

For example, the generic MetaAssociation (Element with note/Note) defined between the "Element with Note" abstract MetaClass and the "Note" MetaClass exists for all concrete

```
MetaClasses inherited from the "Element with Note" abstract
MetaClass.
```

Managing Abstract MetaClasses

Creating an abstract MetaClass

To create an abstract MetaClass:

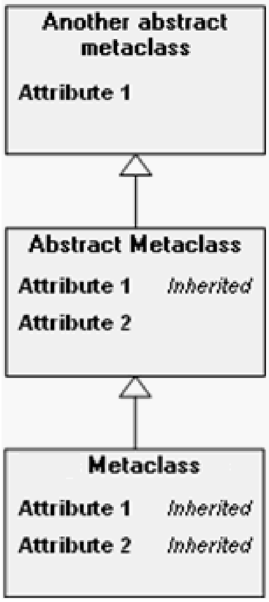
1. Create a new MetaClass (for more details, see "Adding a MetaClass to a Metamodel Diagram)
2. Right-click the MetaClass and open its Properties.
3. Select the **Characteristics** tab and select the **Standard** subtab.
4. In the **MetaClass Layer** field, select **Abstract**.
   Note that the MetaClass changes color in the metamodel diagram and that its icon appears transparent in the navigation tree.

   You must not change an existing concrete MetaClass into an abstract MetaClass. Occurrences of this MetaClass will no longer be accessible. See "Abstract Metamodel Extension Recommendations.

Inheritance relationships

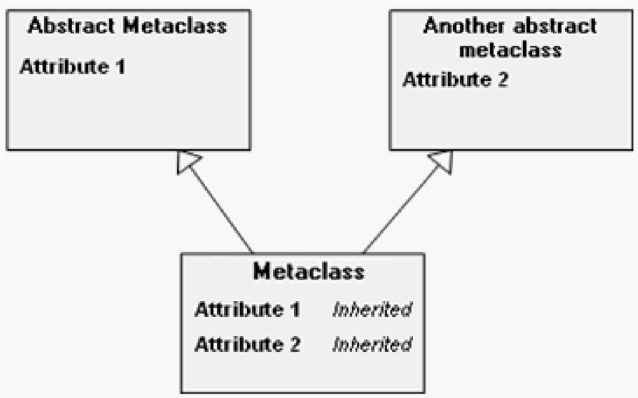
HOPEX inheritance is hierarchical and multiple:

- Hierarchical, since each MetaClass hierarchically inherits all characteristics of the abstract MetaClass from which it inherits.



💣 **Inheritances of several levels are not recommended.**

- Multiple, since a MetaClass can inherit several MetaClasses.



💣 **A MetaClass of the "System" repository cannot inherit an abstract MetaClass of the "Data" repository.**

The list of MetaClasses from which a MetaClass inherits is accessible from the Properties window of the MetaClass, in the **Characteristics** tab, **Standard** subtab. The **SuperMetaClass** field allows you to view and update this list.

To view with the explorer the list of MetaClasses from which a MetaClass inherits (for example: "Element with Note"):

1. Right-click the MetaClass and select **Explore**.
   An explorer window opens.
2. Expand the "SuperMetaClass" folder.

> ☛ *Similarly, by expanding the "SubMetaClass" folder, you obtain the list of all MetaClasses that inherit the current MetaClass.*

Managing Generic MetaAssociations

A generic MetaAssociation is a MetaAssociation defined between an abstract MetaClass and another MetaClass.

All MetaClasses inheriting the abstract MetaClass inherit this "Generic" MetaAssociation by default.

You can specify that a MetaAssociation already existing between two concrete MetaClasses is in fact an alias of a generic MetaAssociation. In this case, this MetaAssociation inherits all the attributes of the generic MetaAssociation. These attributes cannot be redefined.

```
For example, the MetaAssociation "Source Activity" / "Sent
Message" is an alias of the generic MetaAssociation
"Message Source" / "Sent Message" inherited from the
abstract MetaClass "Messaging Participant" by the concrete
MetaClass "Functional activity".
```

> **If you use aliases, you must remember to convert your repositories and comply with the recommendations relating to aliases. For more details, see "Abstract Metamodel Extension Recommendations.**

A MetaAssociation can be defined as an alias of a generic association for two reasons:

- Because the MetaAssociation existed before creation of the generic MetaAssociation representing the same link. This MetaAssociation should be kept for compatibility reasons.
- Because the MetaAssociation represents reduction of the generic MetaAssociation to a concrete MetaClass.

Managing MetaAssociation inheritance

To define that an alias on a MetaAssociation is the alias of a generic MetaAssociation:

1. Open the properties of the MetaAssociation you want to modify.
2. Select the **Characteristics** tab then the **Advanced** subtab.
3. In the **Super MetaAssociation** field, select the generic MetaAssociation that interests you.
4. In the **Meta Specialization Type** field, select:

- **Depreciated** if the alias is defined to assure compatibility.
- **Restrictive** if the alias is defined to reduce the number of MetaClasses concerned by the generic MetaAssociation.

Accessing inherited MetaAssociations

To view the list of MetaAssociations that are aliases of a generic MetaAssociation:

1. Right-click the MetaAssociation and select **Explore**.
2. Expand the "Restricting MetaAssociation" folder to obtain lists of restrictions of this MetaAssociation.

☛ *Similarly, expand the folder "Restricted MetaAssociation" of a MetaAssociation from which it inherits and you obtain the generic MetaAssociation.*

Viewing types in a navigation tree

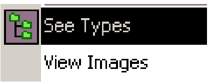
A generic MetaAssociation can connect objects of different MetaClasses.

In a navigation tree, a generic MetaAssociation appears as a branch to a list of objects of heterogeneous MetaClasses.

```
For example, if you use the explorer on an organizational
process described by an organizational process diagram, the
navigation tree will display an "Owned Element" branch.

If you expand this branch, you will see a list of objects of
different MetaClasses.
```

To store these objects in the explorer as a function of their MetaClass:

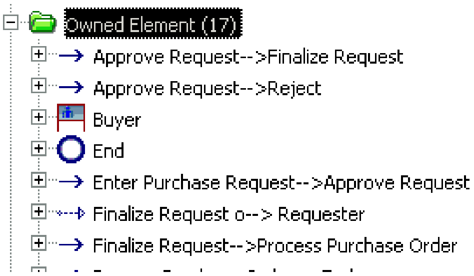
1. Click **View > Option**.
   The list of options appears.
2. Select **View types**.



If you expand the "Owned element" folder, you will see the sub-folders corresponding to each type.

To view icons associated with MetaClasses:

- In the same way, select **View > Option > View Images**.


Owned Element (17)
Approve Request-->Finalize Request
Approve Request-->Reject
Buyer
End
Enter Purchase Request-->Approve Request
Finalize Request o--> Requester
Finalize Request-->Process Purchase Order

☛ *To view repository objects by type, in the* ***Home*** *navigation tree, right-click the repository name and select* ***Regroup by Type****.*

Adapting Development Tools to the Abstract Metamodel

The use of abstract MetaClasses and generic MetaAssociations modifies tools of the **HOPEX** platform.

Property pages

To access the list of MetaAttributes of an abstract MetaClass:

1. Open the Properties window of the MetaClass.
2. Select the **MetaAttribute** tab.
   Note the three subtabs:
   - **MetaAttribute**, groups the list of MetaAttributes specific to the current MetaClass.
   - **SuperMetaAttribute**, groups the list of inherited MetaAttributes.
   - **Standard**, groups the MetaAttributes automatically inherited by any new MetaClass. The format of these attributes is defined as"Standard" in the **MetaAttribute Format** field of the **Characteristics** tab. For more details, see "MetaAttributes.

To access the list of MetaAssociations inherited from an abstract MetaClass:

1. Open the Properties window of the MetaClass.
2. Select the **MetaAssociation** tab.
   Note the two subtabs:
   - **MetaOppositeAssociationEnd**, groups the list of MetaAssociations specific to the current MetaClass
   - **SuperMetaOppositeAssociationEnd**, groups the list of inherited MetaAssociations.

Abstract metamodel diagrams

In an abstract metamodel diagram, graphic functionalities enable visual differentiation of abstract metaclasses, inheritance relationships and generic links.

To view metamodel diagrams proposed in the **HOPEX** platform:

- In the **MetaStudio** navigation window, expand the **Metamodel** folder, then the **MEGA Modeling** folder.

For example, to access the metamodel diagram concerning libraries:

1. In the **MetaStudio** navigation window, expand the **Metamodel** folder, then the **MEGA Modeling** folder.
2. Open the metamodel diagram "Library & Packaging Model".
   Note that:
   - the "Library" MetaClass is grayed
   - the abstract MetaClasses "Owner Packager", "Packaged Element" and "Library Element" appear in color
   - a directional link indicates that the "Library" MetaClass inherits the "Container" MetaClass
   - a generic MetaAssociation is defined between the "Container" abstract MetaClass and the "Containable Element" abstract MetaClass.

Query Tool

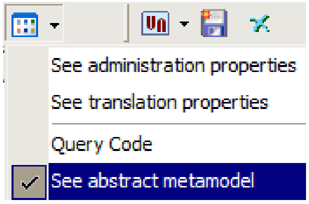
Querying from an abstract MetaClass

The query tool enables access to an occurrence via an abstract MetaClass from which it inherits.

```
For example: Select [Library element]
```

To access abstract MetaClasses in the query tool:

1. Open the query tool.
2. Click **Show** and select **See abstract metamodel**.



You then have access to abstract MetaClasses.

Querying from a generic MetaAssociation

The query language of **HOPEX** has evolved to take account of generic MetaAssociations. Operator "**:**" enables targeting of a query result on a specific MetaClass.

```
For example, the generic MetaAssociation"Message sender"/
"Message sent" connects the MetaClass "Message" to the
abstract MetaClass "Message participant" from which the
"Operation" MetaClass inherits. To access the list of
messages the recipient of which is an operation, it is
possible to use the MetaOppositeAssociationEnd "Message
recipient" and restrict the result using operator ":".
Syntax of the query is as follows:

Select [Message] where [Message Recipient]:[Operation]
```

☛ *When the concrete MetaClass has been specified, you can add to the selection MetaAttributes and MetaAssociations belonging to the concrete MetaClass.*

Evolution of APIs

A consequence of the abstract metamodel is that conventional MetaAssociations, MetaAssociationEnds and MetaAttributes are no longer systematically accessible from a concrete MetaClass.

New APIs, using the compiled metamodel are proposed, to enable optimized and precise access to abstract metamodel concepts. These are generally based on "MegaObject" and "MegaCollection" concepts.

Accessing abstract MetaClasses

Objects handled in **HOPEX** are attached to a concrete MetaClass inherited from one or several abstract MetaClasses.

The **GetType** function allows an object or collection to be considered as an instance of a given MetaClass.

Used from a ***MegaObject***, the **GetType** function enables consideration of the object as a function of the MetaClass given as parameter.

```
Example: MyOrgProc.GetType("BPMN Owner Element").explore
runs the explorer from an object of the Organizational
Process concrete MetaClass, considering it as an object of
the "BPMN Owner Element" MetaClass.
```

☛ *Used without a parameter, operator **GetType** enables consideration of the current object as an element of the concrete MetaClass to which it belongs.*

Used from a ***MegaCollection*** of objects of different concrete MetaClasses, operator **GetType** allows you to obtain a collection restricted to instances of the MetaClass specified as parameter.

```
Example: oCollection.GetType("Sequence flow").explore runs
the explorer on objects of the collection belonging to the
"Sequence flow" MetaClass.
```

If no object of the collection inherits the MetaClass given as parameter, operator **GetType** returns nothing. No error is generated.

Notions of class and collection description

APIs enabling browsing of the abstract metamodel are accessible from two main entry points:

- "ClassDescription" which finds the MetaClass of a "MegaObject" from the function **GetClassObject** .

  `Example:` myMegaObject.GetClassObject.Explore
- "CollectionDescription" which enables discovery of the interface of a "MegaCollection" from the **GetTypeObject** function.

  `Example:` myMegaCollection.GetTypeObject.Explore

API s available from a class description

Among available APIs, APIs relating to the abstract metamodel are:

- **UpperClasses**: returns the list of inherited MetaClasses, viewed as ClassDescription
- **LowerClasses**: returns the list of MetaClasses, viewed as ClassDescription, inheriting the given MetaClass..

☛ *These APIs return a collection accessible by **GetCollection***

APIs available from a collection description

The main APIs available are:

- **TargetClassID**: returns identifier of the target MetaClass (of the collection)
- **SourceClassID**: returns identifier of the source MetaClass (of the collection)
- **TargetTypeID**: returns identifier of the target MetaClass, abstract in the case of a generic MetaAssociation
- **SourceTypeID**: returns identifier of the source MetaClass, abstract in the case of a generic MetaAssociation
- **AliasID**: returns identifier of the alias of the collection if this exists
- **RootID**: returns identifier of the generic association if we are on an alias

☛ *These APIs return a property accessible by* ***GetProp***

- **IsSuperClassOf ()**: returns a positive boolean if the MetaClass passed as argument inherits the MetaClass from which the collection is built
- **IsSubClassOf ()**: returns a positive boolean on elements of the collection that inherits the MetaClass passed as argument
- **IsClassAvailable()**: tests if an object of the MetaClass passed as argument can be inserted in the collection

☛ *These APIs are functions with argument*

Abstract Metamodel Extension Recommendations

Given that an extension of the abstract metamodel impacts MetaClasses and occurrences of the repository, and that modifications can be lost when installing a new version of **HOPEX**, you must comply with certain rules before intervening on the abstract metamodel.

Recommendations to be taken into account are the following:

- You can add abstract MetaClasses.
- Inheritances of several levels are not recommended.
- You must not modify status of MetaClasses of **HOPEX**, change parameters, or add MetaAttributes.
- You can add generic MetaAssociations, but they cannot relate to abstract MetaClasses of **HOPEX**. They can relate to a concrete MetaClass of **HOPEX**.
- You can add MetaAttributes, on condition that they do not relate to abstract MetaClasses of **HOPEX**. They can be connected to a generic MetaAssociation of **HOPEX**.
- You should avoid creating an abstract MetaClass simply because a MetaPropertyPage is common to several MetaClasses. It is preferable to configure different MetaPropertyPages on the adapted MetaClasses.
- You should avoid creating an abstract MetaClass simply because a MetaPropertyPage is common to several MetaClasses. It is preferable to configure different MetaCommands on the adapted MetaClasses.
- When you define a MetaAssociation that already exists as an alias of a generic MetaAssociation, you must convert all repositories that use the MetaAssociation defined as alias.
- If a **HOPEX** MetaAssociation is defined as alias, this alias cannot be removed.
- If you remove an alias on a MetaAssociation that is not **HOPEX**, you risk loss of consistency of your repositories and loss of links.
- The MetaAttributes of the MetaAssociation defined as alias must be identical to the MetaAttributes of the generic MetaAssociation.
- If a MetaAssociation defined as alias has additional MetaAttributes compared with the generic MetaAssociation, these MetaAttributes must be defined at the level of the generic MetaAssociation.

A perimeter enables building a set of objects and links from a root object.

PERIMETERS

See:

- "Introduction to Perimeters
- "Viewing MetaAssociation Behavior Related to a Perimeter
- "Modifying the MetaAssociation Behavior Related to a Perimeter
- "Creating a Perimeter
- "Using a Perimeter in a MetaTool
- "Modifying the MetaTool Default Perimeter
- "Customizing a Standard Perimeter

Introduction to Perimeters

Perimeters enable configuration of the propagation mechanism which enables building of a set of objects from one or several objects called root objects.

This mechanism is used by certain tools to apply processing to this set.

`Examples: Export, Deletion, Duplication.`

☛ *See use of perimeters with MetaTools **Export** or **Compare and Align** in the **HOPEX Administration - Supervisor** guide.*

MetaTool

A MetaTool is an object that enables definition of perimeters that can be used with a tool.

There is a MetaTool for each tool that uses the propagation mechanism.

To build this set of objects, the MetaTool applies a perimeter to the root object.

☛ *You can assign a default perimeter for the MetaTool (see "Modifying the MetaTool Default Perimeter).*
The aim of this perimeter is to provide a standard behavior if no customized perimeter exists in this context.

Propagation

Propagation is the platform mechanism which enables building of a set of objects from one or several objects called root objects.

The propagation principle is to include in the resultant set all the objects connected to an object, then the objects connected to the connected objects, and so on. The resultant set comprises all objects directly or indirectly linked to root objects.

Certain links are taken into account and others not: for a given object, inclusion or non-inclusion in the resultant set of objects linked to the initial object according to link type is called propagation behavior for this link type. Perimeters enable definition of propagation behavior of link types.

Perimeters are associated with MetaTools. MetaTools represent tools that use perimeters to indicate link type propagation behaviors when using a given tool.

Table: Description of propagation behaviors

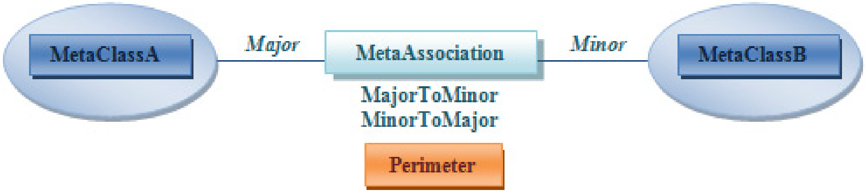
Value	Icon	Propagation description
Deep	🟩	***Recursive complete propagation***: Takes into account this link and the opposite object only. Propagation continues.
Standard	🟧	***Simple propagation***: Takes into account this link and the opposite object only. Propagation stops.
Link	□	***Limited propagation:*** Takes into account this link but not the opposite object. Propagation stops.
Abort	■	***No propagation:*** Does not take into into account this link or the opposite object. No propagation:
Computed	▦	***Propagation dependent on context*** Link type does not enable determination of perimeter behavior of this object. Propagation depends on context; it is defined by a macro and can take values "Deep", "Standard", "Link" or "Abort". See "Creating a Perimeter.

Scope

Propagation behavior is defined by the perimeter.

Each MetaAssociation is linked to a major MetaClass and a minor MetaClass. The perimeter determines the value (Deep, Standard, Link, Abort, Computed) of its MetaAttributes MajorToMinor and MinorToMajor.

☛ *The value of MetaAttributes can also be defined by the MetaAssociationType associated with the MetaAssociation.*



In this example:

- if MetaclassA is taken as root object, propagation behavior to be taken into account is the value of the MetaAttribute MajorToMinor.

  ☛ *Depending on this value, the link and the opposite object MetaClassB are added or not to the set of objects and links.*

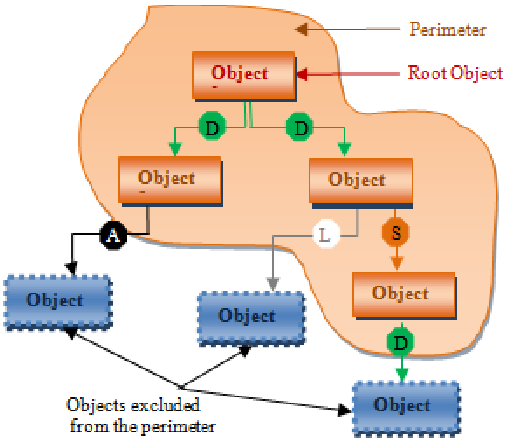
- if MetaclassB is taken as root object, propagation behavior to be taken into account is the value of the MetaAttribute MinorToMajor.

  ☛ *Depending on this value, the link and the opposite object MetaClassA are added or not to the set of objects and links.*

Propagation example

The following example presents object-to-object propagation, from root object "Object 1":

- The perimeter includes object 2 (link ***Deep*** with object 1).
  Propagation continues and excludes object 4 (link ***Abort*** with object 2).
- The perimeter includes object 3 (link ***Deep*** with object 1).
  Propagation continues and:
  - excludes object 5 (link ***Link*** with object 3).
  - includes object 6 (link ***Standard*** with object 3).
    Propagation stops, object 7 is not processed and is excluded despite its ***Deep*** link with object 6.



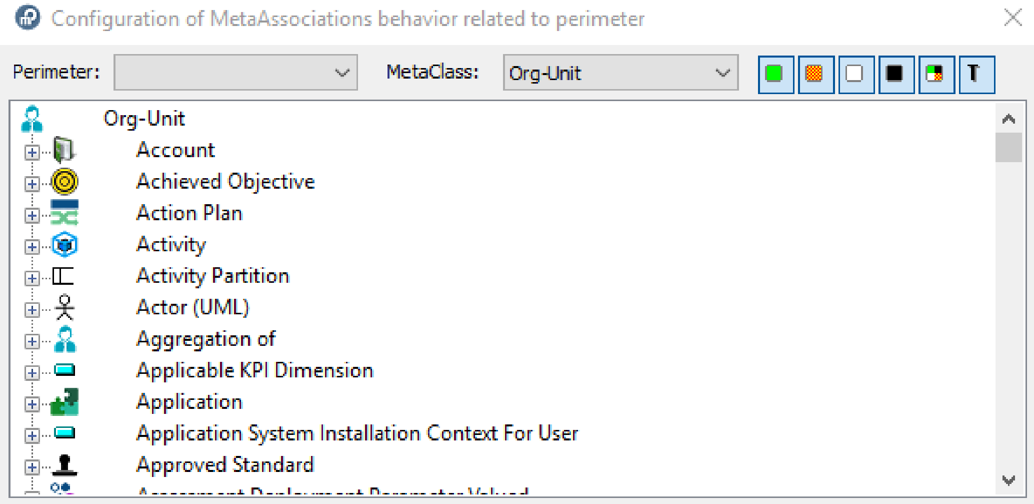
Viewing MetaAssociation Behavior Related to a Perimeter

To view MetaAssociation behavior related to a perimeter:

1. In the **MetaStudio** navigation window, expand the **MetaClass** folder, then the MetaClass type folder.

2. Right-click the MetaClass concerned and select **Manage > Parameterize**.
   The **Configuration of MetaAssociations Behavior Related to Perimeter** dialog box appears.

   ☛ *In the **MetaClass** box, the MetaClass concerned is already selected. You can modify this selection via the drop-down menu.*


Configuration of MetaAssociations behavior related to perimeter

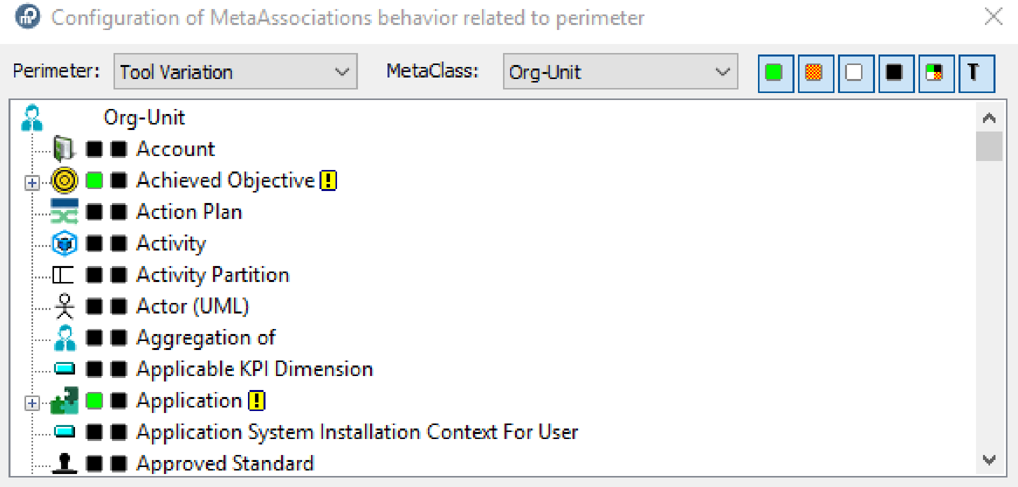
Perimeter:

MetaClass: Org-Unit

3. In the **Perimeter** box drop-down menu, select the perimeter you want to study.

   ☛ *Only those perimeters with _OpPreview characteristic value "Active" are visible.*

   The **Configuration of Behavior of MetaAssociations Related to Perimeter** dialog box is updated.


Configuration of MetaAssociations behavior related to perimeter

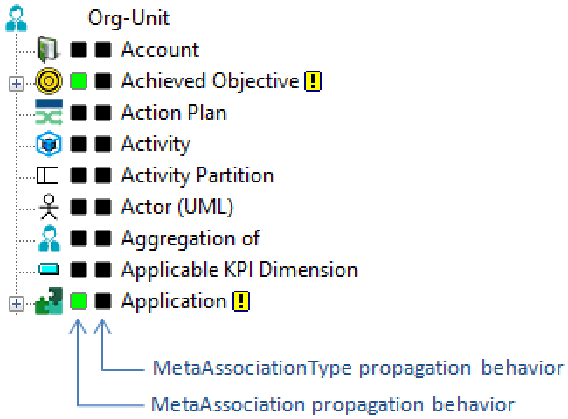
Perimeter: Tool Variation

MetaClass: Org-Unit

The **Configuration of Behavior of MetaAssociations Related to Perimeter** dialog box details for the selected MetaClass (here "Org-Unit") the behavior of the selected perimeter (here "Tool Variation") for each MetaClass concerned:

- propagation behavior of each MetaAssociation (link)
- propagation behavior of each MetaAssociationType (link type)

☛ *For description of icons, see "MetaTool.*



Org-Unit
- ■ ■ Account
- 🟩 ■ Achieved Objective !
- ■ ■ Action Plan
- ■ ■ Activity
- ■ ■ Activity Partition
- ■ ■ Actor (UML)
- ■ ■ Aggregation of
- ■ ■ Applicable KPI Dimension
- 🟩 ■ Application !

MetaAssociationType propagation behavior

MetaAssociation propagation behavior

You can filter display:

- To show only certain propagation types, click **Deep** 🟩, **Standard** ▨, **Link** □, **Abort** ■ and/or **Computed** ◩.
- To show/hide behavior of MetaAssociationTypes (link default behavior before customization), click T.
  A second column appears/disappears at the right of the propagation types column.

☛ ! *indicates that propagation of the MetaAssociation is different from the MetaAssociationType.*

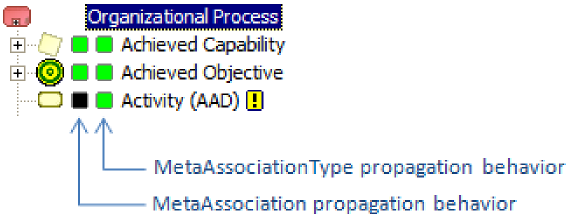
Modifying the MetaAssociation Behavior Related to a Perimeter

You can modify MetaAssociation behavior related to a perimeter.

To modify MetaAssociation behavior related to a perimeter:

1. Open the **Configuration of Behavior of MetaAssociations Related to Perimeter** dialog box, see "Viewing MetaAssociation Behavior Related to a Perimeter.
2. Right-click the MetaAssociationEnd.

3. In the **Behavior** section, select **Propagate**, then the propagation behavior.
   - The icon on the left, representing the behavior of the perimeter on the MetaAssociation, is updated according to the selected propagation behavior.
   - The ! icon indicates that propagation of the MetaAssociationEnd is different from propagation of the MetaAssociationType.


Organizational Process
Achieved Capability
Achieved Objective
Activity (AAD)
MetaAssociationType propagation behavior
MetaAssociation propagation behavior

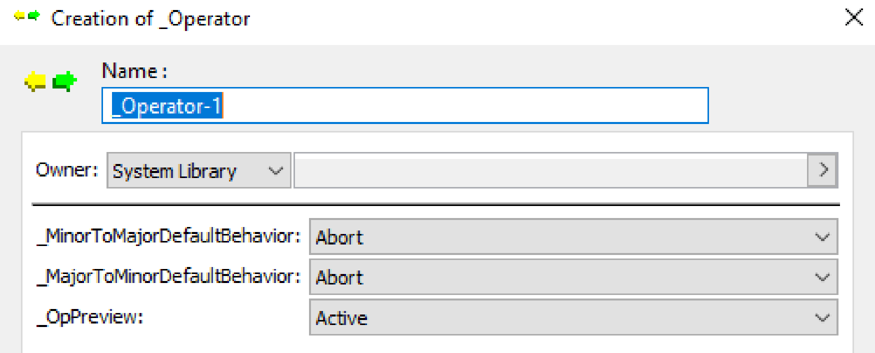
The new behavior is active from next use, except for "Computed" behavior, which required additional configuration.

> ☛ *"Computed" behavior uses a macro (_BehaviorMacro which must be previously created and linked to the current link and perimeter, see "Creating a macro to calculate a MetaAttribute.*

Creating a Perimeter

To create a perimeter:

1. In the **MetaStudio** navigation window (**View > Navigation Windows > MetaStudio**), expand the **Perimeters** folder.
2. Expand the **Perimeters** folder.
3. Right-click the **Custom Perimeters** folder and select **New > _Operator**.



Creation of _Operator

Name : _Operator-1

Owner: System Library

_MinorToMajorDefaultBehavior: Abort

_MajorToMinorDefaultBehavior: Abort

_OpPreview: Active

4. Enter a **Name** for the perimeter.

5. Define characteristics of the perimeter:

   ☛ *For information on how to configure propagation, see "Introduction to Perimeters*

   - **_MinorToMajorDefaultBehavior** represents the default behavior of links of major objects to minor objects.

     ☛ *The default value "Abort" (links not propagated) avoids generating large volumes of objects.*

   - **_MinorToMajorDefaultBehavior** represents the default behavior of links of minor objects to major objects.

     ☛ *The default value "Abort" (links not propagated) avoids generating large volumes of objects.*

   - **_OpPreview** enables display of the perimeter in configuration of the MetaTool (see "Viewing MetaAssociation Behavior Related to a Perimeter).

     ☛ *Default value "Active"*

6. Click **Finish**.
   The new perimeter appears in the list of **Customized Perimeters**.

   ☛ *To use this perimeter with a MetaTool, see "Using a Perimeter in a MetaTool then "Modifying the MetaTool Default Perimeter.*

Using a Perimeter in a MetaTool

To use a perimeter in a MetaTool, you must connect this perimeter to the MetaTool concerned.

☛ *You can connect a perimeter to several MetaTools, enabling reuse of the same perimeter in different contexts.*

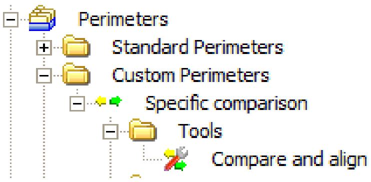
To connect a perimeter to a MetaTool:

1. In the **MetaStudio** navigation window, expand the **Perimeters** folder.
2. Expand the **Standard Perimeters** or **Custom Perimeters** folder concerned (see "Creating a Perimeter).
3. Expand the perimeter concerned.
4. Right-click the **Tools** folder and select **Connect > MetaTool**.
5. Using the **Query** tool, select the MetaTool concerned.

   ☛ *You can select several MetaTools.*

   The MetaTool is added to the **Tools** folder.

```
In the following example, the "Compare and Align" MetaTool
is connected to the "Specific comparison" perimeter
```


Perimeters
Standard Perimeters
Custom Perimeters
Specific comparison
Tools
Compare and align

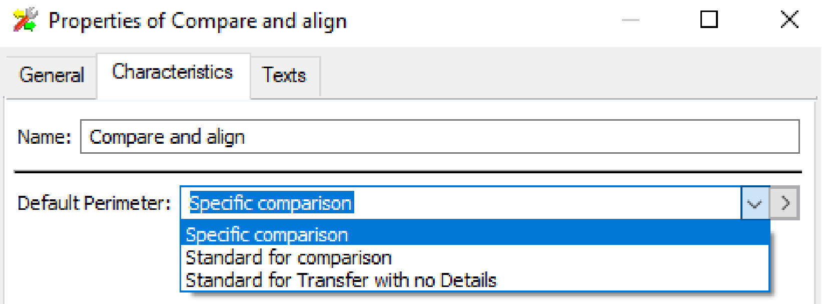
Modifying the MetaTool Default Perimeter

☛ *The default perimeter you want to define for the MetaTool must previously be connected to the MetaTool, see "Using a Perimeter in a MetaTool.*

A MetaTool can be connected to several perimeters, which enables building different sets of objects and links from the same root object.

To modify the MetaTool default perimeter:

1. In the **MetaStudio** navigation window, expand the **Perimeters** folder.
2. Expand the **Perimeters by Tools** folder.
3. Open the properties of the MetaTool concerned.
4. In the **Characteristics** tab, modify the value of **Default Perimeter** via the drop-down menu.



Properties of Compare and align

Name: Compare and align

Default Perimeter: Specific comparison

The default perimeter of the MetaTool is modified.

Customizing a Standard Perimeter

Standard perimeters are stored in the **Perimeters > Standard Perimeters** folder.

It is not recommended to modify these standard perimeters.

Instead, you must create a custom perimeter, which inherits from the standard perimeter you want to customize and configure this custom perimeter.

To modify a standard perimeter:

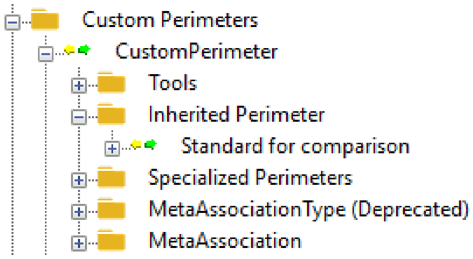
1. Create a perimeter.

   ☛ *See "Creating a Perimeter.*

2. Expand the perimeter concerned.
3. Right-click the **Inherited Perimeter** folder and select **Connect > _Operator**.

4. Select the perimeter you wanted to modify and click Connect.

   ```
   For example: "Standard for comparison" _Operator.
   ```



Custom Perimeters
CustomPerimeter
Tools
Inherited Perimeter
Standard for comparison
Specialized Perimeters
MetaAssociationType (Deprecated)
MetaAssociation

5. Customize your perimeter.

NAMESPACES

The following points are covered here:

- "Managing Namespaces
- "Defining Namespaces
- "Canceling Namespaces
- "Ownership and Use Links

Managing Namespaces

Uniqueness of object name is normally checked against all objects of the same MetaClass in the repository. It is however possible to limit uniqueness check of a name to a particular context known as the namespace.

You can for example use two different conditions with the same name in two different procedures.

```
The condition "If approved" in the "Purchasing Processing"
procedure has complete name "Purchasing Processing::If
approved".
```

```
The condition "If approved" in the "Order Processing"
procedure has complete name "Order Processing::If
approved".
```

These two conditions are two different objects in the repository and can be connected to different operations.

When an object is included in a namespace, two names are presented:

- Local name defined as below of which length is limited to 140 characters.
- Name comprising the local name preceded by the name of the owner object (namespace). Display of this name is limited to 255 characters. If length to be displayed exceeds this limit, "..." are displayed in the middle of the name to replace missing characters.

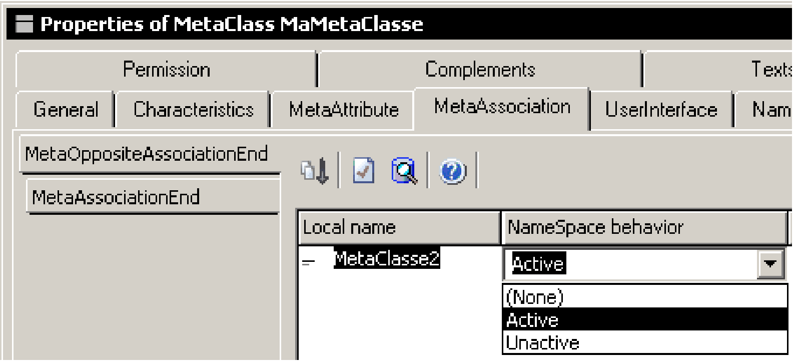
☛ *You can define namespaces in cascade. Example: A message owned by a collaboration itself owned by a business area. In this case, all successive namespaces are displayed in the object name.*

Defining Namespaces

To include a MetaClass in a namespace:

1. Check that you are in "extended metamodel" mode.
2. Open the Properties window of the MetaClass concerned.

3. In the **MetaAttribute** tab, click **Connect** to connect the **Generic Local Name** MetaAttribute.
   A dialog box prompts you to execute a query to find the MetaAttribute concerned.
4. Execute the query and connect the **Generic Local Name** MetaAttribute.
5. Select the **MetaAssociation** tab, then the **MetaOppositeAssociationEnd** subtab.
6. Select **Active** for the **Local Name Behavior** property of the MetaAssociationEnd displayed in the tab.



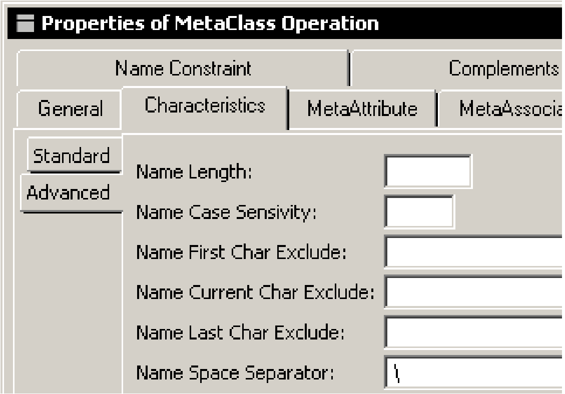
Local name	NameSpace behavior
MetaClasse2	Active

The MetaClass is now in a namespace, with "::" as default separator.

Customizing the separator

To customize the separator, you must modify **NameSpaceSeparator** in the MetaClass Properties window. You can use:

- a simple string to replace "::".



Name Length:

Name Case Sensivity:

Name First Char Exclude:

Name Current Char Exclude:

Name Last Char Exclude:

Name Space Separator: \

- a string containing the expressions "%M" and "%S".
  - "%M" indicating the namespace (the parent)
  - "%S" indicating the object in the namespace (the child)

☛ *Respect character case.*

💣 **When the form of separator with expressions "%M" and "%S" is used, certain query by name functions cannot be used.**

Complementing specification of the MetaAssociation used for the namespace

So that deletion of the namespace also deletes the objects owned by this namespace:

❱ Connect the MetaAssociation between the MetaClass and its namespace using a "Hierarchy" link type.

☛ *This option is recommended.*

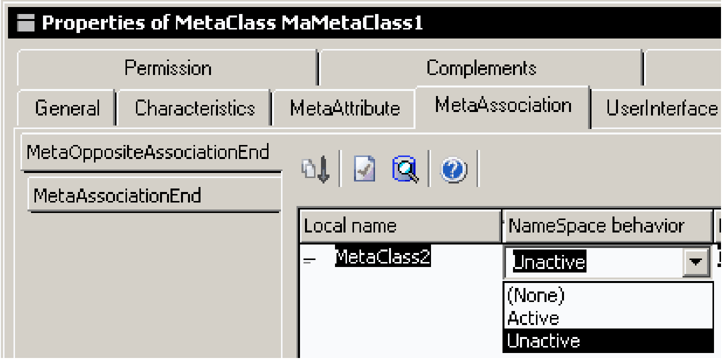
Canceling Namespaces

MetaClasses connected to GenericLocalName

To cancel a namespace on a MetaClass (when the namespace has been created using the method described in "Defining Namespaces"):

1. Open the Properties window of the MetaClass concerned.
2. From the **MetaAssociation** tab select the **MetaOppositeAssociationEnd** subtab.

3. Deactivate local name behavior.



Other MetaClasses

To cancel a namespace on a MetaClass:

1. In the explorer, open the MetaClass for which you want to cancel the namespace.
2. Select the MetaAttribute that has "0000000040000002" as a value for the "Substituted MetaAttribute" attribute.
3. Remove value "0000000040000002" of the Substituted MetaAttribute attribute from the link between the MetaAttribute and the MetaClass.
4. Disconnect the MetaAttribute defined as local name for the MetaClass from "GBML NameSpace Server".

Ownership and Use Links

An ownership link indicates that an object (example: an activity) belongs to an owner object (example: a process).

However, in certain cases we may wish to reuse the same object (an activity) in another context (in another business process). The object (the activity) will then be connected to the other context by a use link. An object can have only one owner.

So as to avoid exploring the two links to find all objects (example: the activities), a spy controller automatically connects via the use link all objects connected by the ownership link.

This mechanism exists between Business Process and Activity, Procedure and Operation, etc.

IMPORT ISSUES

Extensions

Extensions may not be imported correctly for the following reasons:

Text format

The command file can be created with any word processor, such as Word, but it must be saved in Text Only format. If it is in Word format, it will not be recognized by the command interpreter.

☺ *Use Notepad, Write, or Wordpad and avoid formatted text.*

Open private workspaces

Metamodel extensions will be visible in a private workspace only if you dispatch or refresh your workspace.

Dash at the beginning of commands

A dash (-) indicates a comment. Lines beginning with a dash are ignored.

End-of-file character within the commands

Certain file concatenation operations can insert an end-of-file character within the command file. In this case, processing stops at this character.

TRANSLATING THE METAMODEL

You can translate the metamodel of a system repository into another language. MetaClasses, MetaAssociations, MetaAttributes and texts can have a different name depending on the language in which the user is working: this allows a project team to have a single repository for sites working in different languages.

Translating a repository involves the following operations:

- Defining the names in the new language, in a *command file*.
- Importing the command file.
- Translation by the translation utility.

Translating and Compiling Environments

You can translate the metamodel of an environment if several languages are defined in the metamodel. When the metamodel has been modified, it must be compiled. To do this, translate it in the current language.

☛ *For more information on translating and compiling the metamodel, see **HOPEX Administration-Supervisor** guide, chapter "Managing Environments".*

RENAMING HOPEX CONCEPTS

Some standards (e.g.: NAF, DoDAF, Archimate) use their own terminology. **HOPEX** concepts can be renamed according to the context in which they are used.

> For example, the MetaClass called "Application System" in **HOPEX IT Architecture** standard product is called "Application Collaboration" MetaClass in **Archimate** Terminology.

The renaming mechanism implemented in **HOPEX** enables definition of different names carried by the same concept in its different contexts of use. Each user, depending on his/her profile and the context in which he/she is working, uses terminology with which he/she is familiar.

Functionalities proposed here are based on the **Terminology** notion.

The following points are covered here:

- "Defining a Terminology
- "Managing Profiles Associated with Several Terminologies
- "Renaming Concepts
- "Concepts that Can be Renamed

Defining a Terminology

A new context of use of **HOPEX** concepts is defined by all the set of new terms to be used. These terms can be defined in several languages.

> 🕮 *A **Terminology** is a set of terms used in a specific context instead of the names used in basic configuration. This context can be a modeling standard (e.g.: Archimate, TOGAF), the vocabulary used in a specific field (e.g.: audit, internal control) or specific to the company.*

> ☛ *For more details on the list of concepts that you can rename, see "Renaming Concepts.*

Some terms can be specific to a population of users. You can connect a user profile to one or several Terminologies ordered by priority.

> ☛ *For more details regarding profiles connected to several Terminologies, see "Managing Profiles Associated with Several Terminologies.*

Some standards define pictures associated with their concepts. The renaming mechanism implemented in **HOPEX** enables association of MetaPictures with a terminology.

> ☛ *For more details on the use of MetaPictures, see "Defining a shape for the new MetaClass.*

> ☛ *For more details on how to associate MetaPictures with a terminology, see "Connecting MetaPictures of concepts.*

To define a Terminology:

1. Connect to **HOPEX** with the **HOPEX Customizer** profile.

2. Create a Terminology.

   ☛ *See "Creating a Terminology.*

3. Create all the languages you need for the Terminology and associate each of them with the Terminology.

   ☛ *See "Creating a language for a Terminology.*

4. Configure the Terminology:
   - specify the profiles associated with the Terminology

     ☛ *See "Specifying all the profiles associated with a Terminology.*

   - (optional) specify the concept MetaPictures for the Terminology

     ☛ *See "Connecting MetaPictures of concepts.*

5. From **HOPEX Administration**, translate and compile the Metamodel (you do not need to compile the technical data).

   ☛ *For details on the translation and compilation, see **HOPEX Power Supervisor** guide, chapter "Compiling an environment".*

6. Define the terms associated with your Terminology.

   ☛ *See "Renaming Concepts*

7. From **HOPEX Administration**, translate and compile the Metamodel (Technical Data and Metamodel).

   ☛ *For details on the translation and compilation, see **HOPEX Power Supervisor** guide, chapter Compiling an environment.*

Creating a Terminology

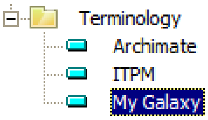
To create a Terminology:

1. Connect to **HOPEX** with the **MEGA Customizer** profile.
2. In the **MetaStudio** navigation window, right-click the **Terminology** folder and select **New > Terminology**.
3. Enter the terminology **Name**.

   `For example: My Galaxy.`

4. Click **OK**.
   The Terminology is created.

   ☛ *If you want your Terminology to inherits from another Terminology see "Inheriting a Terminology.*



Inheriting a Terminology

A Terminology can inherit from another one. Connected to the inherited Terminology, the inheriting Terminology inherits:

- languages

  ☛ *See "Creating a language for a Terminology.*

- MetaPictures

  ☛ *See "Connecting MetaPictures of concepts.*

- terms given to the different concepts in the different languages

  ☛ *See "Renaming Concepts.*

To connect an inherited Terminology:

1. Explore the Terminology you want to be inheriting from another Terminology.

   ☛ *For example explore "My Galaxy" Terminology.*

2. In the **Explore** tool, click **Empty Collections** 🗀.
3. Right-click the **Inherited Terminology** folder and select **Connect**.
4. In the **Terminology** list, select the Terminology you want your terminology to inherit from and click **Connect**.

   For example if you select "Archimate" Terminology, "My Galaxy" Terminology inherits languages, MetaPictures, and terms from "Archimate" Terminology.

Creating a language for a Terminology

To configure your Terminology you need to create the languages you want to be available for your Terminology and associate each language with the Terminology.

☛ *You can create as many languages as you need for your Terminology.*

To create a language for a Terminology:

1. In the **MetaStudio** navigation window, expand the **Language** folder.
2. Expand the language you want to be available for the Terminology.

   For example **English**.

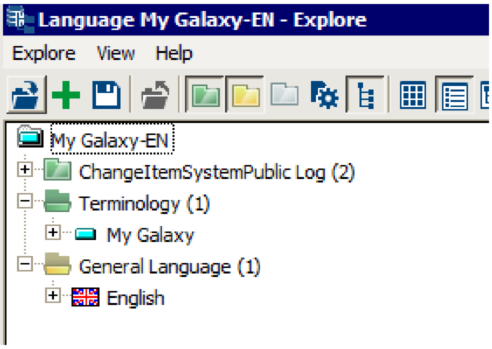
3. Right-click the **Specialized Language** folder and select **New > Specialized Language**.
   The **Creation of Language** dialog box opens.
4. In the **Name** field enter your language name.

   For example "My Galaxy-EN".

5. Right-click the language and select **Explore**.
6. In the **Explore** tool, click **Empty Collections** 🗀.
7. Right-click the **Terminology** folder and select **Connect**.

8. In the **Terminology** list, select your Terminology and click **Connect**.

   `For example "My Galaxy".`



Language My Galaxy-EN - Explore

- My Galaxy-EN
  - ChangeItemSystemPublic Log (2)
  - Terminology (1)
    - My Galaxy
  - General Language (1)
    - English

Specifying all the profiles associated with a Terminology

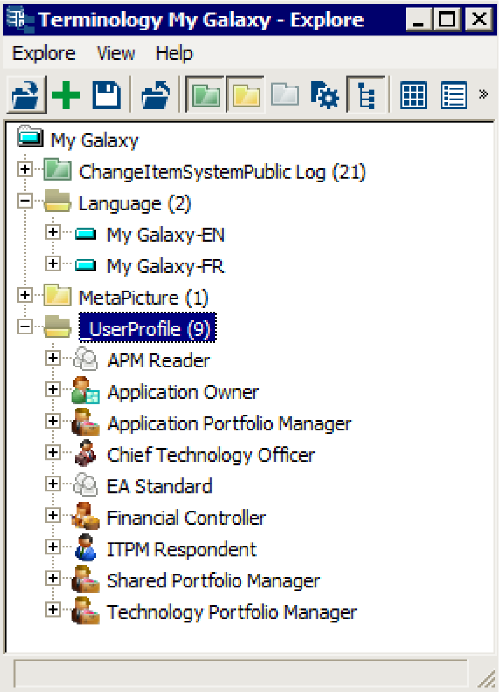
☛ *For more details on definition of users and profiles, see* ***HOPEX Power Supervisor****, chapter "Managing Users".*

To associate profiles with a Terminology:

1. From the **MetaStudio** navigation window, in the **Terminology** folder right-click the Terminology and select **Explore**.
   An **Explore** window opens.
2. Right-click the **_UserProfile** folder and select **Connect**.

3. From the **Connecting** window, select all the profiles you want to associate with the Terminology and click **Connect**.
   All the selected profiles are associated with the Terminology.

   ☛ *To add a profile to a Terminology see also "Adding a Terminology to a profile*



Connecting MetaPictures of concepts

Prerequisite: diagrams specific to the context to which the terminology relates are described and MetaPictures exist.

☛ *For more details on the use of MetaPictures, see "Defining a shape for the new MetaClass.*

To connect a concept MetaPictures to a Terminology:

1. From the **MetaStudio** navigation window, in the **Terminology** folder right-click the Terminology and select **Explore**.
   An **Explore** window opens.
2. Right-click the **MetaPicture** folder and select **Connect**.
3. From the **Connecting** window, in the MetaPicture list, select all the MetaPictures you want to connect to the Terminology and click **Connect**.

Managing Profiles Associated with Several Terminologies

☛ *For details on profiles see **HOPEX Administration - Supervisor** guide, "Managing users" chapter.*

You can associate a profile with several terminologies.

☛ *See "Adding a Terminology to a profile.*

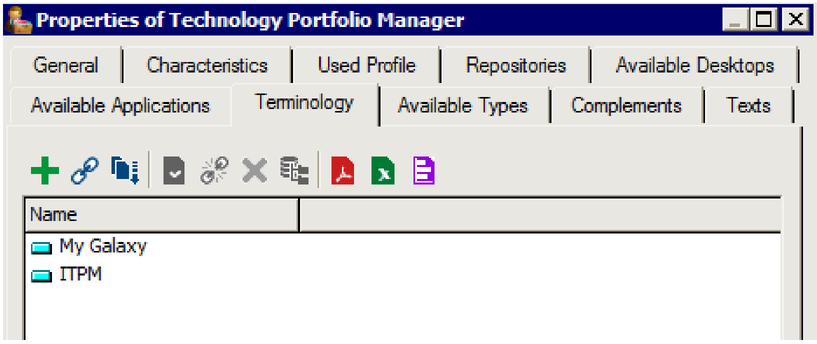
When a profile is associated with several Terminologies, you must order the Terminologies to define which of them must be displayed as priority over the other ones.

☛ *See "Defining the priority Terminology).*

Adding a Terminology to a profile

To associate a profile with a terminology:

1. Access the profile Properties.
2. Select the **Terminology** tab.
3. Click **Connect** to connect the terminology to the profile.

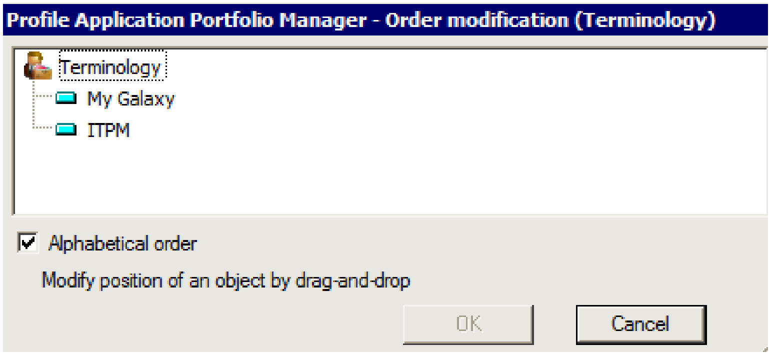


Defining the priority Terminology

By default Terminologies are listed in alphabetical order, so that the default Terminology displayed is the top level one.

To define the priority Terminology for a profile:

1. Access the profile properties.
2. Select the **Terminology** tab.
3. Click **Reorganize** to define the priority order of appearance of terms.


Profile Application Portfolio Manager - Order modification (Terminology)
Terminology
My Galaxy
ITPM
☑ Alphabetical order
Modify position of an object by drag-and-drop
OK
Cancel

4. Drag and drop the Terminologies to order them with the priority Terminology at the top.

   `If My Galaxy Terminology is above ITPM Terminology, ITPM terms are displayed when Galaxy terms have not been specified.`

Renaming Concepts

For each of the **HOPEX** concepts that can be renamed, you can redefine:

- the concept name

  ☛ *See "Changing translatable fields of a concept,*

- other translatable fields

  ☛ *See "Changing translatable fields of a concept.*

Changing a concept name for a specific Terminology language

☛ *For details on the list of concepts that can be renamed and the attribute that corresponds to the concept name, see "Concepts that Can be Renamed.*

Prerequisite: To use the new terminology, you must first translate and compile the Metamodel (you do not need to compile the technical data).

☛ *For more details on translate and compile, see **HOPEX Power Supervisor** guide, chapter "Compiling an environment".*

To change a concept name:

1. Access the concept Properties.

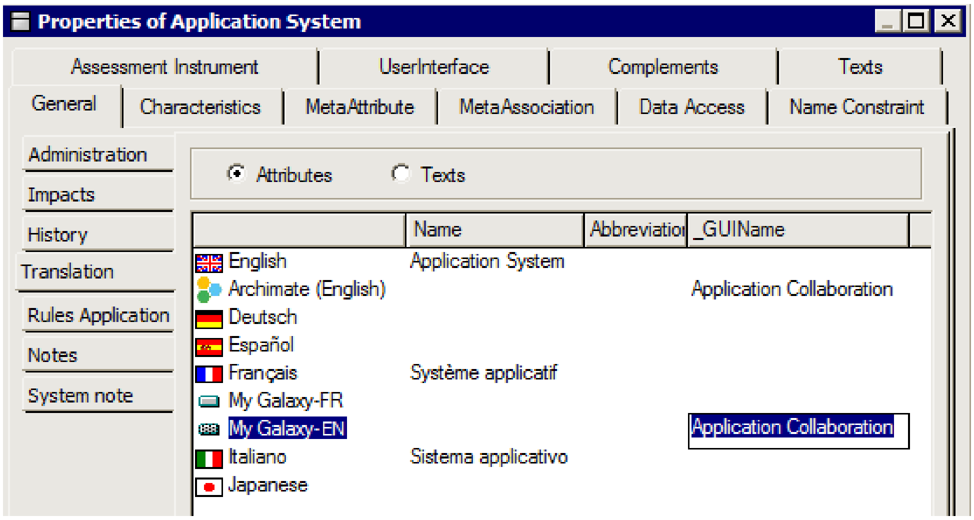
   `E.g.: the Application System MetaClass.`

2. In the **General** tab select **Translation** sub-tab.
3. Select **Attributes**.

4. In the language line, in the **_GuiName** field enter the concept name corresponding to the application context.

    For example you can modify the ***Application System*** Metaclass in the context of "My Galaxy" terminology for English.

    For "My Galaxy-EN" language, in the **_GUIName** field enter "Application Collaboration".



	Name	Abbreviatio	_GUIName
English	Application System		
Archimate (English)			Application Collaboration
Deutsch			
Español			
Français	Système applicatif		
My Galaxy-FR			
My Galaxy-EN			Application Collaboration
Italiano	Sistema applicativo		
Japanese			

5. Click **OK** to finish.

    ☛ *For these modifications to be taken into account you need to compile the Metamodel.*

Changing translatable fields of a concept

The majority of concepts that can be renamed have a "Comment" field, which can be modified to correspond to a context-specific terminology.

Some concepts, such as "_Code Template" have other fields that can be renamed (e.g.: "Comment", "Translatable Code template").

To modify a Code Template field:

    For example a Code Template in the context of Galaxy Terminology and for English language.

1. Access the _Code Template properties.
2. Select the **General > Translation** tab.
3. Select **Texts**.
4. Select the field you want to modify.

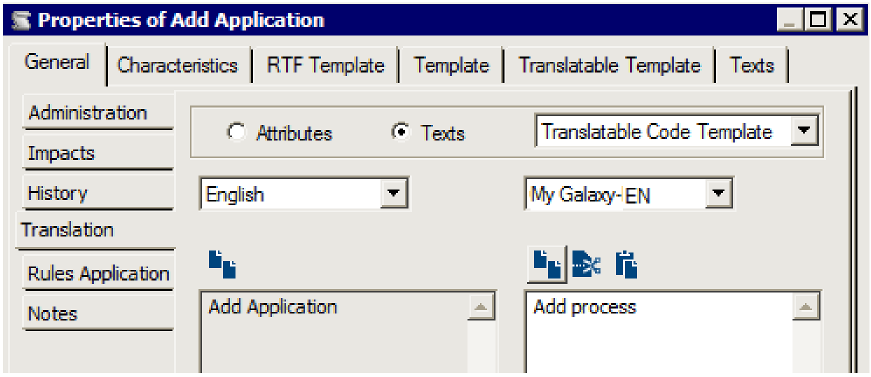
    For example: "Translatable Code template"

5. In the left language field, select the language from which you want to make the modification.

    For example: English.

6. In the right language field, select the language you want to specify.

   `For example: Galaxy-US.`



○ Attributes ● Texts

Translatable Code Template

English

My Galaxy-EN

Add Application

Add process

7. In the right pane, enter the modified string.
8. Click **OK**.

   ☛ *For these modifications to be taken into account you need to compile the Technical Data.*

Concepts that Can be Renamed

The following table lists the **HOPEX** concepts that can be renamed. For each of these:

- the "Main attribute" column indicates the field in which the name of the concept for a terminology and a given language should be specified
- the "Secondary attribute" column lists the other fields that can be modified for a terminology and a given language.

Concept	Main attribute	Secondary attribute
_Code Template	Translatable Code Template	
Business Role	Abbreviation, External Value	Comment
Chapter		Comment
Desktop	_GUIName	
Desktop Container	_GUIName	
DiagramTypeView	DiagramTypeviewName	

Concept	Main attribute	Secondary attribute
DiagramTypeField	_GUIName	
MetaAssociation		Comment
MetaAssociationEnd	_GUIName	Comment
MetaAttribute	_GUIName	Comment
MetaAttributeGroup	_GUIName	Comment
MetaAttributeValue	ExternalValue	ExternalAbbreviation
metaclass	_GUIName	Comment
MetaCommand Group	_GUIName	
MetaCommand Item	_GUIName	
MetaCommand Manager	_GUIName	
MetaField	_GUIName	
MetaListType	_GUIName	
MetaPattern	_GUIName	
MetaPropertyPage	_GUIName	Comment
MetaTree	_GUIName	Comment
MetaTreeBranch	_GUIName	
MetaTreeNode	_GUIName	
MetaWizard	_GUIName	Comment
Modeling Rule	Diagnosis	RuleDescription
Modeling Regulation	_GUIName	Comment
Profile	_GUIName	Comment
Report Chapter	_GUIName	Comment

Concept	Main attribute	Secondary attribute
Report Parameter	_GUIName	Comment
Report Template	_GUIName	Comment
TaggedValue	_GUIName	Comment
Workflow Status	_GUIName	Comment
Workflow Transition	_GUIName	Comment

Metamodel Syntax in Command Files

This section contains examples of syntax enabling understanding of metamodel management operations contained in command files.

Creating MetaClasses

Syntax	.Create MetaClass "Object Type"
Example	.Create .MetaClass "Company Object Type"

For each MetaClass, you can specify name maximum length, systematic conversion and a list of prohibited characters.

<table>
<tr><td><b>Syntax</b></td><td>.Update .MetaClass "MetaClass" ._NameLength Length -
._NameCaseSensivity "Conversion Code" -
._NameFirstCharExclude "Chars To Exclude" -
._NameCurrentCharExclude "Chars To Exclude" -
._NameLastCharExclude "Chars To Exclude"</td></tr>
<tr><td><b>Example</b></td><td>.Modifier .MetaClass "Programme" ._NameLength 16 -
._NameCaseSensivity "U" -
._NameFirstCharExclude "&é(-è_çà)=^$ù*¤!:;,<>?}êîûôöïüÿ" -
._NameCurrentCharExclude "0123456789*?.:,+-"-
._NameLastCharExclude "&é(-è_çà)=^$ù*¤!:;,~#{[|`\^@]}êîûôöïüÿ"</td></tr>
</table>

Typing MetaClasses

Syntax	Connect .MetaClass "MetaClass" ."_Operator" "Operator" ._ScanInit - "Extraction condition" ._ScanType "Processing type"
Example 1	.Connect .MetaClass "tag" ."_Operator" "Extract" ._ScanInit "S"
Example 2	.Connect .MetaClass "Table" ."_Operator" "Extract" ._ScanType "D"

The first command indicates that all repository tags are extracted systematically.

The second indicates that when a table is extracted, all the minor objects describing it (columns, keys, indexes) are also extracted.

Creating MetaAssociations

In a command file, the major MetaAssociationEnd is the first MetaAssociationEnd indicated in the MetaAssociation creation command.

Syntax	.Create .MetaAssociation "NameMetaAssociation" .MetaClass "MetaClassMajor" "MetaAssociationEndMajor" N -"MetaClassMinor" "MetaAssociationEndMinor" N
Example 1	.Create .MetaAssociation "(OU/S)" .MetaClass "Org-Unit" "Org-Unit" N - "tag" "tag" N
Example 2	.Create .MetaAssociation "Org-Unit-Sending" .MetaClass "Org-Unit" "Org-Unit-Sending" N - "Message" "Message-Sent" N
Example 3	.Create .MetaAssociation "Org-Unit-Composition".MetaClass "Org-Unit" - "Composition" N "Org-Unit" "Component" N

Reversing MetaAssociation Orientation

MetaAssociation orientation can be reversed in a command file using the following syntax:

Syntax	.Update .MetaAssociationEnd <MetaClass Name> <MetaAssociationEnd Name> ._Major <Major>
Example 1	.Update .MetaAssociationEnd "Tool" "User-Application" ._Major "0"
Example 2	.Update .MetaAssociationEnd "Application" "User-Tool" ._Major "1"

Modifying Link Type

Link type can be modified in a command file using the following syntax:

Syntax	.Disconnect .MetaAssociation "LinkName" ."MetaAssociationType" "Old Link Type" .Connect .MetaAssociation "NameOfLink" ."MetaAssociationType" "New Link Type"
Example	.Disconnect .MetaAssociation "Org-Unit-Composition" ."MetaAssociationType" "Composition" .Connect .MetaAssociation "Org-Unit-Composition" ."MetaAssociationType" "Hierarchy"

Modifying Object Protection

Syntax	.Update .MetaAssociationEnd "Object Type" "MetaAssociationEnd" ._Protection "U"
Example 1	.Update .MetaAssociationEnd "Operation" "Org-Unit" ._Protection "U" .Modify .MetaAssociationEnd "Org-Unit" "Operation" ._Protection "A"
Example 2	.Update .MetaAssociationEnd "Operation" "Org-Unit" ._Protection "U" .Update .MetaAssociationEnd "Org-Unit" "Operation" ._Protection "U"

Example 1 shows reversal of protections.

Example 2 shows deactivation of protections.

Modifying Link Behavior for a Given Operator

Syntax	.Disconnect .MetaAssociation "LinkName" ."Operator" "Operator"."Minor to major behavior" "MajorProcessing" ."Major to minor behavior" "MinorProcessing"
Example	.Connect .MetaAssociation "Org-Unit-Sending" ."_Operator" "Extract" -."Minor to major behavior" "A" ."Major to minor behavior" "S"

With this command, all the messages sent by an org-unit are extracted, no matter what diagram they are in.

In the standard configuration, the message must be in the diagram to be extracted.

Creating Characteristics

You can also create a MetaAttribute by means of an MGE command file, using the following syntax:

Syntax	.Create .MetaAttribute "Characteristic" ."MetaAttribute Type" "X" ."MetaAttribute Length" "5"
Example 1	.Create .MetaAttribute "Message-Type" ."MetaAttribute Type" "X" ."MetaAttribute Length" "1"
Example 2	.Create .MetaAttribute "Value" ."MetaAttribute Type" "A"
Example 3	.Create .MetaAttribute "ShortName" ."MetaAttribute Type" "X " ."MetaAttribute Length" "8" ."_AtIndex" "U"

Characteristic values may be declared unique for a given object type. To do this, specify the value “U” for _AtIndex in the characteristic definition.

☛ *The characteristic uniqueness is only effective in repositories created after uniqueness has been declared.*

Creating a text type characteristic

Syntax	.Create .MetaAttribute "Characteristic Name" ."MetaAttribute Type" "A"
Example	.Create .MetaAttribute "Ext Objective" .MetaAttribute Type "A"

Creating a tabulated MetaAttribute

Syntax	.Create .MetaAttribute "Characteristic Name" ."MetaAttribute Type" "Format" - ."Update Version" "16643" - .MetaAttribute Length "Length" ."MetaAttribute Format" "T"
Example	Create .MetaAttribute "Ext Role" ."MetaAttribute Type" "X" - "Update Version" "16643" - ."MetaAttribute Length" "20" ."MetaAttribute Format" "T"

Creating a unique index MetaAttribute

Syntax	.Create .MetaAttribute "Characteristic Name" ."MetaAttribute Type" "Format" - .MetaAttribute Length "Length" ._AtIndex "U"
Example	.Create .MetaAttribute "Ext Code" ."MetaAttribute Type" "X" ."MetaAttribute Length" "5" ."_AtIndex" "U"

Connecting a MetaAttribute to a MetaClass

Syntax	.Create MetaClass "Object Type"
Example 1	.Connect .MetaClass "Procedure" ."MetaAttribute" "Procedure-Type"
Example 2	.Connect .MetaClass "Procedure" ."MetaAttribute" "Object"

Example 2 illustrates attachment of a text to an object type.

☛ *The "Comment" text is linked to all object types when they are created. This command is used to connect additional text to a segment.*

Connecting a MetaAttribute to a MetaAssociationEnd

Syntax	.Connect .MetaAssociation "Link" ."MetaAttribute" "Characteristic"
Example	.Connect .MetaAssociation "Operation-Sending" ."MetaAttribute" "Predicate"

Creating an abbreviation

Syntax	.Abbreviate .MetaClass "Long Name" "Short Name"
Example	.Abbreviate .MetaAttribute "Creation Date" "Creation"

ERQL QUERY SYNTAX

This section describes syntax used by ERQL (Entity-Relationship Query Language).

Knowledge of this language is useful if you want to edit repository queries or create your own queries in the **Registered Queries** tab of the query tool.

See:

Query General Syntax: Select

Query notation

Query uses the following notation:

- An expression within square brackets [ ] is optional.
- An expression within brackets ( ) can be repeated.
- Expressions within braces { } are alternative choices.
- The ERQL operators are in bold characters.

Query structure

The syntax of a query is structured as follows:

```
Select MetaClass from @set Into @Result Where Condition
```

You must use the operators "From", "Into" and "Where" in the order indicated in the example above.

Queries are not case-sensitive. Object types and attributes can be entered with or without accents.

☛ *MetaClasses containing blanks must be entered within square brackets ([ ]).*

Query comments

You can insert comments into commands by starting these with "/*" and finishing with "*/".

```
Example: select Message /* Query for messages */
```

Query settings

You can use settings to specify conditions for a query. When you execute this query, a dialog box asks you to enter these settings, with a box for each setting defined in the query.

Setting names are preceded by &. The setting name is used as the title for the field you are asked to complete when specifying the setting value.

In the syntax description, "&setting" indicates that you can define a setting.

```
select Message where type-message = &Type
```

When the setting is processed, you will be prompted for the value of the "Type" field.

The setting name can contain any character if it is between " ".

```
"&Org-Unit Name"
```

Queries with multiple selects

You can use several "Select" clauses in the same query.

"Select" clauses follow the same rules as queries that have only one clause. The only difference is that the name specified after the "From" operator can match the result ("Into") of a previous "Select" clause.

This allows you to use intermediate results without having to save them.

Note that intermediate results are not saved; only the last result is displayed on completion of query execution.

☺ *You can put "Select" clauses on different lines. When entering the query, you can insert a line break by pressing the <Ctrl> and <Enter> keys simultaneously.*

Examples:

- Find the material flows of a project:

```
Select Message into @MaterialFlow where Flow-Type =
"Material Flow"

Select from @MaterialFlow where Diagram.Project = &project
```

- Find the messages of the project diagrams that describe an org-unit:

```
Select Diagram into @Org-UnitDiagram where Described-Org-
Unit and Project = &project

Select Message where Diagram in @ DiagramOrg-Unit
```

QUERY OPERATORS

And: Combines two query criteria.

```
Select [Message] where [Message-Type] = "External Data" and
[Flow-Type] = "Financial Flow"
```

Between: Specifies that a value must be within the range defined by the two given values.

```
Select [Message] Where [Transfer Cost] between 1 And 20
```

Deeply: Performs a recursive query for all objects concerned by a composition link.

```
Select [Org-unit] where Aggregation-Of deeply = &Org-unit
```

Delete: Deletes a previously saved set.

```
Delete @MainOrg-Units
```

From: Restricts the query to a previously defined set.

```
Select [Message] from @TradeFlows where Flow-Type =
“Material Flow”
```

Having count: Restricts the query to objects with the indicated number of linked objects.

```
Select [Org-Unit] where [Message-Sent] having count > 3
```

In: Queries for values in a set or a sub-query.

```
Select Diagram where Org-Unit in @MainOrgUnits
```

Inherited: All links of the query will include objects inherited via this link.

```
Select [IT Service] Inherited Where [Defining-Application]
= 'Myapplication V2.0"
```

Into: Stores the query result in a set for later use.

```
Select [Org-Unit] into @MainOrgUnits where not Aggregation-
Of
```

Keep: Keeps a set for the whole session, or until it is deleted with the Delete operator.

```
Keep @MainOrgUnits
```

Like: Specifies that the name of the queried objects must match the given value, which can include wild cards such as # and !

```
Select [Org-Unit] where [Name] like "!!!!!!!!Managem#"
```

NearMatch: Specifies that the names of the queried objects near match the given value. NearMatch is only available with an indexed repository. As the indexing runs every 10 minutes (by default), you might need to wait for 10 minutes to get the results that would match between two indexing processes.

☛ *To enable repository indexing and customize indexing, see "Managing Repository Indexing", page 39.*

```
Select Application where [Short Name] NearMatch "monage"
```

Returns Applications whose name includes "Management" "Manager" "manage":

```
Select [Software Technology] where [Short Name] NearMatch
"assesori"
```

Returns Software Technologies whose name includes for example "accessories", "accessory".

☛ *NearMatch operator cannot be used with non-indexed MetaClasses (e.g.: Questionnaire MetaClass), non-candidate indexable MetaClasses (e.g.: Action MetaClass), and non-indexable MetaAttributes (e.g.: creation date).*

Not: Negates a query criterion.

```
Select [Message] where [Flow-Type] not= "Financial Flow"
```

Null: The objects queried must not be linked to other objects by the specified link.

```
Select [Org-Unit] WHERE [Message Sent] null
```

Or: One of two query criteria can be true.

```
Select [Message] where [Message-Type] = "Instruction" or
[Flow-Type] = "Financial Flow"
```

Order by: Enables to sort the query results according to a specific MetaAttribute of the MetaClass requested, so as to improve the sorting. This is particularly useful in API to accelerate the sorting.

The MetaAttribute used for the sorting must be indexable.

☺ *Use the Intellisense to get the MetaAttributes indexable only.*

You can sort on several MetaAttributes with an ascending (Asc) or descending (Desc) orientation.

☛ *Ascending is the orientation by default when not specified.*

```
Select [Org-Unit] where [Name] like "b#" order by [Org-Unit
Type][Short Name] Desc [Creation Date] Asc
```

Select: Query command.

```
Select [Message] where [Flow-Type] = "Financial Flow"
```

Unique: Queries for objects linked to only one other object by the specified link.

```
Select [Org-Unit] where [Sent Message Flow] unique
```

Where: Specifies the query criteria.

```
Select [Message] where [Flow-Type] = "Financial Flow"
```

: Wildcard: matches any number of characters.

```
Select [Org-Unit] where [Name] like "Account#"
```

!: Wildcard: matches any single character (use one “!” for each character).

```
Select [Org-Unit] where [Name] like "!!!!!!Management"
```

&: Precedes a variable name.

```
Select Diagram where Org-Unit = &Org-Unit
```

@: Precedes a set name.

```
Select Diagram where Org-Unit in @MainOrgUnits
```

:: Separation character enabling specification of target MetaClass when browsing a generic link.

```
Select Message where [Message Recipient]:[Operation] =
&Operation
```

< > =: Comparison criteria.

```
Select Message Where [Transfer Cost] > 5
```

" " : Characters used to enclose values and names of settings or sets that contain blanks.

[ ] : Characters used to enclose the names of object types, links or characteristics that contain blanks.

```
Select [Report template (MS Word)] where Name = &Name
```

/* */ : Characters used to enclose comments.

```
/* Main Org-Units are not components of another Org-Unit */
```

Help on the Syntax

As you write a query, you can:

- check its syntax
- get help on the *metamodel*

Checking the Syntax

To check the syntax of your query without executing it:

- Click **Analyze** .
  If the query syntax is not correct an error message pops up.

Getting Help on the Metamodel (Intellisense)

The Intellisense tool helps you write a query. Based on the query path already entered, the Intellisense tool provides you the list of metamodel elements you can access at that point:

- MetaClasses
- MetaAssociationEnds
- MetaAttributes

To get help on the metamodel while writing a query:

1. At the inserting point, press the [Ctrl]+[Space] keys.
2. In the list of elements, select the required element.
3. Press [Enter].

   ☛ *(Web Front-End) Click **OK**.*

RESULT: INTO

Use the "Into" operator to name a set of results. This name is used as a title for the dialog box displaying the objects found.

```
Select MetaClass Into @set
```

The Into operator is optional: if it is missing, the result has the same name as the target MetaClass.

```
“select Message where ...” is the same as “select Message
into @Message where ...”
```

The name of the set must be preceded by @.

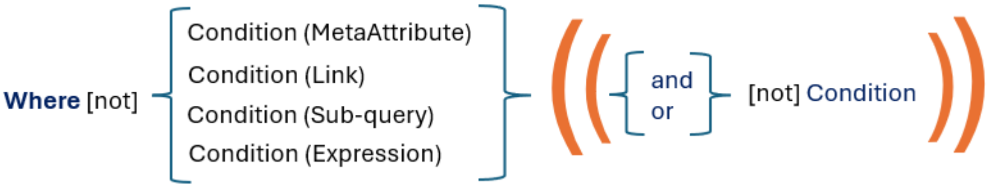
You can save the result using the commands in the **Query** menu of the dialog box displaying the query result. You can apply the "From" operator to this result.

Condition: Where

Conditions that define query criteria can concern the values of object characteristics or links, or the existence or non-existence of a link. Note that conditions can be grouped.

A condition indicates the query criterion or criteria. A condition is optional: if there is no condition, the result is the set or the combination of sets indicated after the From operator. If no set was specified, the result contains all objects of the type indicated.

A condition can consist of basic conditions connected by the ***or***, ***and***, and ***not*** operators. You can use brackets according to the rules indicated for combining sets.



A condition usually consists of:

- a path indicating a link or a characteristic, followed by
- a query operator, followed by
- query parameters

This restricts the query to target objects fulfilling query conditions.

See:

- "Sub-query: In", page 277
- "Browsing the Metamodel", page 278
- "Conditions on Object Characteristics", page 279
- "Conditions on Links", page 280
- "Conditions with Expressions (Code Improvement)", page 283

Sub-query: In

The ***in*** operator allows you to indicate a condition on members of a saved set or to indicate successive conditions.

```
Query Path [not] in @Set,MetaClass where Condition
```

Note that grouped conditions also allow you to indicate successive conditions (see "Grouped conditions", page 282).

Examples:

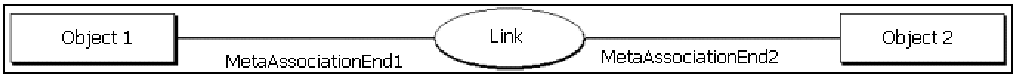
```
select Message where Diagram in @Mod
select Message where Source-Org-Unit in @Org-Unit where
name like “Department#”
```

Browsing the Metamodel

You can concatenate successive MetaAssociationEnds to define a path in the *metamodel*.

Reminder on MetaAssociationEnd names

In the metamodel, objects are connected as follows:



MetaAssociationEnd names follow this rule: seen from an object type, the link is indicated by the name of the opposite MetaAssociationEnd.

```
In this diagram, seen from “Object Type 1”, the link has the
name of “MetaAssociationEnd 2”.
```

When the link is explicit (for example, Source-Message between Org-Unit and Message) both MetaAssociationEnds have names (Source-Org-Unit and Message-Sent). When the link is implicit, the MetaAssociationEnds have the same name as the MetaClass to which they are connected.

When the link is generic, that is when it reaches several different MetaClasses, it is possible to specify a particular MetaClass, separating it with character ’:’.

```
select Message where [Message Recipient]:[Operation] =
&Operation
```

The MetaAssociationEnds of reflexive links have their own specific names.

```
Examples: Component and Aggregation-Of for Compositions,
Previous and Next for Sequences.
```

The object type attached to the last MetaAssociationEnd is called the “source object".

Example:

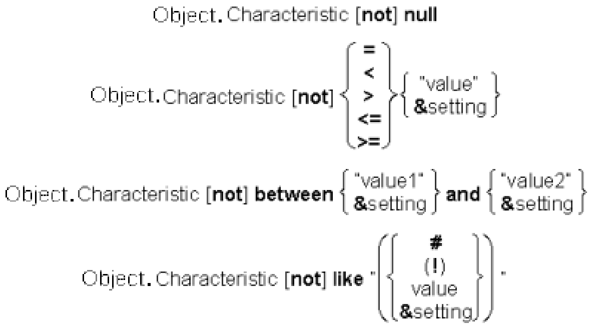
Using the following query, you can obtain the list of Messages Sent by Org-Units in Diagrams of a Project:

```
select Message where Source-Org-Unit.Diagram.Project =
&PROJ
```

The query below gives the same result (see "Conditions on Object Characteristics", page 279):

```
select Message where Source-Org-Unit in Org-Unit where
Diagram in Diagram where Project = &PROJ
```

Conditions on Object Characteristics


Object.Characteristic [not] null
Object.Characteristic [not] = < > <= >= "value" &setting
Object.Characteristic [not] between "value1" &setting and "value2" &setting
Object.Characteristic [not] like " # (!) value &setting "

☛ *Indicates that the characteristic is not entered.*

Comparisons are done on an alphanumeric basis. For example, 2 is greater than 10 or 02.

Wildcards

In the "like" condition, the "# " symbol matches any number of characters. The "!" is the wildcard that matches one character, and can be repeated. For example:

- #x matches all the values beginning with x.
- #x matches all the values ending with x.
- #x# matches all the values containing x.
- !x# matches all the values that have x as the second character.
- x#x matches all the values beginning and ending with x.
- "!!" matches all the values that have two characters.

Examples:

```
select Org-Unit where Name = “Account Manager”
```

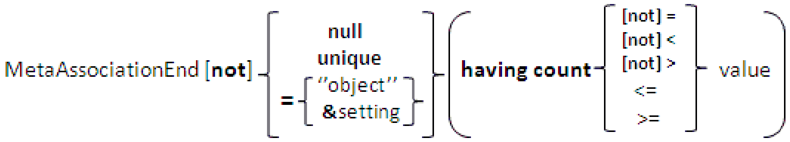
☛ *Here we give an example of a query on a precise name ("Account Manager"). In practice, querying the name of a calculated object can be tedious. It is recommended that a setting be called to write the query. For example: "select Org-Unit where Name = &ManagerName" You then specify the object name when you run the query.*

```
select Org-Unit where Name like "#count#"

select Org-Unit where Name like "!!!!!!! Manager#"

select Org-Unit where Name like "Ac#"

select Diagram where Nature null
```

Conditions on Links

Conditions on links can check link existence and link characteristic values.

Link existence condition: Null, Unique and Having count



- ***Null*** means that the link does not exist. If several MetaAssociationEnds are concatenated, it is the link reached by the last MetaAssociationEnd that is tested.
  For example, "select Project where Diagram.Org-Unit.Message sent null" selects projects where diagrams contain org-units that do not send messages; diagrams with no org-units are ignored.
- ***Unique*** means that the link must exist only once. For concatenated MetaAssociationEnds , the link reached by the last leg is the one counted. (Note: not unique means that at least two links exist.)
- ***Having count*** indicates the number of times the link must exist. For concatenated MetaAssociationEnds , the link reached by the first MetaAssociationEnd is the one counted.

Examples:

```
select Org-Unit where Diagram null
```

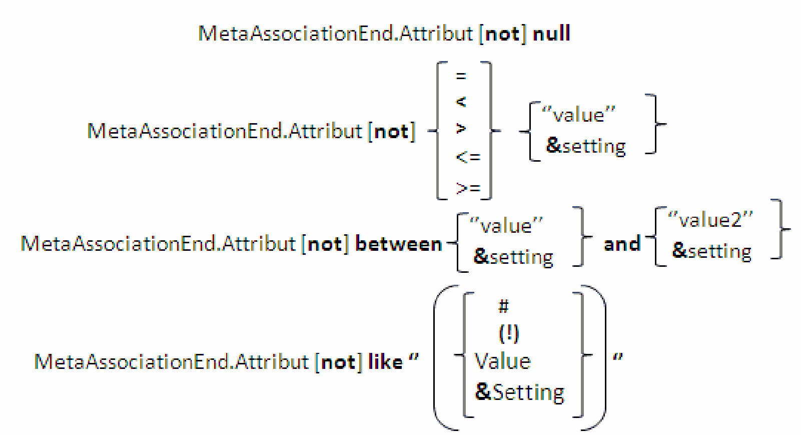
```
select Org-Unit into @MainOrg-Units where Component not
null
```

```
select Diagram where Org-Unit.Message-Sent not null having
count > 2
```

In this last example, diagrams containing at least two org-units that send messages are found.

Link characteristic condition

A condition on a link characteristic is expressed in the same way as a condition on an object characteristic. The name of the link characteristic is concatenated with the name of the link. This name and the characteristic name are separated by a period (.).


MetaAssociationEnd.Attribut [**not**] **null**
MetaAssociationEnd.Attribut [**not**]
=
<
>
<=
>=
"value"
&setting
MetaAssociationEnd.Attribut [**not**] **between**
"value"
&setting
and
"value2"
&setting
MetaAssociationEnd.Attribut [**not**] **like** "
#
(!)
Value
&Setting
"

Example: Query org-units that always send at least one message:

```
select Org-Unit where Message-Sent.Predicate = "always"
```

Source object characteristic condition

You can indicate a condition for a characteristic of the source object in the same way as for link characteristics:

```
select Org-Unit where Message-Sent.Predicate = "always" or
Message-Sent.Flow-Type = "Information Flow"
```

If a characteristic is not specified, the query is based on the name ("Diagram.Name ="zzz"" is equivalent to "Diagram ="zzz""):

```
select Diagram where Org-Unit = &Org
```

Browsing reflexive links Deeply

When the query target MetaClass is identical to the source MetaClass, and the link browsed is a reflexive link (eg. Composition), you can query all levels of the composition using the deeply operator.

```
MetaAssociationEnd deeply Condition
```

Examples:

```
select Org-Unit into @components where Aggregation-Of
deeply = &Org-Unit
```

```
select [Message] where [Org-Unit-Recipient].[Message-Sent]
deeply in [Message]
   where [Org-UnitSender = &"Org-Unit"
```

The first example finds all component org-units of a given org-unit, whether directly or via intermediate org-units.

The second example queries for all Messages produced by target objects when they receive the messages sent by a given org-unit (source).

Restriction concerning Deeply

You cannot combine "Deeply" and "Inherited".

If a link is browsed with the "Deeply" keyword, the Inherited clause enabling inclusion of inherited objects is ignored for this link.

Grouped conditions

When a condition is applied to several characteristics of the source object, it is expressed as a grouped condition. Each of the "and", "or" and "not" operators involves any source object that satisfies the indicated condition. If you want the source object to be the same for different criteria, you must group these criteria with brackets.

As an example, the query:

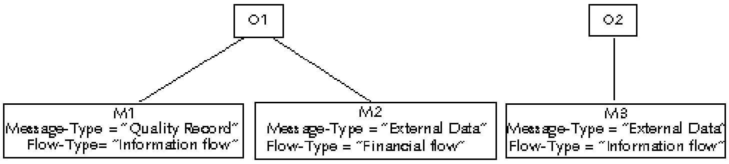
```
select Org-Unit where Message-Sent.Message-Type = “External
Data” and Message-Sent.Flow-Type = “Information flow”
```

finds the org-units that send at least one message of the “External Data” type and at least one message of the “Information flow” type. Nothing in this query indicates that the org-units found must be connected to at least one message of the “Information flow” type which is also of the “External Data” type. To do this, you must rewrite the query as follows:

```
select Org-Unit where Message-Sent.(Message-Type =
“External Data” and Flow-Type = “Information flow”)
```

You must place the period before the brackets. You can also group the conditions in a sub-query:

```
select Org-Unit where Message-Sent in Message where
(Message-Type = “External Data” and Flow-Type =
“Information flow”)
```


O1
O2
M1
Message-Type = "Quality Record"
Flow-Type= "Information flow"
M2
Message-Type = "External Data"
Flow-Type = "Financial flow"
M3
Message-Type = "External Data"
Flow-Type = "Information flow"

The first query gives O1 and O2, while the second only finds O2.

To combine conditions on object characteristics and link characteristics, you must also use grouped conditions.

For example, to query messages always sent by a given org-unit, that is a query on the name of an org-unit and the value of the predicate characteristic between message and org-unit, you have to write the query as follows:

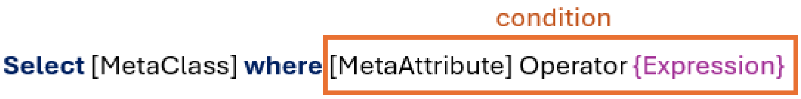
```
select Message where Source-Org-Unit.(predicate = “always”
and name = &Org-Unit)
```

Conditions with Expressions (Code Improvement)

For better performance and improve your coding way, you can add expressions in conditions.

☺ *When possible, it is recommended to use expressions in conditions instead of coding macros.*

An expression in a condition can calculate an attribute value within an expression of different intrinsic attributes, with the following format:



Select [MetaClass] **where** [MetaAttribute] Operator {Expression}

Where:

- MetaAttribute is either:
  - attribute of the MetaClass or MetaAssociation
  - intrinsic or calculated with a trigger
- Operator is either:
  - =, not =
  - <, >
  - <=, >=
  - in, Not in
- Expression can be a mixed of the following:
  - +, -, *, /
  - attribute of the MetaClass or MetaAssociation (intrinsic or calculated with a trigger)
  - SQL Server Function (e.g.: dateadd, log)
  - parameter
  - object variable

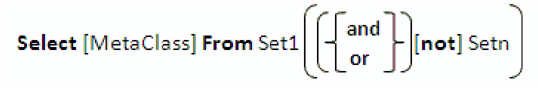
💣 **The expression cannot include calculated attributes nor Tagged Values.**

Examples

Select [Application] **Where** [Cost] > [Expenses]+10

Select [Application] **Where** Not [Property contained].[Link creation date] between dateadd(year, -5, getdate()) And dateadd(year, -1, getdate()) And [Property contained].[Creation Date] > dateadd(year, -5, getdate()) And [Property contained].[Link modification date] = "04/04/2022"

SETS: FROM



You must use the @ symbol as a prefix for set names used in the query.

☺ *The MetaClass preceding the From operator is optional: a set having been defined for a MetaClass, this MetaClass is used by default.*

You can save the result returned by a query as a set. You can use this result in other queries for the duration of your query session. Sets will restrict your query, which can be useful in optimizing response times, executing lengthy queries only once.

Only saved sets can be used by other queries (except sets used in queries containing several Select clauses; see their description below).

Example:

The following query searches for messages of a diagram that satisfy a condition ("Where" operator) in the set of messages “Mod_Messages”.

```
select Message from @Mod_Messages where Condition
```

Sets are kept and named using the **Keep** command of which syntax is keep @set.

You can delete sets with the **Delete** command. The sets are also deleted when you close your work session.

Example:

```
select Org-Unit into @MainOrg-Units where not Aggregation-Of

keep @MainOrg-Units
```

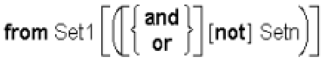
In this query, you build and keep the set of main org-units, that is org-units not aggregated in any other org-unit.

In the next query, you will reuse this set and delete it when you no longer need it.

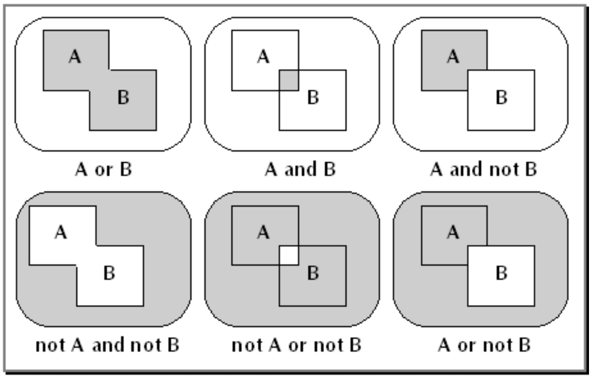
```
select Org-Unit from @MainOrg-Units where Diagram.Project =
&Project

delete @MainOrg-Units
```

This provides you with the list of the main org-units for the project specified by the project setting.

Set Operations



The set used after the "From" operator can be built from a combination of saved sets.



You can use brackets. The rules of distribution and precedence are as follows:

- (A or B) or C = A or (B or C) = A or B or C
- (A and B) and C = A and (B and C) = A and B and C
- (A or (B and C) = A or B and C = (A or B) and (A or C)
- (A and (B or C) = A and B or A and C = (A and B) or (A and C)
- not (A or B) = (not A) and (not B) = not A and not B
- not (A and B) = (not A) or (not B) = not A or not B

Example:

```
select Org-Unit from @MainOrg-Units and not @QualityOrg-
Units
```

QUERY PAGINATION

ERQL syntax enables to retrieve a subset of objects, with a constant display time in list views, whatever the data volume. To do so, use the **offset** and **fetch** keys.

Examples:

```
Select Application order by [creation date] desc offset 1
fetch 1
```

retrieves the last created application

```
Select Application order by [creation date] desc offset 11
fetch 10
```

retrieves the 10 applications of the second page (with 10 objects by page)

You can also use this syntax to search for an object according to a max or min criteria value of a specific attribute.

Query Tips and Examples

See:

- "Tips on Using Queries", page 287
- "Examples of Queries for HOPEX Business Process Analysis", page 291

Tips on Using Queries

Different queries with the same result

You can obtain the same result using different queries.

Query diagrams of a project that describe an org-unit

- And between two conditions

```
Select Diagram where Described-Org-Unit and Project =
&Project
```

- Query in an intermediate set and restrict to this set:

```
Select Diagram into @Org-UnitDiagram where Described-Org-
Unit
```

```
Select Diagram from @Org-UnitDiagram where Project =
&Project
```

- Query in two sets and find their intersection:

```
Select Diagram into @Org-UnitDiagram where Described-Org-
Unit
```

```
Select Diagram into @ProjectDiagram where Project =
&Project
```

```
Select Diagram from @Org-UnitDiagram and @ProjectDiagram
```

Query messages of a project

- Extended form using a sub-queries

```
Select Message where Diagram in Diagram where Project =
&Project
```

- Browsing the links

```
Select Message Where Diagram.Project = &Project
```

Query org-units without a Message-Sent

```
Select Org-Unit where [Message-Sent] null
```

```
Select Org-Unit where not [Message-Sent]
```

Tips with "and", "not" and "or" operators

It is important to ensure that the query corresponds to the desired search, particularly when using the "and", "not" and "or" operators.

The following query:

```
Select Org-Unit where not [Message-Received] = "Order"
```

gives all the org-units except those that receive the “Order” message. The org-units that do not receive a message are also included in the result.

However, the following query:

```
Select Org-Unit where [Message-Received] not = "Order"
```

selects only the org-units that receive a message except those that receive the “Order” message.

To obtain the same result as above, write:

```
Select Org-Unit where [Message-Received] not = "Order" or
[Message-received] null
```

Other query examples

Query messages always sent by an org-unit

```
Select Message into @obl where Source-Org-Unit = &Org-Unit
and Source-Org-Unit.Predicate = ”always”
```

finds the messages sent by the org-unit entered as the setting value, and connected to any org-unit with the source predicate = ”always”.

```
Select Message into @obl where Source-Org-Unit.(name =
&Org-Unit and Predicate = ”always”)
```

finds the messages sent by the org-unit entered as the setting value, with the source predicate = ”always”.

Check validity of link used

Similarly, when counting links, you should check the link:

```
Select Diagram where Org-Unit.[Message-Sent] not null
having count >= 3
```

finds diagrams having at least three messages sent by the org-units in these diagrams.

Use the following query to find diagrams having org-units sending at least three messages:

```
Select Diagram where Org-Unit in Org-Unit where

Message-Sent not null having count >= 3
```

Examples of queries on reflexive links

- Use the following query to find org-units that are part of the "Sales Management" department (and are therefore aggregations of this department):

  ```
  Select Org-Unit where Aggregation-Of = "Sales Management"
  ```

- Query to find org-units that are directly or indirectly part of the “Sales Management” department (which are aggregations of the “Sales Management” department or one of its components):

  ```
  Select Org-Unit where Aggregation-Of deeply = "Sales
  Management"
  ```

- Query to find org-units that are part of an org-unit other than the “Sales Management” department:

  ```
  Select Org-Unit where Aggregation-Of deeply not = "Sales
  Management"
  ```

- Query to find org-units that are not part of the "Sales Management" department:

  ```
  Select Org-Unit where not Aggregation-Of deeply = "Sales
  Management"
  ```

- Query to find org-units that are not components of something or are part of the "Sales Management" department:

  ```
  Select Org-Unit where not Aggregation-Of deeply not =
  "Sales Management"
  ```

Examples with diagram, org-unit, message and keyword

- Use the following query to find diagrams connected to at least one org-unit that receives at least one message with a keyword:

  ```
  Select Diagram Where Org-Unit. [Message received]. Keyword
  ```

- Use the following query to find diagrams connected to an org-unit that receives no messages without a keyword:

  ```
  Select Diagram where Org-Unit.(not Message-Received.(not
  Keyword))
  ```

  OR

  ```
  or Select Diagram where Org-Unit.(not Message-
  Received.Keyword null)
  ```

- Use the following query to find diagrams connected to an org-unit that receives messages that all have a keyword:

  ```
  Select Diagram where Org-Unit.(Message-Received and not
  Message-Received.(not Keyword))
  ```

  OR

  ```
  Select Diagram where Org-Unit.(Message-Received and not
  Message-Received. Keyword null)
  ```

- Use the following query to find diagrams that have only org-units that receive messages without a keyword:

  ```
  Select Diagram where not Org-Unit.(not Message-
  Received.(not Keyword))
  ```

- Use the following query to find diagrams whose org-units all receive messages that all have a keyword:

  ```
  Select Diagram where Org-Unit and not Org-Unit.(not
  Message-Received or Message-Received.(not Keyword))
  ```

Examples of Queries for HOPEX Business Process Analysis

The following query examples are specific to **HOPEX Business Process Analysis**.

- Use the following selector to find all the org-units of an organizational process and its component organizational processes:

  ```
  Select [org-unit] into @root where [Assigned
  Participant].[Containing Diagram].[Described
  Element]:[Process] &process
  ```

  ```
  Select [org-unit] into @sub where [Assigned
  Participant].[Containing Diagram].[Described
  Element]:[Process].[Owner Process]  &process
  ```

  ```
  Select [org-unit] from @root or @sub
  ```

- Use the following selector to find all the org-units described in diagrams of a process:

  ```
  Select [Org-Unit] WHERE [assigned participant].[Containing
  Diagram].[Described Element]:[Process] &Process
  ```

- Use the following selector to find diagrams describing a process:

  ```
  Select [Diagram] WHERE [Described Element]:[Process]
  &process
  ```

- Use the following selector to find the processes impacted by a project:

  ```
  Select [Project] WHERE [Project Deliverable].[Delivered
  Resource]: [Process]
  ```

- Use the following query to find the operation following an operation

  ```
  Select [Sequence flow] into @SF where
  [Predecessor]:[Operation] &ope
  ```

  ```
  Select [Operation] where [Previous Sequence flow] in @SF
  ```

10

CREATING CONSISTENCY CHECKS

Repository build is subject to modeling rules. Depending on the product or products you have, **HOPEX** provides a certain number of rules you can apply to objects you create so as to check their consistency. You can also create new rules.

Each user can run a check on an object. As for rule or regulation modification functions, these require the **HOPEX Power Supervisor** or **Studio** technical modules.

REMINDER: RULES OPERATION PRINCIPLE

Rules apply to **HOPEX** repository objects. They define checks on these objects and are implemented by:

- tests
- macros.

Rules provided by default are accessible in the **MetaStudio** navigation window, available only with the **HOPEX Power Supervisor** and **Studio** technical modules.

You can view rules as a function of the regulations or objects to which they apply. ***Each regulation enables classification of rules according to a specific context or domain***.

You can apply:

- a regulation or a set of regulations by default.
- a regulation to repository objects as and when required.

HOPEX presents a list of regulations that varies depending on products you have available. In **HOPEX Business Process Analysis**, you have for example an organizational type regulation that you can apply by default to check correct sequencing of objects you create.

Displaying check results

Check results are displayed:

- in the properties windows of objects concerned by the rules currently applied.
- in the diagram drawings that you want to check.

☛ *For more information, see the* ***HOPEX Common Features*** *guide.*

REGULATIONS

A regulation enables classification of rules according to a specific context or domain. It can contain not only a set of rules but also sub-regulations.

Displaying regulations

To view regulations provided in **HOPEX**:

1. Select the **MetaStudio** navigation window.

   ☛ *This window is available only with the **HOPEX Power Supervisor** technical module.*

2. Successively expand the "Repository Consistency" and "Modeling Regulation" folders.

The list of available regulations appears.

Activating a regulation

To activate a regulation:

1. In **HOPEX** menu bar, select **Tools** > **Options**.
2. In the dialog box that appears, select **Workspace > Modeling and Methods Regulations**.
3. In the **Active modeling regulation** field, click ... .
4. In the dialog box that opens, select the regulation to be applied and click **OK**.

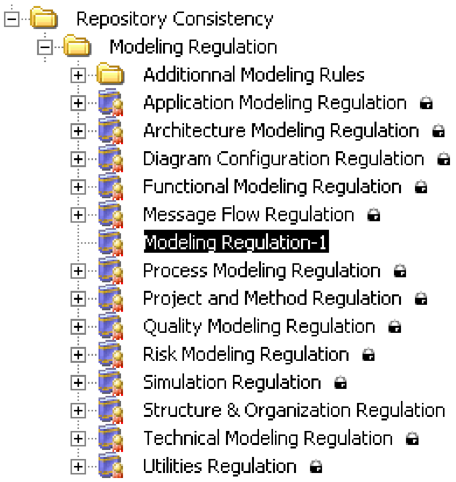
If you have applied several regulations, you must assemble these under a single complex regulation.

Creating a complex regulation

To create a complex regulation:

1. In the **MetaStudio** navigation window, expand the "Repository Consistency" folder.
2. Right-click the **Modeling Regulation** folder and select **New** > **Modeling Regulation**.
3. Name the new regulation and click **OK**.
   This then appears in the navigator.

4. Drag-and-drop the desired regulations onto this regulation.


Repository Consistency
Modeling Regulation
Additionnal Modeling Rules
Application Modeling Regulation
Architecture Modeling Regulation
Diagram Configuration Regulation
Functional Modeling Regulation
Message Flow Regulation
Modeling Regulation-1
Process Modeling Regulation
Project and Method Regulation
Quality Modeling Regulation
Risk Modeling Regulation
Simulation Regulation
Structure & Organization Regulation
Technical Modeling Regulation
Utilities Regulation

5. To take this new regulation into account, select it in the options dialog box as indicated above.

Defining rules of a regulation

Rules provided by default are accessible in the **MetaStudio** navigation window. You can view them as a function of the regulations or objects to which they apply. You can also display all existing rules. They are preceded by the following icon: .

When you run a check on an object, you do this via a regulation. So that a rule can be taken into account in the check, it must be connected to the applied regulation.

To connect a rule to a regulation:

1. Expand the **Repository Consistency** folder.
2. Find the desired rule in the set of rules or from the objects to which it applies.
3. Select the rule and drag it onto the regulation to which you wish to connect it.

Regulation application scope

Regulation application scope corresponds to the MetaClasses concerned by the regulation. Defining MetaClasses for each regulation simplifies selection of regulations to be applied on an object. In fact, when you run a check on an object, only the list of regulations that concern the object will appear.

You can extend application scope of a regulation.

> You can for example connect the "Organizational Process" MetaClass to a regulation concerning operations so that this regulation can apply to operations contained in an organizational process.

To add a MetaClass to regulation application scope :

1. In the **MetaStudio** navigation window, select the regulation.

2. Open its properties.
3. Select the **Scope of Application** tab.
   The first frame lists the MetaClasses concerned.
4. In the second frame, click the **Connect** button 🔗 .
   The Query dialog box appears.
5. Select the MetaClass to be connected, for example "Organizational Process" and click **OK**.

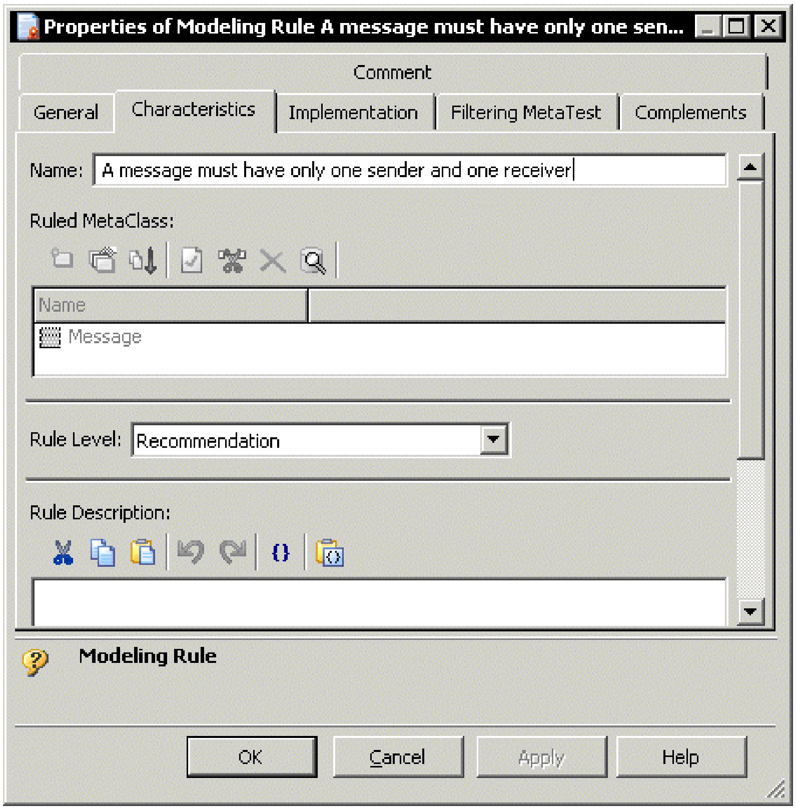
RULE PROPERTIES

To display rule properties:

1. In **HOPEX**, select the **MetaStudio** navigation window.

   ☛ *This window appears only if you have the **HOPEX Power Supervisor** technical module.*

2. In the "Repository Consistency" folder, right-click the rule and select **Properties**.
3. Select the **Characteristics** tab.
   The dialog box that appears indicates:
   - the name of the rule
   - the MetaClass(es) checked by this rule.



Properties of Modeling Rule A message must have only one sen...

Name: A message must have only one sender and one receiver

Ruled MetaClass:

Name	
Message	

Rule Level: Recommendation

Rule Description:

Modeling Rule

Rule level

This box defines importance of the rule. The possible options are:

- **Suggestion**: rule it is advisable to observe, but not a requirement. Non-respect of the rule is indicated in the rules application report but does not produce a warning.
- **Recommendation** important rule. Non-respect of the rule produces a warning.
- **Requirement**: obligatory rule. Non-respect of this rule produces an error and is a source of blocking.

The properties dialog box also displays the test or the macro used for implementing the rule. See "Rule Application Scope".

RULE APPLICATION SCOPE

A rule is applied to a MetaClass. By default, it can be applied to all objects of this MetaClass. However, it is sometimes necessary to define the scope of application of a rule more precisely.

Restricting application scope

Consider the example of the following rule:

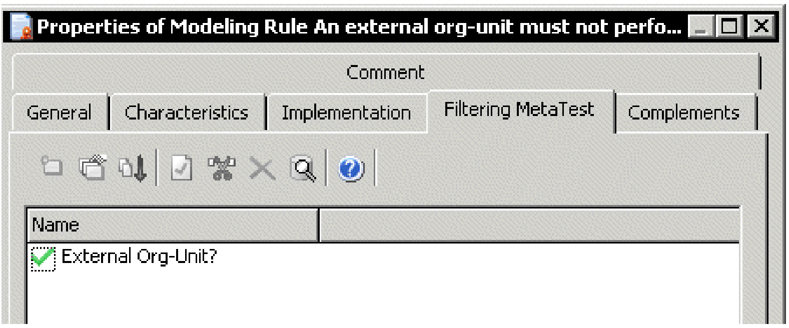
"An external org-unit cannot execute an operation".

This rule is only meaningful for external org-units. It is therefore appropriate to restrict application of this rule on the org-unit MetaClass, so that it executes for external org-units only.

To restrict scope of application of the rule:

- Create a test filtering the scope of application of the rule.

  For example, in the rule you will define a filter so that the rule applies not to all org-units but only to external org-units.



Properties of Modeling Rule An external org-unit must not perfo...

Name
External Org-Unit?

If the filtering test is true, in other words if the org-unit is external, the implementing test executes.

Several filtering tests can be used. In this case, the rule applies if all filtering tests are true.

☛ *See "Defining an Implementation Test".*

Extending application scope

Certain highly specific rules can apply to several MetaClasses.

Consider the example of rule:

"Important objects must have a comment".

MetaClasses considered as "important" are business process, org-unit, organizational process, application, etc. For each of these MetaClasses, the condition applied is: "The comment should not be empty".

Creating a rule for several MetaClasses is of interest only if the condition applied is valid whatever the MetaClass. In our example, the condition refers to the comment which is a property of all MetaClasses, and consequently of all those concerned by the rule.

Conditioning application of a rule

In the modeling regulation around the business process, two rules have been defined for the "Operation" MetaClass:

- An operation must be performed by an org-unit
- An operation must be performed by only one org-unit

For pedagogical reasons, these two constraints have not been grouped in a single constraint (an operation must be performed by an org-unit and by only one org-unit).

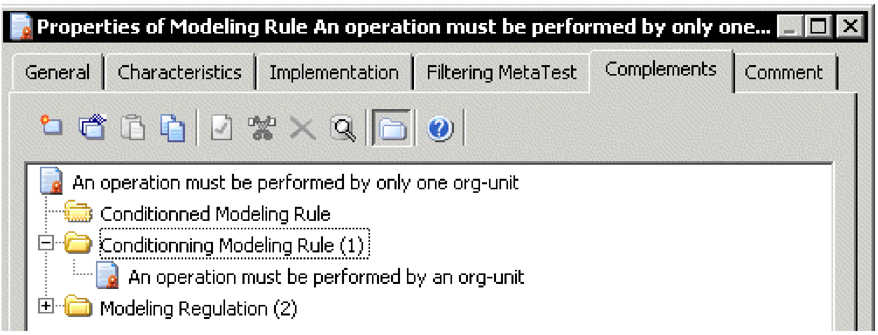
However, it is obvious that if the first rule is not verified, the second is of no interest.

This is why application of the second rule depends on verification of the first. In other words, the second rule is only applied only if the first is verified.

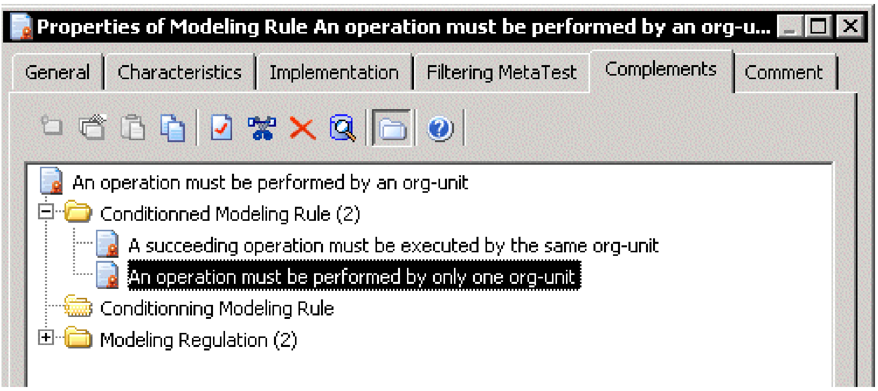
- Open the dialog boxes of these two rules in turn and select the **Complements** tab.

You will see that:

- The rule "An operation must be performed by only one org-unit" is conditioned by the rule "An operation must be performed by an org-unit".


Properties of Modeling Rule An operation must be performed by only one...
General
Characteristics
Implementation
Filtering MetaTest
Complements
Comment
An operation must be performed by only one org-unit
Conditionned Modeling Rule
Conditionning Modeling Rule (1)
An operation must be performed by an org-unit
Modeling Regulation (2)

- The rule "An operation must be performed by an org-unit" conditions the rule "An operation must be performed by only one org-unit".


Properties of Modeling Rule An operation must be performed by an org-u...
General
Characteristics
Implementation
Filtering MetaTest
Complements
Comment
An operation must be performed by an org-unit
Conditionned Modeling Rule (2)
A succeeding operation must be executed by the same org-unit
An operation must be performed by only one org-unit
Conditionning Modeling Rule
Modeling Regulation (2)

RULE IMPLEMENTATION

A rule can be implemented:

- by a test
- by a macro

> ☛ *The use of implementation tests is recommended, test by macros generally being used in complex cases only.*

Implementing a Rule by a Test or Tests

A test expresses a condition in the form of an expression.

Implementing a modeling rule using a test consists of connecting this test to the rule. For a given object, the modeling rule is verified if application of the test that implements it returns the value "true".

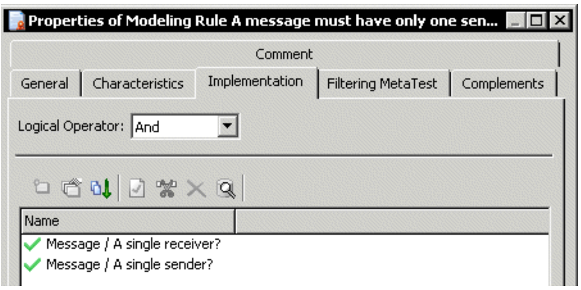
You can connect several tests to a rule. In this case, the **Logical Operator** offers two possible options:

- **And**: the rule is verified only if all tests are positive.
- **Or**: a single test is sufficient to verify the rule.

For example, the rule "A message must have a sender and a receiver" is implemented by two tests :

- One test to verify origin of the message
- One test to verify destination of the message

The logical operator of the test is "And", which indicates that both tests must be positive for the rule to be verified.



Logical Operator: And

✓ Message / A single receiver?
✓ Message / A single sender?

A test can also be used in several rules.

> ☛ *To create an implementation test, see "Defining an Implementation Test".*

Implementing a Rule by a Macro

The macro is written in VBScript and uses **HOPEX** APIs.

To define a macro implementing a rule:

1. Right-click the rule.
2. Select **New** > **Implementing Macro**.

A macro is created and functions to be implemented are automatically declared.

The following is the detail of these functions:

```
Sub RuleAppliableIs (oToBeRuled as MegaObject, oRule as
MegaObject, sParameter as String, bRuleAppliableIs as
Boolean)
```

```
Sub RuleApply (oToBeRuled as MegaObject, oRule as
MegaObject, sParameter as String, bRuleResult as Boolean)
```

☛ *"oRule" is the modeling rule implemented.*

- The RuleAppliableIs function returns "true" in the "bRuleAppliableIs" variable if "oToBeRuled" is a root project. If not, it returns "false".
- The RuleApply function returns "true" in the "bRuleResult" variable if "oToBeRuled" is connected to a project type.
- The "RuleAppliableIs" function is optional. If it is not defined, the rule is implicitly applicable to all instances of the MetaClass (which is the case for the majority of rules).

Another function can be implemented if we wish to control the content of text inserted in the detailed report when the rule is not verified:

```
Sub RuleWithReportApply (oToBeRuled, oRule, sParameters,
bRuleResult, sErrorReport)
```

This function is called on building the detailed report. The last parameter enables sending of a precise error report adapted to the processing carried out.

Implementing Test or Macro?

Consider the example of rule:

```
"A root project has no project type".
```

This rule is applied to the "Project" MetaClass, and more particularly to root projects.

To implement this rule, you must:

- firstly reduce the application scope of the rule.
- carry out the test "This project is not connected to a project type".

Using tests

If we use expression tests to define this rule, we must create two expression tests.

- One to test if the project is a root project, enabling filtering of the rule application scope.
- Another to test if the project is connected to a project type serving to implement the rule.

Using macros

If we use a macro, we have to implement two functions:

- RuleAppliableIs => this function enables restriction of rule application scope.
- RuleApply => this function enables definition of rule implementation.

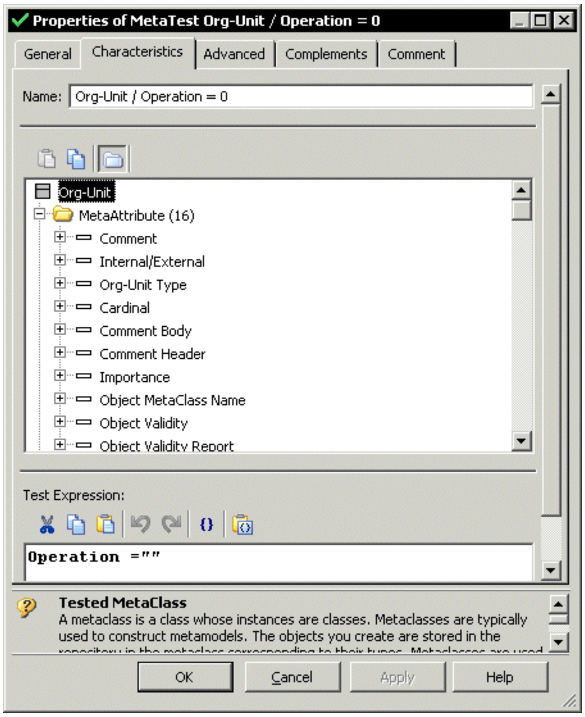
💣 **If a rule is implemented by a macro, tests and filters are ignored. The macro code must manage implementation tests and filters on the application area.**

DEFINING AN IMPLEMENTATION TEST

A test can be implemented by an expression or a macro. The use of expressions is recommended, test by macro generally being used in complex cases only.

Defining an Implementation Test By an Expression

In the test properties dialog box, the **Characteristics** tab presents all MetaAssociationEnds and MetaAttributes of the tested MetaClass.



Name: Org-Unit / Operation = 0

Test Expression:

```
Operation =""
```

Tested MetaClass
A metaclass is a class whose instances are classes. Metaclasses are typically used to construct metamodels. The objects you create are stored in the

By drag-and-drop, you can place in the test expression the repository information to be tested.

> ☛ *Test expression syntax is the same as that used to define conditions in properties dialog box configuration. For more details, see the technical article on properties dialog boxes.*

You can:

- describe a test by expressions and by the functions described below.
- taking account of inheritance in tests (see "Taking account of inheritance in tests")

Expression and logical operator

The expression of a test comprises expressions. These expressions can be logically combined using brackets " () " as well as logical operators.

Logical operators available are:

```
And and Xor
```

Xor is the exclusive "Or", that is "true when one and only one of the combined expressions is true".

It is also possible to use **not** to obtain the negative of an expression

Example:

(Expression1 **and** Expression2) or **not** Expression3

Comparison attribute and operator

The expression of a test can be defined using various functions and comparison operators.

Comparison operators are:

```
= <> > < >= <=
```

These comparison operators can be used to define an expression from the value of one of the MetaAttributes of the tested MetaClass.

```
Attribute operator Value
```

Example

For an org-unit, the expression:

```
Internal/External = "X"
```

Returns "true" if the **Internal/External** attribute is "X" and false if not.

Reminder: here we use a tabulated value attribute and we therefore test its internal value).

For an operation, the expression:

```
Duration >= 40
```

Returns "true" if the **Duration** attribute is greater than or equal to 40.

> ☛ *Operators >, <, <=, >= are of course only meaningful for attributes of a type enabling such comparison.*

Warning: By proceeding in this way, we can test only attributes of the tested MetaClass. To be able to handle the attribute of a link from the MetaClass, the TrueForEach() and TrueForOne() functions subsequently presented should be used.

Function ItemCount() and comparison operator

Comparison operators can also be used to define an expression from the ItemCount function.

This function returns the number of objects found at the end of the MetaAssociationEnd or returned by the selector:

```
ItemCount(LegSel)
```

Where **LegSel** is a field representing a MetaAssociationEnd or a query.

Example

For an org-unit, the expression:

```
ItemCount(Operation)>2
```

Returns "true" if the org-unit is connected to strictly more than 2 operations.

It is possible to use a shortcut to test if at least one object is connected by a MetaAssociationEnd.

Therefore, for an org-unit, the expression:

```
Operation=""
```

Returns "true" if no operation is connected.

```
Operation<>""
```

Returns "true" if at least one operation is connected.

> 🕮 *Use of a query in the ItemCount() function of course requires that the query starts from the currently tested object, or that this does not require an input object.*

Functions TrueForEach() and TrueForOne()

These functions enable definition of an expression applying to an object or objects at the end of a MetaAssociationEnd or of a query.

TrueForEach returns "true" if the expression is true for all objects found at the end of the MetaAssociationEnd or returned by the query.

If no object is found or returned, this function returns "true".

TrueForEach returns "true" if the expression is true for at least one of the objects found at the end of the MetaAssociationEnd or returned by the query.

If no object is found or returned, this function returns "true".

```
TrueForEach(LegSel, Expression)
```

```
TrueForOne(LegSel, Expression)
```

Where **LegSel** is a field representing a MetaAssociationEnd or a query.

Example

For an org-unit, the expression:

```
TrueForEach (Operation , Duration >= 40)
```

Returns "true" if the org-unit is connected only to operations of duration strictly greater than 40, or if the org-unit is not connected to any operation.

For an organizational process, the expression:

```
TrueForOne (Org-Unit, RACI = "R" or RACI = "E")
```

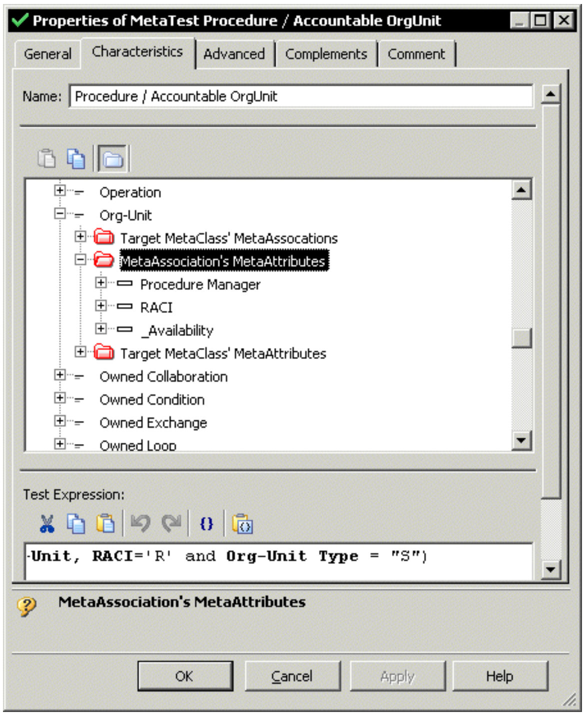
Returns "true" if the organizational process is connected to at least one org-unit with the RACI link attribute internal value "R" or "E".

Note here that when we use the TrueForEach and TrueForOne functions with a MetaAssociationEnd, it is possible to use in the expression :

- The link attributes
  and/or
- The attributes of the object at the end of the link

In the expression used by a TrueForEach or TrueForOne function, it is of course possible to use one of these two functions again so as to take a new MetaAssociationEnd or to trigger a new query.

In the implementation test properties dialog box, the tree presenting the MetaAssociationEnds and MetaAttributes enables navigation in depth of the metamodel. It is therefore possible to drag-and-drop one of the attributes of an object at the end of a MetaAssociationEnd.



Properties of MetaTest Procedure / Accountable OrgUnit

Name: Procedure / Accountable OrgUnit

Test Expression:

```
-Unit, RACI='R' and Org-Unit Type = "S")
```

MetaAssociation's MetaAttributes

Other available functions

```
ItemExist(LegSel, Object)
```

Where **LegSel** is a field representing a MetaAssociationEnd or a query.

This function returns "true" if the specified object is connected via the identified MetaAssociationEnd or is present in the set of objects returned by the query.

```
Available(Object)
```

This function returns "true" if the specified object exists.

Taking account of inheritance in tests

To take account of inheritance in tests on a MetaAssociation, you can suffix the field by:

- @INHERITING: displays objects that are inheriting
- @INHERITED: displays inherited objects
- @INHERITANCE: @INHERITING + @INHERITED

Example

From a MetaTest that relates to the Application MetaClass, you can define the following test:

```
ItemCount(~msUikEB5iGM3[Component]@INHERITED) > 2
```

Tests if the number of component applications of the tested application is more than 2 (including inherited component applications)

☛ *You can also use inheritance with ItemExist.*

Defining an Implementation Test By a Macro

The macro is written in VBScript and uses **HOPEX** APIs.

To define a macro in a test:

1. Open the properties dialog box of the test.
2. Select the **Advanced** tab.
3. Connect the macro to the test.

This macro should show the function:

```
Sub TestApply(oToBeTested as MegaObject, oMetaTest as
MegaObject, sParameter as string, bTestResult as boolean)
```

or:

- "oToBeTested" is the **HOPEX** object.
- oMetaTest is the test implemented.
- "sParameter" is the value of the chain attribute of the link between the test and the macro.
- "bRuleResult" is the result of application of the test.

☛ *"sParameter" is a parameter supplied to the macro. It is the value of the "Macro Parameter" attribute, which can be specified on the link between test and macro. This parameter can enable definition of generic macros, used to implement several tests.*

How to make a concept variable

1 Summary

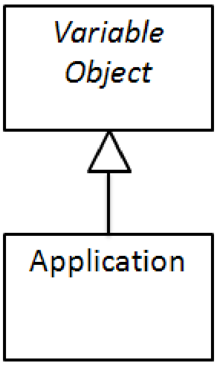
This document is a technical article for people wanting to extend the MEGA repository capability. Technical knowledge about the MEGA metamodel is a prerequisite.

The document details how to make a MEGA concept variable. A variable concept allows the instances to be varied. That means new instances can be created from the previous one inheriting some of the existing connections. Furthermore, inherited objects can be removed and or replaced.

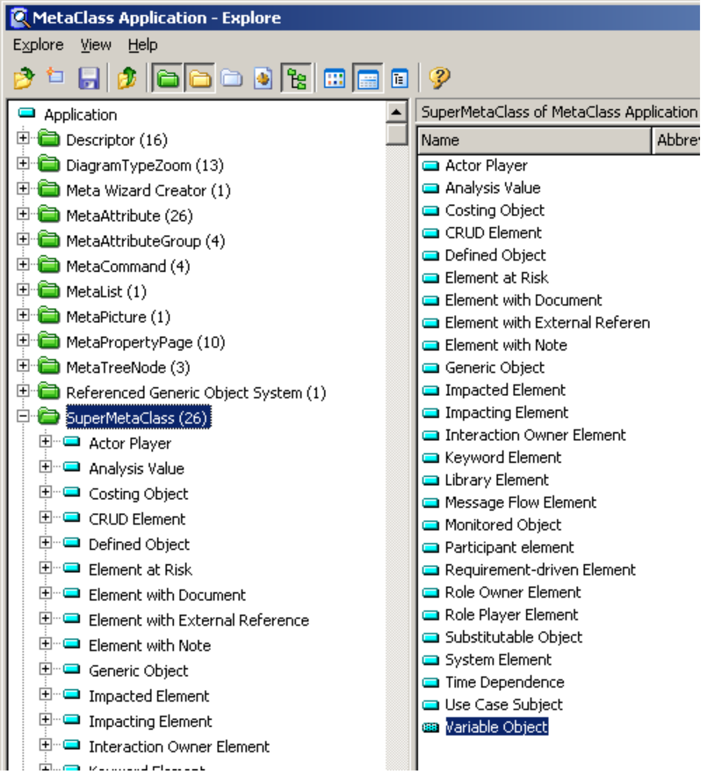
2 Variability

The first step to make an object variable is to inherit from the *Variable Object* Metaclass. The *Variable Object* metaclass is an abstract metaclass; it is not possible to create an instance from it. But, any concrete metaclasses inheriting from it becomes variable.

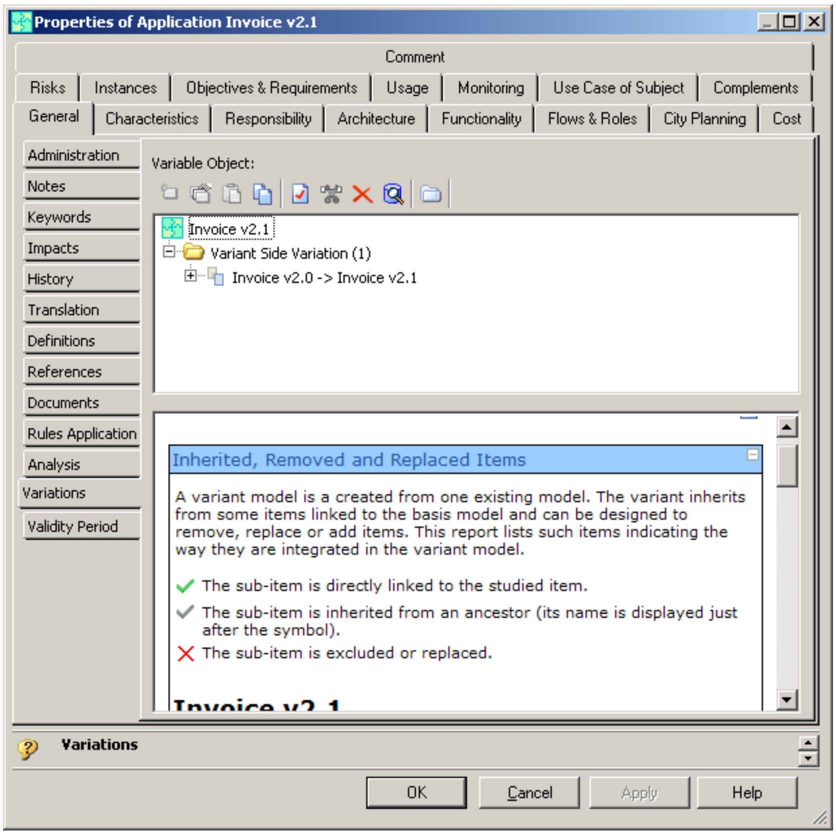
For example, the *Application* metaclass inherits from *Variable Object*.



The inheritance link is performed either tracing the link in a metamodel diagram or adding the *Variable Object* metaclass in the superMetaClass relationship of the concerned concept.



Inheriting from *Variable Object* will add a *General/Variations* tab in the property pages of the extended concept. This tab shows information related to the variation lineage.



Inherited, Removed and Replaced Items

A variant model is a created from one existing model. The variant inherits from some items linked to the basis model and can be designed to remove, replace or add items. This report lists such items indicating the way they are integrated in the variant model.

✓ The sub-item is directly linked to the studied item.

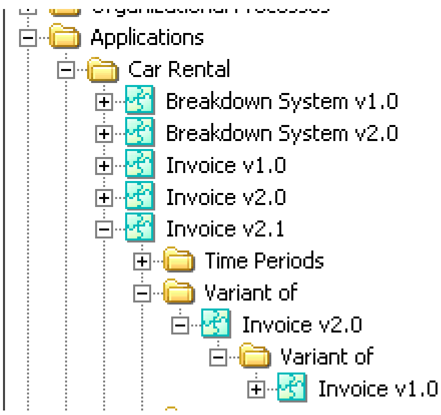
✓ The sub-item is inherited from an ancestor (its name is displayed just after the symbol).

✗ The sub-item is excluded or replaced.

Invoice v2.1

Furthermore, a *New / Variant* command is also added in the popup menu of the variable concept so that variant can be created.

For varied object, a *Variant Of* folder will be displayed in the navigation trees showing the variable concept instances.

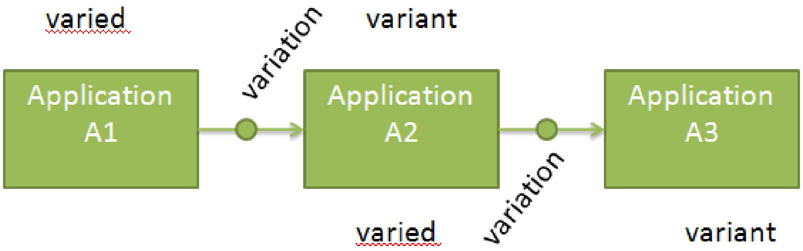


3 Inheritance

The variation mechanism is used to inherit from varied objects. To understand the inheritance we first define the vocabulary:

- **Variation**: the mechanism allowing varying instances. This mechanism is embodied by a MEGA object of type *Variation*.
- **Variant**: any instance created from another existing instance.
- **Varied**: any instance used to create a varied.

Based on those definitions, a variant object of A1 can be a varied of A3.

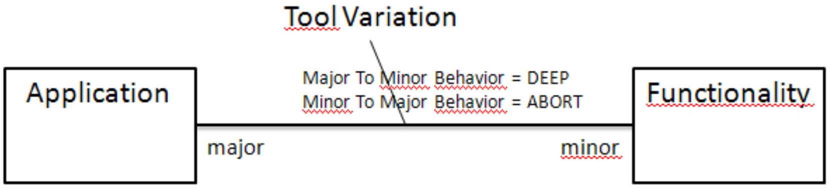


3.1 Inheritance Setting

To be inherited a sub-object must be connected to an inheriting association. Such association is set thanks to the *Tool Variation* operator. This operator is linked to the selected association and the following values must be set:

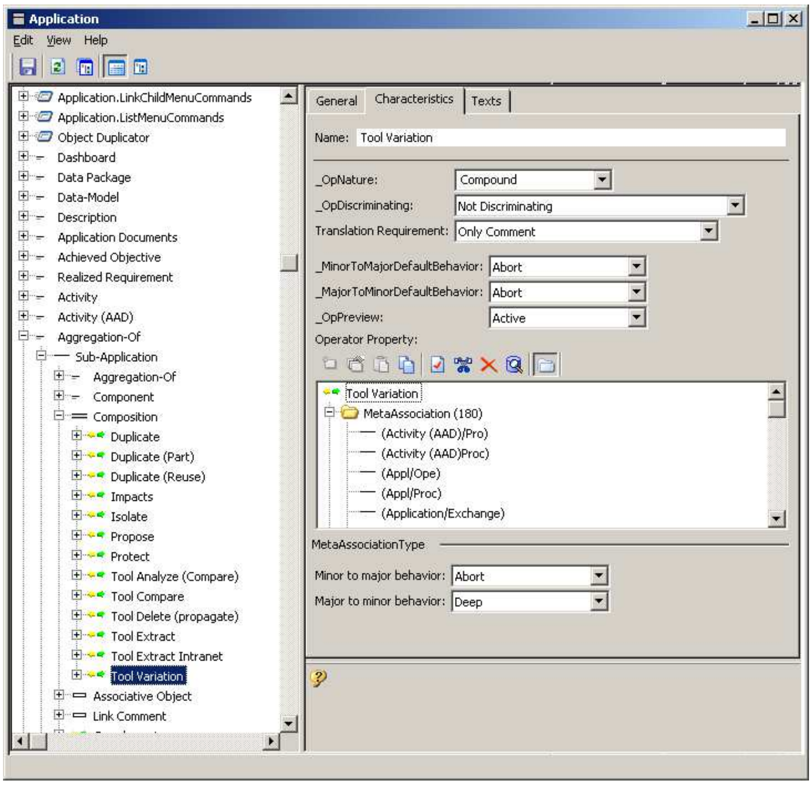
- ***Minor to Major Behavior***
  - **Abort**: there is no inheritance if the sub-object is a major object (yellow folder in the MEGA explorer)
  - **Deep**: the inheritance is applied if the sub-object is a major object (yellow folder)
- ***Major to Minor Behavior***

- o**Abort**: there is no inheritance if the sub-object is a minor object (green folder)
- o **Deep**: the inheritance is applied if the sub-object is a minor object (green folder)


Tool Variation
Application
Major To Minor Behavior = DEEP
Minor To Major Behavior = ABORT
Functionality
major
minor

The following figure shows this operator set for the *Composition* metaassociationtype. In that case, all associations linked to this type behaves the way defined for the operator else the operator is directly set in the association.

For this example, a component application is inherited but a aggregated application is not inherited.



- Application.LinkChildMenuCommands
- Application.ListMenuCommands
- Object Duplicator
- Dashboard
- Data Package
- Data-Model
- Description
- Application Documents
- Achieved Objective
- Realized Requirement
- Activity
- Activity (AAD)
- Aggregation-Of
  - Sub-Application
    - Aggregation-Of
    - Component
    - Composition
      - Duplicate
      - Duplicate (Part)
      - Duplicate (Reuse)
      - Impacts
      - Isolate
      - Propose
      - Protect
      - Tool Analyze (Compare)
      - Tool Compare
      - Tool Delete (propagate)
      - Tool Extract
      - Tool Extract Intranet
      - Tool Variation
    - Associative Object
    - Link Comment

Name: Tool Variation

_OpNature: Compound

_OpDiscriminating: Not Discriminating

Translation Requirement: Only Comment

_MinorToMajorDefaultBehavior: Abort

_MajorToMinorDefaultBehavior: Abort

_OpPreview: Active

Operator Property:

- Tool Variation
  - MetaAssociation (180)
    - (Activity (AAD)/Pro)
    - (Activity (AAD)Proc)
    - (Appl/Ope)
    - (Appl/Proc)
    - (Application/Exchange)

MetaAssociationType

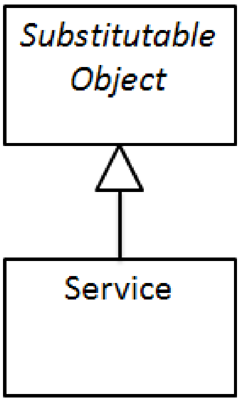
Minor to major behavior: Abort

Major to minor behavior: Deep

3.2 Substitution

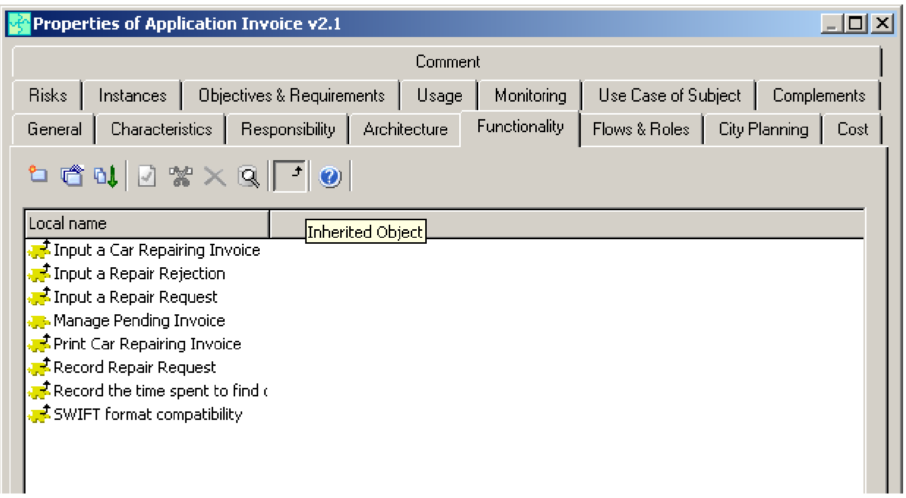
To be removed or replaced a sub-object must be inheritable but also substitutable. The inheritance allows only to grab the sub-object of the varied instance. It does not allow to replace or remove. To do that, the sub-object type must be declared as substitutable. This is done inheriting from the *Substitutable Object* metaclass.

In the following example, the IT Service concept is declared as substitutable. Then, all services inherited in an application can be removed or replaced.



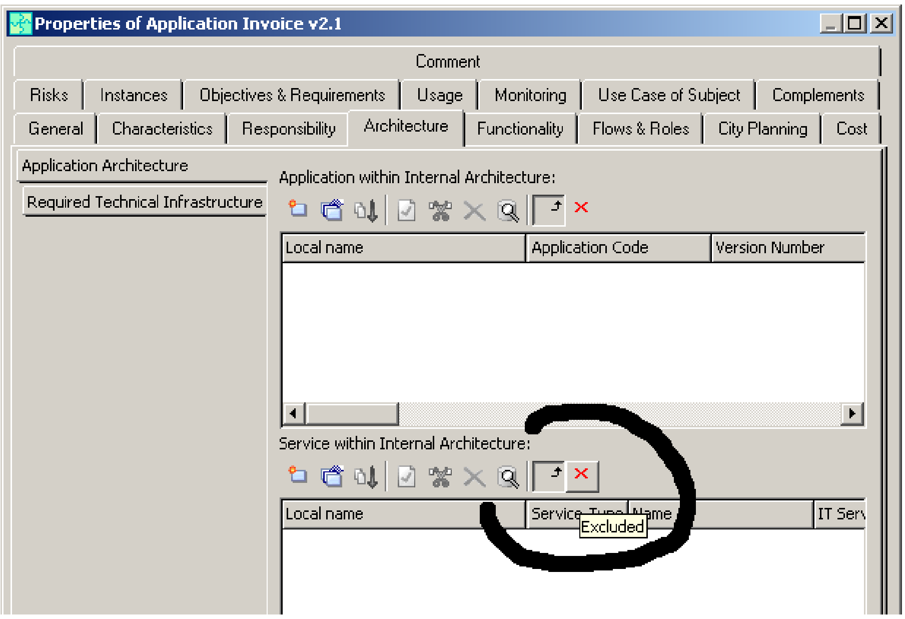
When the replacement and removal is available, all listview of the property pages showing inherited elements allows to perform the removal and the replacement.

For example, the *Functionality* concept is not substitutable. The property page of an application shows only the inheritance (small black arrow).

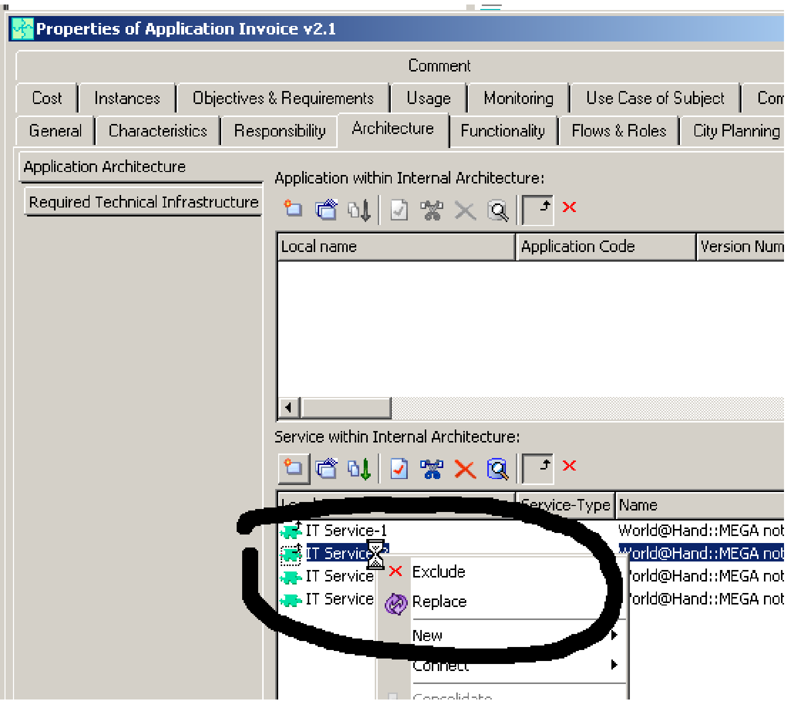

Properties of Application Invoice v2.1

Input a Car Repairing Invoice
Input a Repair Rejection
Input a Repair Request
Manage Pending Invoice
Print Car Repairing Invoice
Record Repair Request
Record the time spent to find ‹
SWIFT format compatibility

The Service concept is substitutable, there is an additional button in the listview of the application property page. The red cross allows to show/hide excluded services.


Properties of Application Invoice v2.1

To remove/replace, the popup menu shows two dedicated commands. The Replace command is available only if there is some objects to be replaced AND some other to do the replacement. The last one must be defined in the variant object and not be inherited.



Customizing Perimeters

Insight. Collaboration. Value.

Introduction to perimeters

This documentation explains how the "perimeter" and "MetaTool" concepts function in HOPEX.

A "perimeter" enables building a set of objects and links from a root object.

A "Metatool" uses a perimeter in the context of a precise functionality.

The "HOPEX Power Supervisor" module is required to configure a perimeter.

Principle

In certain processing, you need to have a set of objects and links around the object that interests you, and this object will be referred to as the "root object". Depending on the context, description of this set can vary by taking into account certain objects or not, certain link types or not.

To solve this problem, the perimeter object has been installed. At the technical level, this is an occurrence of the "Operator" MetaClass, of which the "_OpNature" MetaAttribute is positioned on "Compound".

Configuration

A perimeter groups all the configuration necessary for building the set, and it is important not to confuse the perimeter and the set. Configuration is a behavior specified on the links. Possible behaviors are the following:

- **Deep:** The link and the object are added to the set, then processing continues from this object by propagation.
- **Standard**: The link and the object are added to the set, processing of this branch terminates, and propagation stops here.
- **Link:**The link is added to the set, but not the object. Processing stops at this level.
- **Abort:** Neither link nor object is added, and propagation stops.
- **Computed:**The nature of the link is not sufficient to define behavior on this particular link, and behavior is computed according to context by a macro (for implementation see chapter "Configuring perimeter links").

Defining a MetaTool

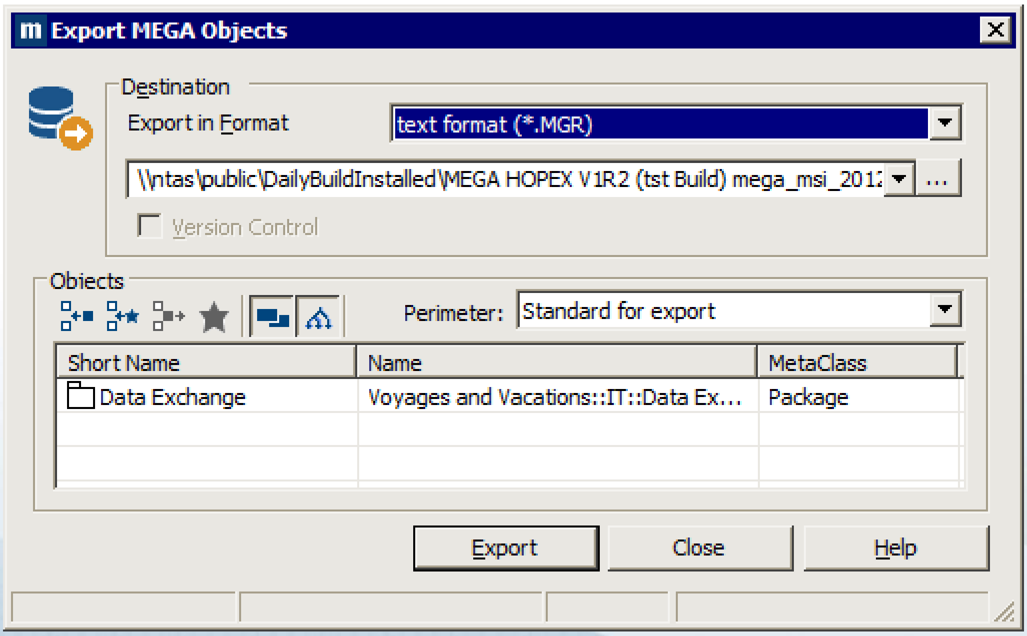
A MetaTool is a functionality which at processing needs to create a set from a root object. In practice, it is not a root object but a set of root objects, but the principle remains the same.
To build this set, the MetaTool applies a perimeter to the root object. It is possible to assign a default perimeter to each MetaTool. The aim of this perimeter is to provide a standard behavior if no customized perimeter exists in this context.

"Export" MetaTool

The "Export" MetaTool enables generation of an extraction file which represents part of the repository. A perimeter is required to build the set of objects and links to be extracted from one or several root objects.

This functionality is available from the object menu by selecting command "Manage\Export", or from the desktop menu by selecting "File\Export\MEGA Objects".

The Export MetaTool dialog box looks like this:



Export in Format: text format (*.MGR)

\\ntas\public\DailyBuildInstalled\MEGA HOPEX V1R2 (tst Build) mega_msi_201

□ Version Control

Perimeter: Standard for export

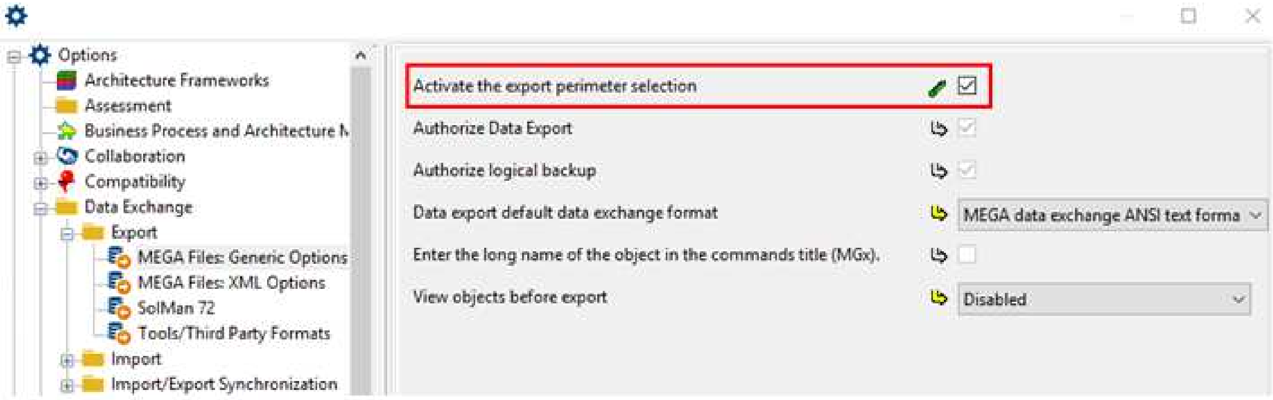
Short Name	Name	MetaClass
Data Exchange	Voyages and Vacations::IT::Data Ex...	Package

By default, a perimeter is already present: "Standard for export", which is the MEGA standard perimeter of the export MetaTool.

The perimeter is not visible as standard in the dialog box.

To activate it:

1. In HOPEX options, expand **Data Exchange > Export** folders.
2. Select **MEGA Files: Generic Options**.
3. In the right pane, select the "Activate the export perimeter selection" option.



Activate the export perimeter selection ☑

Authorize Data Export ☑

Authorize logical backup ☑

Data export default data exchange format: MEGA data exchange ANSI text forma

Enter the long name of the object in the commands title (MGx). □

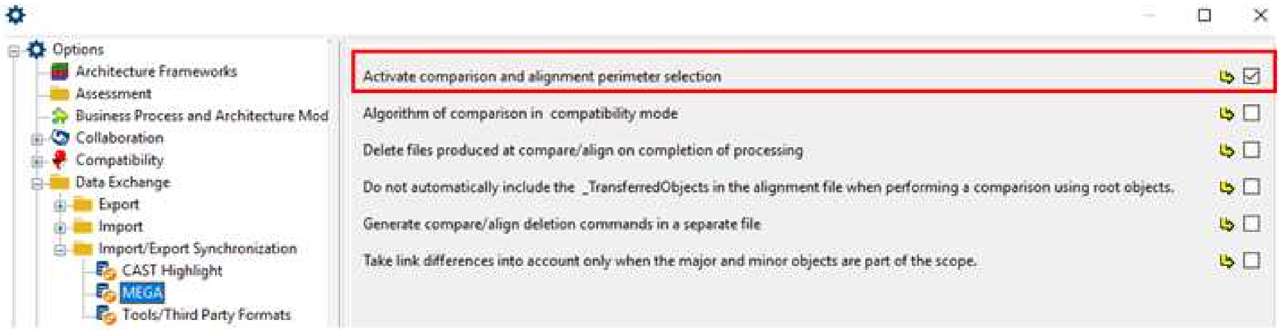
View objects before export: Disabled

"Compare and align" MetaTool

The "Compare and Align" MetaTool enables comparison of objects and links in two MEGA repositories. Processing of compare handles either all objects and links in the repository, or the set of objects and links built from a perimeter and from one or several root objects.

This functionality is available from:

- an object contextual menu by selecting **Manage > Compare and Align**,
- HOPEX desktop menu bar by selecting **Tools > Manage > Compare and Align**,
- an environment menu by selecting **Compare and Align**



Activate comparison and alignment perimeter selection ☑

Algorithm of comparison in compatibility mode □

Delete files produced at compare/align on completion of processing □

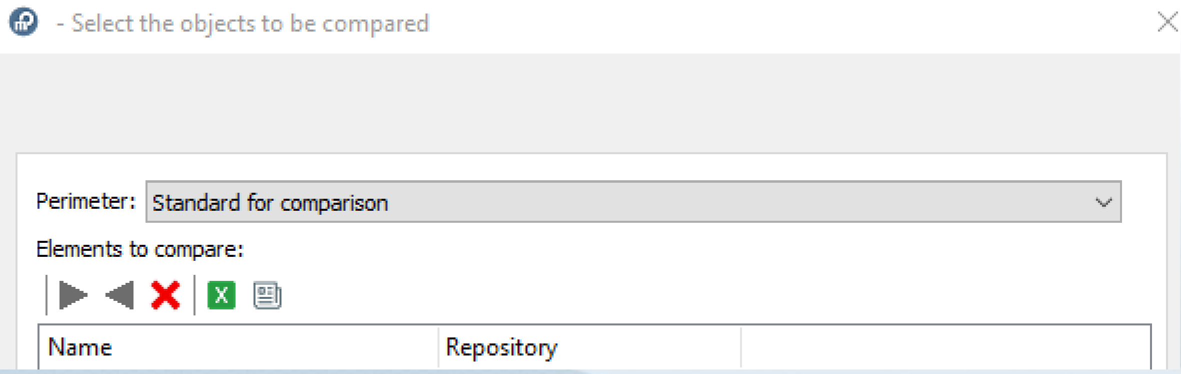
Do not automatically include the _TransferredObjects in the alignment file when performing a comparison using root objects. □

Generate compare/align deletion commands in a separate file □

Take link differences into account only when the major and minor objects are part of the scope. □

Perimeter visibility in this MetaTool is managed by the "Activate compare and align perimeter selection" available in options:

The "Standard for comparison" perimeter is the MEGA standard perimeter for use of the "Compare and align" MetaTool.



- Select the objects to be compared

Perimeter: Standard for comparison

Elements to compare:

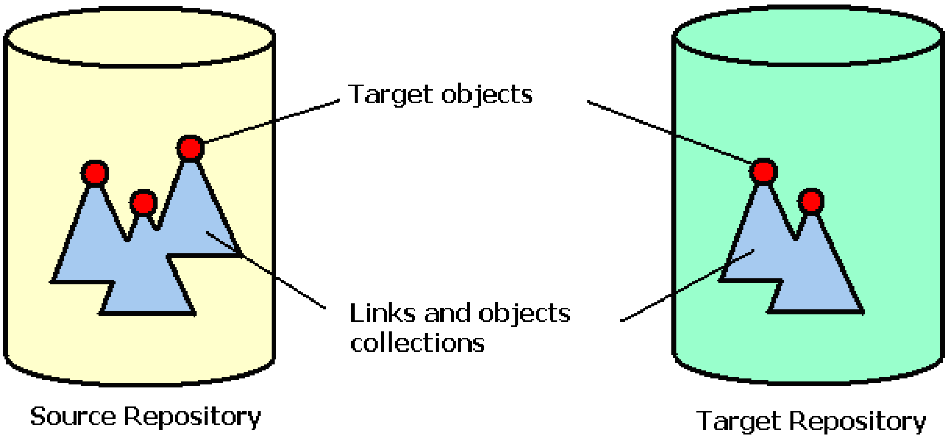
Name	Repository

Building a set by propagation

When we install a perimeter configuration, we need to know how it will be interpreted by the propagation processing which builds the set of objects and links.

Root objects list

The user inputs a set of "Root" objects of the source repository, and a method of creating the set by propagation: the perimeter.

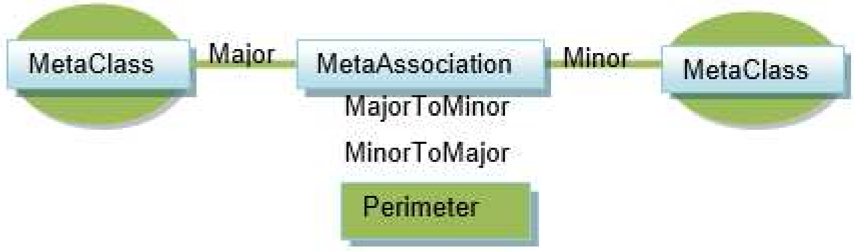


Root objects are not necessarily in the two repositories to be compared. The result will differ depending on whether processing is applied on a set of root objects, or if processing is applied to each root object.

Set of objects

The set of objects is built from root objects by propagation. A perimeter groups behavior of all MetaAssociations, and enables addition or not of an object to the set of objects opposite the MetaAssociation.

The MetaAssociations are directional. Each MetaAssociation has a "Major" MetaClass and a "Minor" MetaClass. Behavior is specified for each perimeter on "MajorToMinor" and "MinorToMajor" MetaAttributes of a MetaAssociation. Behavior can also be specified on the MetaAssociationType connected to the MetaAssociation.



Taking MetaClassA as the root object, we browse the MetaAssociation in direction Major to Minor. The behavior taken into account will be that specified on the MajorToMinor MetaAttribute. Depending on the value of this behavior, the opposite object (ie. MetaClassB) will be added or not to the set with this behavior. If the opposite object is accessible via another MetaAssociation or set of MetaAssociations with different behavior, behavior of this object in the set of objects is updated with the least restrictive behavior (orders being "DEEP", "STANDARD" and "LINK"). Only objects of « DEEP » or « STANDARD » behavior are used in the « Comparison » MetaTool. Objects with "LINK" behavior are used in link processing only.

Root object: by default, this object is added to the set with "DEEP" behavior.

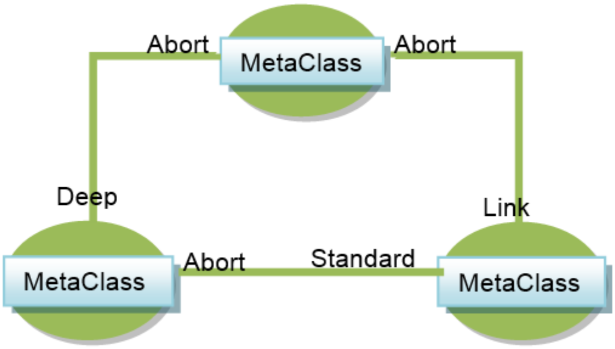
"DEEP" behavior: the opposite object is placed in the set with "DEEP" behavior and propagation continues on this object.

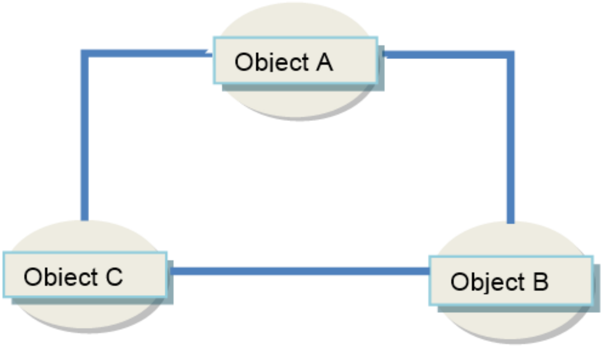
"STANDARD" or "LINK behavior: the opposite object is placed in the set with this behavior and propagation processing stops.

"ABORT" behavior: the opposite object is not placed in the set and propagation processing stops.

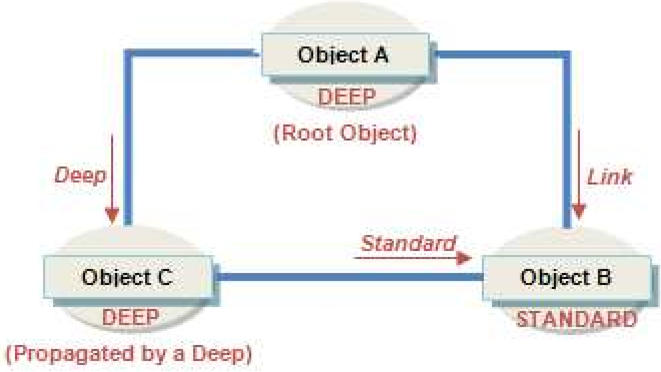
"COMPUTED" link: MetaAssociation behavior is not sufficient to define behavior on this particular link and behavior is computed by a macro which gives behavior related to the current object, which can be "DEEP", "STANDARD", "LINK" or "ABORT". The object is processed in the same way as other objects with the same behavior.

The next setting with the next data





Gives us the following path with behavior of objects; we note that the "Object B" is browsed by "LINK" and by "STANDARD", finally taking "STANDARD" behavior.



Set of links

It is then important to see the repository as a chart rather than a tree, ie. that an object can be browsed by different links. The set of links is built from behavior of objects contained in the set of objects obtained in the previous paragraph.

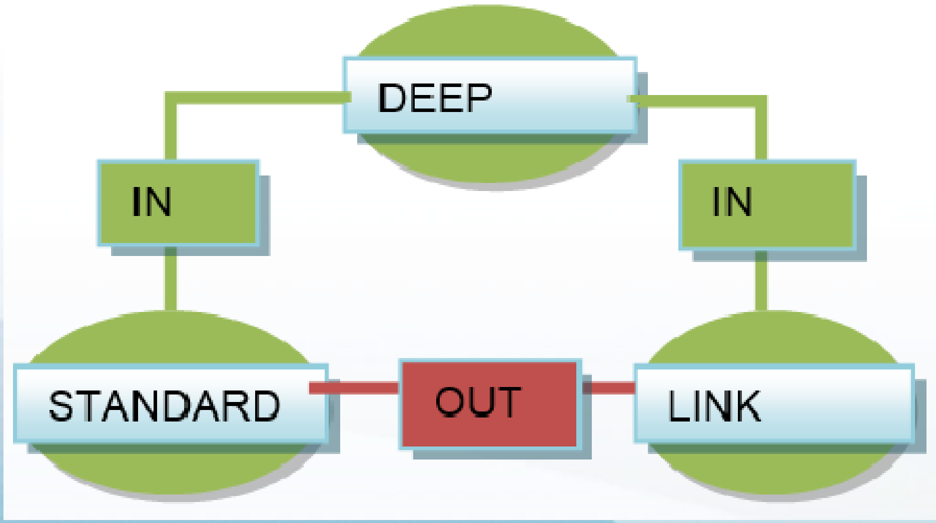
We list all links contained in the perimeter, ie. all occurrences of links between two objects belonging to the set of objects. Depending on the nature of these two objects ("DEEP", "STANDARD" or "LINK"), the link is added or not to the set according to the following table:

		Object		
		Deep	Standard	Link
Opposite object	Deep	IN	IN	IN
	Standard	IN	IN	OUT
	Link	IN	OUT	OUT

IN: the link is added to the set.

OUT: the link is not added to the set.

Example:



The (IN) link between the "DEEP" object and the "STANDARD" object is taken into account in building the set of links. The (IN) link between the "DEEP" object and the "LINK" object is taken into account in building the set of links. The (OUT) link between the "STANDARD" object and the "LINK" object is not taken into account in building the set of links.

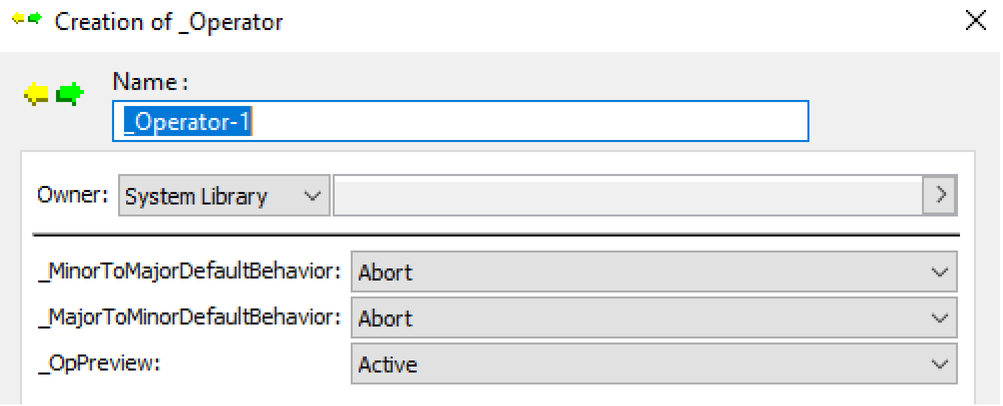
In the "Compare" MetaTool, the "IN" links are taken into account, but not the "OUT" links.

Creating a perimeter

To create a perimeter:

1. From HOPEX, display the **MetaStudio** navigation window (**View > Navigation Windows > MetaStudio**).

   Note: you must be in "Expert" metamodel access.
2. Expand the **Perimeters** folder.
3. Right-click the **Custom Perimeters** folder and select **New > _Operator**.



Creation of _Operator

Name : _Operator-1

Owner: System Library

_MinorToMajorDefaultBehavior: Abort

_MajorToMinorDefaultBehavior: Abort

_OpPreview: Active

4. Enter a **Name** for your perimeter.

The following properties are already specified:

- _MajorToMinorDefaultBehavior

  This is the default behavior to give to links in direction Major Object to Minor Object. This property is set to "Abort". The strategy is to not propagate links so as to avoid a high-volume object set. You then specify behavior on the links you want to browse.

- _MinorToMajorDefaultBehavior

  This is the default behavior to give to links in direction Minor Object to Major Object. This property is also set to "Abort".

- _OpPreview: (Active\Inactive)

  This property enables activation of perimeter display in the configuration MetaTool, see Configuring a perimeter.

5. Click **Finish**.

Customizing a Standard perimeter

Standard perimeters are stored in the **Perimeters > Standard Perimeters** folder.

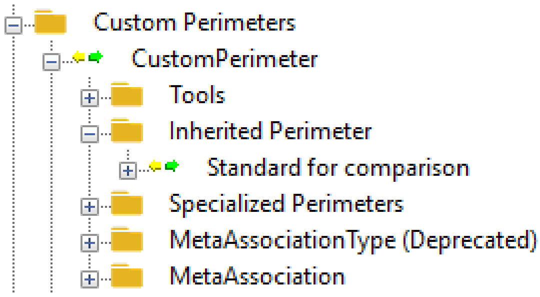
It is not recommended to modify these standard perimeters.

Instead, you must create a custom perimeter, which inherits from the standard perimeter you want to customize and configure this custom perimeter.

To modify a standard perimeter:

1. Create a perimeter, see Creating a perimeter.
2. Expand the perimeter concerned.
3. Right-click the **Inherited Perimeter** folder and select **Connect > _Operator**.
4. Select the perimeter you wanted to modify and click **Connect**.

   For example: “Standard for comparison” _Operator.



Custom Perimeters
CustomPerimeter
Tools
Inherited Perimeter
Standard for comparison
Specialized Perimeters
MetaAssociationType (Deprecated)
MetaAssociation

5. Customize your perimeter, see Configuring a perimeter.

Configuring a perimeter

Using a perimeter in a MetaTool

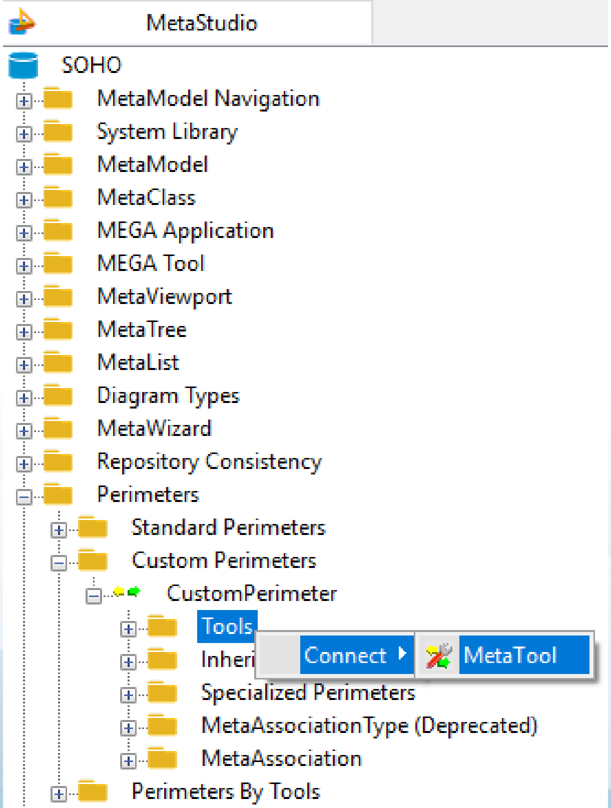
To use a perimeter in a MetaTool, you must connect the perimeter to the MetaTool concerned. A perimeter can have several MetaTools, enabling reuse of a perimeter in several different contexts.

Similarly, a MetaTool can have several perimeters, enabling you to build several different sets of objects and links from a root object.

To connect a perimeter to a MetaTool:

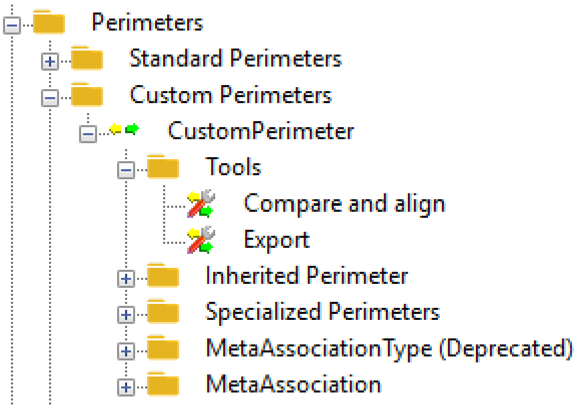
1. Create a perimeter, see Creating a perimeter.

   E.g.: “CustomPerimeter.
2. Expand the perimeter concerned.
3. Right-click **Tools** folder and select **Connect > MetaTool**.



4. Select the MetaTools you want to connect.

   For example, connect it to the "Export" and "Compare and align" MetaTools.



- Perimeters
  - Standard Perimeters
  - Custom Perimeters
    - CustomPerimeter
      - Tools
        - Compare and align
        - Export
      - Inherited Perimeter
      - Specialized Perimeters
      - MetaAssociationType (Deprecated)
      - MetaAssociation

Default perimeter in a MetaTool

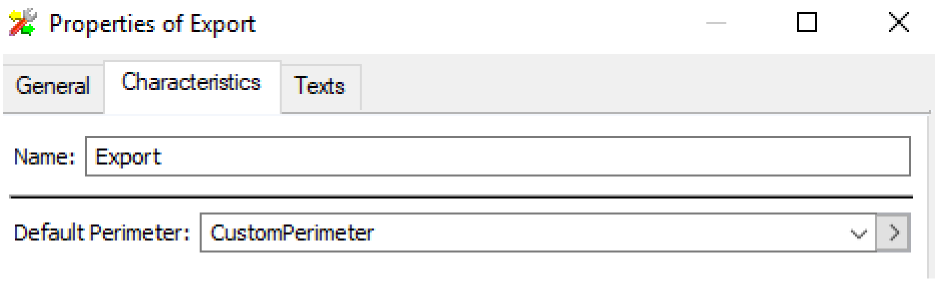
You can modify the default perimeter for a MetaTool.

Prerequisite: The default perimeter you want to define for the MetaTool is connected to the MetaTool, see Using a perimeter in a MetaTool.

To modify the MetaTool default perimeter:

1. In the **MetaStudio** navigation window, expand the **Perimeters** folder.
2. Expand the **Perimeters by Tools** folder.
3. Open the properties of the MetaTool concerned.
4. In the **Characteristics** tab, modify the value of **Default Perimeter** via the drop-down menu.

   For example, you can change the default perimeter of the "Export" MetaTool to insert the perimeter you have created.



Properties of Export

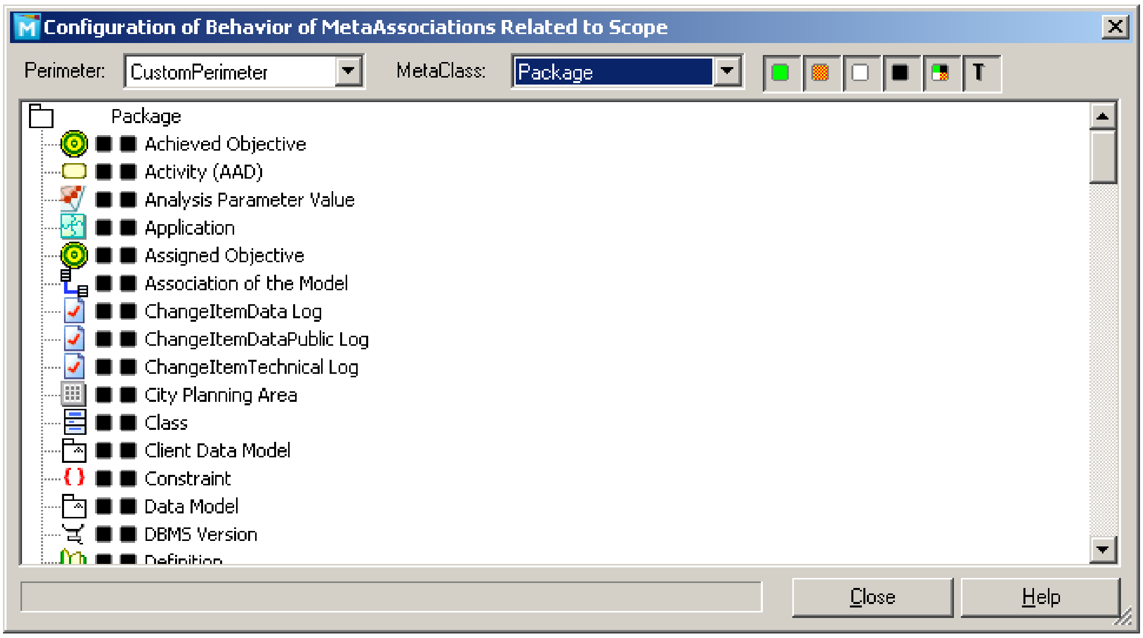
Name: Export

Default Perimeter: CustomPerimeter

Configuring perimeter links

This section covers the configuration of links required for propagation of links and objects in building a set from a perimeter.

To open the configuration dialog box, select **Manage > Parameterize** in the perimeter menu:



Configuration of Behavior of MetaAssociations Related to Scope

Perimeter: CustomPerimeter

MetaClass: Package

Package
- ■ ■ Achieved Objective
- ■ ■ Activity (AAD)
- ■ ■ Analysis Parameter Value
- ■ ■ Application
- ■ ■ Assigned Objective
- ■ ■ Association of the Model
- ■ ■ ChangeItemData Log
- ■ ■ ChangeItemDataPublic Log
- ■ ■ ChangeItemTechnical Log
- ■ ■ City Planning Area
- ■ ■ Class
- ■ ■ Client Data Model
- ■ ■ Constraint
- ■ ■ Data Model
- ■ ■ DBMS Version
- ■ ■ Definition

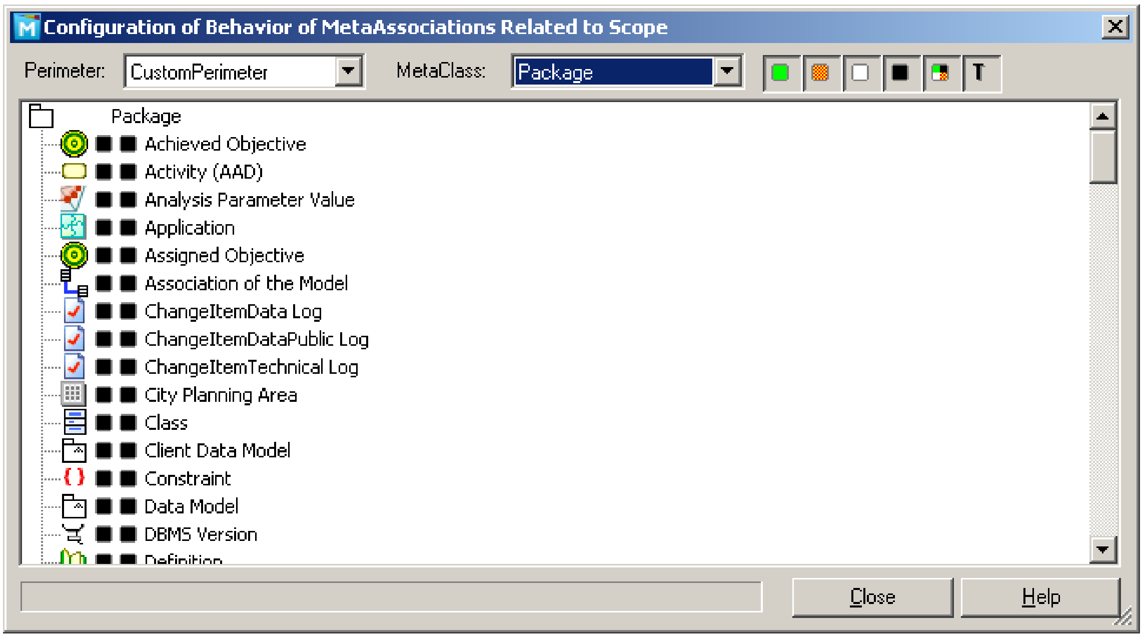
The "Perimeter" box enables selection of the perimeter we want to configure (only perimeters of which the"_OpPreview" MetaAttribute is "Active" are visible.

The "MetaClass" box enables selection of the MetaClass from which links will be browsed, and we begin by using the MetaClass of the root object, for example "Package".

The toolbar includes buttons to filter links (MetaAssociationEnd) according to their behavior:

- 🟩 Filter the links that have "Deep" behavior.
- ▨ Filter the links that have "Standard" behavior.
- □ Filter the links that have "Link" behavior.
- ■ Filter the links that have "Abort" behavior.
- ◩ Filter the links that have "Computed" behavior.

The "T" button enables display in the tree of default behavior of the link before customization, which is "Abort" in our example:



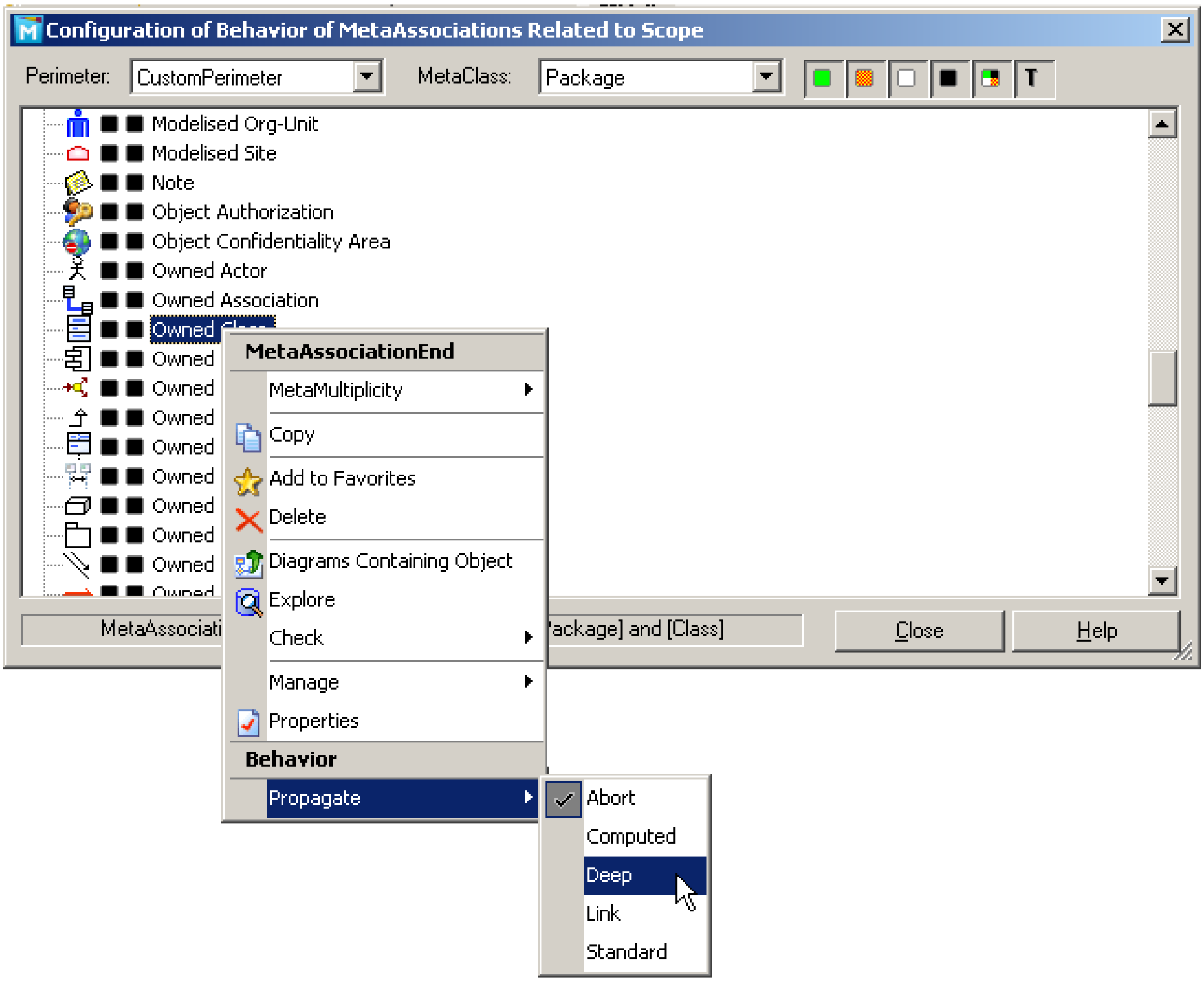
Configuration of Behavior of MetaAssociations Related to Scope

Perimeter: CustomPerimeter

MetaClass: Package

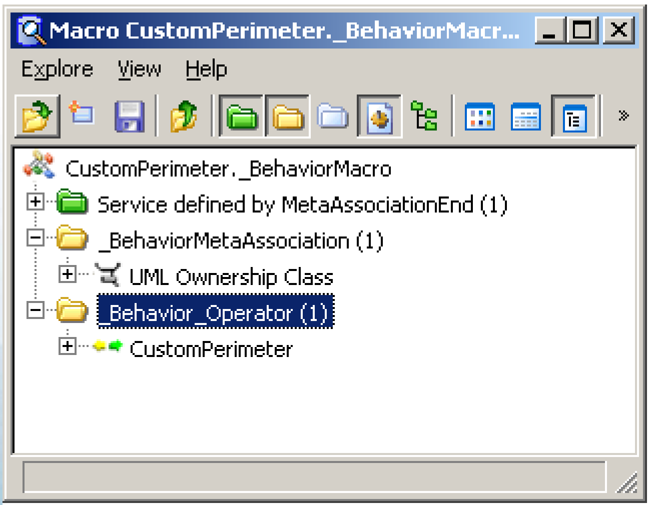
Package
- ■ ■ Achieved Objective
- ■ ■ Activity (AAD)
- ■ ■ Analysis Parameter Value
- ■ ■ Application
- ■ ■ Assigned Objective
- ■ ■ Association of the Model
- ■ ■ ChangeItemData Log
- ■ ■ ChangeItemDataPublic Log
- ■ ■ ChangeItemTechnical Log
- ■ ■ City Planning Area
- ■ ■ Class
- ■ ■ Client Data Model
- ■ ■ Constraint
- ■ ■ Data Model
- ■ ■ DBMS Version
- ■ ■ Definition

To modify propagation behavior of a link, select the new behavior in the "Propagation" menu. The current behavior is represented by a tick alongside the behavior. The standard menu of a link is also accessible.



The new behavior is active for the next use, except for "Computed" behavior which requires additional configuration.

This behavior uses a macro which must be present, and for this you must create a macro and then connect it to the current link and perimeter as here:


Macro CustomPerimeter._BehaviorMacr...
Explore View Help
CustomPerimeter._BehaviorMacro
Service defined by MetaAssociationEnd (1)
_BehaviorMetaAssociation (1)
UML Ownership Class
Behavior_Operator (1)
CustomPerimeter

You must then add VB code in the macro which uses the following method:

```
Function BehaviorCompute(oObject As MegaObject, oPerimeter As Variant) As String
```

Parameter:

- *oObject* is the object which is processed when building the set. It is obtained from propagation of another object: its parent; it is possible to recover this and the link used for propagation.

- *oPerimeter* is the perimeter object, enabling reuse of a macro for several perimeters if required.

Return:

- BehaviorCompute: This function returns the behavior of the link. It could have been computed in several ways, as shown in the example below.

"Computed" behavior example:

We position behavior of the link [Package/Owned Class] on "Computed" to restrict the list of classes owned by a package. We want to build a set of objects with owned classes that are not interfaces.

The following code is applied to the macro:

```
'MegaContext(Fields,Types)
Option Explicit

' ------------------------------------------------
' --  Variables
' ------------------------------------------------
Const cStereotypeInterface = "(NHm3AzqouC0"

' ------------------------------------------------
' --  BehaviorCompute
' ------------------------------------------------
Function BehaviorCompute(oClass As MegaObject, oPerimeter As Variant) As String
  ' Default Behavior : DEEP
  BehaviorCompute = "D"

  If (oClass.GetProp("~wzN3o0nDp840[Stereotype]") = cStereotypeInterface) Then
    ' Behavior : ABORT
    BehaviorCompute = "A"
  End If

End Function
```

"Compare and align" MetaTool filter

The "Standard for comparison" perimeter, which is the default perimeter of the "Compare and align" MetaTool, has a particularity specific to the "Compare" functionality. On this perimeter these is a specific configuration that enables:

- Management of the MetaAttributes used for comparison of objects and links.
- Exclusion of certain MetaClasses and MetaAssociations.

Technical installation

The configuration must be declared in the _Settings text of the perimeter used in the "Compare and align" MetaTool.

If we use the comparison functionality on the entire repository, the configuration used is that present on the default perimeter of the MetaTool. If this text is empty, configuration is taken on the _Settings text of this MetaTool.

If configuration is defined on a perimeter, that of the MetaTool will not be taken into account. It is advisable to filter Meta Attributes too technical :

```
[CmpFilteredMetaAttributes]
~510000000L00[CreationDate]=0
~710000000T00[LinkCreationDate]=0
~610000000P00[UpdateDate]=0
~810000000X00[LinkUpdateDate]=0
~(10000000v30[_Creator]=0
~720000000T40[_LinkCreator]=0
~b10000000L20[_Modifier]=0
~920000000b40[_LinkModifier]=0
~520000000L40[_CreateVersion]=0
~620000000P40[_UpdateVersion]=0
~f20000000b60[Update Log]=0
~a20000000H60[LanguageUpdateDate]=0
~b20000000L60[LinkLanguageUpdateDate]=0
```

Filtering MetaAttributes

To compare an object or link, all MetaAttributes are taken into account, which in certain contexts is pointless and produces too many differences.

Two general filter strategies are installed: either processing takes all MetaAttributes into account, or it takes none. To specify, position the CmpFilterMetaAttributes variable on 1 or on 0 in the "CmpFilterGeneral" section as here:

```
[CmpFilterGeneral]
CmpFilterMetaAttributes= {0|1}
```

If CmpFilterMetaAttributes= 1 (default value), all MetaAttributes of the object or link are used for comparison.

If CmpFilterMetaAttributes= 0, no MetaAttribute is taken.

When strategy has been defined, we add an additional filter, increasingly detailed to suit requirements. We can begin by specifying the MetaAttributes we want to always or never use for all objects and links at comparison. For this, we use the "CmpFilteredMetaAttributes" section, with the following syntax:

[CmpFilteredMetaAttributes]

<Field of MetaAttribute>= {0|1}

Ex: ~510000000L00[Date de creation]=0

If <Field of MetaAttribute>=0, the MetaAttribute is not taken into account for all MetaClasses and MetaAssociations.

If <Field of MetaAttribute>=1, the MetaAttribute is taken into account for all MetaClasses and MetaAssociations.

It is then possible to similarly configure for the MetaAttributes we want to manage or not according to the MetaClasses in the "CmpFilteredMetaAttributesByMetaClasses" section, with the following syntax:

[CmpFilteredMetaAttributesByMetaClasses]

<Field of MetaClass>, <Field of MetaAttribute>= {0|1}

Ex: ~d20000000U20[_Collection],~510000000L00[Creation date]=0

If <Field of MetaClass>, <Field of MetaAttribute>=0, the MetaAttribute is not taken into account for this MetaClass.

If <Field of MetaClass>, <Field of MetaAttribute>=1, the MetaAttribute is taken into account for this MetaClass.

We can similarly filter MetaAttributes on the MetaAssociations in the "CmpFilteredMetaAttributesByMetaAssociations" section, with the following syntax:

[CmpFilteredMetaAttributesByMetaAssociations]

<Field of MetaAssociation>, <Field of MetaAttribute>= {0|1}

Ex: ~d20000000U20[_Collection],~510000000L00[Creation date]=0

If <Field of MetaAssociation>, <Field of MetaAttribute>=0, the MetaAttribute is not taken into account for this MetaAssociation.

If <Field of MetaAssociation>, <Field of MetaAttribute>=1, the MetaAttribute is taken into account for this MetaAssociation.

Filtering MetaClasses and MetaAssociations

It is possible to filter the MetaClasses we do not want to take into account at comparison. In the "CmpExcludedMetaClasses" section, we list the MetaClasses we want to exclude, with the following syntax:

```
[CmpExcludedMetaClasses]
<Number>=<Field of MetaClass>
1=~o20000000A30[_Dispatch]
```

The <Field of MetaClass> MetaClass is excluded from comparison.

As for MetaClasses, we can filter MetaAssociations we want to exclude by listing these in the "CmpExcludedMetaAssociations" section as below.

```
[CmpExcludedMetaAssociations]
<Number>=<Field of MetaAssociation>
1=~oXRVCA8Dp8i1[Owned class]
```

The <Field of MetaAssociation> MetaAssociation is excluded from comparison.

Standard configuration

A configuration is already present on the "Compare and align" MetaTool, excluding comparison of technical MetaClasses such as "_Dispatch" for example, and not take into account technical MetaAttributes such as"Logfile" for example.

The following is complete standard configuration:

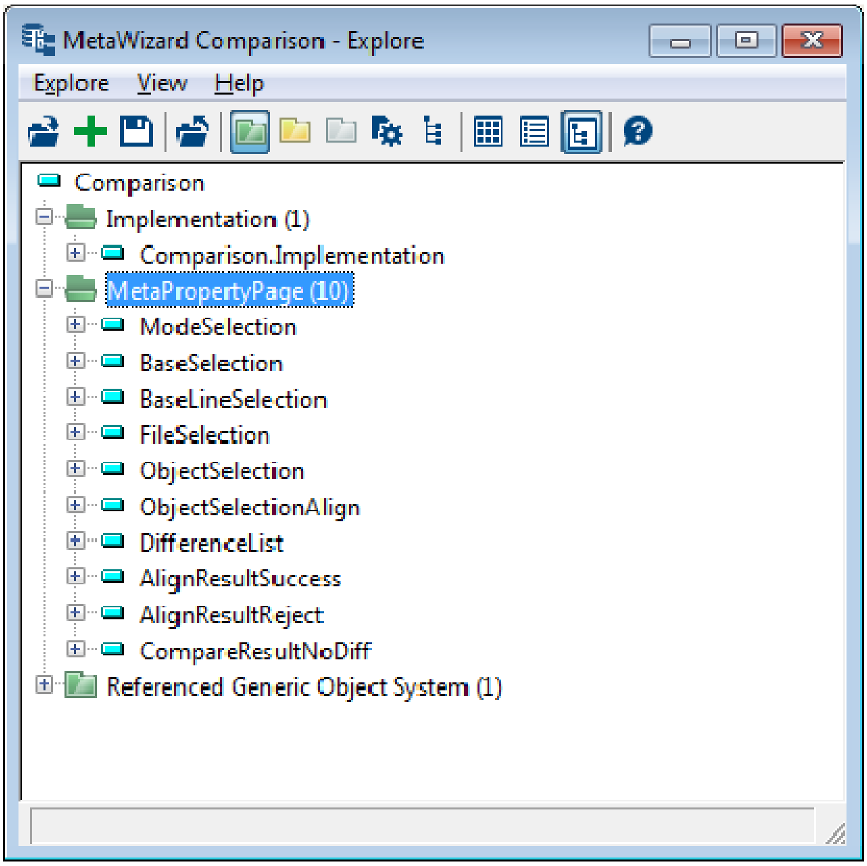
```
[CmpFilterGeneral]
CmpFilterMetaAttributes=1

[CmpFilteredMetaAttributes]
~510000000L00[Creation date]=0
~610000000P00[Modification date]=0
~(10000000v30[Creator]=0
~b10000000L20[Modifier]=0
~520000000L40[Creation version]=0
~620000000P40[Modification version]=0
~f20000000b60[Logfile]=0
```

```
[CmpExcludedMetaClasses]
1=~o20000000A30[_Dispatch]
2=~n20000000630[_DispatchData]
3=~nFhC9FZc0P30[Recent query]
4=~d20000000U20[_Collection]
5=~e20000000Y20[_CollectionItem]
6=~f20000000c20[_CollectionItemLink]
7=~h20000000k20[ChangeItemData]
8=~l20000000(20[ChangeItemDataPublic]
9=~r20000000M30[ChangeItemDataTechnical]
10=~i20000000o20[ChangeItemSystem]
11=~m20000000230[ChangeItemSystemPublic]
12=~t20000000U30[ChangeItemSystemTechnical]
```

Compare Tool API (to compare and align)

CompareTool is a MEGA object enabling management of object comparison between the current private workspace and a target repository in VBScript. Comparison enables creation of an update file designed to align the target repository. This functionality is available with the "HOPEX Power Supervisor" module.

NOTE: The comparison wizard provided as standard is a comprehensive and well maintained example of API use.



Implementation to VbScript

MegaObject can be installed using the following methods:

- Function OpenCompareToolEx(ByVal oParameterSource As Object, ByVal oParameterTarget As Object, ByVal strOption As String) As Object

This method enables instancing of the CompareTool object. Comparison is between a source and a target repository. You can either open the repository you wish to compare or provides the required parameter to let the compare tool opening the repository for you.

The two first parameter can be either a MEGARoot or a String. In case of a MEGARoot it'll be used by the CompareTool. If it is a String, it should describe the repository to be used:

strParameter
This is a string option of the form:
<optionName1>=<optionValue1>,<optionName2>=<optionValue2>,…
DbTarget = it can be either a base name, a base id or "TRANSACTION" or "FILE". If you select "TRANSACTION" the current private workspace will be used. If you select "FILE", a temporary repository will be created and your data file imported into it.
FileTarget = When specifying "FILE" as DbTarget, here is the path of the data file to import.
DbSource = Like DbTarget but for the source
FileSource = Like FileTarget but for the source
BaselineSource = When selecting a base name or a base id as DbSource you can open the repository according to an archived state.
BaselineTarget = Like BaselineSource for the target. Of course when a repository is opened according to an archived state, it is read only. You will not be able to align comparison result.

The last parameter is for option. You can leave it empty or use the previous syntax: <optionName>=<optionValue>
Compatibility = 0 or 1 lets you select the old comparison algorithm for compatibility purpose.

- **Sub Close()**
  Frees the target repository private workspace and frees the MegaObject variables.

Attributes

- **ContinueOnConfidential:**
  Enables specification of whether comparison processing should be aborted if a confidential object is present in the collection of objects to be compared (in source or target repository).
  TRUE: comparison processing is not aborted.
  FALSE: processing is aborted; this is the default value.

- **Perimeter:**
  Enables modification of the perimeter of extraction of sets of objects and links to be compared in repositories. This attribute is not used when the "CompareAll" method is used.

- **FoundConfidential:**
  This attribute is specified by comparison processing to indicate presence of a confidential object in one of the two object collections to be compared.

Methods

Comparison between complete repository or object sets is necessary to enable generation of an alignment file.

- **Sub CompareAll()**
  Compares all objects and links of the source repository related to the target repository.

- **Sub CompareObjects(mgobjcolFromSource as MegaCollection, mgobjcolFromTarget as MegaCollection)**
  Compares the a set of objects and links from repository source and/or target.
  Parameter:
  mgobjcolFromSource*: Object set of the source repository enabling, with assistance of the perimeter, constitution of a set of objects and links for the two repositories to be compared (source and target).*

mgobjcolFromTarget: Object set of the target repository enabling, with assistance of the perimeter, constitution of a set of objects and links for the two repositories to be compared (source and target).

- **Sub SetGenerateOutputOption(strOptionName As String, varOptionValue As Variant)**
  Enables specification of variables used at alignment file generation.
  Parameter:
  strOptionName*: Name of variable identifying the Option. The following three options are available: "UpdateExtract", "DeleteFile", "IgnoreAttributes" and "TransferredObjects".*
  varOptionValue: *Option value. The value depends on Option:*
  ** "DisconnectExtract": This option enables specification of whether the alignment file generates "disconnect" commands rather than "delete" commands; default value of this option is FALSE.*
  ** "DeleteFile: A second file name can be configured using this option in order to manage "delete" commands. The alignment file takes into account all actions, and if a file name is defined for this option, all "delete" commands are sent to this file.*
  ** "IgnoreAttributes": Processing compares all object MetaAttributes, but with this option it is possible to ignore certain MetaAttribute types. Each element is separated by " ; ". MetaAttribute types are as follows: "Date" ("Date" MetaAttributes are not taken into account at generation of alignment files, for example: Modification Date),"Creator" (MetaAttributes linked to object creation are ignored), "Authorization" (MetaAttributes linked to authorization are not managed), "Comment" (Comment MetaAttributes are ignored), "Order" (Order MetaAttribute is not taken into account). A MetaAttribute can be specifically ignored by adding it to the list. This MetaAttribute will be ignored for all objects and links,*
  for example: oCompareTool.SetGenerateOutputOption "IgnoreAttributes", "Date; *~610000000P00[Date de modification];~b10000000L20[Modificateur];Order"*

** " TransferredObjects": This option lets the user choose to include or not objects of merging (_TransferedObject). By default this option is true. This value is a Boolean.*

- **Sub ResetGenerateOutputOptions()**
  This method enables reinitialization of options that were specified using the previous method: "SetGenerateOutputOption".

- **Sub GenerateOutput(strOutputFileName As String, strOutputType As String)**
  This method generates the alignment file named "strOutPutFileName". If the "DeleteFile" option was specified a file with "delete" commands is created.
  Parameters:
  strOutputFileName*: Alignment file name.*
  strOutputType*: enables modification of alignment file format. Default format is "MGR".*

- GetRootSource

Returns the MEGA Root used for comparison as Source Repository. This is convenient especially if you opened is through the call to OpenCompareTool

- GetRootTarget

Just like GetRootSource, returns the MEGA Root of the target repository.

- Align

This function will align the differences with the target repository. If an error occurs it returns a path (string) to the rejected commande file. If there are no rejected command an empty string is returned. You can only align if the target repository is writable. As an instance, you cannot use align if the target repository is an archived state based on a dispatch.

- Commit

This method must be called after the align function and before closing the target repository. Commit will save every update on the target repository after the alignment.

- Rollback

This method can be called instead of “Commit” to cancel every update processed in the target repository with the “align” function.

- SetMatchingObjectMacro(strMacroId)

You can call this method specifying a macro Id.

When comparing two repositories, the standard behavior to find the matching object in the other base is to look for an object with the same idabs.

The macro must implement this function:

Function GetMatchingObject(oContext, oReference)

oReference is the MEGAObject that is searched in the other repository

oContext is an object with the following available methods:

- GetDirection: returns a String with two possible values:
    - "ST": oReference is from the Source repository and the macro should look for the matching object in the Target repository.
    - "TS": oReference is from the Target repository and the macro should look for the matching object in the Source repository.
- GetMatchingObjectRoot: returns the MEGARoot of the repository where the matching object is searched.

This macro must always return a MEGAObject (even if no matching object have been found).

Here is a sample function with quite the same behavior as the standar.

```
Function GetMatchingObject(oContext, oReference)

        Dim oMatchingObjectRoot

        set oMatchingObjectRoot = oContext.GetMatchingObjectRoot()

        Dim szDirection

        szDirection = oContext.GetDirection()

        Dim oMatchingObject

        Set oMatchingObject =
oMatchingObjectRoot.GetCollection(oReference.GetClassId).Item(oReference.GetId())

        Set GetMatchingObject = oMatchingObject

End Function
```

Here, ".item" on a MEGACollection always returns a mega object even if the item is not found.

- GetDiffResultCollection

After a CompareAll or CompareObjects this methods returns a MEGACollection containing all differences.

This MEGA Collection contains object with attributes described with the following Informal Query: "~CcRkodhCELgI[Compare.DiffResult.Collection]"

- “~410000000H00[Order]” Number: This is the order of the current difference.

- “~310000000D00[Absolute Identifier]” Idabs: This is the identifier of the current difference.

- “~snk(al)kEboV[MetaPicture]” Idabs: this is the picture’s idabs of the current difference (according to its type: create, delete, link, …).

- “~K3Rv(lEmEvlI[DiffTypeMetaPicture]” MegaIdentifier: this metapicture depends if it is a update, a connect, a delete…

- “~3SqlLqFEEPoF[Kind]” Enumeration: this is the kind of the difference, a link or an object. Many attributes are available only link or object kind.

- ”~)TqlXLHEEn3G[SourceAvailable]” Boolean: true if the element is available in the source repository.

- ”~PUqliLHEEj4G[TargetAvailable]” Boolean: true if the element is availbale in the target repository.

- ”~ZApdg(vjEX67[Difference]” Enumeration: It is the difference type: Connect, create, delete,…

0: Aucune différence

1: Créé

2: Modifié

3: Supprimé

4: Relié

5 : Délié

- ”~CVJie9xjEzg5[Target]” String : The metaclass name of the object or the metaassociation name of the link.

- ”~ASJi1AxjEHj5[Object 1]” String: the object name if the difference kind is an object. The object next to the link if the difference is a link.

- ”~BTJiGAxjELm5[Object 2]” String: if the difference is a link the other object name next to the link.

- ”~lTqlPgcDEb8F[ObjectSourceIdabs]” Mega Idabs: if the difference is about an object, this property is the idabs of the object in the source database when it exists.

- “~jSqlClcDEzIF[ObjectSourceMetaclassIdabs]” Mega Idabs: if the difference is about an object, this property is the idabs of the metaclass of the object in the source database when it exists.

- “~sTqlmgcDEP9F[ObjectTargetIdabs]” Mega Idabs: if the difference is about an object, this property is the idabs of the object in the target database when it exists.

- “~ETqlBtcDEzKF[ObjectTargetMetaclassIdabs]” Mega Idabs: if the difference is about an object, this property is the idabs of the metaclass of the object in the target database when it exists.
- “~LSqlvKfDE1OF[LinkSourceIdabs]” Mega Idabs: if the link exists in the source database, this property is the metaassociation idabs.
- “~9TqlULfDEzOF[LinkSourceMajorIdabs]” Mega Idabs: If the link exists in the source database, this property is the major metaassociationend idabs.
- “~5Vql)qGEErwF[LinkSourceMajorMetaclassIdabs]” Mega Idabs: if the link exists in the source database, this property is the metacalss idabs on the major side of the link.
- “~jTqlpLfDEvPF[LinkSourceMinorIdabs]” Mega Idabs: If the link exists in the source database, this property is the minor metaassociationend idabs.
- “~FSqlhrGEEjyF[LinkSourceMinorMetaclassIdabs]” Mega Idabs: if the link exists in the source database, this property is the metacalss idabs on the minor side of the link.
- “~OVqliuKEE59G[LinkTargetIdabs]” Mega Idabs: if the link exists in the target database, this property is the metaassociation idabs.
- “~LUqlCMfDErQF[LinkTargetMajorIdabs]” Mega Idabs: If the link exists in the target database, this property is the major metaassociationend idabs.
- “~uSql5sGEEfzF[LinkTargetMajorMetaclassIdabs]” Mega Idabs: if the link exists in the target database, this property is the metacalss idabs on the major side of the link.
- “~1UqlwOfDEfUF[LinkTargetMinorIdabs]” Mega Idabs: If the link exists in the target database, this property is the minor metaassociationend idabs.
- “~OTqlMsGEEb(F[LinkTargetMinorMetaclassIdabs]” Mega Idabs: if the link exists in the target database, this property is the metacalss idabs on the minor side of the link.

Sample (VBScript)

```
Option Explicit

Const cszDbSourceName = "Dev"
Const cszDbTargetName = "Prod"

Function DifferenceTypeName(strDiffValue)
  DifferenceTypeName = ""
  dim mgDiffTypeCurrent
  For Each mgDiffTypeCurrent In
GetObjectFromId("~ZApdg(vjEX67[Difference]").GetCOllection("~uGZC89p5ua00[_ParameterValue]")
    If strDiffValue  = mgDiffTypeCurrent.GetProp("~L20000000L50[Valeur interne]") Then
    DifferenceTypeName = mgDiffTypeCurrent.Name
    Exit For
   End If
  Next
End Function

Dim mgCmpTool
Set mgCmpTool = GetRoot.OpenCompareToolEx("DbSource=" & cszDbSourceName, "DbTarget=" &
cszDbTargetName, “”)
If Not mgCmpTool Is Nothing Then
  print "Repository " & mgCmpTool.GetRootSource.Name & " opened as source"
  print "Repository " & mgCmpTool.GetRootTarget.Name & " opened as target"
  mgCmpTool.CompareAll
  Dim mgcolDiff
  Set mgcolDiff = mgCmpTool.GetDiffResultCollection
  Dim mgDiff
  For Each mgDiff In mgcolDiff
   Dim strDiff
   strDiff = mgDiff.GetProp("~ZApdg(vjEX67[Difference]")
   Dim strtarget
   strTarget = mgDiff.GetProp("~CVJie9xjEzg5[Cible]")
   Dim strObject1
   strObject1 = mgDiff.GetProp("~ASJi1AxjEHj5[Objet 1]")
   Dim strObject2
   strObject2 = mgDiff.GetProp("~BTJiGAxjELm5[Objet 2]")
   print DifferenceTypeName(strDiff) & ": [" & strtarget & "] " & strObject1 & "/" & strObject2
  Next
  mgCmpTool.Close
End If
```

Sample (Java)

```
final MegaRoot mgRoot = mgobjSource.getRoot();
final MegaCurrentEnvironment mgEnvironment = mgRoot.currentEnvironment();
final MegaToolkit mgToolkit = mgEnvironment.toolkit();
final MegaCollection mgcolOrgProcess = mgRoot.getCollection("~gsUiU9B5iiR0[Organizational Process]");
final MegaObject mgNewOrgProcess = mgcolOrgProcess.create();
final MegaObject mgobjPerimeter = mgRoot.getObjectFromID("~cj6s5ij3q400[Standard for comparison]");
if ((null != mgobjPerimeter) && mgobjPerimeter.exists()) {
  final MegaCollection mgcolObjectToCompare = mgRoot.getSelection("");
  mgcolObjectToCompare.add(mgNewOrgProcess);
  final MegaCompareTool oCompareTool = new MegaCompareTool(mgRoot, "DbSource=TRANSACTION",
"DbTarget=" + mgRoot.getProp("~Z20000000D60[Short Name]"), "");
  oCompareTool.perimeter(mgToolkit.getString64FromID(mgobjPerimeter.getID()));
  oCompareTool.compareObjects(mgcolObjectToCompare);
  final String strRejects = oCompareTool.align();
  if (0 == strRejects.length()) {
   oCompareTool.commit();
  } else {
    oCompareTool.rollback();
  }
  oCompareTool.close();
  mgcolObjectToCompare.release();
  mgobjPerimeter.release();
}
mgNewOrgProcess.release();
mgToolkit.release();
mgEnvironment.release();
mgcolOrgProcess.release();
mgRoot.release();
```

Export Tool API (to Export)

ExportTool is a MEGA object enabling management of standard export of an object or collection of objects in VBScript or JAVA. This functionality is available with the "HOPEX Power Supervisor" module.

Implementation to VbScript

MegaObject can be installed using the following methods:

Syntax:

Function OpenExportTool () As MegaObject

This method returns an instance of the "ExportTool" MegaObject.

Attributes

- **Propagate:** Allows you to specify if object roots must be propagated.
  - TRUE: To propagate; this is the default value.
  - FALSE: To take object roots only.

- **TransferedObject:** Takes account of transfered objects.
  - TRUE: The transfered objects are part of object roots; this is the default value.
  - FALSE: No transfered objects taken.

- **Format:** Choice of format of an exchange of data.
  - MGR: Ansi text data exchange; this is the default value**.**
  - XML: XML data exchange.

- **Perimeter:** Perimeter is used in Export Processing to extract sets of objects and links to generate the export file. The default perimeter is defined on MetaTool “Standard of Export”.

- **ContinueOnConfidential:** Enables specification of whether export processing should be aborted if a confidential object is present in the collection of objects to be exported.
  - TRUE: Export processing is not aborted; this is the default value.
  - FALSE: Export processing is aborted.

- **FoundConfidential:** This attribute is specified by Export processing to indicate presence of a confidential object in object collection. You cannot modify this attribute.

Methods

Export Function

Export an object or a collection of object in data exchange file.

Syntax:

```
Sub Export (
   oObject As MegaObject,
   szFilename As String
)
```

Parameters:

oObject [In]

MegaObject

This object is root of Export processing.

szFileName [In]

String

This is the name of the result file. It is the full name: local or network, directories, short name of the file and Extension.

Syntax:

```
Sub Export (
   oObjectColl As MegaCollection,
   szFilename As String
)
```

Parameters:

oObjectColl [In]

MegaCollection

This collection of objects is root of Export processing.

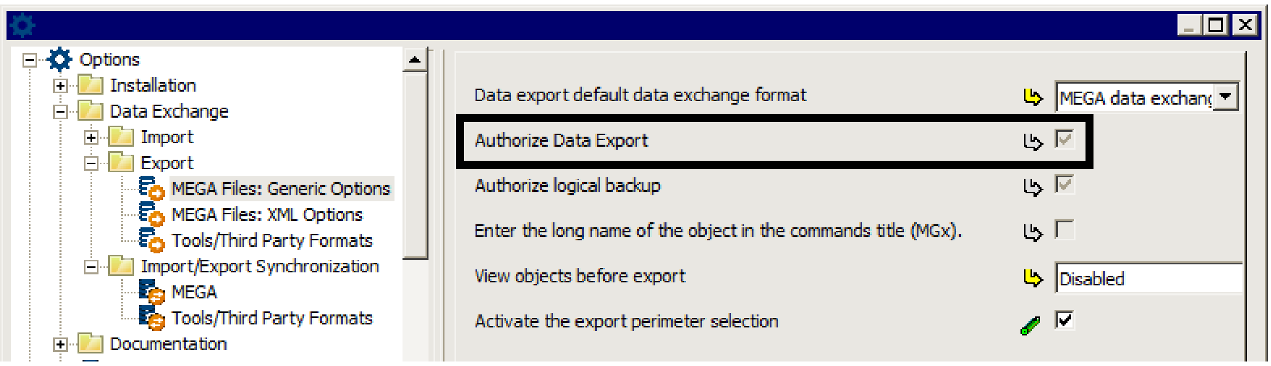
szFileName [In]

String

This is the name of the result file. It is the full name: local or network, directories, short name of the file and Extension.

Remarks

The option “Authorize Data Export’ must be activated to use the ExportTool component.


Options
Installation
Data Exchange
Import
Export
MEGA Files: Generic Options
MEGA Files: XML Options
Tools/Third Party Formats
Import/Export Synchronization
MEGA
Tools/Third Party Formats
Documentation

Data export default data exchange format: MEGA data exchanç

Authorize Data Export ☑

Authorize logical backup ☑

Enter the long name of the object in the commands title (MGx). □

View objects before export: Disabled

Activate the export perimeter selection ☑

Sample

```
Option Explicit

' -----------------------------------
' -- ExportTool                V1.0
' -----------------------------------
' Samples of ExportTool object installation

' Creation of 'Component ExportTool' :
Dim oExportTool
Set oExportTool = OpenExportTool

' Display parameters :
Print "Default Parameters :"
Print "- Perimeter            = " & oExportTool.perimeter
Print "- Format               = " & oExportTool.Format
Print "- Propagate            = " & oExportTool.Propagate
Print "- TransferedObject       = " & oExportTool.TransferedObject
Print "- ContinueOnConfidential = " & oExportTool.ContinueOnConfidential
```

```
' -----------------------------------
' -- Sample 1 :
' -----------------------------------
' Export a package with no transfered object
' By Default : Propagation is Active and Format is 'MGR'
' -- Constants :
Dim strObjectFileName
strObjectFileName = "C:\TEMP\ObjectExport.MGR"
oExportTool.TransferedObject = FALSE
' --
Dim oPackage
Set oPackage = GetRoot.GetCollection("~h8rEkjZmo400[Paquetage]").Item(1)
If (oPackage.GetID <> 0) Then
  oExportTool.Export oPackage, strObjectFileName
  ' Dispaly result file in NotePad :
  Dim oObjWShell
  Set oObjWShell = CreateObject("WScript.Shell")
  oObjWShell.Exec """C:\WINDOWS\system32\notepad.exe"" " & strObjectFileName
End If
Set oPackage = Nothing
oExportTool.TransferedObject = TRUE


' -----------------------------------
' -- Sample 2 :
' -----------------------------------
' Export all packages with No Propagation
' and the 'XML' Format
' -- Constants :
Dim strCollectionFileName
strCollectionFileName = "C:\TEMP\CollectionExport.XML"
oExportTool.Propagate = FALSE
oExportTool.Format    = "XML"
' --
Dim oPackageColl
Set oPackageColl = GetRoot.GetCollection("~h8rEkjZmo400[Paquetage]")
oExportTool.Export oPackageColl, strCollectionFileName
' Dispaly result file in Internet Explorer :
```

```
Dim oCollWShell
Set oCollWShell = CreateObject("WScript.Shell")
oCollWShell.Exec """c:\program files\internet explorer\iexplore.exe"" " & strCollectionFileName

Set oPackageColl = Nothing

' Display "FoundConfidential" is need :
If (oExportTool.FoundConfidential) Then
  Print "--> Confidential object is found."
End If


Set oExportTool = Nothing
```

The following text shows the output from the preceding code example:

```
Default Parameters :
- Perimeter             = ~Bav0cNnAjyQR[Standard for export]
- Format                = MGR
- Propagate             = True
- TransferedObject      = True
- ContinueOnConfidential = False
```

Implementation to JAVA

Constructor Detail

MegaExportTool

`public` **`MegaExportTool`**`(MegaRoot mgRoot)`

This method enables instancing of the ExportTool object.

Method Detail

propagate

```
public boolean propagate()
```

Allows you to specify if object roots must be propagated.

Returns:

- TRUE: To propagate; this is the default value.

- FALSE: To take object roots only.

---

propagate

```
public void propagate(boolean bNewPropagate)
```

Allows you to specify if object roots must be propagated.

Parameters:

`bNewPropagate` -

- TRUE: To propagate ; this is the default value.

- FALSE: To take object roots only.

---

transferedObject

```
public boolean transferedObject()
```

TransferedObject takes account of transfered objects.

Returns:

- TRUE: The transfered objects are part of object roots; this is the default value.

- FALSE: No transfered objects taken.

---

transferedObject

```
public void transferedObject(boolean bNewTransferedObject)
```

TransferedObject takes account of transfered objects.

Parameters:

`bNewTransferedObject` -

- TRUE: The transfered objects are part of object roots; this is the default value.

- FALSE: No transfered objects taken.

---

format

```
public java.lang.String format()
```

Format Choice of the format of an exchange of data.

Returns:

- MGR: ANSI text data exchange; this is the default value.

- XML: XML data exchange.

---

format

```
public void format(java.lang.String NewFormat)
```

Format Choice of the format of an exchange of data.

Parameters:

`NewFormat` -

- MGR: ANSI text data exchange; this is the default value.
- XML: XML data exchange.

---

perimeter

```
public void perimeter(java.lang.Object oPerimeterID)
```

Perimeter is used in Export Processing to extract sets of objects and links to generate the export file. The default perimeter is defined on MetaTool "Standard of Export".

Parameters:

`oPerimeterID` -

New perimeter is used in Export processing.

---

perimeter

```
public java.lang.Object perimeter()
```

Perimeter is used in Export Processing to extract sets of objects and links to generate the export file. The default perimeter is defined on MetaTool "Standard of Export".

Returns:

Perimeter is used in Export processing.

---

continueOnConfidential

```
public boolean continueOnConfidential()
```

Enables specification of whether export processing should be aborted if a confidential object is present in the collection of objects to be exported.

Returns:

- TRUE: Export processing is not aborted; this is the default value.
- FALSE: Export processing is aborted.

---

continueOnConfidential

```
public void continueOnConfidential(boolean bContinueOnConfidential)
```

Enables specification of whether export processing should be aborted if a confidential object is present in the collection of objects to be exported.

Parameters:

`bContinueOnConfidential` -

- TRUE: Export processing is not aborted; this is the default value.
- FALSE: Export processing is aborted.

---

foundConfidential

```
public boolean foundConfidential()
```

This attribute is specified by Export processing to indicate presence of a confidential object in object collection. You cannot modify this attribute.

Returns:

- TRUE: A confidential object is detected.

- FALSE: No confidential object is found.

Export

```
public void Export(MegaObject moObject,
                   java.lang.String strFileName)
```

Export an object in data exchange file.

Parameters:

`moObject` -

This object is root of Export processing.

`strFileName` -

This string is the name of the result file. It is the full name: local or network, directories, short name of the file and Extension.

Export

```
public void Export(MegaCollection mcObjects,
                   java.lang.String strFileName)
```

```
   final String strFileName = "C:\\TEMP\\MegaExport.Log";
```

Export a collection of objects in data exchange file.

Parameters:

`mcObjects` -

This collection of objects is root of Export processing.

`strFileName` -

This string is the name of the result file. It is the full name: local or network, directories, short name of the file and Extension.

Sample

```
// -----------------------------------
// -- ExportTool              V1.0
// -----------------------------------
// Samples of ExportTool object installation

// Creation of 'Component ExportTool' :

MegaExportTool oExportTool;
```

```
 oExportTool = new MegaExportTool(mgRoot);

 // Display parameters :
 String szMes;
 szMes = "Default Parameters : \r\n";
 szMes = szMes + "Propagate : " + oExportTool.propagate() + "\r\n";
 szMes = szMes + "Format : " + oExportTool.format() + "\r\n";
 szMes = szMes + "Propagate : " + oExportTool.transferedObject() + "\r\n";
 szMes = szMes + "continueOnConfidential : " + oExportTool.continueOnConfidential() + "\r\n";
 Object oPerimeterID = oExportTool.perimeter();
 szMes = szMes + "Perimeter : " +
mgRoot.getObjectFromID(oPerimeterID).getProp("~210000000900[Nom]") + "\r\n";

 // -----------------------------------
 // -- Sample 1 :
 // -----------------------------------
 // Export a package with no transfered object
 // By Default : Propagation is Active and Format is 'MGR'
 // -- Constants :
 String strObjectFileName = "C:\\TEMP\\ObjectExport.MGR";
 oExportTool.transferedObject(false);
 // ----
 MegaObject oPackage = mgRoot.getCollection("~h8rEkjZmo400[Paquetage]").item(1);

 if ((oPackage != null) && (oPackage.exists())) {
   szMes = szMes + "Sample1 : Package(" + oPackage.getProp("~210000000900[Nom]") + ")\r\n";
   oExportTool.Export(oPackage, strObjectFileName);
   szMes = szMes + "--> Found Confidential Object : " + oExportTool.foundConfidential() + "\r\n";
 }

 // -----------------------------------
 // -- Sample 2 :
 // -----------------------------------
 // Export all packages with no propagate
 // By Default : Propagation is Active and Format is 'XML'
 // -- Constants :
 String strCollectionFileName = "C:\\TEMP\\CollectionExport.XML";
 oExportTool.transferedObject(true);
 oExportTool.propagate(false);
 oExportTool.format("XML");
 oExportTool.continueOnConfidential(true);
 oExportTool.perimeter("~lCRBhWpi6DF0[Standard for export MEGA product (internal)]");

 // ----
 MegaCollection oPackageColl;
 oPackageColl = mgRoot.getCollection("~h8rEkjZmo400[Paquetage]");

 szMes = szMes + "Sample2 : All Packages\r\n";
 szMes = szMes + "Default Parameters :\r\n";
 szMes = szMes + "Propagate : " + oExportTool.propagate() + "\r\n";
 szMes = szMes + "Format : " + oExportTool.format() + "\r\n";
 szMes = szMes + "Propagate : " + oExportTool.transferedObject() + "\r\n";
 szMes = szMes + "continueOnConfidential : " + oExportTool.continueOnConfidential() + "\r\n";
 oPerimeterID = oExportTool.perimeter();
 szMes = szMes + "Perimeter : " +
mgRoot.getObjectFromID(oPerimeterID).getProp("~210000000900[Nom]") + "\r\n";

 oExportTool.Export(oPackageColl, strCollectionFileName);

 this.writeInFile(strFileName, szMes);
```

Customizing the User Interface

Forms

Forms

1 INTRODUCTION

1.1 What is FORMS?

Forms is a *framework* for design of forms used to exploit MEGA repository data. These forms are stored in the MEGA repository.

They can be used in both Windows Front-End and Web Front-End.

1.2 Accessing forms

Forms are accessed:

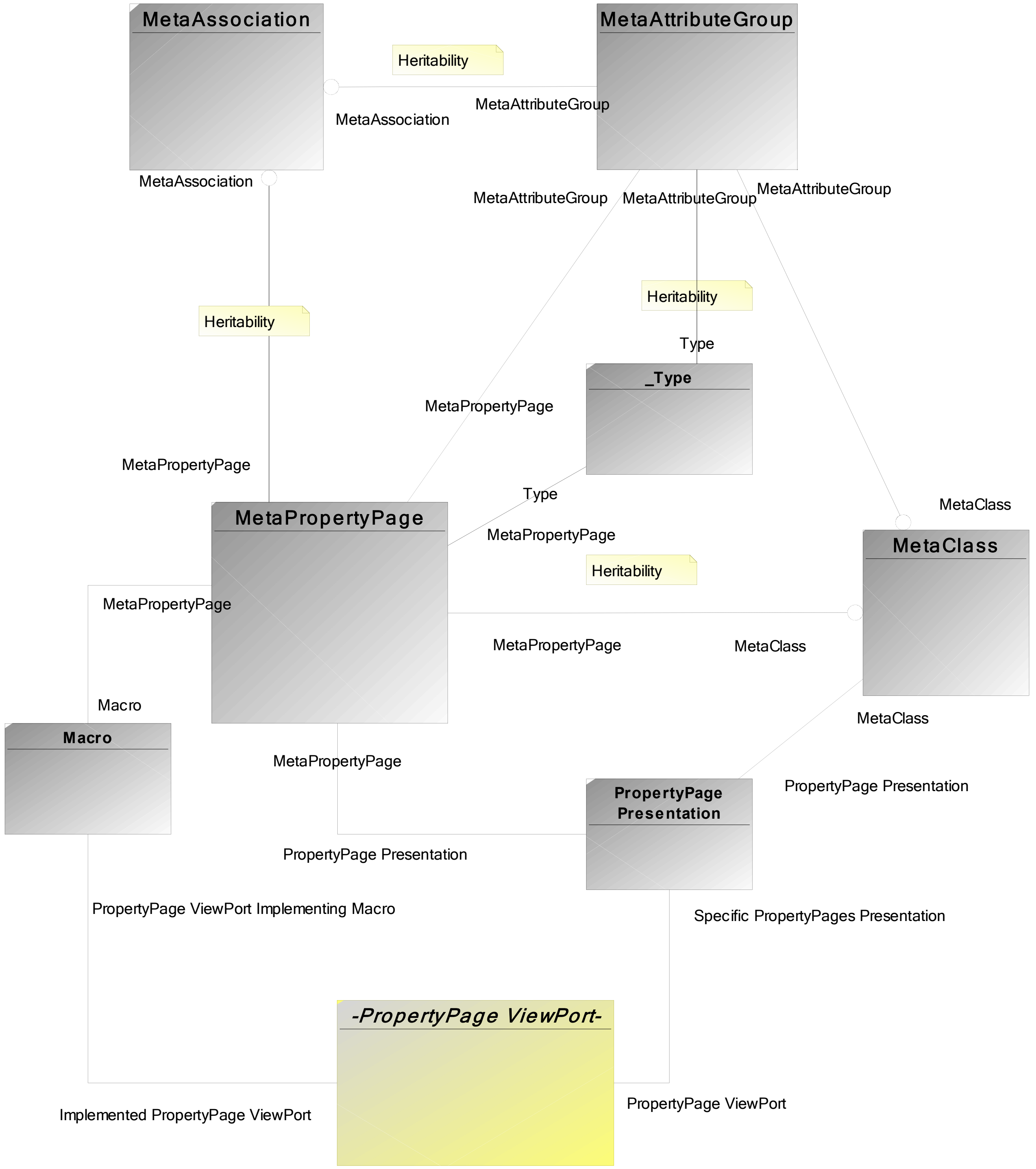
- via the "properties" menu available on MEGA objects
- in Web Front-End using a Mega Parameterized Tool
- by API, using AddPanel and PropertiesDialog functions

1.3 FORMS and wizards

MEGA wizards (modeled by the MetaWizard MetaClass) are made up of an assembly of MetaPropertyPages. Extensions specific to wizards have been integrated in forms to offer improved interaction between wizards and pages.

2 Specification of a form for a MEGA MetaClass

2.1 Metamodel



2.2 Standard form

2.2.1 General principles

From the time of its creation, every *MetaClass* in the MEGA repository has an implicit form without it being necessary for anything to be defined. This implicit file is built as a function of the *MetaAttributes* associated with it. As a function of their type, these attributes are broken down into several properties pages. Other more technical generic pages are added, for example a page enabling translation, and another displaying object history.

Finally, a MetaClass inherits property pages defined on the abstract MetaClasses it inherits, if the "heritability" attribute has been activated.

This standard form can be modified by parameterization. Parameterization can be carried out:

- by reorganizing presentation of the MetaAttributes of the MetaClass: the proposed breakdown may not be suitable for the desired ergonomics. This breakdown can be modified by defining specific groups of attributes on this MetaClass, modeled by the MetaAttributeGroup concept.
- by modifying the visual aspect of the main properties page (the Characteristics page), or of a pages resulting from one of the MetaAttributeGroups mentioned above.
- by inserting specific pages not concerning editing of a MetaAttribute.

2.2.2 Attribute groups and *MetaAttributeGroups*

2.2.2.1 Implicit groups of a Metaclass

Attributes of a MetaClass are broken down generically into the following implicit groups:

<Administration>
This group contains all administration attributes. These MetaAttributes are characterized by activation of the "*In administration tab*" flag of their *extended property*, but certain conventional *MetaAttributes* appear automatically. It is associated with the General type, has order number 1 and is therefore the page that appears first.

<Translation>
This group contains all translated attributes and texts of the *MetaClass*. It is associated with the General type and has order number 70.

<Texts>
This group contains all texts of the *MetaClass* not connected to an explicit *MetaAttributeGroup* . It is associated with the Text type and has order number 32000.

<Characteristics>
This group contains all attributes not assigned to a generic group or a specific group. It is associated with the Characteristics type and has order number 2.

<Extensions>
When the <Characteristics> group has been replaced by a specific group (see below), the <Extensions> group contains all attributes not assigned to a generic group or a specific group. It is not associated with a type and has order number 31990.

2.2.2.2 Explicit groups: MetaAttributeGroups

Generic organization of attributes and the form that results from this may not be suitable for the desired ergonomics. This is particularly the case when:

- there are a large number of MetaAttributes and/or you want to organize these as a function of the subject they cover.
- you want to see texts appear with the attributes and not in the Texts page.
- you want to define a properties page visually, or extend its content to something other than attributes.

To define a group specific to the MetaClass:

1. Create an occurrence of the MetaAttributeGroup.
2. Connect this to the MetaClass.
3. Associate with it the desired MetaAttributes.

In associating a MetaAttributeGroup to a _Type, you specify the tab in which it will appear.

A generic page is created for each MetaAttributeGroup, displaying its content. This page can be redefined by creating a MetaPropertyPage and connecting it to the MetaAttributeGroup.

It is also possible to associate with a MetaAttributeGroup TaggedValues (taken into account if they are also connected to the MetaClass) or MetaAssociationEnds (taken into account if they are legAttributes)

The MetaAttributeGroups defined on abstract MetaClasses are inherited by the sub-MetaClasses, depending on the value of the Heritability attribute.

2.2.2.3 Modification of <Characteristics> group - <Extensions> group

You may want to parameterize the Characteristics page of a *MetaClass*:

- to include attributes not originally included (eg: texts),
- to add elements not directly related to a MetaAttribute of the MetaClass (eg: a list of objects connected to a tree), or
- to specify a specific user interface for each page.

To do this:

1. Create a *MetaAttributeGroup.*
2. Connect the *MetaAttributeGroup* to the *MetaClass.*
3. Connect to it the 'Characteristics' generic *_Type*.

   This association with the *_Type* is sufficient to identify the *MetaAttributeGroup* as substitute for the generic group.

If the list of attributes to be included in the page is not modified, it is not necessary to connect the *MetaAttribute* to this *MetaAttributeGroup*: in this case, all the *MetaAttributes* that the corresponding generic group would have included are connected "virtually".

However, if at least one *MetaAttribute* is associated with this *MetaAttributeGroup*, the list is considered as complete: any *MetaAttribute* not included in a generic or specific *MetaAttributeGroup* will be added in the <Extensions> generic group. If you do not want it to appear and if implementation of the Characteristics page allows, it is sufficient to connect the *MetaAttributes* included in the Extensions page to the Characteristics *MetaAttributeGroup*.

2.2.3 MetaClass properties pages

The list of pages displayed in a properties dialog box is determined specifically for each MetaClass.

Added to deduced pages of implicit and explicit groups defined for the MetaClass are:

- Standard pages of links or menu trees.
- Specific pages directly associated with the MetaClass.

2.2.3.1 Properties page tab and order

At the time of their collection, pages are classified as a function of their type, which enables creation of the list of tabs to be displayed in the form. The order of pages serves not only to sort pages within a tab, but also to sort the tabs themselves. The order number of a tab is defined by the order number of its first page. The General tab has order number 1, the lowest number possible, and which is the number of the "Administration" page. When a tab contains only one page, this page appears directly in the position of the tab, unless explicitly specified otherwise in the corresponding *_Type*.

When the page is deduced from a group, it inherits the type and order number defined for this group as well as its name: the *_GUIName MetaAttribute* of the *MetaAttributeGroup* is used for this, or by default its name.

A type and order number are defined for each non-dependent standard page of a group. These cannot be modified.

With each page explicitly defined via an occurrence of _PropertyPage, you can associate a specific type that will enable definition of its tab. Generic implementations of pages enable redefinition of the order number.

2.2.3.2 Standard pages relating to MetaAssociations

Parameterization of certain operators related to the MetaClass enable control of display of the following pages, which display the associated objects in tree form using the MetaAssociations concerned:

- Correspondences page: displays a tree parameterized by the _Operator "Correlate". This page is added when a MetaAssociation with a behavior relating to this operator other than 'Abort' has been found on the MetaClass. It is General type and has order number 50.
- Impacts page: displays a tree parameterized by the _Operator "Impact". This page is added when a MetaAssociation with a behavior relating to this operator other than 'Abort' has been found on the MetaClass. It is General type and has order number 48.
- Complements page: displays a tree parameterized by the _Operator "Complement". This page is added when a MetaAssociation with a behavior relating to this operator other than 'Abort' has been found on the MetaClass. This page has no type and therefore appears as the main tab. It has order number 30000. Like all pages relating to an *_Operator*, the page name is by default that of the operator. It is however possible to define another name for this tab, by connecting the operator to the *MetaClass* and by specifying in the desired language the *_GUIName* attribute present on this link.

2.2.3.3 Customizing a page relating to a *MetaAttributeGroup*

The mere presence of a *MetaAttributeGroup* is sufficient to cause appearance of a specific properties page for a *MetaClass*. This page may not however be suitable:

- You want to be able to hide the properties page according to criteria specific to the occurrence.
- Presentation of attributes may not be satisfactory; you want to modify look, presentation order, title, appearance conditions, etc.
- You want to add to this page elements not derived from an element of the *MetaAttributeGroup* (for example a tree, schedule, list, attribute from another connected occurrence, etc.)

Page specific implementation has been provided by MEGA to satisfactorily manage this *MetaAttributeGroup*.

The page corresponding to this MetaAttributeGroup can be customized. This customization will be provided by an occurrence of *_PropertyPage* connected to the *MetaAttributeGroup*.

2.2.3.4 Adding a page not dependent on a *MetaAttributeGroup*

One or several pages displaying no element relating to a *MetaAttributeGroup* can be added to the properties dialog box of a *MetaClass*. This customization will be provided by an occurrence of *_PropertyPage* directly connected to the MetaClass. Customization possibilities offered by MEGA for such pages are covered later.

2.2.4 Property page filtering according to product

The list of properties pages, as well as their content depends on products installed and the view the current user has of the metamodel.

Implicit property pages cannot be specifically filtered. However, they do not display attributes hidden to the user; and an empty generic page is not inserted in the dialog box.

Depending on the case, appearance of a page is also conditioned by filtering of its corresponding *_PropertyPage* as well as by that of its *MetaAttributeGroup*: if one of these two concepts is filtered, the page does not appear.

2.2.5 Pages relating to MetaAttributes of MetaAssociations

MetaAttributes connected to *MetaAssociations* are visible either:

- from the properties dialog box of an object seen from its link, or
- in a form specific to the link: this dialog box is presented in Diagram when you want to read properties of a link, and it can also be accessed by APIs.

2.2.5.1 Form of a link

Standard behavior classifies attributes in generic groups following the same principle as for *MetaClasses*:

- <Administration> group for administration MetaAttributes.
- <Location> group for translated MetaAttributes of the MetaAssociation.
- <Text> group for MetaAttributes of Text type.
- <Characteristics> group for other *MetaAttributes*.

You can define MetaAttributeGroups on MetaAssociations.

You can define MetaPropertyPages on MetaAssociations.

2.2.5.2 Form of an object seen from a link

In the absence of parameterization, the *MetaAttributes* of the *MetaAssociation* by which the object is seen are presented in the properties dialog box. Standard processing distributes *MetaAttributes* of the *MetaAssociation* conforming to their MetaAttributeGroup. They are presented at the end of the page, preceded by a bar containing the name of the opposite *MetaAssociationEnd*.

Administration attributes, as well as the Order attribute, are displayed in the Administration page, while text type attributes are inserted in the Texts page. Other attributes are displayed in the Characteristics page.

This processing is carried out by page generic implementations only, or specific implementations processing the problem explicitly.

Specific MetaPropertyPages defined on the MetaAssociation will also be seen in this form if the heritability attribute has been activated.

2.3 Form defined by a ViewPort

The form associated with a MetaClass can be redefined in a precise use context. This use context corresponds to the *PropertyPage ViewPort* context, implemented by the *Desktop* and *Web Site Template* MetaClasses.

This redefinition is made by creating a PropertyPagePresentation associated with the ViewPort and the MetaClass. You can then associate with the PropertyPagePresentation the MetaPropertyPages you want to see in the form of the MetaClass for this ViewPort. In addition, the PropertyPage Definition attribute enables specification of whether the PropertyPagePresentation replaces the standard form – in which case the form will comprise only those pages associated with the presentation – or whether it is an addition – in which case the pages associated with the presentation will be added to the standard form.

A Macro can be associated with the ViewPort. In this macro you can implement page filtering or page addition.

You can explicitly invoke the form of an object defined by a PropertyPageViewPort using the PropertiesDialog function:

```
MegaObject.PropertiesDialog "ViewPort=<ViewportID>]"
```

where **ViewPortID** is a field containing the absolute identifier of the ViewPort.

2.3.1 Implementing page filtering

This implementation hides standard form pages that you do not want to see in the specific form. It has the following prototype:

```
Function FilterPage(Context as MegaPageLoader,PageID,TypeID) As Boolean
```

PageID is the identifier of the Page to be filtered (typically a MetaPropertyPage identifier or MetaAttributeGroup identifier, if the page is implicit)

TypeID is the identifier of the page Type, which defines the folder in which it will appear.

Context carries information on the object being loaded and the handling tools.

This function returns True if the page is visible, False if it should be hidden.

The **MegaPageLoader** component has the following methods and properties:

.Object as MegaObject: object to which the form being loaded relates (read-only)

.Title as String: title of page to be filtered (read/write)

.ViewPortID: identifier of viewport being loaded (read)

.Holder as MegaPageHolder obtains the Form component being loaded (see below)

.MacroCLSID: macro implementing the page to be filtered This enables discrimination of conventional pages if necessary

.SameID(id1 [, id2]) as Boolean: with one parameter, compare the supplied identifier with that of the properties page, with two parameters, compare the identifiers

2.3.2 Implementing a Page Loader

When you want to add specific or dynamically defined pages to a form, you must implement a method using the **InvokeOnObject** function.

This component also enables redefinition of certain form elements and should implement:

```
Sub InvokeOnObject(Source As MegaObject,Holder As MegaPageHolder,InitString as
String)
```

Source: contains the object to which the form relates

Holder: component that manages the form, to which pages can be added

InitString: initialization string, empty in the case of a call from a ViewPort

The **MegaPageHolder** component has the following functions and methods:

.Caption as String (read/write) form title

.Object as MegaObject: object to which the form relates (can be empty if you load an associative object)

.Relationship as MegaObject: associative object to which the form relates, exclusive with .Object

.Description as MegaObject: description of the object to be displayed (can be empty)

.Insert(PageID[,PageName as String, PageType as Object, Obj as MegaObject, Parameterization As String, Props as MegaCollection, Options As Integer])

PageID: identifier of the MetaPropertyPage to be displayed. If the page is calculated, this can be any identifier, but different from other identifiers of the form page.

The following elements are optional:

PageName: name of the page (mandatory if the page is not a MetaPropertyPage)

PageType: _type of the page, enabling definition of the tab in which it will appear

Object: MegaObject to which the page relates if it is not the same as that of the form

Parameterization: when the page is calculated, contains the parameterization text enabling its supply. This text corresponds exactly to content of a _parameterization text of a MetaPropertyPage

Props as MegaCollection: when the page is calculated and displays properties in a standard way, this collection can replace Parameterization. It must contain a list of properties conforming to a description.

Options: bit field containing the following details:

```
#define PAGEOPTION_PROPPAGESTANDARD 0x1000
#define PAGEOPTION_PROPPAGEADMIN    0x2000
#define PAGEOPTION_PROPPAGETEXT     0x4000
#define PAGEOPTION_METAPROPPAGE     0x8000
```

PAGEOPTION_METAPROPPAGE is implicit if the Caption is not defined. Otherwise this flag indicates that the PageID corresponds to a MetaPropertyPage.

.TabStyle as String: (read/write) tab display mode (Windows Front-End only). Possible values are:

"SideList": tabs are displayed vertically on the left of the form

"ComboBox": tabs are displayed at the top in a drop-down list

"BottomTabs": tabs are displayed at the bottom

"ScrollTabs": tabs are displayed at the top in a single line

"Hidden": When there is only one page in the form

.IsStatusBarHidden as Boolean: (read/write) display of status bar

.IsCaptionHidden as Boolean: (read/write) display of caption

2.3.3 Overloading standard tabs

Standard tabs are presented with a name, a certain order and with a predefined appearance that can be modified in the context of a ViewPort. To do this, use the (ViewPort/_Type) link, and more specifically the "Parameterization Link" parameterization text in which you can redefine elements of this tab. To do this, use the Folder section of this parameterization text:

```
[Folder]
Permanent =  { YES | NO }
Selector = { TabStyle }
Order = nnn
Name = { name | <Field> }
```

Permanent YES indicates that the tab is always displayed, even if it contains only one page. NO indicates that when the tab contains only one page, we display this directly

Selector (Windows Front-End) enables redefinition of the appearance of the tab (see **TabStyle** function described above)

Order enables redefinition of the tab order number. In this case the order number of its component pages is ignored.

Name enables modification of tab name. If you want to translate this name, you must use a <Field> pointing to a system repository object of which you display the short name.

Note: You can modify standard appearance of a tab by applying this parameterization to the *_Parameterization* text of the *_Type* itself.

2.3.4 Overloading form behavior

To define generic behaviors of elements of forms:

- ➔ Parameterize the MetaViewPort by means of its *_Parameterization text*. You can particularly parameterize the default behavior of listViews (cf 3.5.3.1.2.10).

This enables reusing forms in different contexts.

```
[ListViewDefault]
    MultiSelection=1
    ModelessPropPage=1
    PropPageAffinity=<AffinityID>
[ListViewExport]
    <ExportName>=Item(<ButtonID>),Param(<extraParameter>),Name(<Name>),Picture=<P
ictureID>,Method=<MethodID>
```

MultiSelection indicates that listViews are by default in multiselection mode.

ModelessPropPage indicates that « Properties » command invoked from the listViews s the corresponding form in docked mode (default behavior is Popup mode). In that context, you can define the affinity to be used for these forms – that means the container in which they will be displayed – by means of **PropPageAffinity.**

ListViewExport section enables defining default export buttons in the listViews (see 3.5.3.1.2.5)

2.4 Generated form

A form can be loaded from a Macro; this Macro dynamically defines the list of pages to be added in a form, in the same way as for the macro of a ViewPort described above, and implements the method:

```
InvokeOnObject(Source As MegaObject,Holder As MegaPageHolder,InitString as String)
```

In this case InitString can contain an initialization string enabling transmission of a context to the macro.

This form is invoked using the PropertiesDialog function:

```
MegaObject.PropertiesDialog "Loader=<macroID>[initstring]"
```

MacroID being a MegaField corresponding to the loading macro.

This field can be followed by initString characters that will be transmitted to the macro.

If the form to be invoked contains only one page, and this page is a MetaPropertyPage, it is not necessary to implement a Macro, and you can simply call:

```
MegaObject.PropertiesDialog "PropPage=<pageID>"
```

2.5 Page general parameterization

Components supplied by MEGA that can be used in customization are the following Macros:

_PropertyPageStandard: standard characteristic page. This implementation should be used to customize the Characteristics page

_PropertyPageExtension: standard page derived from a *MetaAttributeGroup*. This implementation should be used to customize the *MetaAttributeGroup* page.

_PropertyPageComment: standard text page. This implementation should be used to customize the Texts page.

_PropertyPageLink: page displaying a list of objects associated by a *MetaAssociation* specific to the dialog box object. This implementation is depreciated, the functions offered by _PropertyPageExtension to manage lists - Control(ListView) - being considerably more comprehensive; in addition, certain generic behaviors (such as export buttons) have not been implemented on this type of list.

_PropertyPageTree: page displaying a tree of which the dialog box object is the root. Standard pages displaying a tree. This implementation is depreciated, the functions offered by _PropertyPageExtension – Control(TreeView) - to manage lists being considerably more comprehensive.

_PropertyPageTreeViewOption: page displaying an options box. Used in particular in the Procedure properties dialog box to manage ISOxxx tabs. **This implementation is not compatible with HOPEX**.

The following parameterizations are to be inserted in the *_Parameterization* text of the *_PropertyPage*

2.5.1 Page name and order number

The name of the tab representing the page is the _GUIName of the _PropertyPage. If this attribute is not specified, the _GUIName of the MetaAttributeGroup that it displays is used, if this exists. If not, the Name of the _PropertyPage is used. Text pages implemented by **_PropertyPageComment** have a specific name logic: if only one text is displayed (for example Comment), this text name replaces the page name.

The page order number is defined by the option:

```
[Page]
  Order = <Nombre>
```

2.5.2 Page active by default

It is possible to specify a particular page as active by default in the dialog box.

```
[Page]
```

```
IsDefActive = { 0 | 1 }
```

Note: This parameterization applies only to initial display of the dialog box, subsequent displays taking account of the last active tab selected by the user.

If several pages are specified as being active by default, the page effectively active will be "one of these… ".

2.5.3 Filtering a page as a function of the object

It is possible to control appearance of a page as a function of a condition relating to the form object. To do this, the following option should be specified:

```
[Filter]
  Condition = <Bool-Expression>
```

The condition relates to the object. Texts relating to MetaAttributes, TaggedValues or MetaAssociations of this object can be included. Comparison operators =, <>, <=, >=, < and >, or/and boolean operations and brackets are supported.

Attributes and occurrences can be included in hexaidabs form prefixed by # or in the form of fields.

At a binary comparison, the first element (if it is a field) is interpreted as the value of a *MetaAttribute* or a *TaggedValue* of the current object.

Comparison can be numerical or alphanumerical as a function of the variables used. If the first element is an attribute, the test type is determined by the attribute type. It should be remembered that 1000 is larger than 2 (numerical comparison) while '1000' is smaller than '2' (alphanumeric comparison).

Condition syntax description:

- Bold non-italic (and red) elements are keywords or separators of the language
- Elements between square brackets are optional
- Elements between brackets are optional and can be repeated
- Elements between curly brackets are alternative choices

```
<Bool-Expression> :: <And-Expression> ( OR <And-Expression> )
<And-Expression> :: <Condition> ( AND <Condition> )
<Condition> :: [ NOT ] {
( <Bool-Expression> ) | < Comparison> |
ItemExist( <Field> , <Field> ) |
Available( <Field> ) |
ContainString( <Field> , <Field> ) |
SeenFrom( <Field> ) |
TrueForOne( <Field> , <Bool-Expression> ) |
TrueForEach( <Field> , <Bool-Expression> ) |
IsAdminMode | IsConfidentialityCustomized | True | False | IsWindowsMode |
IsPermission( { Update | Delete | Change | Unlink | Lock } ) |
IsMetaPermission( <Field> , { Update | Delete | Change | Unlink | Create | Read }
) |
IsClass( <Field> ) |
IsType( <Field> ) |
IsChildAvailable( <Field> ) |
IsToolAvailable( <Field> ) |
IsConcreteType( <Field> ) |
ApplyTest( <Field> ) |
AccessLevel( <Constant> )
}
<Comparison> ::
{ <Field> | <Fonction> | <Constant }
    { = | <> | <= | >= | < | > }
{ <Field> | <Fonction> | <Constante }
<Function> :: ItemCount( <Field> ) |
Number( <Numerical-Expression> ) |
SessionValue( { CurrentLanguage | DataLanguage | SystemLanguage | User } )
<Numerical-Expression> :: numerical expression

<Constant> :: ' ( <Letter> ) > ' | "( <Letter> ) " |
[ - ] <Number> ( <Number> )
<Letter> :: any ANSI character other than the separation character
<Number> :: { 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 }
```

2.5.3.1 Boolean functions

ItemExist(Collection,Occurrence): true if the occurrence cited belongs to the collection.

Available(**Object**): true if the object is visible depending on products available on the key and filters defined for the user.

TrueForEach(**AssociationEnd,Condition**): true if the condition is true for all objects accessed by the collection (the condition applies to the accessed object)

TrueForOne(**AssociationEnd,Condition**): true if the condition is true for at least one of the objects accessed by the collection (the condition applies to the accessed object)

ContainString(*Champ*,Test): true if the test string is contained in the value represented by the field.

SeenFrom(Collection): true if the form object is seen from the collection

IsAdminMode: true if you are in the administration module

IsConfidentialityCustomized: true if confidentiality of the current environment has been customized

True: always true

False: always false

IsWindowsMode: true if you are in Windows Front-End

IsPermission({Update|Delete|Change|Unlink|Lock}): tests permission required on the object

IsMetaPermission(Collection,{Update|Delete|Change|Unlink|Create|Read}): tests permission required on the collection seen from the object

IsClass(MetaClass): true if the object is of the class

IsType(MetaClass): true if the object inherits the class

IsChildAvailable(Collection): true if the collection is visible from the object for the connected user

IsToolAvailable(Tool): true if the tool is visible from the object for the connected user

IsConcreteType(MetaClass): true if the object is compatible with the class

ApplyTest(MetaTest): executes the metatest on the object and returns the results

AccessLevel(Constant): tests visibility level of the current user. The first letter of the constant corresponds to the level tested: **B**eginner, **S**tandard, **E**xpert …

2.5.3.2 Other functions:

ItemCount(Collection): returns collection object number

Number(Expression): converts expression to number. Useful if you want to force numerical comparison between two values

SessionValue(value): returns absolute identifier of the requested session value, which can be:

CurrentLanguage	: session current language
DataLanguage	: session repository language
SystemLanguage	: session system repository language
User	: user connected to the session

3 FORMS PROPERTIES PAGE DESIGN

3.1 Basic principles and initialization

A Forms MetaPropertyPage is implemented by one of the two following macros:

_PropertyPageStandard: this implementation is used in "Characteristics" pages.

_PropertyPageExtension: this implementation is used in other cases.

These implementations use the _parameterization text defined on the MetaPropertyPage, and more specifically the [Template] paragraph.

They also use the MetaAttributeGroup associated with the MetaPropertyPage: the page is automatically populated by the properties defined in this MetaAttributeGroup

In addition, the _MetaPropertyPageStandard will automatically add to the form:

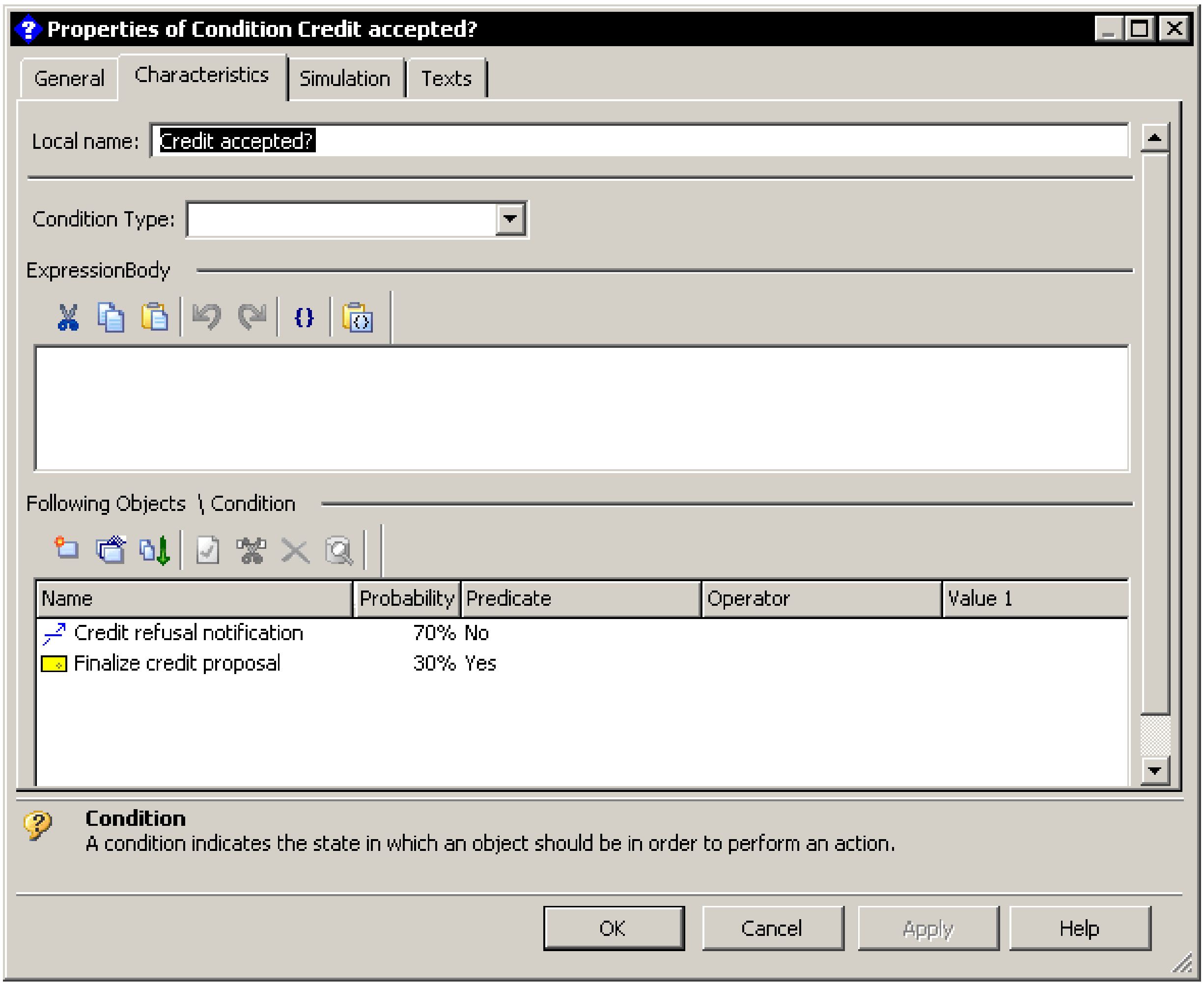
- the object name
- its owner
- the MetaAttributes defined in the accessed MetaAssociation (if the object is seen from a MetaAssociation) and not associated with a specific MetaAttributeGroup of the MetaAssociation.

These implementations also use other sections of the _parameterization text, enabling definition of page behavior.

The name of the properties page is the *GUIName* of the *_PropertyPage*, except for the Characteristics page of which the name is predefined. If not specified , the *GUIName* of *MetaAttributeGroup* is used.

The following example indicates programming of the condition page shown below.

```
[Template]

AttGroup=Group(Bar),Pos(Top)

FObjGroup=Group(Bar),Pos(Bottom),Name(Following      objects)

ExprGroup=Group(Bar),Pos(Middle),Name(ExpressionBody)

FObj=Map(Following      objects)

TypeAtt=Item(Condition      Type),In(AttGroup)

FObjList=Item(Following
objects),Contains(FObj),In(FObjGroup),Control(ListView),Title(No)
ExpressionAtt=Item(ExpressionBody),In(ExprGroup),Control(Text),Title(No)
```


Properties of Condition Credit accepted?
General
Characteristics
Simulation
Texts
ExpressionBody
Following Objects \ Condition
OK
Cancel
Apply
Help

Local name: Credit accepted?

Condition Type:

Name	Probability	Predicate	Operator	Value 1
Credit refusal notification	70%	No		
Finalize credit proposal	30%	Yes		

Condition
A condition indicates the state in which an object should be in order to perform an action.

Page minimum size can be defined in another section of the parameterization text:

```
[Page]
MinWidth = <width>
MinHeight = <height>
<width> and <height> are specified in DialogUnits.
```

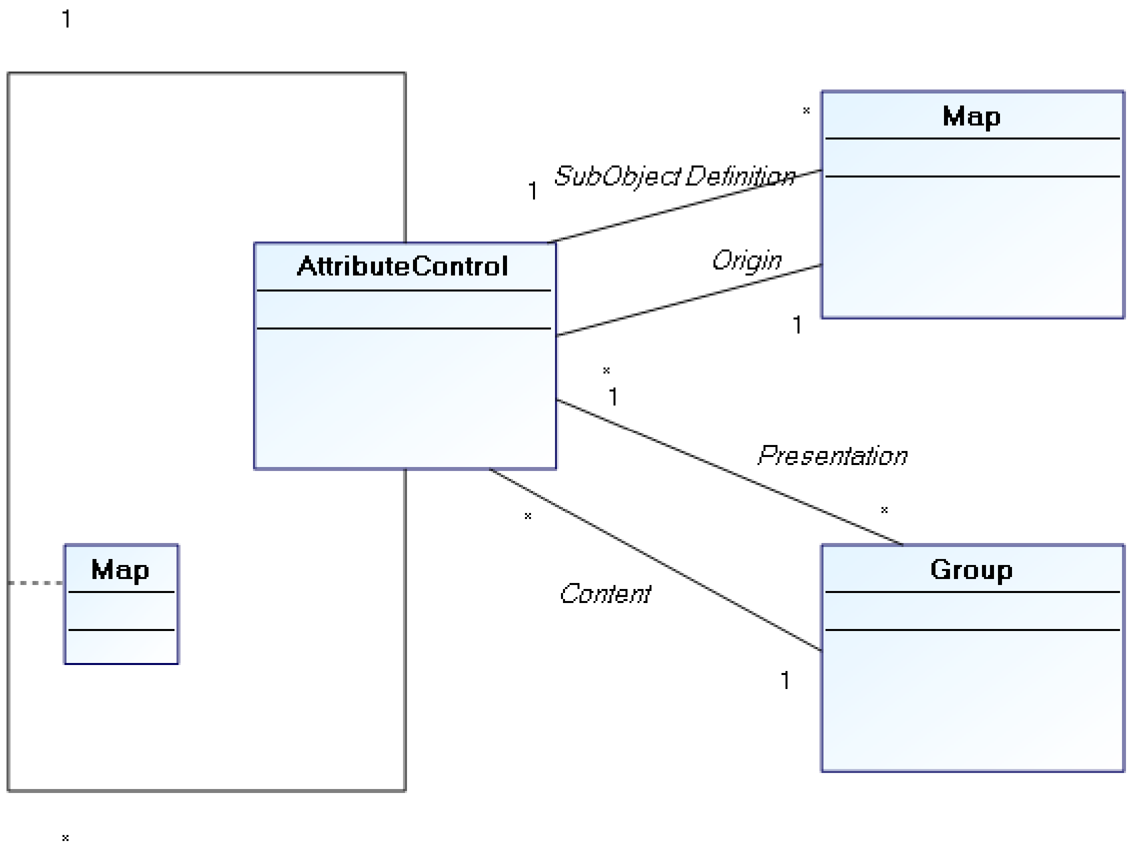
This parameterization can be essential if you want to define specific deformations of elements at page resizing: in this case, the minimum page size is used as the basic size in the parameterization.

For information, DialogUnits is a unit independent of font size. In this unit, a medium character of the font used occupies a rectangle 4(width) x 8(height). To obtain the effective size of the page, you need to know the medium character size of the font used to display the form.

Note: As MinWidth and MinHeight are different in Windows and Web Front-Ends, adjustments must be performed accordingly.

➔ For information regarding conversions, refer to Microsoft documentation: https://learn.microsoft.com/en-us/previous-versions/windows/desktop/bb226789(v=vs.85)

3.2 Properties page logic model



3.2.1 AttributeControl

A page is defined as being an AttributeControl hierarchy. In the model of the above object, the page is itself an AttributeControl.

An AttributeControl is therefore a page element. It relates to a MEGA object and is associated with a metamodel concept representing information on this object.

There are three main AttributeControl types:

- **Implicit AttributeControls**

  These are automatically associated with MetaAttributes, TaggedValue (or more generally AbstractProperties) and MetaAssociationEnd _LegAttributes, depending on their type and format. They therefore reference properties directly defined on the object.

  Implicit AttributeControls are suitable in the vast majority of cases to adequately display object properties, as well as most extensions (AbstractProperties or MetaAssociationEnd). These extensions are not directly associated with the object, but are defined in the System repository and must have a value for the object. The keyword XRef(True) enables indication of an element as extension, and directs the template analyzer to the correct AttributeControl type to be displayed.

- **Explicit AttributeControls**

  These propose specific data entries on object properties, or display of calculated data obtained from the object and defined by a metamodel concept (for example a matrix, an HTML formatter or a MetaTree), or display of presentation elements (titles, comments) referencing system repository elements without direct reference to the displayed occurrence (for example codeTemplates or resources).

- **Composite AttributeControls**

  These are explicit AttributeControls of which definition is complex and included in the template. This definition can reference system repository objects, which will be associated with the composite element by a composition map. For example, to define a ListView, you may explicitly define its columns, which are mainly MetaAttributes: The latter will be cited as the composition map element associated with the ListView.

3.2.2 Maps

Maps are used to logically group AttributeControls. They are associated with Page type AttributeControls. There are two types of map:

- object maps
- composition maps

3.2.2.1 Object maps

These enable redefinition of the object associated with an AttributeControl. In this way it is possible to display in the same form properties from several distinct objects. These maps are explicit elements of page type AttributeControls, indicating the list of objects to which the page relates.

A map of this type points to a specific object, and does this in several ways:

Pilot maps: these are commanded by a composite AttributeControl (for example a ListView or a TreevVew). In this case the Pilot(AttCtlName) keyword associated with the map indicates the AttributeControl that will pilot it. The object of the map therefore corresponds with the object selected in the piloting control. When the selection changes, all AttributeControls of the page are reinitialized with the new object of the selection. If no object is selected, map elements are deactivated. In a map of this type, the identifier used in the Map(ident) keyword is not used.

Unique collection maps: these maps have a collection as identifier. In this case the first element of the collection is considered as the object of the map. If the collection is empty, map elements are deactivated.

Maps on object: these maps are for calculated pages, since they reference an explicit object of the repository, by keyword Child(childID). The object pointed by the map must be connected to the object of the page by the collection identified in the Map, and correspond to the absolute identifier supplied.

Maps on root: identified by the keyword Root(Yes) , these maps point to the repository root, and are therefore independent of the displayed object.

Context maps: these maps enable use of connections to tools displaying pages. For wizards, the "Context" name implicit map points to the collaborative object of the wizard.

3.2.2.2 Composition maps

Composition maps enable definition of content of a composite AttributeControl: for example, columns of a ListView or branches of a TreeView are defined asAttributeControls, defined in a composition map. This map will then be associated with the MetaAttributeComposite. These maps do not have a real existence in the object model of the form, since they merely participate in definition of the hierarchy of the AttributeControls.

3.2.3 Groups

Groups are used to define appearance of a form. They are associated with Page type AttributeControls.
Groups defined for the page are assembled vertically in a defined order. A certain number of groups are implicit and are created automatically when Attributes are included.
The page analyzer first lists the composition of the *MetaAttributeGroup* (if this exists). We then search in the configuration for any possible elements that could overload it. If there is no overload, or if the overload does not specify a group, these attributes are arranged in standard groups as follows:
Naming groups (short name, long name) are associated by default with the **NameGroup** implicit group as follows:

- Extensions are associated with the **ExtensionGroup** implicit group
- Texts are arranged in a control associated with the text group
- Attributes derived from the association are associated with the **LinkGroup** implicit group
- Other attributes are associated with the standard group.

NameGroup, **ExtensionsGroup** and **LinkGroup** groups can be used in the parameterization: you can then associate specific elements.
Groups are positioned in the page in the following way, **Pos(xxx)** being a syntactic element enabling definition of the general position of a group. It should be noted that Groups of the same position are ordered alphabetically according to name.

Properties page group positioning

```
NameGroup
PosGroups (Top)
StandardGroup
PosGroups (Middle)
ExtensionsGroup
PosGroups(Extension)
LinkGroup
PosGroups (Bottom)
TextGroups
```

The text group is always positioned at the end; only the last zone displayed can be resized vertically, and texts are the most likely elements for resizing. However, if you want to display a ListView that can be resized, it can be positioned alone in a Group in 'Bottom' position. For example, this is what is proposed by the condition page above. It explicitly positions the *ExpressionBody* text in another group so the ListView will effectively be the last zone displayed.

3.3 Specifying templates

3.3.1 Syntax

Page programming is carried out in the *_Parameterization* text, which is a configuration text: it must therefore conform to configuration text compatible syntax. The `[Template]` paragraph of this text is used.

All elements included in this paragraph should be defined in a single line.

Each element is named to conform to configuration text syntax (name = definition). The name is never included in the properties page, but is used to order groups.

It is essential that the order of parameters of each element be respected.

Syntax of elements is detailed below. Elements in bold (red) are keywords; optional parameters are between curly brackets. Alternative elements are between curly brackets separated by vertical bars

For certain elements it is possible to specify positions and sizes. Characteristics should always be expressed in *Dialog Units*.

Elements recognized by this syntax are:

3.3.1.1 Conditions

Conditions enable conditioning of the appearance of groups or of elements. The condition relates to the properties dialog box object occurrence.

Condition relating to the form object:

```
<Name> = Condition(<Bool-Expression>)
```

Condition related to the object pointed by a map:

```
<Name> = ConditionFrom(<MapName>[:{True|False}],<Bool-Expression>)
```

If the map does not point to any object, the Condition is evaluated as False, unless otherwise specified after the colon.

Syntax of conditions is described in section 2.5.3.

3.3.1.2 Groups

These enable positioning of elements with respect to each other. In addition, it is possible to condition appearance or deactivation of the elements included.

Elements appearing in the same group are canonically presented in their order of declaration. It is however possible to include them in the order defined by the metamodel using the MetaClass/MetaAttribute, MetaClass/TaggedValues and MetaClass/MetaOppositeAssociationEnd links using the following option:

```
[Page]
NoGroupOrder = 1
```

The groups are declared in the template using the following syntax:

```
<Name> = Group(<Aspect>)[,Pos(<Position>)][,Name(<Field>)][,At(<column>[,<line>])]
[,Size(<width>,<height>)][{,HiddenOn(<ConditionName>)|,DisabledOn(<ConditionName>)}]
[,Initially(<ConditionName>)]
```

<Aspect> :: {Bar|Frame|VisibleBar|Hidden }: Characterizes physical representation of the group - single bar at top, frame around attributes or nothing. **VisibleBar** indicates that the bar is visible, even if the group is the first on the page. Groups of **Frame** can be collapsed in Web Front-End.

<Position> :: {Top|Bottom|Extensions|Middle}: Characterizes overall positioning of group in page. Default is **Middle**.

<Field>: Field relating to any MEGA occurrence of which the name, preferably translatable, will be displayed in the page as the group name. You can also include a simple character chain, which will be non-translatable.

<column>: Numerical value in Dialog Units indicating horizontal shift of elements included in the group. This data can be included alone.

<line>: Numerical value in Dialog Units indicating vertical position of the top of the group.

<width> <height>: Group height and width. As for <line>, these parameters are optional and need only be used when the specifier wishes to precisely control the position of a group. If not specified, group size is calculated as a function of the cumulative size of groups included, and group position as a function of other groups present in the page.

<ConditionName>: Name of a previously defined condition. If the condition is not fulfilled, the group and its entire content are either masked (**HiddenOn**), or deactivated (**DisabledOn**). The keyword **Always** can be used instead of a condition. Applied to the keyword **Initially**, the condition indicates the initial appearance of a collapsible group (this is only effective in Web Front-End).

3.3.1.3 Maps

```
<Name> = Map(<Field>){,Root(Yes)|,Cookie(<Field>)|,Child(<Field>)}
[,Visibility(<Visibility>)][,Advisor(Yes)][,Pilot(<ItemName>)]
<Field> : field referencing a MEGA object
```

<Visibility> : {Always|IfExist|IfFrom}: Enables conditioning of display of Map elements. Default is **Always**, and Map elements are then systematically displayed (they are disabled if the connected object is not found). If not specified, the elements obtained defined from this Map are masked if the connected object does not exist (**IfExist** case) or if the object cannot be seen from the MetaAssociationEnd opposite the specified MetaAssociationEnd (**IfFrom** case). In this case the Map points to the parent object.

For composition maps, no parameterization (including <Field> of the map) is taken into account. These maps serve only to group AttributeControls defined for a composite AttributeControl, and it is the latter that uses the elements associated with it as required.

For the other maps, <Field> specified after the map keyword represents the Collection enabling access to the object pointed by the map from the form object. This collection is therefore generally a *MetaAssociationEnd*, a *Query* or any form of *Abstract Collection* accessible from this object. The object pointed is determined as for the following API function:

```
obj.GetCollection("<Field>").Item(1)
```

When the Collection contains several elements, the result is therefore unspecified, except if the keyword **Child** indicates the occurrence to which the Map should point.

In this case, the object pointed is determined as for the following API function:

```
obj.GetCollection("<MapField>").Item("<ChildField>")
```

This use is however generally reserved for pages generated dynamically, since such an identifier is rarely available at specification.

The keyword **Advisor(Yes)** enables definition of a Map that is not directly use to display elements but that causes the page refresh when a modification is detected in the collection with which it is associated.

The keyword **Refresh(Yes)** indicates that the collection must be reselected at page refresh. It must be used when the collection associated with the Map is a request or a calculated association, and when its content might change while the form is displayed.

The keyword **Pilot(<ItemName>)** enables to indicate the form element, which will define the object pointed by the map.
When an object is selected in the element, this notification is processed by the map that modifies, as appropriate, its pointed object. Only elements that enable selection of an element can be pilots, either listViews, treeViews, or Metatrees.

Maps also enable access to objects not directly associated with the form object:

The keyword **Root(Yes)** indicates that the map points to the repository root:

```
obj.GetCollection("<Field>").getRoot
```

The keyword Cookie enables referencing of a context object. By compatibility, this keyword enables access to the collaborative object of a wizard (`Context.cookieObject.` In this case the Cookie corresponds to the identifier of the *MetaClass* MetaWizard (~DutIllKV5X40[MetaWizard]), and the identifier of the Map corresponds to that of the collaboration of the cookie.

This access mode has however been replaced by the **Context** implicit map, which enables access to this object without needing to know the identifier of the collaboration.

The keyword **Visibility(**<Field>**)** enables definition of a Map pointing to the parent object of the form object, when the latter is obtained from a collection. In this case <Field> must correspond to the identifier of the accessed MetaAssociationEnd.

The **FromLeg** implicit map indicates the associative object of the form object; when the latter is obtained from a Collection (`obj.getRelationship`). By using it you can include MetaAssociation generic attributes in the page.

By compatibility, the keyword **LegObject** is synonymous with **Map**.

3.3.1.4 AttributeControls

AttributeControls enable definition of page content, or composite elements of this page. Within the same Map (keyword **From**), all AttributeControls must have distinct <Field> identifiers, since it is not desirable to display the same element in the same form several times (at update in the two elements, we cannot determine which value will finally be used). If necessary, the keyword **Duplicate(Yes)** enables explicit specification of a duplicate.

When the AttributeControl is not defined in a map, either a composite or calculated element will appear, but more generally a property of the form object. Elements defined in the MetaAttributeGroup associated with the page are implicitly added; if an element is redefined here (its <Field> then corresponds to the identifier of a property listed in the MetaAttributeGroup), it replaces the implicit element: In this way you can therefore modify implicit display of an intrinsic property of the form object. If an implicit element appears in the page and is not connected to a MetaAttributeGroup, this may mean that it appears in a page other than that actually defined: You should therefore check that these two pages are not present simultaneously in the form, or that redundancy of the elements is not problematic (which is the case for example when one of the two elements is read-only).

When the element concerns a property of the object concerned, it is not generally necessary to include the control type to be displayed, this latter being deduced from the metamodel.

The keyword **XRef(True)** offers this implicit processing of properties not directly associated with the object (for example *TaggedValues*). This keyword indicates that the interpreter of the page can consider that the definition of the <Item> is accessible in the repository (for example via the **GetPropertyDescription** API function) and that it can be used to determine the implicit control type to be used.

```
<Name> = Item(<Field>)[,From(<MapName>)][,Duplicate(Yes)]
  [,{Contains|Remoting}(<MapName>)]
  [,In(<GroupName>)]
  [,Control(<ControlName>)]
  [,At(<left>[,<top>])][,Size(<width>,<height>)][,MaxSize(<width>,<height>)]

  [,VClip(<VClipMode>)][,HClip(<HClipMode>)]
  [,Visibility(<Visibility>)][,Name(<NameField>)]

  [,Title(<TitlePos>)][,At(<left>,<top>)][,Size(<width>,<height>)]
  [,Param(<Param>)]

  [,{HiddenOn|DisabledOn}(<ConditionName>)]

  [,Mandatory(<ConditionName>)]

  [,XRef(True)]
```

<Field>: References the MEGA element mapping the object. This can generically be a MetaAttribute, a TaggedValue, a MetaAssociationEnd. For implicit AttributeControls, the corresponding value must be accessible by the API function **obj.getProp("<Field>")**. For explicit or composite AttributeControls, other object types are possible, and use of <Field> depends on the control type.

<MapName>: Name of a Map relating to the element. It must be explicitly defined, but can be implicit (map **Context** map or **FromLeg** map)

The **From** Map indicates that the AttributeControl does not relate to the object, but to the object pointed in the map

The **Contains** Map is used on composite AttributeControls and enables definition of the list of its component AttributeControls. Implicit maps cannot be used in this case.

(compatibility): The **Remoting** Map indicates that the AttributeControl pilots a map; in this case the Control should not be specified, since the keyword only operates for **Control(DropDownSelection).** This specification has been replaces by keyword **Pilot** defined on the Map.

<GroupName>: Name of the Group in which the element will be physically located. The group must be defined before the element if it is explicit.

<ControlName>: Identifies the name of the type of graphic element to be used. The list of controls and their specificities are define in chapter 3.5. This parameter is not necessary in the case of an implicit control (property defined on the object, or external reference indicated by keyword **XRef(True)**.

<left>, **<top>**: Horizontal and vertical coordinates, relative to the group, expressed in DialogUnits (except exceptions). The vertical coordinate can be omitted; in this case the horizontal coordinate expresses shift relative to the group. These coordinates are optional, the element being positioned below the previous element by default.

<width>, **<height>**: Element height and width, expressed in DialogUnits. If not specified, a default size appropriate to control type and property characteristics is used.

<VClipMode>, **<HClipMode>**: These parameters enable management of deformation of a control at page resizing.

VClipMode enables definition of vertical deformation. It can have the following values:

> **No**: No vertical deformation – distance from top of element to top of page is fixed.
>
> **Yes**: Deformation proportional to that of page: distance from top of element to top of page is fixed, as is distance from bottom of page to bottom of element.
>
> **TopToBottom**: Equivalent to **Yes**.
>
> **Bottom**: No vertical deformation of element, distance from bottom of element to bottom of page is fixed.
>
> **Center**: No vertical deformation of element, distance from bottom of element to middle of page is fixed.
>
> **TopToCenter**: Distance from top of element to top of page is fixed, as is distance from bottom of element to middle of page. Vertical deformation of element in proportion half of page deformation.
>
> **CenterToBottom**: Distance from top of element to middle of page is fixed, as is distance from bottom of element to bottom of page. Vertical deformation of element in proportion half of page deformation.

HClipMode enables definition of horizontal deformation. It can have the following values:

> **No**: No horizontal deformation. Distance from left limit of element to left of page is fixed.
>
> **Yes**: Distance from left limit of element to left of page is fixed, as is distance from right limit of element to right of page. Stretch of element is therefore proportional to that of page.
>
> **LeftToRight**: Equivalent to **Yes**.
>
> **Right**: No horizontal deformation. Distance from right limit of element to right of page is fixed (element remains aligned on right).
>
> **Center**: No horizontal deformation. Distance from left limit of element to middle of page is fixed (element remains centered horizontally).

LeftToCenter: Distance from left limit of element to left of page is fixed, as is distance from left limit of element to center of page. Stretch of element is therefore in proportion half of page deformation.

CenterToRight: Distance from left limit of element to center of page is fixed, as is distance from right limit of element to right of page. Stretch of element is therefore in proportion half of page deformation.

⚠ Clipmodes must be defined with great care. An incorrect specification can cause overlap or covering of elements.

If you use clipmodes on several group elements, you must explicitly define minimum page size: this is the size on which elements are based to be correctly positioned. In Web Front-End, the size of the group in which the element appears is taken into account for element docking, if it is specified.

If clipmodes are not specified, standard behavior depends on control type. Most controls are of fixed maximum size and are resized horizontally when this maximum size is greater than effective page size. Composite controls and texts are resized by default to the maximum allowed by page size when they are located at the end of the page.

<Visibility>: Determines presence or absence of the element as a function of the user view of the metamodel. It can have the following values:

Standard: Applies the mapping visibility level (default value)

Always: Overrides systematic display

Admin: Displays only in extended view

Hidden: Systematically hides the zone and should therefore be used as overload of an unwanted attribute.

<NameField>: Enables replacement of the title of the element. Can be a field, and in this case the name of the referenced MEGA object will be displayed as zone title. A 'hard' non-translatable character chain can also be included. By default, the title of the zone is the name of the concept used as <Field>.

<TitlePos>: Indicates position of the zone title related to the zone. It can have the following values:

Up: The title is positioned above the element

Left: The title is positioned to the left of the element

No: The title is not displayed.

Positioning is ignored if title coordinates are explicitly given by the **At** and **Size** parameters defined immediately after. If these parameters are not defined, titles are aligned and sized to the maximum title size of the group concerned so that group elements can themselves be aligned.

<Param>: Optional configuration specific to control type. Can be a character string or a reference to a **Parameter** (in this case the parameter name is preceded by '@').

<ConditionName>: Name of a previously defined condition. If the condition is not fulfilled, the zone is either masked (use with **HiddenOn**), or deactivated (use with **DisabledOn**). However, unlike masking of groups, zone location remains reserved in the case of **HiddenOn**. If the page is not Synchronized (in this case, application of the condition does not cause page redesign). Keywords **Always** or **Never** can be included instead of a condition.

Applied to keyword **Mandatory** and to a modifiable element, the condition enables definition of whether the obligation constraint should be applied to the element.

Conditions are reassessed when displayed objects are modified in the repository, or when you modify their value (in the case of synchronized pages). If the value of a condition changes at these updates, the page can be redesigned.

By compatibility, the keyword **Attribute** is synonymous with **Item**.

3.3.1.5 Parameters

Can be used in the Param() field of an element when this field is long, complex, or includes brackets.

The chain supplied begins at the first bracket (after Param) and ends at the final bracket of the line.

In this case, the Parameter name should be included, preceded by @.

Example:

```
MyParam = Parameter(Long Parameter, with (Parenthesis) and « specials chars[>)
MyAtt = Attribute(…) …,Param(@myParam)
```

3.3.1.6 Inclusions

In page parameterization, reference can be made to another parameterization. This enables:

- factorization of elements common to several parameterizations.
- making page content dependent on elements specific to the form object.
- calculation of part of page content using a Macro.

Origin of inclusion is defined in parameterization by the keyword Origin.

Static inclusions use a parameterization included in a system repository object explicitly cited in the inclusion. In this elementary case there is no origin.

In dynamic inclusions, the system repository object in which we read the parameterization is deduced from the displayed object.

Inclusions can call a macro that will generate the parameterization to be included.

The lines of definition obtained by inclusion are inserted in the template of the page in the position where the keyword is located.

This device is recursive.

```
<Inclusion> = IncludeProfile(<ObjField>)[,DefaultOf(<InclusionName>)]
   [,From(<MapName>)][,In(<GroupName>)]
   [,Origin({FromLeg|Service|Object|ServiceList|Value|Macro})]
   [,Text(<TxtField>)][,Paragraph(<PrgName>)]
```

<ObjField>:

If the inclusion does not depend on an Origin, identify the system on which the parameterization will be queried. This object is not necessarily a **_PropertyPage**. If not, use of this identifier depends on the origin.

<InclusionName>:
Indicates that this inclusion is only effective if the <InclusionName> dynamic inclusion has found no system object on which to read a parameter. It therefore enables definition of a default content.

<From>:
When the inclusion depends on an origin, enables calculation of the origin from the object pointed by the map rather than the form object.

<GroupName>:
Includes elements of the inclusion for which no group was defined in the specific group.

<TxtField>:
Identifies the text in which configuration is included as appropriate. By default, this is the **_Parameterization**.

<PrgName>:
Name of the paragraph in which configuration will be read. By default, this is the **Template** paragraph.

The keyword **Origin** enables determination of the calculation mode allowing us to obtain the parameterization. It can have the following values:

- **FromLeg**: in this case we read the parameterization text on the MEGA object corresponding to the description of the form object. (in API: obj.gettypeobject.getID). In particular, this allows us to include elements relating to the MetaAssociationEnd from which we observe the form object.
- **Service**: in this case, <ObjField> is a MetaAssociationEnd or a property of object type, and we read the parameterization from the object corresponding to this property, (in API: obj.getProp("<ObjField>") if it is specified and belongs to the system repository.
- **Object**: in this case parameterization is read directly on the form object (in API: obj.getID). This is only possible of the observed object is in the system repository.
- **ServiceList**: parameterization is similar to **Service**, except <objField> here is a Collection identifier allowing us to obtain system repository objects from the form object. Parameterizations of accessed objects are concatenated when there are several of these (in API: for each srv in obj.getCollection("<ObjField>") …)
- **Value**: in this case *<ObjField>* is a listed property; when the value of this property for the form object corresponds to an listed value (*MetaAttributeValue* or more generally *Abstract PropertyLiteralValue*), we read parameterization on this occurrence of the literal value.
- **Macro**: in this case *<objField>* is a macro. The macro should implement the following function:

```
Function InvokeOnObject(
obj As MegaObject,
idText,
Page as MegaPropertyPageStandard) As String
```

obj being the form object, **idText** the identifier of the requested text (*_Parameterization* by default, or *<TxtField>* if specified, and **Page** the page in preparation.

This macro should send a character string which will be analyzed as a parameterization text. The paragraph [Template] (or [<PrgName>] if specified) of this text will be included in the parameterization.

3.3.2 Other options of standard pages

On a standard page you can define options that enable definition of its general behavior. These parameterizations can be declarative in the *_parameterization* text of the Page, or programmatic (see 3.4.1 and 3.4.2.4)

[Dialog]
CommandHandler=<MacroID>: Enables definition of a Macro that will be connected to the page and can listen to notifications from AttributeControls or the form (see 3.4.3)

[Page]
CheckVisibility = 1 (default) or 0: This option enables conditioning of page display due to the fact that its content is not empty at creation of the form. It should be deactivated if content of the page is calculated dynamically, but also for performance reasons; calculation of page content can be costly, and if this option is activated, should be carried out at creation of content (therefore at initialization) and not at display of the page. This option should also be deactivated if presence of the page is determined by the product or the ViewPort, and not by its content.

ReloadOnActivate = 0 (default) or 1: If activated, indicates that the page should be recalculated each time it is redisplayed. This can be useful when page content is determined dynamically and can vary depending on updates made from other pages.

ImmediateUpdate = 0 (default) or 1: When this option is inactive, the page is validated transactionally, that is the object is not updated with values modified in AttributeControls by clicking a button of the form (OK or Apply) or the wizard (Previous, Next or Finish. Otherwise the object is updated as soon as the AttributeControl is modified; in certain cases however this modification is effective only after a certain delay (Web Front-End) or when the modified AttributeControl loses focus.

IsSynchronized = 0 (default) or 1: If activated, indicates that the page is synchronized; in this case the conditions in its parameterization are evaluated with values in the form rather than with the effective values of the object attributes. This option is pointless if ImmediateUpdate = 1, since in this case these values coincide.

ActiveControl = 0 (default) or 1: When this option is activated, the validity control of an AttributeControl is executed at its modification; otherwise it is not executed until validation. This option is pointless if ImmediateUpdate = 1.

ContinueApply = 0 (default) or 1: If this option is activated, the page can be validated even if certain of its AttributeControls have detected errors. This mode can be indispensable in the case of a docked form without buttons, since in this case it can be impossible to close the form.
LabelTemplate = string: Enables redefinition of a page name. This string can contain a reference to an object of the system repository (in the form of a field), and in this case the name of this object will be used as the page name.

3.3.3 Synchronized pages - Immediate pages

3.4 Programmatic access to forms

This section describes possibilities of accessing page content by API code, so as to be able to specify a specific style on the form.

3.4.1 Executing a form by API

A form can be directly executed from APIs by the PropertiesDialog method. This method applies to the object that is the form object:

```
myObject.PropertiesDialog [option]{[,option]}
```

The following options are exclusive and enable determination of form content:

"PropPage=<Field>" displays the <Field> page in the form. By default, the ShowContextMenu option is deactivated.

"ViewPort=<Field>" displays the form of the object, conforming to PropertyPageViewPort <Field>. By default, the ShowContextMenu option is activated.

"Loader=<Field>initString" displays the form defined in the <Field> macro (see 2.3.2). By default, the ShowContextMenu option is deactivated.

The following options enable modification of form appearance or behavior:

"ApplyMode=Continue": enables continuous mode activation. In this mode, page modifications can be applied and the form closed, even if there are errors. This can be indispensable in docked mode.

"Immediate={0|1}": activates immediate mode; in this mode, application of the modification is instantaneous.

"Title=*caption*": enables modification of the form title.

"CheckAction={0|1}": enables display (1) or not (0) of the button for actions related to workflow of the form object (as appropriate) in the button/status bar.

"TabStyle={styles}": enables modification of tab display style (Windows Front-End) (see 2.3.2).

"ShowContextMenu{0|1}": enables activation (1) or deactivation (0) of display of object menu from the form title bar.

"HideStatus={0|1}": hides (0) or shows (1) the form button/status bar. If hidden, the form automatically passes to mode Immediate=1 – in absence of button it is not possible to apply modifications consistently outside this mode.

"ActivePage=*pageid*": enables definition of the initial page of the form.

"DefaultSize=*width,height*": enables definition of default size of the form (in pop-up mode).

"Position=*left,top*": enables definition of the position of the form in pop-up mode (Windows Front-End only).

As many arguments can be transmitted to the method as there are options. It is nevertheless possible to include several options in the same argument, separating these by a semicolon. The following two commands are therefore equivalent;

```
myObject.PropertiesDialog "ApplyMode=Continue;Title=my caption"
myObject.PropertiesDialog "ApplyMode=Continue" , "Title=my caption"
```

This is not however possible for the *Loader=* option, the *initString* string being totally free and can therefore contain semicolons.

The above options concerning form pages can be specifically redefined on each of the pages (see 3.3.2).

3.4.2 Access interface

3.4.2.1 Access to form content

You usually access interfaces relating to properties pages via a handler, that is via functions called at processing during execution of a form or wizard. It is however possible to directly access the component of a form using the PropertiesDialog function.

```
Set oList = oObject.PropertiesDialog([option]{[,option]}[,] "Description")
```

The string "**Description**" must be the last parameter.

The **oList** object returned is an enumerator enabling listing of form pages. The enumerated objects are **MegaPropertyPage** components.

3.4.2.2 MegaPropertyPage

The **MegaPropertyPage** enumerated above have the following properties:

.getId: GUID of the page (corresponds to the GUID MetaPropertyPage property of the page when it corresponds to a MetaPropertyPage).

.GUIName: page name that can be displayed in the user interface language.

.name: page name in the user language.

.Default: True if the page is the default page (active tab of form).

.IsTab: if True, the page in question is not a page with content, but a tab containing sub-pages.

.level: level of the page in hierarchy of tabs. If higher than 1, the page appears in a tab.

.ParentId: when the page is in a tab (level > 1), contains the identifier of the page containing the tab.

.sourceID: identifier of the MetaPropertyPage corresponding to the page. When the page is calculated, it does not correspond to a MetaPropertyPage, and in this case the identifier (if it exists) is the identifier of the system repository object that motivated appearance of the page; this is generally a MetaAttributeGroup.

.TypeID: identifier of _Type associated with the page.

.Component: when the page is a described page, returns the corresponding **MegaPropertyPageComponent** component. Otherwise returns *Nothing*.

3.4.2.3 MegaPropertyPageComponent

The **MegaPropertyPageComponent** object enables access to page content via an informal MegaObject (ie. it does not correspond to an occurrence of the MEGA repository) describing the page as an *AttributeControl*.

.Content enables access to this MegaObject.

.QueryVerb indicates if this **MegaPropertyPageComponent** implements the requested method. By this means, you can determine if it is a **MegaPropertyPageStandard**: in this case the **connect** function (for example) is implemented, and therefore QueryVerb(**"Connect"**) returns True.

3.4.2.4 MegaPropertyPageStandard

When the page is a standard page, the **MegaPropertyPageComponent** is therefore also a **MegaPropertyPageStandard**. It is using this component that wizard triggers access content of pages. It implements the following functions and properties:

Functions:

.Refresh(Optional reset as Boolean) requests the page to refresh. By default, this consists of requesting refresh of the AttributeControls it contains. If **reset** is present and is *True*, the page is totally recalculated. It is preferable to avoid using this option unless absolutely necessary, since the calculation can be costly and cause loss of contextual data.

At refresh of an AttributeControl:

- either the AttributeControl value was modified from the last application of updates: in this case the value is not modified, so as not to lose the entry made by the user.
- or the AttributeControl was not modified, in which case the displayed value is recalculated.

.Connect(connection As Object) enables association with the page of a component which can listen to notifications sent by AttributeControls.

.Disconnect(connection As Object) enables dissociation of a component associated by **Connect**.

.SetModify(val As Boolean) enables forcing of status of the page to be “modified” or “not modified” depending on the value *val*. In Windows Front-End in particular, the effect of this can be to "gray" or "ungray the "Apply" button: if after a SetModify(False) no form page is in "modified" state, the button will be grayed. It will be ungrayed at a SetModify(True).

.SetButton(button As String,Action As String) enables modification of appearance of the button of a form or wizard displaying the page. This function should be used with care, since buttons of wizards and forms are not identical:

For a wizard, buttons are named CANCEL, PROCEED, NEXT, PREV.

For a form, buttons are named CANCEL, OK, APPLY, HELP.

Possible actions are:

SHOW: shows button
HIDE: hides button
ENABLE: activates button

DISABLE: deactivates button
FOCUS: gives focus to button (if possible, and Windows Front-End only)
DEFAULT: defines the button as default button (Windows Front-End only)
TOCLOSE: applied to CANCEL button in a form, and in Windows Front-End only, renames the "Cancel" button to "Close" and deletes the "OK" and "Apply" buttons. To be used when an irreversible action has occurred (in this case it is no longer possible to cancel).

.OnCommand(Cmd as String) as String: enables simulation of sending a notification to the page. This function should only be used in specific contexts – for example to interact an updateTool with a dialog Handler (see 3.4.3 and 3.4.4). The command can contain a string in JSON format, and in this case the notification is considered as a DesktopCommand. It can also simulate a specific notification sent by an element of the page, in which case it is of form "ntfname:itemname". The value returned depends on the notification.

The following properties are accessible in read-only:

.getId As String: returns CLSID of the page. Equivalent to **MegaPropertyPage.getId**.

.getPageId: returns identifier of the MEGA object modeling the page. This is generally a MetaPropertyPage identifier, but not necessarily so.

.getRoot As MegaObject: returns repository root.

.template As MegaObject: returns the object on which the page was activated.

.isConnected As Boolean: indicates that the page has connections that can listen to notifications.

.labelTemplate As String: page title.

The following properties can be modified at form initialization (for example in a wizard), with the exception of statusMessage and isContinue properties, which can be modified at any time. If not, they can be consulted at any time. In most cases you can define these properties statically in page parameterization (see 3.3.2), but they can be overloaded in this way.

.transactional As Boolean: if True, the page is transactional and updates only at an explicit command (for example a click on "Apply" or "OK" buttons, or in a wizard “Previous”, “Next” or “Finish” buttons. Otherwise, update is carried out at each modification of an AttributeControl.

.activeValidation As Boolean: if True, the validity control of the page is carried out at each modification of an AttributeControl. Otherwise, it is only carried out at validation. Pointless in the case of a non-transactional page.

.statusMessage As String: indicates to the page container an error message or status to be displayed, as appropriate.

.isSynchronized As Boolean: Indicates that the page must be synchronized, that is that conditions defined in its template apply to data assigned in AttributeControls, and not to effective value of the object associated with the form. Pointless in the case of a non-transactional page.

.isContinue As Boolean: indicates that the page accepts validation, even if certain of its AttributeControls detect an error.

The following properties are not accessible at processing of a notification (see 3.4.3)

.currentControl as MegaObject: returns the AttributeControl sender of the notification.

.currentSelectedItem as MegaObject: when the sender AttributeControl is a list, a tree, or more generally an element handling a list of objects, and when the notification in progress references an element selected in this list, then this element is accessible using this property.

.currentNotification as String: returns the name of the current notification.

3.4.2.5 AttributeControl MegaObject and associated model object

The object model of a page (and more generally of a form) can be accessed via a hierarchy of MegaObjects. These objects do not correspond to occurrences of the database. The list of properties, methods and collections defined for these MegaObjects is not defined in the repository, so they can be accessed by their name without risk of incompatibility.

3.4.2.5.1 Properties defined for this MegaObject

Nature (String): name of AttributeControl type corresponding to the element (see 3.5).

Kind (Integer): completes nature of AttributeControl. This is a bit field containing in particular the following values:

ReadOnly: 0x2000 (8192): indicates that the element is read-only.

Numerical: 0x0100 (256): indicates that the element corresponds to a numerical value.

WidthDefault: 0x0100 (2048): Indicates that the element manages display of default value.

Mandatory: 0x4000 (16384): indicates that the element is mandatory.

NoEdit: 0x0400: indicates for a composite element with a data entry zone (ComboBox, EditButton…), that this zone is read-only.

ResetOnRefresh: 0x0200 (512): This flag is for drop-down lists and elements displaying a menu, and indicates that the menu or content of the list can change if the element has undergone modifications, and therefore needs to be recalculated.

ValidateInput: 0x8000 (32768): indicates that the updated value in the element must be specifically checked when the page is in active validation mode.

ItemName (String): element internal name. This name is stable and unique and corresponds to the name of the element declared in the template. In the case of an implicit element, this name is based on the absolute identifier of the implicitly listed property. The name can be used when you are querying an element in a collection using the Search function. It is also used to build the name of notification functions relating to this element.

Style (Integer): bit field defining element style.

Styles relating to element docking:

CLIPLEFT	0x00000002	dock at left (default)
CLIPRIGHT	0x00000004	dock at right
CLIPTOP	0x00000008	dock at top (default)
CLIPBOTTOM	0x00000010	dock at bottom
CLIPMLEFT	0x00000020	dock at left and at middle
CLIPMRIGHT	0x00000040	dock at right and middle
CLIPMBOTTOM	0x00000080	dock at bottom and middle
CLIPMTOP	0x00000100	dock at top and middle
NOVCLIP	0x00001000	ignore vertical docking
NOHCLIP	0x00002000	ignore horizontal docking
BOTTOMALL	0x00080000	dock at bottom if element is last on page

Styles relating to element title:

NOTITLE	0x00010000	element without title
TITLEUP	0x00020000	element with title above
SETTITLE	0x08000000	reserved for RadioButtons, to calculate their size

Styles relating to positioning and size:

AUTOPOS	0x00000800	element position has been calculated
AUTOSIZE	0x00004000	element size can change dynamically and is calculated by the element itself
SCREENUNITS	0x00008000	element size expressed in pixels and not in DialogUnits
BORDERED	0x00100000	element has border

Styles relating to element availability

DISABLED	0x00400000	element is deactivated
HIDDEN	0x00800000	element is hidden
MANDATORY	0x04000000	element entry is mandatory

Other styles

DEFAULTED	0x00040000	indicates that element manages default value
COMPUTEDHELP	0x20000000	indicates that element help must be calculated
ALWAYSUPD	0x10000000	indicates that element must be systematically validated, even if it has not been explicitly modified
DYNAMIC	0x01000000	indicates that element visibility can change during data entry
REMOTING	0x02000000	indicates that element can pilot other elements

ClipMode (String): property calculated from **Style** and describing docking mode. This string is empty if no docking is defined for the object, otherwise it takes the form:

"vertical":"<Vclip>","horizontal":"<Hclip>"

If vertical or horizontal docking is not defined, the corresponding element is not present in the string.

Vclip can be `"TopToBottom", "Bottom", "Center", "TopToCenter", "CenterToBottom"`

Hclip can be `"LeftToRight","Right","Center","LeftToCenter","CenterToRight"`

ID: element identifier. This identifier is based on the dynamic identifier and cannot be used to make information relating to this element persistent.

MapID: identifier of the Map in which the element is defined. This identifier allows you to find the Map in the collection associated with the parent AttributeControl of the element.

GrpID: identifier of the Group in which the element is positioned. This identifier allows you to find the Group in the collection associated with the parent AttributeControl of the element.

Name (String): element title in user language.

GUIName (String): element title in interface language.

HTMLTitle (String): element title in HTML format in interface language. This title can be "rich".

Order (Integer): element order umber, corresponding to the element dynamic identifier. It determines order of access to elements at browsing by tab key.

Width, Height, Left, Top (Integer): indicate position of the element relative to its group, as well as its size. This data cannot be specified if positioning is automatic (ie. not defined explicitly) or if its size corresponds to default size. This data is supplied in DialogUnits (see 3.1) except in special cases.

TitleRect (String): when title position has been explicitly defined, this property contains a JSON indicating this position. The JSON takes the form:
`{ "x" : x, "y" : y, "width" : w, "height" : h }`
If one of these properties is not defined, it does not appear in the JSON.
If this property is not defined, the space occupied by the title is included in the size of the element.

SourceID: contains the absolute identifier defined for the element. Generally corresponds to the absolute identifier of the displayed property, and more specifically to the identifier associated with the element in the template by Item(<SourceID>).

ObjectID: identifier of the object with which the element is associated. This is the form object if the element is not in a Map; otherwise it is the object pointed by the Map.

Options (String): contains options defined for the element. This value generally corresponds to parameterization contained in the Param() field in the element definition.

This property can be modified: certain control types can adapt their behavior at modification of this parameterization.

Value (String): value displayed by the element. If the element does not explicitly a value - this is for example the case for a CheckBox or RadioButtons – it contains the value of the property associated with the element for the source object. For composite elements not proposing a consistent value, this property cannot be used for specific purposes.

This property can be modified.

TargetID: contains an absolute identifier relating to the value displayed in the element, when this has a sense, and in particular:

When the element is designed to display a MEGA object, or more generally an attribute of object type, TargetID contains the identifier of this object.

When the element displayed has an enumerated value attribute, TargetID contains the identifier of the enumerated value.

When an element is updated by value, TargetID corresponds to identifier null if this value has not been identified as an enumerated object or value. This comparison is only made at validation of the element.

This property can be modified.

Data (String): character string containing information relating to the current value of the element. Its content depends on the control type and is generally presented in JSON format.

This property can be modified.

Format (Integer): bit field specifying nature of the element value, and information enabling use or specification of this value.

FORMAT_IDABS	0x0001	value corresponds to an object identifier and must be synchronized with **TargetID**
FORMAT_TRANSLATION	0x0010	value is not in current language
FORMAT_NEUTRAL	0x0020	value is in "Neutral" language
FORMAT_USERLANGUAGE	0x0040	value (enumerated) is in user language
FORMAT_HASDEFAULT	0x0080	for an enumerated attribute, the value (empty) indicating an unvaluated property is replaced by the value (default)
FORMAT_NUM_FLOAT	0x0100	value is a floating comma number
FORMAT_NUM_SIGNED	0x1000	value is a signed number
FORMAT_NUM_DURATION	0x2000	value is a number representing a duration
FORMAT_NUM_PERCENT	0x2000	value is a number representing a percentage

```
    FORMAT_NUM_EFFECTIVE 0x8000 value is an unformatted floating comma number
```

Read Only (Boolean) or short form **RO**: indicates that the element is read-only or deactivated.

Length (Integer): when the element displays a character string, indicates maximum string size. This property can be used to limit data entry or determine element size if this is not provided.

HelpKey (String): character string enabling determination of content of help associated with the element.

ErrorMessage (String): when the value of the element is invalid, or more generally an error has been detected related to this element, this string contains the corresponding error message.

IsDirty (Boolean): indicates that the element has been modifies and requires validation.

IsDefaultValue (Boolean): when the element manages default value, indicates that the current value corresponds to the default value.

IsInitialized (Boolean): indicates that the element has been initialized. Otherwise, the page has not yet been displayed and the values of properties that depend on the form object (for example Value, TargetID, Data) cannot be used.

Tick (Integer): numerical value incremented each time the element is modified.

Group, GroupGUIName (String): name of the group in which the element is located.

GroupOrder (Integer): order number of the group in which the element is located. Enables ordering of page elements.

ControlID (String): contains CLSID of the update Tool attached to the element.

3.4.2.5.2 Methods defined for this MegaObject

GetObject as MegaObject: object with which the element is associated. This is the form object if the element is not in a Map; otherwise it is the object pointed by the Map.

myCtl.GetObject.GetID corresponds to *myCtl.ObjectID*. We pass via this function when the element object is informal, since in this case we cannot use *GetObjectFromID* to access this object with its identifier only.

Search(itemIdent{,optional mapoption1, mapoption2 }) as MegaObject: When the element is a page or composite element, this function finds a sub-element according to its name (itemName).

It is also possible to find an element using its SourceID with an identifier (internal format) or a field as itemIdent argument. In this case several page elements can have the same SourceID, in particular if they are in different Maps. In this case you can specify the Map in which to search.

```
Search(sourceID, "MapID", mapID)
```

Otherwise, or more explicitly if you write:

```
Search(sourceID, "AnyMap")
```

the function returns the first element of the page with this sourceID.

You can recursively search for a sub-element from a composite sub-element by using an itemName path made up follows:

```
ItemNameMain>SubItemName
```

Refresh(optional Reset as Boolean) refreshes the element; if Reset, the element is recalculated.

Page as MegaPropertyPageStandard: when the element is defined in a Standard Page, this function returns the component corresponding to this page.

Component as Object: certain elements have an evolved access level, implemented by a specific component accessible via this function. For example, SubPages enable access via this function to the MetaPropertyPage component associated with the SubPage (when it is instanced).

3.4.2.5.3 Collections defined for this MegaObject

GetCollection("AttributeControl"): returns the list of sub-elements of the element.

GetCollection("Map"): when the element is a page (or SubPage), returns the list of maps defined on the page. These maps are described by an AttributeControlMap MegaObject described below. You can search for a map in this collection using the MapID property of an AttributeControl (**`mapCollection.Item(attcontrol.MapID)`**)

GetCollection("Group"): when the element is a page, returns the list of groups used in this page. These groups are described by an AttributeControlGroup MegaObject described below. You can search for a group in this collection using the GrpID property of an AttributeControl (**`mapCollection.Item(attcontrol.MapID)`**)

3.4.2.5.4 Main properties of AttributeControlMap MegaObject

ID: map identifier. This identifier is dynamic and cannot be used to make information relating to this element persistent. It corresponds to the MapID property of AttributeControls defined in this map.

Name as String: name of the map as defined and used in the template.

TargetID: absolute identifier associated with the map.

PilotID as String: when the map is piloted, contains the itemName of its pilot.

3.4.2.5.5 Properties of AttributeControlGroup MegaObject

ID: group identifier. This identifier is dynamic and cannot be used to make information relating to this element persistent. It corresponds to the GrpID property of AttributeControls defined in this map.

Name: name of the group as defined and used in the template.

Order as Integer: group order number, enabling definition of order of groups in the page.

GUIName: group title.

HTMLTitle: group title in HTML format.

SourceID: when the group title has been defined from a repository object, contains the identifier of this object.

Style as Integer: bit field indicating group style.

```
STYLE_BAR     0x10000      : separator not collapsible
STYLE_FRAME    0x20000      : frame, collapsible
STYLE_ALWAYS  0x40000      : separator present, even if in first position
STYLE_CLOSED  0x80000      : group is initially closed
```

Size as String: contains initial size of this group when defined, otherwise returns an empty string. The returned string corresponds to the following format:

```
"width" : w, "height" : h
```

If one of these two values is not defined, it does not appear in the string.

3.4.3 Dialog Handler

The dialog Handler system enables improvement of specific interaction possibilities on a properties page by allowing you to add interactions at events occurring when editing the form. These interactions are made using an interaction component implemented by a macro, and connected to the page.

Connection can be:

- either declarative in parameterization of the PropertyPage (see 3.3.2)

  ```
  [Dialog] CommandHandler=<MacroID>
  ```

- or by code - in wizards – using the Connect function of the MegaPropertyPageStandard component (see 3.4.2.4)

When the macro is connected, it can carry out processing on notifications from AttributeControls of the page, or on events relating to the form or page.

3.4.3.1 Notifications

```
Sub On_ctlName_ntfName(Page as MegaPropertyPageStandard)
```

When it is implemented, this method is called when the AttributeControl of which the itemName property is ctlName sends the notification ntfName.
At this call:

> *Page.currentControl* corresponds to the notification sender element. Therefore *Page.CurrentControl.ItemName* corresponds to ctlName
>
> If the sender manages an object list and if the notification concerns the element selected in this list, *Page.currentSelectedItem* corresponds to the selected object.
>
> *Page.currentNotification* corresponds to ntfName

If this method is not implemented, the OnCommand method (see below) is then called.

```
Sub OnCommand(Page as MegaPropertyPageStandard,ntf as Integer,ctl as Integer)
```

This method is called each time a notification sent by an AttributeControl of the page is not specifically processed by a method.

> ntf is a notification identifier number. Standard notifications are named and their name is accessible via Page.currentNotification.
>
> ctl corresponds to the Order property of the sender AttributeControl. It can change if the page is built dynamically, or if elements can be hidden and the page is synchronized or in immediate mode. For these reasons, it is preferable to use Page.currentControl.itemName, corresponding to the name of the element as defined in the template, to determine the notification sender.

At this call:

> *Page.currentControl* corresponds to the notification sender element.
> If the sender manages an object list and if the notification concerns the element selected in this list, *Page.currentSelectedItem* corresponds to the selected object.

3.4.3.2 Other events

```
Sub PropertyPage_Initialization(Page as MegaPropertyPageStandard)
```

This method is called at page initialization.

```
Function CheckPropertyPage(Page as MegaPropertyPageStandard) As String
```

This function is called before each page validation. It allows you to indicate an error preventing page validation. In case of an error, this function returns an explanatory error message. If there is no error, it returns nothing or an empty string.

In case of error, validation is cancelled and no update is carried out.

```
Function OnPropertyPageApply(Page as MegaPropertyPageStandard) As String
```

This function is called after each page validation. In particular, it allows you to carry out supplementary updates. It is possible to indicate an error preventing page validation. In case of an error, this function returns an explanatory error message. If there is no error, it returns nothing or an empty string.

In case of error, if the page is not in continuous mode, if we are in:

- a form, it is not closed,
- a wizard, the transition is stopped and we remain on the page.

```
Sub PropertyPage_OnDesktopCommand(Page as MegaPropertyPageStandard, JSONCommand
As String)
```

This function is specific to Web Front-End and enables sending of a notification from Desktop to the form or wizard, if scopes of tools will allow this. The notification is only sent on the active page of the tool.

3.4.4 UpdateTools

An UpdateTool is a component designed to manage update of a property (MetaAttribute, _AbstractProperty or MetaAssociationEnd seen as an attribute).

This component is only invoked through an AttributeControl, whether in a form, a wizard or an in-place data entry; it is not designed to parameterize display of an attribute in a regeneration.

An UpdateTool enables:

- definition of the AttributeControl to be used to update the property.
- definition of options and complements of this AtrributeControl, as could be done in page template configuration.
- definition of content of the drop-down list, when the attribute type includes one.
- if appropriate, definition or completion of the menu and processing of commands, or implementation of button processing.
- definition of the value effectively displayed by the element.
- definition or completion of validation of the element, in particular update of the repository induced by this validation, and application of specific controls.

An updateTool cannot be associated with a varchar type property (text). Updates of these properties are defined by their associated _type.

3.4.4.1 Implementing an updateTool for a property

An updateTool is implemented by a Macro. To be recognized as such, this component must contain the function.

```
Function AttCtl_GetDefaultKind() As String
```

This macro should be associated with the property according to the property type. If it is:

- a MetaAttribute, the MetaAssociationEnd "*MetaAttributeUpdateTool*" should be used.
- a MetaAssociationEnd *_LegAttribute*, the MetaAssociationEnd "*_LegUpdateTool*" should be used.
- an *Abstract Property* (for example a *TaggedValue*), the MetaAssociationEnd "*Update Tool*" should be used.

3.4.4.2 Defining AttributeControl type

The first role – the only mandatory role – of an updateTool is to define the type of AttributeControl that will be used to update the property.

- If this type does not depend on the edited occurrence, and if it is not necessary to define options, you can simply implement the function **AttCtl_GetDefaultKind()** mentioned above.
- Otherwise, you must add the function to it:

  ```
  Function AttCtl_GetKind(Context As MegaUpdateToolContext) As String
  ```

  Unlike the previous case, this function enables different behavior according to the edited object, and specification of options forAttributeControl, using the context component **MegaUpdateToolContext**.

These two functions should return a character string of form:

```
<ControlName>{:<option>}{,<option>}
```

ControlName is the name of the AttributeControl type (see **Error! Reference source not found.**)

Possible options are:

- type complements such as lists in the **Kind** property of the AttributeControl MegaObject (see **Error! Reference source not found.**),
- options that specify the AttributeControl use context:

ManageReadOnlyMenu: in the case of an EditMenu, indicates that the updateTool takes into account the fact that the object or property is read-only and adapts the menu or its behavior to this fact. By default, the framework considers that read-only is not managed and specific commands of the updateTool are not inserted in the menu.

ManageValueID: in the case of an attribute with enumerated values, indicates that the TargetID property, systematically corresponding to the enumerated value absolute identifier, is perhaps used to update the attribute when it is of the corresponding type.

SingleOnly: indicates that the AttributeControl does not operate in multiselection mode and that attribute entry should be deactivated in this case.

In the AttCtl_GetKind function it is also possible to modify AttributeControl style and options.

3.4.4.3 MegaUpdateToolContext component

MegaUpdateToolContext enables access to updateTool context. It includes:

- the following objects:

  .MegaObject As MegaObject: occurrence to be modified. In the case of an entry derived from MultiSelection, this is a calculated MegaObject aggregating the collection of selected objects.

  .AttributeID As Variant: attribute identifier (resp. Abstract Property or MetaAssociationEnd) to which the updateTool relates.

  .GetRoot As MegaRoot: access to properties page MegaRoot.

  .AttributeControl As MegaObject: returns the component corresponding to the element controlled by the updateTool. This component is AttributeControl type (see **Error! Reference source not found.**) and provides direct access to the page element, as well as the page itself (where appropriate).

  .ComboListCollection As MegaCollection: when the element manages a drop-down list, this collection corresponds to the list of elements appearing in the list. It is designed to be populated by the updateTool in the *AttCtl_FillCombo* function (see below).

- the following properties:

  .EditText as String: value displayed when the element contains a text area (in editing or read-only). This property can be modified and corresponds to *.AttributeControl.Value*.

  .IsInPlaceMode As Boolean: is True if the element is invoked in the context of in-place entry. This property can be accessed after object initialization: The type of element displayed can differ if you are in in-place mode.

  .IsMultiSelection As Boolean: is True if the element is invoked in the context of multiselection entry. This property can be accessed after object initialization, for example when the updateTool does not know how to manage multiselection and in this case wants to force mode to "read-only".

  .ValueID: where appropriate, contains an identifier corresponding to displayed value, corresponding to *.AttributeControl.TargetID*; this identifier can correspond to the identifier of an object (when the property is object type) or to that of a MetaAttributeValue (when the object is enumerated). The updateTool may have to explicitly manage the displayed value (*EditText*) and the identifier pointed (*ValueID*), in particular when implementing a command modifying the value of the element. This is the case for elements of object type, for which it is highly desirable to have the identifier if known, as well as the name to be displayed. It should be noted that modification of *EditText* results in invalidation of *ValueID*: The latter should therefore be updated after *EditText*.

  In the case of Web Front-End, update from a combined list does not update the value of this variable if the property concerned is not of Object type.

  .ValueTypeID: When updateTool implements a generic MetaAssociationEnd, attribute or object type abstract property, you can use this property to consult or update the MetaClass type of the target object. Where appropriate, this property can be used to create the standard menu of the element.

- the following functions:

.Invalidate: notifies that the element has been modified. Among other things, this enables ungraying of the "apply" button in properties pages.

.SetDirty: should be used at menu command or button press processing, when you want to indicate to the element that the current value should be considered as having been modified and therefore that the element should be updated.

.Recompute(Item as String): invalidate the menu (if Item contains "Menu") or drop-down list (if Item contains "DropList") of the element and results in menu recalculation at the next invocation (ie. click on menu or drop-down list button. It is possible to transmit ("Menu,DropList") as parameter.

.AddAccelerator(key as String,cmd as Integer{,altkey as String}): (Windows Front-End only) can be used in the *AttCtl_GetAccelerators* function of the updateTool, to define accelerators (key combinations triggering actions when the element has focus).

key corresponds to the keyboard key,

altkey indicates the combined keys and can contain values "ALT", "SHIFT" and "CONTROL" (value "ALT,CONTROL" - for example – is also included) ;

cmd is the number identifying the command and transmitted in the *AttCtl_AcceleratorInvoke* function when the corresponding accelerator is activated.

3.4.4.4 Generic functions

These functions concern initialization, display and update of an element. If they are present in the updateTool, they are invoked whatever the element type.

```
Function AttCtl_SetText(Context,editText) As Boolean
```

This function is called at supply of the element, and can be used in particular when you want to display on the element a value not corresponding to the effective value of the property.
The editText argument corresponds to the value that will be displayed if the updateTool does not carry out any processing. This argument can be modified, and if the function returns True, then the text display area is updated with the new value.

```
Function AttCtl_SetDefaultText(Context,editText) As Boolean
```

This function is called when you want to display the element default value (for elements with default value view system). Its operation is identical to that of AttCtl_SetText.

```
Function AttCtl_UpdateText(Context,editText) As Boolean
```

This function is called after element validation, when we are preparing to display the new value after element update. Its operation is identical to that of AttCtl_SetText.

```
Functon AttCtl_UpdateCheck(Context,Status,ErrorMessage) as Boolean
```

This function is called at element validity check – when the page is in "*ActiveValidation*" mode. It should not execute any update.
In the case where the check detects an error, this error should be documented by updating the *ErrorMessage* argument with a character string explaining the error; this string will then be

displayed to the user. If there is no error, an empty string should be returned, or the argument not modified.
The Status variable contains the return code, which in this case can be:

STATUS_ERROR (1): an error has been detected. If ErrorMessage is not empty, then STATUS_ERROR is automatically activated.

STATUS_NOCHANGE (2): indicates that the element will not result in property update.

If this function does not return value FALSE, standard validity checks of the property are called.

```
Function AttCtl_Update(Context,Status,ErrorMessage) as Boolean
```

This function is called at element validation; it should execute validity checks, then update the property – except if element default update is suitable.
In the case where the check detects an error, this error should be documented by updating the *ErrorMessage* argument with a character string explaining the error; this string will then be displayed to the user. If there is no error, an empty string should be returned, or the argument not modified.
The Status variable contains the return code, which in this case can be:

STATUS_ERROR (1): an error has been detected. If ErrorMessage is not empty, then STATUS_ERROR is automatically activated.

STATUS_NOCHANGE (2): indicates that the element will not result in property update.

STATUS_OK (4): the property has been correctly updated (in this case False should be returned so that standard update is not executed).

If this function does not return value False and status is not STATUS_ERROR, property standard update is called.

3.4.4.5 Specific functions implemented by UpdateTool

```
Function AttCtl_FillCombo(Context,ComboListColl) as Integer
```

If present, this function is called to populate the drop-down list of an AttributeControl.
To do this:

➔ Add in ComboListColl collection the values you want to include in the drop-down list.

This collection is a collection of enumerated Values, corresponding to the type used in MegaObjects accessing the compiled Metamodel, as used in particular by the GetPropertyDescription(<propID>).Values collection.
These MegaObjects have in particular the properties:

Rpbid: identifier in base 64. For properties of object type, this identifier is assigned at selection in the drop-down list to **ValueID**, and is therefore used to update the property at validation (except of course if validation is specifically handled by the updateTool). Otherwise, where appropriate, this is the identifier of a MEGA enumerated value.

InternalName: internal value, used as standard to execute update for attributes not of object type. For an object type attribute, this can be a field referencing the listed object.

GUIName: name displayed in the drop-down list.

So that elements will be effectively integrated in the list, they can be:

- MegaObjects representing enumerated values obtained from a description. It is only in this case that you can display bitmaps, derived from modeling of enumerated values (note: there are no bitmaps in ComboEditMenus)

- MegaObjects explicitly created in the collection, not directly corresponding to repository objects, and not outliving this collection.

Return value can take the following values:

0: call default processing: typically, the default enumerated list corresponds – for the enumerated properties – to the following code

Context.getRoot.getPropertyDescription(Context.AttributeID).Values

For _legAttributes, the default code supplies the drop-down list by means of favorite candidate query if it has been defined.

1: display list from collection - case of a standard enumerated property

2: display list from collection, taking account of absolute identifier of the enumerated value - applicable to properties of object type

-1: as 1, if you want to display (Default) rather than (Empty) to indicate empty value in the combo

-2: as 2, if you want to display (Default) rather than (Empty) to indicate empty value in the combo

Note: if there is inconsistency between the return value and the property type, behavior risks being not as expected. In Windows Front-End, an error will be activated if the properties pages debugging option has been activated.

The following example enables management of an object type property of which the target is compatible with the 'Org-Unit' MetaClass. The list is supplied with all repository org-units. Note that this code is similar to standard code supplying a drop-down list from a favorite candidate query (in this case the query is used in GetCollection in place of the "Org-Unit" MetaClass).

```
Function AttCtl_FillCombo(oContext As MegaUpdateToolContext,oFillCollection As MegaCollection) As Integer
  Dim oOrgUnit
  for each oOrgUnit in oContext.MegaObject.GetRoot.GetCollection("OrgUnit")
    Dim oAdded
    Set oAdded = oFillCollection.Create(oOrgUnit.GetID)  ' assignment of absolute identifier of org-unit to value created.
    oAdded.GUIName = oOrgUnit.ShortName
    oAdded.InternalName = oActeur.MegaField()  ' warning, internal value should not exceed 20 characters… We can put simple counter in this case since in the case of an object the internal value is not used
  Next
  AttCtl_FillCombo = 2
End Function
```

```
Function AttCtl_ImplementsMetaCommand(Context) As String(or Integer)
```

Presence of this function indicates that the updateTool will handle all or part of the AttributeControl pop-up menu – and therefore applies only to elements that have such a menu.

It is first possible to redefine the *capability* applicable for the element (see **Error! Reference source not found.**). To do this, the function can return either a numerical value corresponding to the required *capability*, or a character string containing keywords **LINK**, **LIST**, **SEARCH**, **NEW, EMPTY** to activate the corresponding capabilities **Integrate link specification** (0x20), **List** (0x1), **Search** (0x2), **Create** (0x8), **NoEdit** (0x1000).
It is then possible to add specific commands in this Popup menu by implementing in the updateTool the functions allotted to MetaCommandManager, in particular:

```
Sub CmdCount(obj,count)
Sub CmdInit(obj,num,name,category)
Sub CmdInvoke(obj,num)
```

On calling these methods, the updateTool context is made available to the macro implementer by the *AttCtlContext* property of *PropertyBag* of the global MegaMacroData

This system enables insertion of specific menu commands. It does not however enable insertion of pop-up sub-menus corresponding to object menus as the standard ChildMenu does; this possibility is offered by a specific system. You should:

1. Define capability in the form of a character string and insert in it the keyword **SPECIFICCHILDMENU.**
2. Implement in the updateTool the following function:

   ```
   Function AttCtl_GetMenuObject(Context,Indice,Name As String) As MegaObject
   ```

   This function should return the MegaObject of which you want to display the menu.

   It is possible to specify *Name* with the pop-up menu label; if this is not done, the pop-up will have as label the name of the MegaObject description.

   Several object sub-menus can be inserted; the function is called as many times as it returns a MegaObject different from the previous one; the Index argument is incremented at each call (initial value is 1).

The following is an example of updateTool displaying a menu, applicable to a MegaIdentifier type property.

```
'MegaContext(Fields,Types)
'Uses(Components)
Option Explicit

Function AttCtl_GetDefaultKind()
AttCtl_GetDefaultKind = "ComboBoxMenu:ValidateInput"
End Function

Function AttCtl_ImplementsMetaCommand(AttCtlContext As MegaUpdateToolContext)
AttCtlContext.ValueTypeID = "~BEy8SnY(yKk0[City Planning Area])"
AttCtl_ImplementsMetaCommand = 7
End Function

Sub CmdCount(obj,count)
count = 3
End Sub

Sub CmdInit(obj,num,name,category)
name = "Command " & num
category = 4
End Sub

Sub CmdInvoke(obj,num)
Dim AttCtlContext as MegaUpdateToolContext
Set AttCtlContext = MegaMacroData.GetBag.AttCtlContext
  Dim oR
  Set oR = AttCtlContext.MegaObject.GetRoot.GetCollection("Acteur").SelectQuery( -
           "Invoke Command #"  & num & " on Attribute " &  -
           AttCtlContext.AttributeControl.Page.GetID,True)
if oResult.Count = 1 Then
AttCtlContext.ValueID = oResult.Item(1).GetID
AttCtlContext.EditText = oResult.Item(1).ShortName
```

```
         end                       if
    End Sub
```

Also applicable in the case where an AttributeControl can give access to a PopupMenu, it is possible to define and manage accelerators, that is keys able to execute commands when pressed while AttributeControl has focus; these accelerators are only currently available in Windows Front-End:

```
Function AttCtl_GetAccelerators(Context)
```

This function enables definition of accelerators. To do this use the **AddAccelerator** function of *Context*. The return value of this function is not significant, and in VbScript a Sub can be used. When an accelerator has been defined, the following function is called to activate it:

```
Function AttCtl_AcceleratorInvoke(Context,cmd) as Boolean
```

Return value of this function is optional and, if specified, indicate the the AttributeControl has been modified (value True) at accelerator call.

```
Function AttCtl_OnPush(Context) as Integer
```

This function is called when a button is clicked (if this button does not open a pop-up or drop-down). For controls of "Button" type, this function is only called if the "NoCall" option is defined. It is also called for an Image type control, if the action has not been deactivated (by option Click=Inactive) or handled by a menu (Click=PopupMenu). In this case the data field of the AttributeControl can contain click coordinates in form "cursorPos" :"line,col" .

If return value is not null, this indicates that the updateTool has processed the command. If it is positive, this indicates that the element has been updated by the command.

3.4.4.6 Processing notifications sent by the element

When an AttributeControl is modified or must handle events, it can notify the updateTool if the latter implemented the function

```
Function AttCtl_OnCommand(Context,ntfCode,item) as Boolean
```

item is a numerical value indicating notification context, and *ntfCode* the notification code, when context is an interface element. In Windows Front-End, notifications relating to elements are retransmitted to the updateTool, and Windows documentation contains the list. In Web Front-End, only certain notifications are sent, interactions being less detailed. Only notifications valid in Web Front-End are listed below. Note here that there can be significant differences between Web Front-End and Windows Front-End:

- in Web Front-End, you do not send detailed interactions specific to internal operation of Windows.
- successive notifications can be sent in a different order.
- in Web Front-End, it is not always possible to interact with the rest of the form at an internal notification – in particular Page.Refresh will only operate the notification comes from an effective client command.

For this, it is recommended to act only on clearly identified notifications and to be aware – if appropriate – that the code may not operate in Web Front-End.

3.4.4.6.1 Internal notifications

These notifications are not necessarily sent at an interaction. They can also be sent at page creation. This is why during their processing, execution of actions using the rest of page content should be avoided (for example refresh operations), the rest of page content not necessarily being operational at the time of notification.

Item values for these notifications are:

- **ITEM_INIT** (0x8000 – 32768) sent at element initialization.
- **ITEM_SET** (0x8007 - 32775) sent when the element value has been modified, either at a direct interaction or by code.
  ntfCode is 0 if the element is reinitialized (not specified)
- **ITEM_READONLY** (0x8006 - 32778) notification sent when element read-only property has been changed – This notification is only currently effective for EditMenus.
  ntfCode is 1 if the element is deactivated.

3.4.4.6.2 Notifications resulting from user action

Item values for these notifications are:

ITEM_DROPDOWN (1) notification concerning element drop-down list. In this case, notification code to be used is:

CBN_SELENDOK (9) This notification is sent when a new element has been selected from the drop-down list

ITEM_MENU (2) notification concerning element pop-up menu. In this case, notifications to be used are:

MNU_DROPDOWN (7): sent before pop-up menu display
MNU_ENDOK (9): sent after menu display, when a command has been executed.
MNU_ENDCANCEL (0xA – 10): sent after menu display, if no command has been executed.

ITEM_EDIT (3) notification concerning edit area. In this case, notification code to be used is:

EN_CHANGE (0x300 – 768) – sent when content of the area has changed.

ITEM_BUTTON (4) notification concerning button. In this case, notification code to be used is:

BN_CLICKED (0) – button has been pressed.

ITEM_TITLE (5) should not normally be used.

3.4.4.7 Multiselection MegaObject

When an updateTool is instanced in a multiselection framework, it has (context.MegaObject) a virtual MegaObject calculated from the list of selected objects.

In addition, the Context.IsMultiSelection property is True.

On this object, the GetProp() function calculates the property value according to that of the listed objects: if a value is identical on all objects, then GetProp()returns this value. Otherwise, the value is considered as unspecified. Unique or indexed properties (such as the absolute identifier for example) are not therefore significant for the multiselection object.

Regarding the SetProp() function, it applies the update on all objects of the selection; it is therefore not possible to update a unique index.

If the updateTool implementer needs to explicitly know the list of selected objects, this can be obtained from the multiSelection MegaObject using the ad hoc collection megaObject.GetCollection("~p1qjEch)GnxA[SelectedObjects]")
However, other collections accessible from this MegaObject are empty (in particular they are not created from collections of selected objects, as properties are).

3.5 AttributeControl types

This section covers the different control types that can be displayed in a page. For each of these controls, the following is explained:

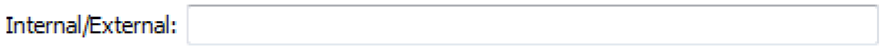
- Parameterization options, which can be specified in the Param() keyword.
- Use of properties and, as appropriate, collections and methods of the MegaObject corresponding to this type of element.
- Explicit notifications sent by this type of element.

3.5.1 Implicit AttributeControls

These types are automatically associated with properties depending on their type and format. Under certain constraints, it is possible to redefine the control displaying a property.

Unless otherwise indicated, all the elements listed below are presented preceded by their title.

3.5.1.1 Edit



Internal/External:

This type is the default type used for a modifiable property. Its default size is adjusted to the declared length of the property and the number of characters that can be entered.

The value displayed is the external value of the attribute.

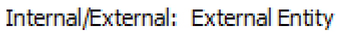
Properties:

Value corresponds to the displayed value.

Notifications sent:

Change: sent when edit zone content has been modified.

3.5.1.2 Static



Internal/External: External Entity

This type is the default type used for a read-only property. Its default size is adjusted to the declared length of the property.

The value displayed is the ''display' value of the attribute.

Properties:

Value corresponds to the displayed value.

3.5.1.3 CheckBox

Default type for Boolean properties. By default, it has the NOTITLE style, and the title value is displayed in the static zone located after the button; default size of the element is calculated depending on the size of this title.

By default, this control uses the internal value of the property to execute update; to do this, it considers that this internal value is the number 1 for check value and 0. This control therefore operates natively with boolean type properties, short et long. In other cases, correct operation can only be assured with a *CheckOn* option.

This control has only two values, and does not distinguish unspecified properties from 'False' properties (which correspond to value 0). If you wish to distinguish these values, use 3StateCheckBox type.

Option:

CheckOn=<UnCheckValue>/<CheckValue>: This option enables display of a property with only two valid values in the form of CheckBox, and this whatever its format. It is particularly adapted to simplify entry of attributes with two enumerated values. To do this, specify (in ASCII format) the value playing the "uncheck" role and the value playing the "check" role.

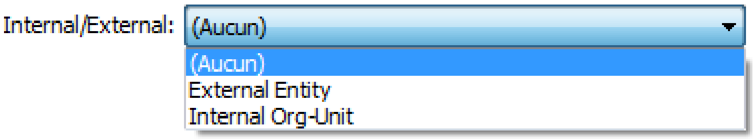
Properties:

Value: corresponds to ASCII value "0" or "1".

Notifications sent:

Click: sent when we click the box.

3.5.1.4 DropDownList



Drop-down list, used as standard for closed list enumerated properties. This element does not therefore allow entry of a value, which can only be modified from a drop-down list.

When the value of the property does not correspond to an available enumerated value (this can also occur if the enumerated values are filtered), the effective value of the property is nevertheless added to the drop-down list content. In principle, it should be possible to activate an entry without modifying the previous value, and this in any circumstances.

When the property is mandatory, the value (None) is not proposed.

The displayed value corresponds to the external value of the property; this value is normally presented in the user interface language, but it can be parameterized so that this display is in the current language by activating the "Keeps user language" indicator (0x40) in its 'Extended properties' MetaAttribute.`

Default size of this zone is calculated according to the size of the external values of the list of enumerated values.

Properties:

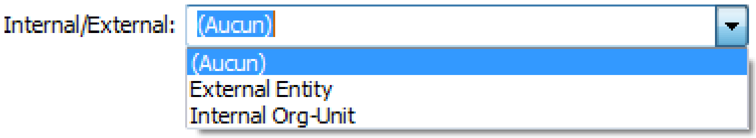
Value: displayed value.

TargetID: absolute identifier of the enumerated value corresponding to the displayed value.

Notifications sent:

SelEndOK: sent when a new value has been selected in the list.

3.5.1.5 ComboBox



Internal/External: (Aucun)

(Aucun)
External Entity
Internal Org-Unit

Drop-down list combined with edit zone, used as standard for open list enumerated properties. The value presented in the edit zone does not necessarily correspond to an enumeration value.

It is possible to directly enter the value to be modified in the edit zone; when it corresponds to an external or internal value of the enumerated values declared on the property, the physical update is executed with the internal value of the latter. The language used for external enumerated values can be parameterized by the "Keeps user language" indicator in their 'Extended properties' MetaAttribute (see 3.5.1.4).

Default size of this zone is calculated according to the size of the external values of the list of enumerated values, and the size of the attribute itself.

Properties:

Value: displayed value.

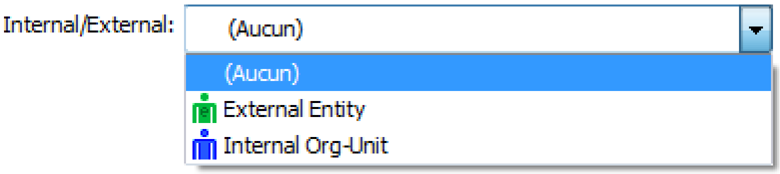
TargetID: absolute identifier of the enumerated value corresponding to the displayed value, if applicable. When the displayed value has been directly modified in the edit zone, and as long as the zone has not been validated, this value is null. It is at validation, most often at update, that comparison is made between an enumerated value and this entered value.

Notifications sent:

SelEndOK: sent when a new value has been selected in the list.

Change: sent when the edit zone has been modified.

3.5.1.6 ComboBitmaps



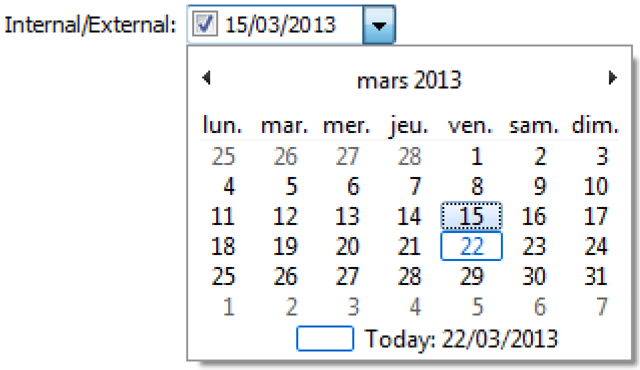
Internal/External: (Aucun)

(Aucun)
External Entity
Internal Org-Unit

Drop-down list associated with an image. This type of element is associated with enumerated properties when the "Activate images" indicator (0x4000) has been activated in their 'Extended properties' MetaAttribute.

Operation is similar to a DropDownList (3.5.1.4). The image displayed is defined in the enumerated value.

3.5.1.7 DatePicker



Internal/External: ☑ 15/03/2013

Edit zone associated with date entry via a calendar. This type of element is associated with properties of date type.

The date is displayed in external format.

This element can be used to edit times using the *TimeFormat* option.

This element operates correctly only with Date type properties.

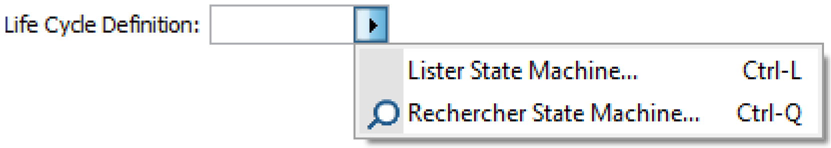
Option:

TimeFormat: entry of a time.

Properties:

Value: value of date entered in ASCII format (GMT yyyy/mm/dd hh:mm:ss)

3.5.1.8 EditMenu (and other controls designed for object type properties)



Life Cycle Definition:

Lister State Machine... Ctrl-L
Rechercher State Machine... Ctrl-Q

Edit zone associated with a button enabling opening of a pop-up menu. This type of element is associated with properties of object type (including in particular _legAttributes). These properties can also be associated with StaticMenus, ComboBoxMenus, DropDownListMenus, even ComboBoxes or DropDownLists, depending on parameterization of their capacity, as well as the definition of candidate queries. This parameterization also enables definition of menu content.

The edit zone displays as standard the short name in *display* format of the object pointed by property when it exists; otherwise the ASCII value of the absolute identifier pointed is displayed.

3.5.1.8.1 Specification of update capacity and possibilities

Capacity (Capacity) is a MetaAttribute defined on the _AbstractProperty MetaClass and the _LegAttribute MetaAssociationEnd enabling piloting at metamodel level of behavior of the modification tool concerned. This attribute is a bit field for which the following values are defined:

- **List (1)**: presence of "list" menu command. If the entry zone has been modified, List presents only those objects with names beginning with the value entered in the edit zone.
- **Search (2)**: presence of the "search" menu command, which enables object selection using the MEGA standard query tool.
- **ChildMenu (4)**: presence of the menu of the associated object as a sub-menu.
- **Create (8)**: presence of "create" menu command.
- **NoEdit (4096 - 0x1000)**: the edit zone is read-only. If the element is not required, the "delete" menu command is added.
- **Integrate link specification(default) (32 - 0x20)**: in the case of a _LegAttribute, specifications defined on the MetaAssociationEnd relating to an entry, and in particular candidate definition, are taken into account for presentation of the element.
- **Ignore link specification (16 – 0x10)**: in the case of a _LegAttribute, specifications defined on the MetaAssociationEnd relating to the entry are ignored.
- **AutoCreate (256 – 0x100):** when the user has modified the edit zone of the element, and therefore the element does not make explicit reference to an existing object, validation of the element results in creation of an object whose name corresponds to the entered value.
- **AutoList (128 – 0x80):** when the user has modified the edit zone of the element, and therefore the element does not make explicit reference to an existing object, but the name entered in this zone can correspond to repository objects, the list of these objects is proposed at validation.

When taking into account of MetaAssociationEnd entry specification has not been deactivated, we search on this latter:

- value of MetaAttribute LinkOptions. If this value is odd (bit 1 activated), the Search menu is hidden.
- the list of Candidate queries which have been associated with it, as well as the value of the "Favorite" MetaAttribute defined on this link. If we find a favorite Candidate query (that is with value "yes" in the "Favorite" MetaAttribute), the control presented displays a drop-down list button enabling listing of occurrences defined by the favorite candidate query. The other candidates will be the subject of specific menu elements: if we select these menu elements, the corresponding query will be executed, enabling object selection.

When the property cannot be modified, or when user rights on MetaAssociationEnds are restricted, the menu and the capacities used are correspondingly adapted; if the zone cannot be modified, only the ChildMenu will be presented as appropriate.

The type of element presented therefore depends on these specifications, and in particular on:

- the presence of a menu enabling selection or consultation of the object presented in the element.
- the definition of a favorite candidate query, which will display drop-down list behavior (and therefore possibly a second button).
- the fact that the edit zone is read-only or not.

Menu	*favorite query*	*read-only*	*control type*
Yes	No	no	**EditMenu**
Yes	No	yes	**StaticMenu**
Yes	Yes	yes	**ComboBoxMenu**
Yes	Yes	no	**DropDownListMenu**

No	Yes	yes	**ComboBox**
No	Yes	no	**DropDownList**

3.5.1.8.2 Target MetaClass

The behavior and appearance of the element menu depends on the MetaClass of the property target object. In the case of a MetaAssociationEnd, this MetaClass corresponds to the opposite MetaClass. In the case of a MetaAttribute of Object type, you can define this MetaClass using the "*Referenced MetaClass*" MetaAssociation. In the case of an *Abstract Property* of Object type, it is the MetaAssociation (*Property Object / MetaClass*) that enables definition of this MetaClass.

When the target MetaClass is defined, the search, listing and query tools will be confined to this MetaClass.

If the target MetaClass has not been defined in the metamodel of the displayed property, it can be specified in the parameterization of the element using the Target option. This option also enables specialization of an element to a specific concrete MetaClass, in this case a generic MetaAssociation.

If the target MetaClass is not defined, we consider that the object can be of all MetaClasses of the repository. In this case, the AutoCreate and AutoLink capacities cannot be effectively used.

3.5.1.8.3 Editing name and automatic behavior

When the editMenu zone is not read-only, the user can enter a name in this zone. This enables:

- specification of the search key of the "list" command.
- initialization of the object name in the "create" command.
- update execution by finding the object whose name corresponds to the entered value.

This name can however be ambiguous. On the one hand it is assimilated in the short name of the object, and can therefore correspond to several objects when the target MetaClass has a namespace. on the other, when the target MetaClass is abstract, objects of different MetaClasses can have the same names.

Validation of an entry zone in this way has therefore been globally confined to concrete target MetaClasses without namespace; when there is risk of ambiguity, zone validation is refused and the element is considered as invalid.

It is however possible to alleviate this behavior using AutoLink and AutoCreate capacities. AutoLink enables removal of ambiguity by proposing at zone validation a dialog box listing all objects corresponding to the name entered in the element; update is executed with the object explicitly selected in this list. AutoCreate enables automatic object creation if there is no object corresponding to the selected name. It is preferable to combine AutoLink and AutoCreate so as not to create the object until possible ambiguities have been resolved.

3.5.1.8.4 Options

It is possible to redefine all or part of the capacities defined in the metamodel using options on the element:

Target=<MetaClassId> enables redefinition or specification of the target MetaClass.

Capabilities=<HexaNum> enables redefinition of capacity of the property. HexaNum is understood as a hexadecimal number.

DefaultList=<QueryId> enables definition or redefinition of the query to be used to supply the drop-down list, as appropriate. This query is not necessarily a candidate.

3.5.1.8.5 Properties

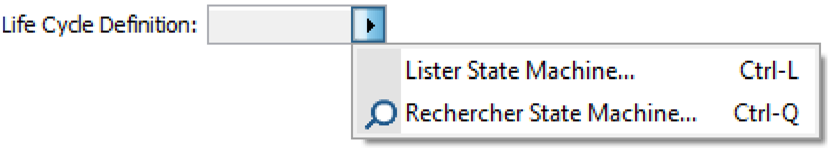
Value: displayed value. It corresponds, if its content has not been modified by the user, to the short name of the object pointed by the element in *Display* format. If the pointed absolute identifier does not correspond to an existing object of the repository, **Value** contains the ASCII value of this identifier.

TargetID: absolute identifier of the object pointed by the element. When the user modifies content of the edit zone, this property is deleted. When a menu command allowed explicit selection of a new object, this property contains the identifier of this new object. When capacity allows, the sub-menu relating to the object menu (ChildMenu) corresponds to the repository object pointed by this property. When specified, this property is used to validate the element. Otherwise, and in the absence of AutoLink or AutoCreate capacity, element validation uses the value defined in **Value**, considering it in external format. This value can be used consistently if it corresponds to the ASCII value of an absolute identifier.

3.5.1.8.6 Notifications

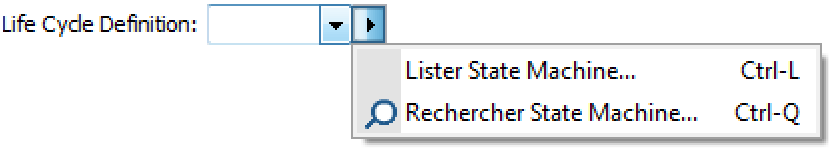
Change: sent when the edit zone has been modified.

3.5.1.9 StaticMenu


Life Cycle Definition:
Lister State Machine... Ctrl-L
Rechercher State Machine... Ctrl-Q

Element associated with properties of object type, with specific capacities. See 3.5.1.8

3.5.1.10 ComboBoxMenu

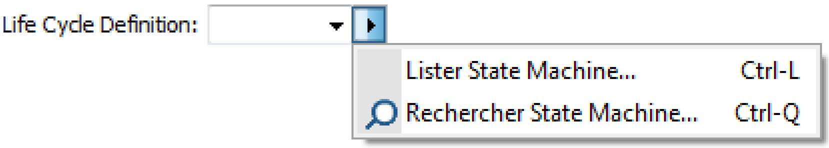

Life Cycle Definition:
Lister State Machine... Ctrl-L
Rechercher State Machine... Ctrl-Q

Element associated with properties of object type, with specific capacities. See 3.5.1.8

The first button enables expand of a drop-down list.

The second enables menu opening.

3.5.1.11 DropDownListMenu


Life Cycle Definition:
Lister State Machine... Ctrl-L
Rechercher State Machine... Ctrl-Q

Element associated with properties of object type, with specific capacities. See 3.5.1.8

The first button enables expand of a drop-down list.

The second enables menu opening.

3.5.2 Explicit AttributeControls

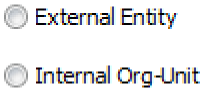
These types are never proposed as standard, and are therefore always defined in the framework of a template or an updateTool. They enable either proposal of alternative entries for certain property types, or definition of form presentation elements (titles, comments, images).

3.5.2.1 3StateCheckBox

CheckBox with third state including 'not specified' state. Can be used if you are concerned by being able to de-specify the attribute, or if you want to explicitly manage a required boolean attribute.

Options, properties and notifications: see 3.5.1.3.

3.5.2.2 RadioButtons



The RadioButtons type can replace the DropDownList type (see 3.5.1.4). The main difference is the fact that the list of enumerated values, which serves to build the list of buttons displayed, is requested just one time at element creation, and cannot subsequently be modified.
This element is presented by default without title.
When element size is not specified:

- Element height is deduced from the number of buttons to be displayed.
- Width is calculated according to the number of text characters associated with each button, as well as the number of characters of the title. Since this calculation is made in DialogUnits and cannot therefore take account of characteristics of the displayed font, the effective width may not be appropriate.

3.5.2.3 EditButton

This type is a simplified version of an EditMenu, which enables execution of only one command.

A parameterization of this element enables use of this button for file or directory selection, using FileSelection and DirectorySelection options.

In others, use of an updateTool or dialogHandler is necessary to implement the command corresponding to button press.

Options

- **Icon=<idPicture>**: enables replacement of <…> of button by a medium-sized image extracted from the indicated MetaPicture.
- **Bitmap=<idPicture>**: enables replacement of <…> of button by a small image extracted from the indicated MetaPicture.

- **FileSelection**: button press enables file selection. In this case you can use the following options to specify required operation:

  Open indicates that we want to open an existing file; if not, we want to indicate a file to be created or modified (**Save** mode by default).

  FileExists=YES|NO (NO by default) indicates if the file should exist. This option is superfluous in **Open** mode.

  DefaultExtension<xxx> enables definition of file default extension.

  Filter<xxx{;yyy …}> enables definition of filtering on file extensions listed in defining acceptable file extensions.

  PathExist=YES|NO (default YES): when in Save mode and we enter the file name in the edit zone, enables check that directories specified in the path should exist (case YES). In the case of validation, detects an error if the path references non-existent directories.

  In Windows Front-End, button press displays the file selection Windows user interface, in Save mode or in Open mode depending on the selected option.

  In Web Front-End, this check enables piloting of a file upload to the server (Open mode).- in this case the value of the property (available on the server) will contain the name of the downloaded file - or a download in the case of Save - in this case the value of the property will contain the name of the file to be downloaded; if this element is used in a wizard and has not been explicitly downloaded during execution of the wizard, the download starts automatically when the wizard finishes.

  A wizard using this type of element could therefore operate in the same way, whether in Web Front-End or Windows Front-End, in:

  - Open mode, we could consider that the value corresponding to the property is the name of an existing file which we could then open.
  - Save mode, we could use this property as being a file name to be created (or modified) by the system process; at termination of the wizard, the file will be available to the end user (who has specified the directory in Windows Front-End, or it will be downloaded in Web Front-End).

  By code, it is possible to indicate a root directory using the "**root**" field of the **Data** property. This directory will be used in Windows Front-End to initialize the user interface directory.

  DirectorySelection: button press enables directory selection. In this case you can specify using the **PathExist=YES|NO** option if we must check the existence of the specified path or not. This parameterization does not function in Web Front-End.

Properties

- **Value**: value displayed, except if **FileSelection** option is activated and we are in Web Front-End: in this case the value will correspond to the uploaded or downloaded file (this file name not known from the user interface).
- **Data**: in this case **FileSelection** or **DirectorySelection**, this property can contain a string in JSON format containing a "root" value corresponding to the initial directory.

  Example: **{ "root" : "D:\MyDirectory"** }

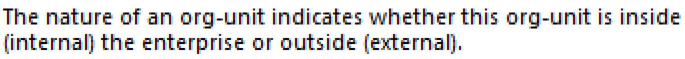
  If this variable is not in JSON format, we consider that it contains the directory name.

  This value is consulted at Windows user interface opening, enabling file or directory selection.

Notifications:

- **Change**: sent when the edit zone has been modified.
- **Click**: sent when the button is clicked.

3.5.2.4 HelpComment



The nature of an org-unit indicates whether this org-unit is inside (internal) the enterprise or outside (external).

This element is used to display a help or explanation in a form. It is designed to display a text on several lines. It can display rich text. It does not enable data entry and is read-only.

This text can be the comment of a system repository object; in this case the element identifier corresponds to the identifier of this object, and can therefore point an object of any type.

It can also, by option, display a text. In this case the element identifier corresponds to that of the text to be displayed.

In Windows Front-End, it can enable navigation to other system repository comments if the system objects are referenced in the form of a field in the initial text.

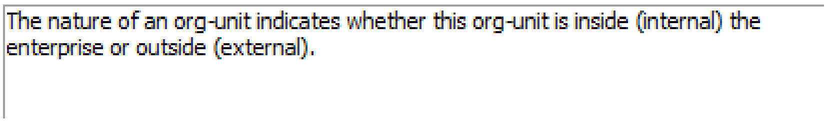
This element is presented by default without title.

Options:

TextMode: indicates that the element must display a text of the object pointed by the element; this text corresponds to the element identifier. Otherwise, the comment of the system object corresponding to the element identifier is displayed.

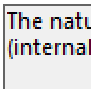
The following options are not taken into account in Web Front-End:

Edge: displays the comment in the form of a simple framed text. In this case the following options are ignored, and navigation by field is deactivated.



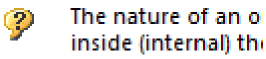
The nature of an org-unit indicates whether this org-unit is inside (internal) the enterprise or outside (external).

Border: the comment is framed.



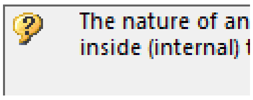
The natu
(internal

Icon or **Info**: the help icon is displayed alongside the comment.



The nature of an o
inside (internal) th

Icon and Border can be associated.



The nature of an
inside (internal) t

Properties:

Data: text content in ASCII format.

Value: text content in HTML format.

3.5.2.5 Button

Display of a clickable button enabling execution of an action.

If the element identifier points to an _Operator of method type, or a macro implementing a method, the method is executed when the button is pressed.

This element is presented by default without title. The name displayed in the button corresponds to the value of the title, except if the button displays an image.

Options:

NoCall: the method is not called when the button is pressed. Used when the action triggered by the button is processed by the **Click** notification, and not by the method.

Icon=<idPicture>: enables replacement of the name displayed in the button by a medium-sized image extracted from the indicated MetaPicture.

Bitmap=<idPicture>: enables replacement of the name displayed in the button by a small image extracted from the indicated MetaPicture.

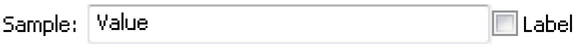
When the button displays an image, the size of the button should correspond to that of the image, and is therefore expressed in pixels. The button is therefore SCREENUNITS style. If the size of the button is specified in the template, it will be considered as a size in pixels.

Units=Dialog: in the case where the button displays an image, enables sizing of the button in DialogUnits by deactivating the SCREENUNITS style; this can be used if the button must be aligned with other elements. In this case its size can conform to that of the other elements.

Notifications:

Click: sent when the button is clicked.

3.5.2.6 MegaEditCheck



A MegaEditCheck includes 3 areas:

- a value, which could be editable
- a check box, which could be clickable
- a label.

It is without exception necessary to implement an UpdateTool to be able to use a MegaEditCheck. Without this, a MegaEditCheck behaves like a simple edit area: label value is not specified and check box is not used.

To consult or modify label and check box values, the Data property of the AttributeControl should be used.

This property describes the value and status of these areas in the form of a character string.

It is of form

"label":"<Label>","check":"<checkValue>","show":"<showValue>"

- <Label> corresponds to the string displayed in the label static area.

- <checkValue> corresponds to check box value and can be C (checked), U (unchecked) or I (indeterminate).
- <showValue> enables configuration of check box display and can be E (enabled), D (Disabled) or H (Hidden). In this last case, the check box is not visible.

As indicated in paragraph **Error! Reference source not found.**, it is possible to modify only one of these elements. For example AttCtl.Data = """check"":""C""" forces check mark display without affecting the values of "label" or "show"

If you want to access specific information of the Data string, you can use a MegaToolkit function: JSONStringSearch

```
Set myToolkit = myRoot.CurrentEnvironment.Site.Toolkit

CheckValue = myToolkit.JSONStringSearch("{" & AttCtl.Data & "}", "check")
```

CheckValue contains C, U or I.

Properties:

Value: edit area content

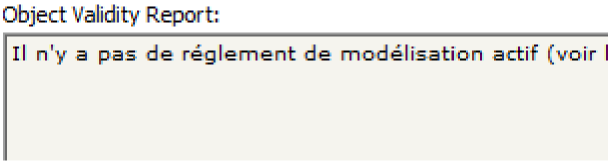
Data: check box and label content and value (see above)

Notifications:

Change: sent when the edit zone has been modified.

Click: sent when the check box is clicked.

3.5.2.7 Viewer



Object Validity Report:

Il n'y a pas de réglement de modélisation actif (voir l

A viewer enables display of an HTML Formatter in a page; it is presented in the form of an HTML browser inserted in the page.

By default, the identifier associated with the element corresponds to the identifier of the formatter; this formatter is launched on the object pointed by the element.

You can also directly launch a macro to generate this HTML content by means of **DirectMacro option**. In that case the identifier associated with the element corresponds to the identifier of the macro to be executed. This macro must implement the **Generate** or **GenerateStream** method which will be call to generate the HTML content of the viewer.

```
Sub Generate(obj As MegaObject, ctx As MegaObject,data As String,strout As String)

Sub GenerateStream(obj As MegaObject,ctx As GenerationContext,data As String,stream)
```

Data corresponds to the Data value of the AttributeControl.

This system enables to specify viewers that run on informal objects, associative objects, or whose content can depend on the association browsed from the object pointed by the element, as appropriate.

Formatter execution is asynchronous.

The element is displayed by default with its title above.

It has style BOTTOMALL and therefore is deformed to use all the page when it is the last element.

Being an HTML component, this element offers great freedom in what can be displayed. However, it is advisable to consult HTML formatter documentation if you want to trigger interactions that can operate in Web Front-End and in Windows Front-End.

However, if you want to use a viewer to display systems designed to execute updates, it is advisable to avoid direct update execution, which is not easy in Web Front-End. By configuration it is possible to synchronize updates from the viewer with form validation. The principle used is to designate a Macro to handle updates from the viewer at form validation. To do this, an interaction mode between the viewer and its container is defined.

Interactions are by means of a callback function, which should be presented in the header of the HTML document. This function has (javascript) as prototype

```
function onContainerCmd(prm,attctl)
```

prm is a character string indicating the interaction type

attctl is a parameter referencing the viewer container in Web Front-End. In Windows Front-End, this parameter is empty. In particular it enables identification of whether the use context of the HTML document is Web Front-End or Windows Front-End.

Calls to this function are carried out if the following options are present:

Initialize: the callback function is called at initialization of the form, with value prm = "initialize".

Refresh: indicates that the viewer should not be recalculated at refresh; instead of recalculation, the onContainerCmd function is called, with value prm = "refresh".

Appliable: indicates that callback should be called at form validation; the onContainerCmd function is called, with value prm = "apply". By default, this function is only called if the element is ‘‘dirty’, that is considered as modified. It is however systematically called at each form validation if the **alwaysCall** option is specified. In the current version, this option is essential for Web Front-End operation.

The onContainerCmd("initialize" function is systematically called when any of the options enabling callback call is specified.

When calling onContainerCmd("apply", it is possible to specify the **Data** property of the element to control updates to be executed.
To be usable, this property must contain a JSON format character string including the macro to be called on the server:

```
"applyMacro":"<MacroID>"
```

At validation on the server, if this field is found in the **Data** property of the element, the <MacroID> macro is instanced and we call on this macro the function

```
Function InvokeOnObject(object,idText,Page)
```

Where **object** is the object of the AttributeControl, **context** the component corresponding to the AttributeControl, and **param** the JSON string corresponding to the **Data** value containing data allowing the macro to know updates to be executed.

To specify the Data property from the onContainerCmd("apply",attctl) function, proceed differently according to whether you are in Windows Front-End (attctl = "") or Web Front-End context.

- In Windows Front-End, you can directly update the AttributeControl using the context object:

```
external.ContextObject("PageContext").SetData(result.data)
```

- In Web Front-End, the value returned by the onContainerCmd function should be a JSON format string. Present in this string can be the fields:
  - data: is assigned to the **Data** property.
  - reload: causes reloading of the formatter if assigned to True.
  - errorMessage: is assigned to the **ErrorMessage** property and cause – if not null – an error on the page.

The **context** component, accessible in Windows Front-End from the **external** object and in the InvokeOnObject function of the validation macro, has the following properties:

.setData(string): updates the **data** property of the AttributeControl.

.setDirty(bool): generates an "Update" notification if true and positions the dirty value.

.setError(string): updates the **ErrorMessage** property of the AttributeControl. If this string is not empty, indicates that the element is in error and the form cannot be validated.

.refresh(): refreshes the viewer.

.notify(action): generates a notification on the page.

The following example proposes an implementation enabling generic update of object properties via a viewer.
To do this, consider that the validation macro uses an ApplyParameter field of JSON Data. This property should be an object table, each object containing the following fields:

id: <identifier of property to be updated>

value: <property value>

format: (optional) update format

The validation macro is implemented in javascript to use a JSON format parameter as simply as possible It browses the table defined above and executes the corresponding update.

```
//Language:javascript
function InvokeOnObject(obj,device,param){
  obj.getRoot().print(param);
  var vparam = eval("(" +param + ")");
  obj.getRoot().print(vparam.applyMacro);
  for(var i = 0; i < vparam.applyParameter.length; i++) {
    var itemObj = vparam.applyParameter[i];
     obj.getRoot().print("ID=" + itemObj.id + ", VALUE=" + itemObj.value + ",
FORMAT = "+ itemObj.format);
    obj.SetProp(itemObj.id, itemObj.value , itemObj.format);
  }
}
```

This macro having been defined, you must now create an HTML document able to manage this interaction. This document is displayed and will enable update of ShortName and Cardinal MetaAttributes of a MEGA object (for example a Site, an Operation, an Org-Unit or any other MetaClass using the Cardinal MetaAttribute).

The HTML document of this example is of form:

```
<hmtl>
  <head>
    <script language='javascript'>
      function onContainerCmd(prm,attctl) {
        var fieldVal1 = document.forms[0]['input1'].value;
        var fieldVal2 = document.forms[0]['input2'].value;
      var result = {'reload': 'true','errorMessage':'null','data':
        {'applyMacro':'#MACROID#','applyParameter':
            [{'id':'~MtUi79B5iyF0[Cardinal]','format':'','value':fieldVal1}
            ,{'id':'~Z20000000D60[Short Name]','format':'','value':fieldVal2}
            ]}};
        if(attctl==='')
          external.ContextObject(""PageContext"").SetData(result.data);
        alert('test:'+prm+':'+attctl);
        return result;
      }
</script>
  </head>
  <body>
    <form>
       <table border='0' align='CENTER'>
                   <tr><td><b>Cardinal      :</b></td><td><input      id='input1'
type='text'
               value='"#CARDINAL#"' /></td></tr>
         <tr><td><b>Short Name :</b></td><td><input id='input2' type='text'
               value='"#SHORTNAME#"'/></td></tr>
       </table>
     </form>
  </body>
</html>"
```

In this HTML, #MACROID# should be replaced by a field representing the absolute identifier of the validation macro, #CARDINAL# by the value of the Cardinal MetaAttribute for the form object - `oMegaObject.GetProp("~MtUi79B5iyF0[Cardinal]")` and #SHORTNAME# by the value of the short name - `oMegaObject.GetProp("~Z20000000D60[Short Name]")`

This HTML document generates a warning at validation.

For operation

Options:

HideStatusBar: in Web only. Hides the status bar (and therefore the button enabling element refresh).

Initialize, Refresh, Appliable: indicates presence of an onContainerCmd(prm,attctl) callback function in the HTML document header, enabling interactions between the HTML document and the form.

alwaysCall: when the **Appliable** option is defined, indicates that the validation macro must be called, even if the component is not considered as being modified. This option is currently necessary for Web Front-End validation operation, since it is not possible to dynamically modify component status.

Properties:

Value: HTML content of viewer (currently non-operational)

Data: contains a string in JSON format containing data on interaction between the HTML document and the validation function (see above).

Specific component

The specific component accessible from the AttributeControl object includes the interaction functions described above.

SetData(string): attctl.component.SetData string corresponds to attcl.data = string

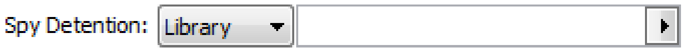
SetDirty(bool)

SetError(string): attctl.component.SetData str corresponds to attcl.errorMessage = str

Refresh()

Notify(action)

3.5.2.8 ComboLinks



This element enables creation of the "owner" of an object. It therefore handles the problem of a data entry containing exclusive MetaAssociationEnds, from card-max to 1. The MetaAssociationEnds list included in the drop-down list is discriminated by an _operator of "compound" type, the MetaAssociationEnds presented being tagged "deep" for this operator.

This element is not designed to be overloaded.

Properties:

Value: edit area content (corresponds to short name of associated object)

TargetID: Identifier of associated object

SourceID: identifier of selected MetaAssociationEnd.

Notification:

Change: sent when associated object is changed.

3.5.2.9 Shape

The Image AttributeControl enables display of an image, possibly followed by a static area.

It enables representation of an object; to do this you can define a specific object for this element.

Click on an image can trigger an action, which can be to display the object menu.

The image displayed is a MetaPicture format 32x32, corresponding

- to the image of the object associated with the element with the **ImageFrom:Object** option
- to the image associated with the property of the element with the **ImageFrom:Attribute** option. In this case, if the property is MegaIdentifier type, it is considered as containing the identifier of a *MetaPicture* which is then displayed. If not, it should be an enumerated property of which we display the *MetaPicture* associated with the enumerated value corresponding to the value of the property.

The static area is not displayed if the **Value:void** option is specified. If not, it corresponds

- to the title of the area with the **ValueFrom:Title** option
- to the value of the property of the element with the **ValueFrom:Attribute** option
- to the value of a property of which we specify the field with the **ValueFrom:<PropID>** option

When the area displays its title, or when the **Value:IncludeTitle** option is specified, the value of the static area is concatenated with the title.

When we click the image, no action or notification is triggered with the **Clik=Inactive** option.
The **Click=PopupMenu** option displays the object menu.

If you want to display by this element an object seen from the main object of the page, creation of a map is not necessary. You need only specify the collection browsed with the option **FromCollection:<CollectionID>{TypeName|ClassName}** and associate with the element the absolute identifier of the object to be displayed. Most often this requires call to a page generator (see 3.3.1.6). In this context it is possible to include the name of the object type (TypeName) or the name of its MetaClass (ClassName).

This element is present by default without title; it being possible to include the title in the value. Its default style is SCREENUNITS, meaning that its size is expressed in pixels in the configuration (unless otherwise specified).

Options:

FromCollection:<CollectionID>{TypeName|ClassName}: displays image of an object seen from the collection

Click={Inactive|PopupMenu}: disables the click notification, or displays the object pop-up menu

Value :{void|IncludeTitle}: specifies display of the static area: hidden or preceded by the element title.

ValueFrom:{Title|Attribute|<PropID>}: specifies the displayed value (area title, element attribute, or particular property of the object indicated by its <PropID>) field.

ImageForm:{Object|Attribute}: specifies if the image is the image of the object or corresponding to the image associated with the property of the element.

Units=Dialog: enables dimensioning of the image in DialogUnits by disabling the SCREENUNITS style; this can be used if the image should be aligned with other elements – in this case its size can conform to that of the other elements.

Properties:

Value: content of the static area

Data: enables definition or determination of the image to be displayed, in form

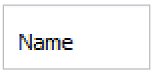
"image":"<moniker>"

The moniker generally corresponds to the identifier of the MetaPicture, preceded by character '~'

Notification:

Click: sent when the image is clicked

3.5.2.10 Label


Name

Displays a label. As standard, this label corresponds to the name of the object whose identifier is associated with the element. It is however possible, by option, to display the element title or a property value.

This element is presented by default without title.

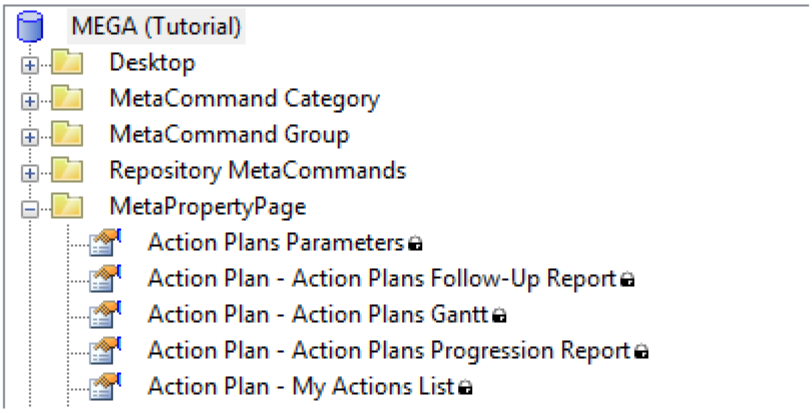
Options:

ValueFrom:{Title|Attribute|<PropID>} indicates that the displayed value corresponds to the element title, to the property value associated with the element, or to another property of which we pass the identifier in the form of a field.

Properties:

Value: content of the static area

3.5.2.11MetaTree


MEGA (Tutorial)
Desktop
MetaCommand Category
MetaCommand Group
Repository MetaCommands
MetaPropertyPage
Action Plans Parameters
Action Plan - Action Plans Follow-Up Report
Action Plan - Action Plans Gantt
Action Plan - Action Plans Progression Report
Action Plan - My Actions List

This AttributeControl enables display of a MetaTree in a properties page.

The identifier associated with the element corresponds to the identifier of the MetaTree. The root of this MetaTree is the object pointed by the element.

A MetaTree can be used to control a Map.

MetaTree options are described in a JSON format character string (character case should be scrupulously respected). Note that an option string containing a bracket must be specified using a Parameter.

Example:

```
treeParam = Parameter({"metalist":"OEb0XL0zF93Q","checkbox":true})

testTree = Item(~ulSYFlOK1XB0[Objects]),Control(MetaTree),Param(@treeParam)
```

The JSON can contain the following parameters:

"checkbox":boolean enables to have check boxes opposite each element. The default value is false.

"selPropagate":{"mode":string,"direction":string,"autoExpand":string} This option is only effective in the framework of a checkbox tree, and enables definition of the propagation mode of a selection of elements in the tree.

mode can be **on**, **off**, **defaultOn**, **defaultOff**. In the two default modes, a button allows the user to enable or disable this behavior as required.

direction can be:

down (default value): when a node is selected or deselected, all children are immediately selected or deselected. Behavior is recursive and should therefore only be activated on trees that are truly hierarchical (not of cycle).

up when a node is selected, all parents are selected, up to the tree root. When a node is deselected, its parents are deselected, except those that have another descendant selected.

both combines both behaviors.

autoExpand can be **on**, **off** (default), **defaultOn, defaultOff** but is only valid for descending propagation. This enables automatic expand of nodes to show selected child elements, with or without a button available to the user. This option should be used with care, since expanding the tree can be costly in terms of time.

Propagation of the selection is not supported in Windows Front-End.

"metalist":string: enables definition of additional columns on the MetaTree. The parameter supplied is a MetaList identifier, of which MetaFields will be interpreted as tree columns. MetaField now inherits from Abstract Property. This enables connection by the Implementation MetaAssociationEnd of a macro implementing the calculated attributes interface.
When value of a MetaField with macro is requested in the framework of a tree, the MegaObject is enhanced by addition of a "virtual" attribute "~MJoWUFOzF92Q[Tree Node Full Path]", which enables access to the full path of the object in the tree, formatted in the same way as the selection.

In Web Front-End, in-place edit is possible in columns by double-clicking the value you want to modify.
Limitations:

Presentation in columns is available in Web Front-End only.

In Windows Front-End, these fields are concatenated with the name of the element.

MetaFields of columns operate only for MetaFields based on a macro or on a MetaAttribute (not MetaAssociationEnd or query)

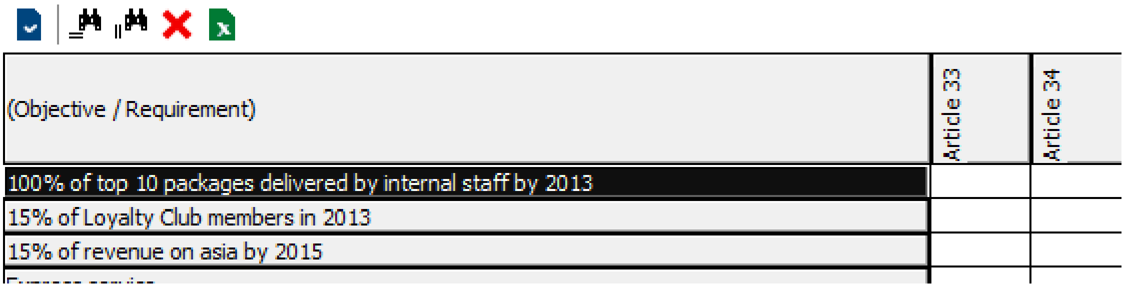
Properties:

Value: lists selected objects. This value is a JSON format character string that gives the full path of objects in the tree. In assigning the value we modify the selection.

Notifications:

sent when the selection changes.

3.5.2.12Matrix



(Objective / Requirement)	Article 33	Article 34
100% of top 10 packages delivered by internal staff by 2013		
15% of Loyalty Club members in 2013		
15% of revenue on asia by 2015		

This AttributeControl enables display of a Matrix in the properties page. This matrix can be either display of the content of a Matrix or System Matrix occurrence, or display of the matrix resulting from application of a matrix template to the object pointed by the element (default mode).

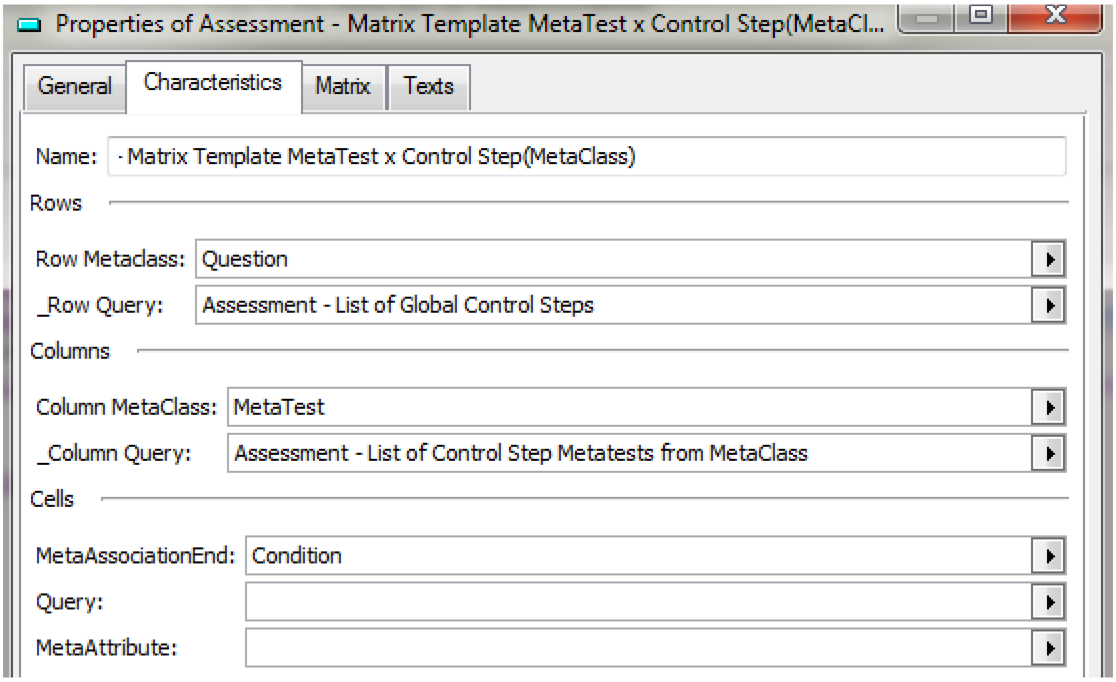
To display content of a Matrix (or System Matrix), the **Object** option should be present – in this case the object pointed by the element is the matrix, and its content will be displayed – or the **Data** option – in this case the identifier associated with the element corresponds to the occurrence of the matrix to be displayed.

- In the case of the Object option, the identifier associated with the element is not taken into account; in the case of the Data option, content of the matrix does not depend on the object pointed by the element.
- The Matrix (or System Matrix) occurrence contains the list of selected rows and columns as well as those that may have been added using addition buttons. These collections are serialized in a technical attribute of the matrix at validation of the element. The matrix occurrence also memorizes display modifications (for example column width).

In the case of matrix template display, the identifier associated with the element corresponds to the identifier of the Matrix Template to be used. A matrix template enables definition of:

- an object collection defining rows of the matrix. This collection is obtained from the object pointed by element.
- an object collection defining columns of the matrix. This collection is obtained from the object pointed by element.
- a MetaAssociationEnd defining cell content: cell content corresponds to the association between column object and row object according to this MetaAssocationEnd.
- a MetaAttribute of the MetaAssociation defining the value displayed in the cell: if this attribute is not defined, the cell will display a checkbox indicating link presence or absence.

- It is possible to specify a query instead of a MetaAssociationEnd: in this case we display in the cell a checkbox indicating presence of the row object in the collection obtained from this query from the column object.


Properties of Assessment - Matrix Template MetaTest x Control Step(MetaCl...
General
Characteristics
Matrix
Texts

Name: - Matrix Template MetaTest x Control Step(MetaClass)

Rows

Row Metaclass: Question

_Row Query: Assessment - List of Global Control Steps

Columns

Column MetaClass: MetaTest

_Column Query: Assessment - List of Control Step Metatests from MetaClass

Cells

MetaAssociationEnd: Condition

Query:

MetaAttribute:

Unlike the case of matrix display, the list of rows and columns corresponds exactly to the content of row and column collections. You must be able to modify content of these collections to be able to modify the list of displayed rows and columns.

Options:

NoBar: the matrix toolbar is not displayed.

Object, Data: indicates that the matrix displays content of a Matrix or System Matrix occurrence.

AutoLink: simple click on a cell enables creation or deletion of the corresponding association.

Component methods:

Specific methods of the AttributeControl can be called from the component object (AttributeControl.Component)

.Fetch(Input As String) As String: this function obtains complete content of the matrix in the form of a JSON format string. The input parameter is ignored.

The JSON restored by this function contains two elements:

- an **hdr** element containing description of the matrix, and
- a **content** element containing the rows.

In **hdr**, columns are presented as a **columns** table of form

```
{ "id": "<idcol>", "format": "<format>" , "image":bool , "name":name };
```

<idCol> contains absolute identifier of the column element.

Also included are elements **columnTotal** (number of columns), **total** (number of rows), **cornerName** (title, displayed in cell top left corner) and **checkBoxCell :true** if cells include checkboxes, indicating presence of the association.

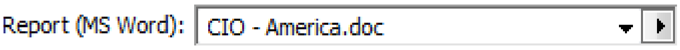
content is a table, each element including a row, of form

```
{ "id" : "<idLine>", "image": "<image>", "<columnid>": "<value>" … }
```

idLine corresponds to the identifier of the row object; only columns effectively specified are present,

<columnid> being the column object identifier and value to be displayed.

3.5.2.13DropDownSelection



Report (MS Word): CIO - America.doc

This element enables display of content of a selection in the form of a drop-down list. The menu button enables addition of elements to the collection, and access to the menu of the object selected in the list.

The identifier associated with the element is the identifier of a collection (MetAassociationEnd, query or other abstract collection).

Although of appearance identical to a DropDownListMenu (**Error! Reference source not found.**) the two elements are fundamentally different.

- The DropDownListMenu manages an Object type property. The drop-down list enables display of update proposals. Edit area content corresponds to the value (current or after validation) of this property. This element resembles an edit area containing update help features.
- The DropDownSelection manages an object collection directly associated with the object of the element. Edit area content is an object selected in this collection. This element is similar to a single-selection list view presented in compact form.

Like a ListView, a DropDownSelection enables updates on the collection (occurrences addition and deletion). It also enables (it is even its main use) control of a Map pointing to the object selected in the collection.

This element is not designed to be overloaded; in particular it does not generate public notifications. You can however use private notification of number 1, corresponding to change of selection. You cannot directly change the selected element in the list.

Options:

- **NoMenu**: in this case, the menu button is not displayed and the element has the appearance of a simple drop-down list.
- **SelectOnRefresh**: indicates that the collection should be recalculated when the element is refreshed; to be used when we want to take into account collection updates when corresponding to a query – in particular an ERQL query.

Properties:

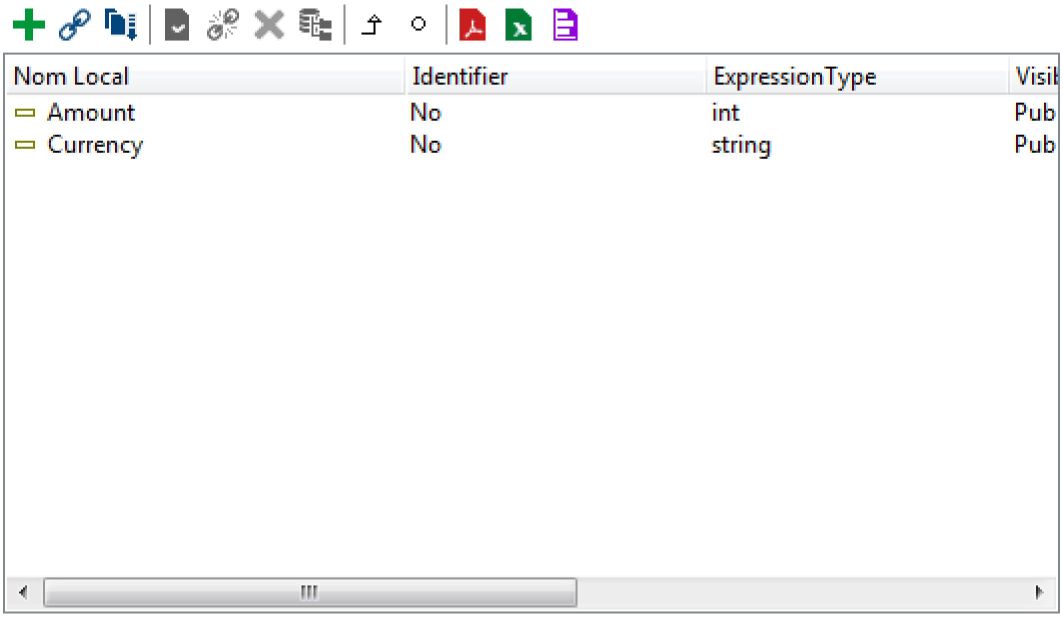
- **Value**: edit area content (corresponds to short name of selected object) This value cannot be modified.
- **TargetID**: Identifier of selected object This value cannot be modified, (except by command SELECT-<id> from the properties page).

3.5.3 Composite AttributeControls

Composite AttributeControls enable presentation of complex data. To define this data, you may decide to associate it with sub-AttributeControls using a composition Map.

Example: sub-AttributeControls, amongst other things, enable definition of columns of a ListView.

3.5.3.1 ListView



Nom Local	Identifier	ExpressionType	Visil
▭ Amount	No	int	Pub
▭ Currency	No	string	Pub

A ListView enables display of an object collection. This collection is obtained from the object associated with the ListView element. The columns – properties of displayed objects – can be configured. A ToolBar is defined by default for the ListView, enabling triggering of actions relating to the collection (creating a new element for example) or to an element selected in the list. The ListViews can be single-selection or multiple-selection. Finally, it is possible to define a ListView displaying several selections, known as alternative selections.

A ListView can control a map: in this case the object selected in the list is assigned to the controlled map, and therefore becomes the object associated with items contained in this map.

On this subject, the ListView content map (enabling ListView specification) should not be confused with the map controlled by the ListView (enabling object path definition).

In the basic case, the identifier associated with the ListView is that of the collection to be displayed: the list displayed corresponds to the path by api of the collection `obj.GetCollection(<ItemID>)`

3.5.3.1.1 ListView "content" map

The listview content map enables definition of buttons, columns and alternative collections. Options defined for these elements indicate for which use it is reserved:

- An element corresponding to a toolbar button contains the "**PushButton**" option.
- An element corresponding to an alternative collection contains the "**AlternateSelection**" option.
- An element corresponding to an additional column contains the "**Extra**" option.

- The other elements are considered as column overloads.

3.5.3.1.2 Configuration elements

3.5.3.1.2.1 Specification of displayed columns

Displayed columns of a ListView are obtained from description of the displayed collection; it is possible to define filtering so as not to display all properties defined for this description. For information, it is possible to obtain by api the elements of this description by `obj.GetCollection(« ItemID »).GetTypeObject.Properties`

With each column is associated the identifier of the property represented. This identifier is unique, so two columns displaying the same property cannot be inserted.

If no column filtering option is present, all columns of the description are displayed, except:

- columns corresponding to Properties not visible according to metamodel access of the current user
- columns corresponding to MetaAttributes hidden at MetaClass level for the profile of the current user
- columns corresponding to translations of MetaAttributes (only the root MetaAttribute is displayed, corresponding to display of the value in the user language)
- columns corresponding to an administration property, a local property, or a property hidden in edit – corresponding to flags "in administration tab" (0x4), "never appears in list column" (0x1) and "hide on edit" (0x20000000) of the "extended properties" attribute.
- The "ShortName" attribute - we directly display the attribute that substitutes the name, beside which ShortName would be redundant.

If this list is not suitable, you can define generic filtering using the following options:

NoDefaultColumn: in this case, only the naming attribute is displayed; this is the one serving as name for description of the collection. It can then serve:

- as name if the name attribute is not substituted, or if the collection displays the long name by default. This is determined by the "**Name Appearance**" MetaAttribute which is read either on the browsed *MetaAssociationEnd* or on the *MetaClass*.
- as the attribute that substitutes the name if the collection is configured to display the short name. The name of this attribute generally being esoteric, we give it the conventional name ("Local Name").
- of the ShortName conventional attribute for collections derived from an abstract class or calculated
- of another attribute in the case of informal collections for which a naming attribute has been explicitly defined.

StandardColumn: filtering is identical to default filtering, but only the default columns (properties having enabled the "Default Column" (0x2) flag in their "extended properties") are visible by default.

AssocColumn: besides the naming attribute, the listview displays the properties defined on the browsed MetaAssociation.

More precise filtering can be defined by overloading columns in the ListView content map. When the identifier of a content map element of overload type (that is, not having as option "**Extra**", or "**AlternateSelection**", or "**PushButton**", or "**Cookie**") corresponds to a property identifier of the collection description, a new configuration is applied on this column according to the following rule:

- If the element is hidden (keyword **HiddenOn**()), it is filtered. Note that if the hide condition is likely to change at object update, the columns list should be recalculated at ListView refresh; this is not done automatically, and to do this you must supply the ListView (and not the element) with the **RefreshColumns** option.
- otherwise, and if the column was filtered by default, it is made visible; this is not however done if filtering followed metamodel filtering or was defined by user rights.
- If the element is disabled (keyword **DisabledOn**) the column is not forced in read-only mode
- If the element is defined with keyword **Title(Up),** the column title is replaced by the **Name()** of the element.
- If the element is defined with a particular size (**Size(x)**) this size is used as initial size of the column. If the size is empty (**Size(0)**) the column is initially hidden.
- If the element is defined with a **Control(CheckBox)** it is displayed in the form of a check box; with a **Control(ComboBitmaps)** it is displayed preceded by its image.

Columns not included in the description can be added, known as "ExtraColumns". These columns are defined by content map elements provided with the **Extra** option. See 3.5.3.1.2.4. Finally, you can define columns specifically calculated for the ListView which are also content map elements, but provided with the **Cookie** option.

Default columns appear in the order defined for the description; it is however possible to reorder all or certain of the listview columns using the **ReOrder** option. When this option is specified, it is content map element declaration order that is used to sort columns, elements not obtained from this map being listed afterwards. Note that the name is always placed first.

3.5.3.1.2.2 Images associated with listed objects

By default, the first element of each line in the list is the image of the listed object. If you do not want this image to appear, specify option **NoLineBitmap**.
The image to be displayed can be redefined, either by stamping or by replacement. To do this, you must have available on the list element one or several properties for which an image is defined; these can be:

- enumerated properties of which values are associated with a MetaPicture
- calculated properties of megaidentifier type, of which values correspond to a *MetaPicture*.

To replace the image of the line:

➔ specify the **BitmapSelector=*Field*** option where <*Field*> is a reference to the property to be used.

To stamp the image of the line:

➔ specify the **BitmapStamper=*Field {,Field}*** line. Several stamps can be used. Stamped images do not replace the list image, but are placed above transparently.

3.5.3.1.2.3 Element sorting

At initial display, elements in a ListView are sorted:

- according to their order number
- at equal order number, according to name.

The order number and name are properties defined as those of the collection description. Name calculation has already been mentioned in section 3.5.3.1.2.1 and mainly depends on the target MetaClass and the collection browsed. The order number is the 'Order' MetaAttribute when the collection displayed is a MetaAssociationEnd. When it is an ERQL Query – which does not allow order number retrieval – the object creation date is used.

To sort initially in lexicographical order, that is not use the order number and sort directly on the name:

➔ specify the **NameSort** option.

To redefine the attribute used to determine order number:

➔ specify the **OrderOverloadProp=*PropID*** option.

3.5.3.1.2.4 Definition of Columns outside description

You can insert in a ListView columns not included in description of the browsed collection. These columns are represented by listview content map elements, with options **Extra** or **Cookie**.
An **Extra** column has a corresponding property identifier which should be accessible for all listed elements, independent of the ListView. Similarly, if you define the following element in the content map:

```
Extracol = Item(PropID),From(contentMap),Param(Extra)
```

Then for all elements listed in the collection, it should be possible to execute (in script)

```
oListItem.GetProp(«<PropID>»)
```

It is for this reason that these columns are not in principle MetaAttributes, since these are normally included in descriptions derived from the metamodel. However, it is possible to include _AbstractProperties (generally TaggedValues).

- If listed elements are 'Elements with TaggedValue' or 'System Elements with TaggedValue', all non-calculated _AbstractProperties can be applied to them.
- If an _AbstractProperty is calculated, the possibility of inclusion as a column depends on its implementation, and more specifically on its compatibility with the browsed elements.

You can include a column whose calculation is specifically executed in the ListView: in this case, the content map element has the **Cookie** option, and other elements enabling definition of the calculation mode of this column as defined below.

```
Param(Cookie{,BasedOn=BaseId}{,IsTrigger=n},DataSource=DSId{,DSOptions})
DSOptions that can be: From:CollectionId
                        Path:ObjectPath
                       Macro:MacroId InitString
```

The principle of this calculation is to search for the column value not on the line object, but in another collection, called here DataSource.
In the standard case, this collection is constituted from the source object of the ListView.

To obtain the value of the column, we first search this DataSource for an object whose identifier corresponds to that of the line object.

If the column is defined as trigger (keyword **IsTrigger=1** or **IsTrigger=2**), the collection derived from DataSource is managed as a link controlled by the column. If **IsTrigger=2**, the column does not reference a property, but is considered as a boolean value, displayed by a check box, whose value indicates presence of the object in the DataSource.
In other cases:

- if the object is not present in the DataSource, the column is unspecified.
- otherwise, value of the column corresponds to the value of a property of the DataSource object, corresponding either to the absolute identifier defined in <BaseId>, or by default to the absolute identifier of the column.

The following is the equivalent of the column value calculation in script.

```
' oRootObject  : object associated with ListView  (ListView.GetObject)
' oLineObject : object corresponding with line
' sColumn : column value

Dim cDataSource as MegaCollection
Set cDataSource = oRootObject.GetCollection("<DSId>")
  'this example relates to a simple DataSource, that is without <DSOptions>

Dim oDataSourceObject
Set oDataSourceObject = cDataSource.Item(oLineObject.GetID)
If oDataSourceObject.Exists Then
 if "<BaseId>" = "" Then
  sColumn = oDataSourceObject.GetProp("<ColumnId>")
 Else
  sColumn = oDataSourceObject.GetProp("<BaseId>")
 End If
Else
 sColumn = ""  ' No reference to object in DataSource, column not specified
End If
```

Definition Options of a DataSource enable management of a DataSource not corresponding to a collection directly accessible from the source object of the ListView

From :*CollectionId* enables access to the DataSource collection indirectly from a link considered unique. It could be that the DataSource collection is not defined.

```
Dim cDataSource as MegaCollection
Set oInter = oRootObject.GetCollection("<CollectionID>").Item(1)
If oInter.Exists then
 Set cDataSource = oInter.GetCollection("<DSId>")
Else
 Set cDataSource = Nothing ' in this case DataSource is not defined
End If
```

Path:*ObjectPath* In this case the DataSource collection is not obtained from the source object of the ListView, but from the object of which we specify the *MegaPath*. It could therefore be that the DataSource collection is not defined if the MegaPath does not correspond to an object.

```
Set oInter = oRoot.GetObjectFromPath ("<ObjectPath>")
If oInter.Exists then
  Set cDataSource = oInter.GetCollection("<DSId>")
Else
  Set cDataSource = Nothing ' in this case DataSource is not defined
End If
```

Macro:*MacroId InitString* enables access to the DataSource collection from a Macro implementing a standard collection as follows:

```
Function GetStandardCollection(     MegaObject as MegaObject,
RetType as Variant,
InitString as String) as MegaCollection
```

MegaObject is the source object of the ListView

InitString is the character string specified in the option

RetType is an optional parameter enabling overloading of the returned collection type, when it is specified with an identifier corresponding to a collection type.

When a column derived from a DataSource can be modified, update affects the DataSource collection. If the column is not defined as being a trigger:

- When an object corresponding to the line element exists in the DataSource, the value of the property of this object corresponding to the column is updated with this value.
- When this object does not exist, we try to insert it in the DataSource collection by providing the identifier of the line object (except of course if the update value is empty: in this case we do nothing). If creation is successful, we update the property corresponding to the column of this new object with the update value.

If the update column corresponds to a trigger:

- In the case of an indicator (**IsTrigger=2**), we insert or remove the collection object according to the boolean value of the update (nothing is done if the value supplied corresponds to the calculated value).
- In the case of a simple trigger, update with an empty value removes the object from the collection if it was included; in other cases, update corresponds to the standard case (insertion if required and property update).

In the case where the DataSource collection is not defined, update fails.

3.5.3.1.2.5 ToolBar content specification

A ListView toolbar includes:

- standard buttons, which can be filtered
- specific buttons, which should be defined in the content Map.

The ListView toolbar can be hidden using the **NoBar** option.

Standard buttons of a ListView are:

Create (**C**): Starts the tool for creation of a new element in the list.

Link (**L**): Starts the tool for creation of a new element in the list (connect).

Reorder (**R**): Starts the tool enabling reordering of elements of the list according to standard order (see 3.5.3.1.2.3).

The following buttons concern the occurrence selected in the list:

Destroy (**D**): Starts the tool to delete the selected object.

Unlink (**U**): Starts the tool to remove an element from the list (disconnect).

Properties (**P**): Display of the properties dialog box of the object selected in the list.

Explore (**E**): Starts the explorer on the selected object (available only on Windows Front-End).

Open (**O**): Executes the default command of the menu of the selected object. This command is not present by default in the toolbar.

Standard buttons are automatically disabled when the action concerned is not possible or prohibited.
To filter standard buttons use the **ToolBar[]** option. Between the toolbar brackets are the names (or abbreviations) of buttons concerned, preceded by symbol "+" if you want to show these, and "-" if you want to hide them.

Example: the **ToolBar[-RDU]** option specifies hiding of buttons Reorder, Destroy and Unlink.

When a standard button is hidden, the corresponding command in the pop-up menu of the selected element is automatically removed (if it exists). If the toolbar is integrally hidden by the **NoBar** option, but you do not want commands corresponding to the buttons to be removed from the pop-up menu of the selected object, you must include the **ToolBar[]** option which will in this case enable forcing of display of corresponding menu commands.

When alternative lists have been defined in the ListView, selection of the list to be displayed can be by means of dedicated buttons (see 3.5.3.1.2.6)

Export buttons enable production of a document using content of the ListView. Export in Excel (X) and PDF (P) are available as standard. The **ExportCommands=P|X|0** option enables control of display of these buttons.
You can redefine the list of these buttons in a given ViewPort PropertyPage (see 2.3.4). In this context, the default list of buttons will follow configuration of this viewPort, defined in its *_Parameterization.* text.
In this configuration, we define export buttons in the **[ListViewExport]** section:

```
[ListViewExport]

   <ExportName>=Item(<ButtonID>),Param(<extraParameter>),Name(<Name>),Picture=<PictureID>,Method=<MethodID>
```

Each export button is identified by its **ExportName**. The first letter of each of these ExportNames should be distinct, since it identifies the button in the toolBar and can be used in ExportCommands to specifically filter export buttons in the ListView.

Example: if you define an export named "Html", **ExportCommands=H** will indicate that only this export button will be present in the listView.

Identifier of the button will be <ButtonID>; the name displayed will be <Name>, which can be a character string or a field – in this case the name will be that of the referenced object -. The image associated with the button corresponds to <PictureID>, and the macro to be invoked <MethodID>. <extraParameter> can be used to specify an initialization string for this macro.

You can define additional buttons or modify behavior of standard buttons, using the content map. To do this, elements with the **PushButton** option should be added to the content map. Clicking one of these additional buttons generates a notification TBN_DROPDOWN (64826) relating to the ListView, which can be intercepted by the trigger macro of the property page (see 3.4.3). The **PushedButton** ListView specific function enables determination of which button has been clicked: It returns the identifier of the content map element associated with the button.

The name associated with the button corresponds to the title defined for the element.
Configuration of a button element uses the following options:

```
Param(PushButton{,OnSelection}{,CheckFlags}
     {,Picture=PicId}{,Method=MacroId}
     {,ReplaceStandard[<ButtonName>]}{,OverloadStandard[<ButtonName>]})
```

OnSelection: indicates that the button only ungrays if an occurrence is selected in the list.

CheckFlags: indicates that the button should gray if the element is disabled (for example by DisabledOn).

Picture=*PicId*: defines the image associated with the button.

Method=*MacroId*: indicates that the button should not generate a notification on the trigger macro of the property page, but on an instance of the *MacroId* macro created specifically for this button.

ReplaceStandard[] and **OverloadStandard[]** enable overload of a standard button. With **OverloadStandard**, only the image of the standard button is affected. With **ReplaceStandard**, standard processing of the button is replaced by the specific processing associated with the element. The name of the standard button to be overloaded corresponds to the name defined in **ToolBar[]**

3.5.3.1.2.6 Alternative list specification

Specification of alternative lists enables a unique ListView to display several distinct collections. Advantages here are more compact page size and faster loading, since only the selected collection will be queried.
When a ListView has alternative collections, we add to it a user interface enabling selection of the current collection:

- This selection can be made using additional buttons added to the ToolBar of the ListView. These buttons are RadioButtons – only a single selection can be made. This is default behavior.
- It can be achieved by means of a DropDownList inserted in this same ToolBar; to do this, we should include the **ShowAlternate=DropDown** option.
- It can be done – in Web Front-End only, by means of a list of buttons including tabs located above the ListView (and therefore outside the ToolBar). To do this, include

the **ShowAlternate=Folder** option. This function not being available in Windows Front-End, the option is ignored in this case and buttons are displayed. If you want to display tabs in Web Front-End and a DropDownList in Windows Front-End, you can include the **ShowAlternate=FolderOrDropDown** application.

To define alternative collections:

➔ Define in the ListView content map an element that has the **AlternateSelection** option. The identifier of this element will be that of the browsed collection. Options of such an element are:

```
Param(AlternateSelection{,Picture=<PicId>}{combinateOptions}{plugin})
```

<PicID> picture associated with the button. If this option is not specified, the picture defined for the collection will be displayed.

combinateOptions concerns combine alternative lists, see 3.5.3.1.2.7.

Plugin enables definition of a specific macro enabling population of the collection to be displayed, see 3.5.3.1.2.8

Column Filter enables definition of generic column filtering specific to the alternative Selection. This option is of form:

```
ColFilter=NoDefColumn|StandardColumn|AssocColumn|None
```

When this option is present, it replaces the column filtering defined on the ListView (**NoDefaultColumn**, **StandardColumn**, **AssocColumn**). The value **None** deactivates filtering defined on the ListView.

When alternative collections are defined for a ListView, the ListView identifier itself is no longer used to define a collection, except if for questions of visibility none of the alternative selections is available. In this case the ListView becomes a standard list. However, visibility of the ListView itself will be subject to availability of this identifier.

Although this ListView can display several distinct collections, it has only one content map. Elements contained in this map (additional columns, buttons) may apply to only one of the alternative collections, and filtering should be defined so that they are only taken into account in suitable collections. Filtering is carried out by means of the **ForAlternate[*collectionID{,CollectionID}*]** option, which should appear in the element to be filtered. If this option is specified, the element (column or button) will only be visible when a *collectionID* defined in the option corresponds to the identifier of the current alternative collection.
Note: in Windows Front-End, the platform does not allow dynamic hide of a button in a toolbar: inappropriate buttons will not be hidden but simply disabled.

3.5.3.1.2.7 Combined alternative lists

This system is a variant of alternative lists, affecting their selection mode. It proposes replacement of RadioButtons by CheckBoxes. In this case, the total number of alternative selections corresponds to the number of possible combinations. In this system, the total number of alternative collections therefore corresponds to the square of the number of CheckBoxes.

Example: two CheckBoxes enable selection of four collections (no selection, first button selected, second button selected, both buttons selected).

To specify a combined alternative list:

- ➔ define as many **AlternateSelection** elements in the content map as there are possible selections, minus one: the identifier associated with the ListView is that of the collection used when none of the CheckBoxes is selected.

We then distinguish between elements corresponding to unitary collections (corresponding to click of a single button) and elements corresponding to combined collections (corresponding to a combination of buttons).

Elements corresponding to unitary collections have the **PushMask=n** option, where n is the bit value (1,2,4,8).

Elements corresponding to combined collections have the **MaskValue=c** option, where c corresponds to the combination of buttons.

Example: when there are two buttons:

- ***PushMask=1*** *indicates the collection activated by selection of the first button*
- ***PushMask=2*** *indicates the collection activated by selection of the second button*
- ***MaskValue=3*** *indicates the collection activated by selection of both buttons*

(If no button is selected, the collection activated corresponds to the identifier associated with the ListView).

These ListViews necessarily using CheckBoxes, the **ShowAlternate** option is not taken into account.

Note: the generic system of display of inherited and substituted objects, which applies as standard to all ListViews configured by a MetaAssociationEnd for which inheritance of variations is enabled, automatically uses the system of combined alternative lists. This system is exclusive, and will be disabled if alternative collections are explicitly defined for the ListView.

3.5.3.1.2.8 Specific collections: plug-ins and ERQL queries

You can define a ListView that does not browse a collection defined in the metamodel. Two cases may require such a system:

- *You may want to display the result of a dynamically defined ERQL query.*
- *You* may want to define a collection defined by a Macro.

To display the result of an ERQL query:

- ➔ Supply the ListView with the **RequestMode** option.

  In this case, the identifier of the ListView must correspond to a property of type VarChar. The text contained in this property should correspond to an ERQL query, which will be evaluated when completing the ListView: the query result is then displayed in the list. In HOPEX 1.0 version, queries with parameter cannot be executed.

To display a collection calculated by a macro:

- ➔ Supply the ListView with the option:

```
PlugIn<macroID{:InitString}>
```

macroID corresponds to a Macro identifier. This macro should implement a standard collection as follows:

```
Function GetStandardCollection(      MegaObject as MegaObject,
                                      RetType as Variant,
                                      InitString as String) as MegaCollection
```

MegaObject is the source object of the ListView

InitString is the character string specified in the option

As *InitString* can be anything, it is possible to generate pages with macros having dynamically defined behaviors.
Another difference between such a Macro and one used in a Query or *AbstractCollection*, is that this macro can use the association browsed by a source object of the ListView. This is in fact the case for the *BridgeCollection* macro supplied as standard by MEGA, which can be used in all ListViews. This macro enables display of objects according to two links, one of which is from the source object of the ListView object.

```
PlugIn<BridgeCollection:MainMAEID,SecondaryMAEID>
```

In this case, the collection browsed is oObject.getCollection(«MainMAEIe»)
If the ListView object is derived from browsing of a link (oObjet.getSource.Exists): we therefore have the collection oObject.getSource.getCollection(«SecondaryMAEId»).
In this case the attributes of the MetaAssociation SecondaryMA are added as columns in the ListView, and we search for the corresponding values of these attributes in the second collection.
If the object is not a source object, only the first collection is displayed.

3.5.3.1.2.9 Web Front-End specificities

ListViews displayed in Web Front-End benefit from functionalities not accessible in Windows Front-End. The main innovation is that lists are by default presented in paginated mode, enabling faster loading. It is also possible to implement filtering via columns. Finally, presentation mode benefits from more comprehensive configuration.

- Display of alternative collections (**ShowAlternate**) is more complete, since it uses Folder mode.
- The **hidePagingToolbar** option deactivates pagination. It can be used on lists which we know will contain only a few objects.
- **hideLabelButton=1**: in Web Front-End, toolbar buttons have a label. This option enables their removal
- **GridMode**: indicates that use of ListView is data entry oriented. In particular, cell modification access is simplified.

3.5.3.1.2.10 MultiSelection list

The **MultiSelection** option enables definition that a ListView is multiselection. This new operating mode has been introduced in HOPEX.

In this mode, the ‘'Property' command displays a box containing all columns in update not corresponding to Unique Index Attributes: this box enables simultaneous modification of all selected objects.

In Web Front-End, entry of a cell in this mode modifies the column value for all selected objects.

When the column is managed by a specific updateTool, the latter may not adapt to the fact that it is not instanced on a specific object, but on an object collection.

MultiSelection mode is not the default mode, but it is possible to configure the active PropertyPageViewPort to make it the default mode for all ListViews.

For this:

➔ Configure the PropertyPageViewPort concerned by inserting the following keyword in its *_Parameterization* text (as 2.3.4).

```
            [ListViewDefault]
                  MultiSelection=1
```

Note: with this configuration, certain ListViews may exhibit inconsistent behavior, in particular when they control elements, or when you have defined specific buttons in their toolbar that do not operate correctly in this mode. You can alleviate these problems by providing such ListViews with the **MonoSelection** option, which deactivates multiselection.

3.5.3.1.2.11 Specific Configurations

Filtering: the **ShowAbstract** option applies to object collections of an abstract class, and enables deactivation of filtering relating to visibility of the concrete class of the element, filtering applied as standard. For lists displaying system repository objects, the **NoMetaFilter** option enables deactivation of filtering relating to visibility of these objects.

Entry mandatory: in a wizard, it can be useful to impose constraints on a ListView. The Mandatory parameter, applied to the list, indicates that a selection must be made in the list to validate the page. A less restrictive option, **NotEmpty**, indicates that the list should not be empty for the page to be validated.

A list can be used specifically to reorder its elements. Any other form of update is prohibited, and it is not possible to resort the list according to a column. The new sort order is saved in the repository when the page is validated. This mode is activated by the **ReOrderDrop** option.

The pop-up menu of elements displayed in the ListView can be redefined. For this, define a *MetaCommand Manager* and associate this with the ListView by means of the **ExtraCommands=<CommandAccessorID>** option. Commands defined in this manager will be inserted in the pop-up menu of objects in the list.

3.5.3.1.2.12 Cell Edit

When you edit a cell of a ListView, you initiate a system called *in-place editing*. This triggers specific start of an editing tool superimposed on the ListView without directly affecting it.

This editing tool operates according to the updateTool defined for the column of the cell to be edited. If necessary, the UpdateTool will adapt to in-place mode; it can adapt its behavior when in-place mode imposes data entry space limited to cell size.

A column declared in read-only does not allow activation of editing of these cells (excepting the particular case where the AttributeControl enables actions other than cell modification; in particular, this is the case for editMenus displaying the object menu).

The **NoInPlaceEdit** option enables deactivation of cell editing for the complete ListView

3.5.3.1.2.13 List view content refresh

ListViews can display collections of different types: certain are "live" and can react directly to repository updates that could modify their content; this is the case for example for collections derived from a non-calculated MetaAssociationEnd. Other collections however cannot react to these updates, as for example the result of an ERQL query. The query must be run again to reflect the consequences of these updates in the list.

The ListView cannot know behavior of the collection displayed at update, and using an option you should indicate to it the required refresh modes if necessary. Available options are:

> **RefreshOnChange,RefreshOnInsert,RefreshOnDelete**: these options indicate that when the collection is modified, we must reselect before refreshing content.

RefreshOnCommand: indicates that the **Refresh** function applied to this ListView results in reselection of this list. Otherwise, the **Refresh** function only results in fetch of the existing collection, without reselection.

Reselection of a collection can be costly in terms of performance, and options should be specified carefully.

3.5.3.1.3 Options

Column Filter Definition

NoDefaultColumn: only the naming attribute is displayed.

StandardColumn: displays only standard columns, other presentable columns are initially hidden.

AssocColumn: displays the naming attribute and properties of the browsed association.

ReOrder: orders columns according to content map.

RefreshColumns: recalculates column list when page is refreshed.

Definition of image associated with each listed object:

NoLineBitmap: no image.

BitmapSelector=*PropID*: defines property serving to calculate image.

BitmapStamper=*PropID* {,*PropID*}: adds stamps on image

Definition of initial sort:

NameSort: sort by name, order number is not used.

OrderOverloadProp=*PropID*: enables order number attribute redefinition.

Refresh collection: indicates when collection should be re-queried.

RefreshOnChange

RefreshOnInsert

RefreshOnDelete

RefreshOnCommand

Configuration of background loading of the ListView (Windows Front-End).

NoBackgroundLoading: loading is synchronous.

NoBackgroundEscape: background loading cannot be interrupted by Escape key.

Defining Toolbar:

NoBar enables toolbar hide.

ToolBar[] enables hide/show of ListView standard buttons.

hideLabelButton=1 (Web Front-End only): does not display button labels.

ExportCommands=P|X|0: export buttons display commend (0: no button, X: Excel, P: Pdf).

Specific Selections:

RequestMode: Selection obtained from ERQL query.

PlugIn<*MacroId{:InitString}*>: Selection obtained from macro.

Filtering objects displayed in list:

ShowAbstract: deactivates filtering related to visibility of concrete class of the element

NoMetaFilter: deactivates filtering related to visibility of system objects.

Alternative collection selection display mode:

ShowAlternate=DropDown | Folder | FolderOrDropDown

Other options:

hidePagingToolbar: (Web Front-End only): deactivates pagination

MultiSelection: activates multiselection mode

NotEmpty: validiity condition of the ListView imposing that this should not be empty

EnableDropOrder: ListView designed for sorting its elements

NoInPlaceEdit: deactivates cell editing

GridMode: (Web Front-End) indicates that the ListView is principally designed for editing.

AssociativeCollection: collection obtained from RelationShip object
ExtraCommands=<CommandAccessorID> redefines menu of listed objects

3.5.3.1.4 Properties

Value: this property contains absolute identifiers of the selected object in the ListView. In the case of a list with multiselection, the concatenation of selected absolute identifiers is returned, separated by a space. In assigning the value you modify the selection.

3.5.3.1.5 Notifications

Select: sent when the selection changes.

Open: sent at double-click (Windows Front-End only)

BarClick: sent at click on one of the specific buttons of the toolBar. The **PushedButton** method enables determination of which button has been clicked.

3.5.3.1.6 Component Methods

Specific methods of the AttributeControl can be called from the component object (AttributeControl.Component)

.PushedButton As Variant: this function should be called at processing of a **BarClick** notification and returns the identifier of the clicked button. This identifier corresponds to the Item of the button or to a conventional button identifier (for example for PDF Export)

.GetCollection As MegaCollection: returns the collection displayed in the ListView

.GetAlternate As Integer: returns a distinct number according to the alternative collection displayed.

.Fetch(JSONCommand As String) As String: Obtains content of the list as it is displayed in the ListView. This function returns a JSON format string. The Command is a JSON format string indicating what you want to obtain. If you want to obtain the complete list with all columns displayed, this parameter should be exactly **{"currentView":true}**. If not, it should contain information relating to what you want to obtain conforming to the **hdr** structure as described below. This can be useful if you want to partially retrieve the collection by specifying fields **firstLine** and **lineCount**.

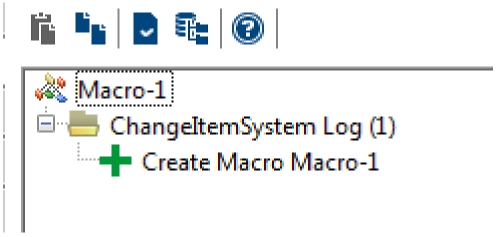
The returned JSON contains two structures:

- An **hdr** structure containing restored data (list of columns **columns**, sort method **sortCriteria**), as well as the number of restored rows **lineCount**, the number of elements of the collection **total**, and the index of the first restored row **firstLine**.
- A **content** structure, table containing the list of rows. Each row is a structure for which conventional fields are specified **id** (object absolute identifier, **classId** (object class), **image** (object image moniker) and as many fields as columns specified for the object (the name of the field corresponding to the column identifier) of which the value corresponds to the value to be displayed (when the column displays an image, its value is an object containing the image moniker and the string to be displayed). If the ListView is multiselection, we also indicate if the object is selected with the boolean field **isSelected**

Example of a restored JSON:

```
{
 "content":[

{"id":"f10000000b20","classId":"O20000000Y10","image":"~gYNHjd5mzKQ1","2100000009
00":"Commentaire"},

{"id":"f20000000b60","classId":"O20000000Y10","image":"~gYNHjd5mzKQ1","2100000009
00":"Journal"}],
  "hdr":  {
   "firstLine":0,"lineCount":2,"total":2,"noEmpty":true,

"columns":[{"id":"210000000900"},{"id":"Q10000000f10"},{"id":"S20000000n50"}}],
   "sortCriteria":[{"id":"210000000900"}]
  }
}
```

3.5.3.2 TreeView



The TreeView element enables tree display. Unlike MetaTree, it is not necessary to define a MetaTree to use it. In addition, it also operates with informal objects.

This object enables display of Collections from the object associated with the element. These collections are presented in the form of folders. In expanding these, you can list elements of a collection.

3.5.3.2.1 Specifying collections to be displayed

The identifier associated with the TreeView can be:

- the identifier of an _Operator.

  In this case Collections are filtered according to behavior of this operator related to the collection.

  By default, only collections with "Abort" behavior are filtered. If you specify the **ScanStandardOp** option, "Abort" and "Link" behaviors are filtered. With the **ScanDeepOp** option, only "Deep " behaviors are displayed.

- the identifier of a MetaAssociation type.

  In this case collections of this type are displayed.

- the identifier null (~000000000000[Null]).

  In this case no collection is filtered.

To explicitly define collections:

➔ Use the content map associated with the TreeView.

Identifiers associated with elements of this map can be *MetaAssociationEnd* or *_Abstract Collection* identifiers (therefore in particular of *queries*).

When an object is expanded in the tree, we collect in the content map the elements of which the source is compatible with this object, in order to create the list of collections to be displayed.

Whatever the collection display definition mode, only those collections visible from the user technical level viewpoint, and from the viewpoint of his/her associated rights are displayed.

3.5.3.2.2 Tree configuration

3.5.3.2.2.1 Multi-level trees

By default, the tree only expands at a Collection level. The **IsDeep** option enables specification of a multi-level tree. In this case, the filtering mode on child levels is identical to that of the root; in particular the content map is also used for child elements.

To increase or decrease the number of expanded element levels at element display:

➔ Use the **UnfoldedAtStart=n** option, where n is the number of levels to expand (by default, n = 2).

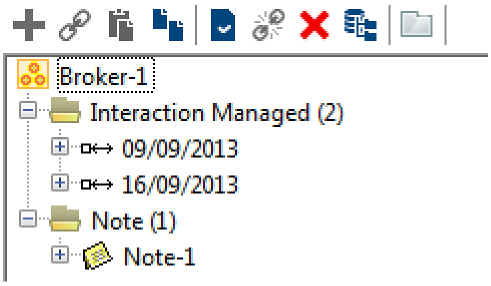
This option should be used with caution, since the number of levels open to initialization significantly affects tree loading time…

When the tree is configured with a content map, it is possible for only certain types of element to be multi-level by specifying the **IsDeep** option on the element of the map itself. On the other hand, **IsLeaf** on an element enables deactivation of the **IsDeep** option global to the tree.

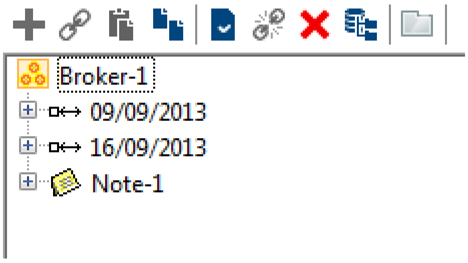
3.5.3.2.2.2 Direct display

When there are no ambiguities on displayed objects, it is possible not to display folders corresponding to collections, but to directly display objects listed in collections. For this use the **HideFolder** option.

ShowFolder



HideFolder



When you expand the element, all visible collections are therefore expanded; nodes however remain grouped by collection.

This option results in disappearance of folder level, commands appearing on folders (in particular create and connect menus) are no longer accessible; in addition it is no longer possible to explicitly drag-and-drop objects in the folder representing the collection. **HideFolder** mode must therefore take account of this.

When the tree is configured with a content map, it is possible for folders to be hidden only for certain types of element, by specifying the **HideFolder** option on the map elements themselves. On the other hand, the **ShowFolder** option on an element enables deactivation of the **HideFolder** option global to the tree.

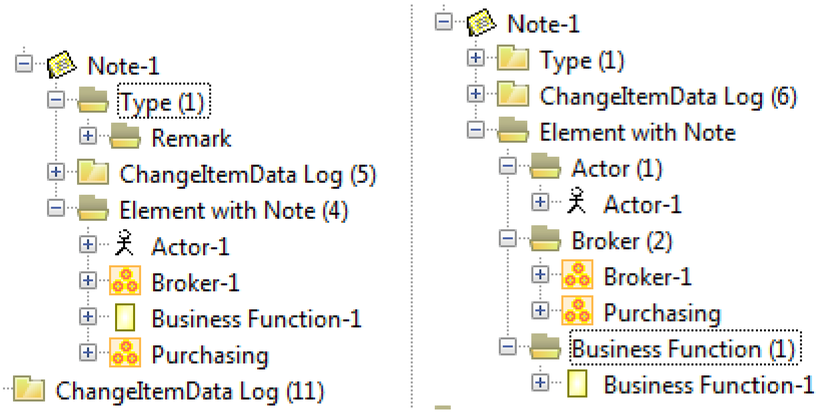
3.5.3.2.2.3 *Displaying generic collections*

When target of a browsed collection is an abstract MetaClass, objects presented under a folder can be of different types. By default, the tree displays objects on a single level, without grouping by MetaClass.
To group them by MetaClass:

➔ Use the **ShowConcrete** option.

In this case, another level is added, grouping child nodes by MetaClass.

ShowConcrete



When the tree is configured by a content map, you can define this display specifically for each collection using the **ShowSpecialized** and **HideSpecialized** options specified on the map element.

3.5.3.2.2.4 Object sort and display

As standard, elements expanded in the tree are sorted according to the order number defined for the collection (generally the Order attribute). When two elements have the same order number, the second sort criterion is the label associated with the node. This label is itself the collection naming property.
For folder level, we use sort order defined for MetaAssociations in the Metamodel.

To display a label other than the object name:

- ➔ Use the option NameComputer=<PropID>.

  In this case, the PropID property will be used as label. When the tree is configured by a content map, you can deactivate this behavior for a given element by specifying the **NoComputedName** option.
- ➔ In this case you can also deactivate sorting by order number using the option **NoOrder** (objects will then be ordered lexicographically).

Folders are represented by homogeneous folders.
To differentiate their display:

- ➔ USe the **DefaultFolderBitmap** option.

  In this case the color of the folder depends on collection type (major, minor, calculated).

3.5.3.2.2.5 Defining the toolbar

The toolbar of a treeview cannot be extended and displays only standard commands. These commands differ depending on whether the selected element is a folder or an object. In particular, commands "New", "Connect" and "Paste" are only available at Folder level, and are grayed when the selected node is a MEGA object.

Standard buttons of a treeview are:

Create (**C**): Start the tool for creation of a new element in the collection corresponding to the selected folder

Link (**L**): Start the tool for insertion of a new element (connect) in the collection corresponding to the selected folder

Reorder (**R**): Starts the tool enabling reordering of elements of the selected folder according to standard order (see 3.5.3.1.2.3). This option is not present by default.

Destroy (**D**): Start the tool for deletion of the selected object. When a Folder is selected, proposes the mass deletion tool for all collection objects.

Unlink (**U**): Start the tool for disconnection of the selected object. When a Folder is selected, starts the mass disconnection tool for all collection objects.

Properties (**P**): Display of the properties dialog box of the object selected in the list.

Explore (**E**): Starts the explorer on the selected object or collection (available only on Windows Front-End).

HideEmpty (H): Enables show or hide of folders corresponding to empty collections. The **HideEmptyAtStart** option enables initial hide of empty collections.

The **ToolBar[]** option enables show or hide of elements of this toolbar. It is configured similarly to the ListView toolbar (see 3.5.3.1.2.5**Error! Reference source not found.**)

3.5.3.2.2.6 Notifications

Select: sent when a new node in the tree is selected.
Generally this notification is only sent when the selected node is an object. A folder does not correspond natively to an object, since it corresponds to a collection. It is however possible to make a folder selectable using the **SelectFolder=<CollectionID>** option. When this option is activated, an informal object is created corresponding to the folder description, to which is added the <CollectionID> collection corresponding to the object collection seen from the folder. This object is then considered as the selected object. Such a system enables specification of a user interface displaying a list corresponding to the list of objects of the Folder.

3.5.3.2.2.7 Options

ScanStandardOp, ScanDeepOp: enables modification of filtering of collections when the tree is configured by an operator.

IsDeep: multi-level tree

UnfoldedAtStart=n: number of levels initially expanded.

HideFolder: hides folders level

ShowConcrete: displays folder level for concrete MetaClasses

NameComputer=<PropID>: tree elements label calculation

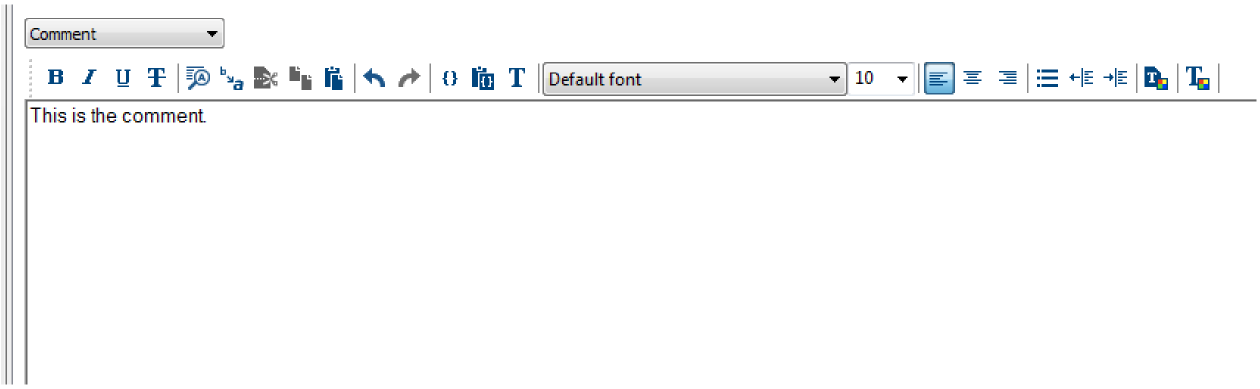
DefaultFolderBitmap: color display of folders.

HideEmptyAtStart: enables initial hide of empty collections.

ToolBar[]: specifies toolbar content.

SelectFolder=<CollectionID>: makes folders selectable

3.5.3.3 Text



This is the comment.

This element enables display and editing of text type properties. The text editor displayed depends on the type of text associated with the property.

The identifier associated with the element corresponds to the editor property.

You can group editing of several texts in the same screen area. In this case a dropdownlist located above the edit area enables selection of displayed text. This is default behavior, when several texts are associated with the displayed MetaAttributeGroup.

When you want to explicitly specify a text grouping, you can do this using the content map of the element; for each map element there is a corresponding property to be edited. This map also enables forcing read-only display mode.

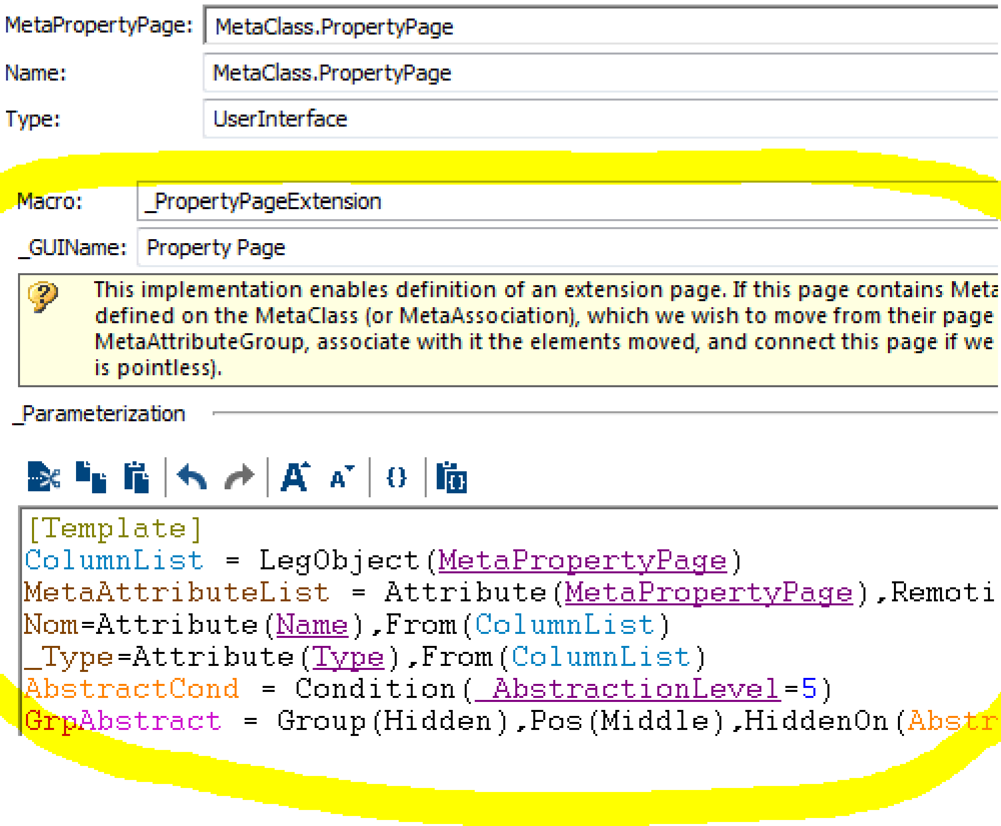
Options:

NoBar: removes the text editor toolbar.

Notifications:

Change: sent when text content has been changed

3.5.3.4 SubPage



MetaPropertyPage: MetaClass.PropertyPage

Name: MetaClass.PropertyPage

Type: UserInterface

Macro: _PropertyPageExtension

_GUIName: Property Page

This implementation enables definition of an extension page. If this page contains Meta
defined on the MetaClass (or MetaAssociation), which we wish to move from their page
MetaAttributeGroup, associate with it the elements moved, and connect this page if we
is pointless).

_Parameterization

```
[Template]
ColumnList = LegObject(MetaPropertyPage)
MetaAttributeList = Attribute(MetaPropertyPage),Remoti
Nom=Attribute(Name),From(ColumnList)
_Type=Attribute(Type),From(ColumnList)
AbstractCond = Condition(_AbstractionLevel=5)
GrpAbstract = Group(Hidden),Pos(Middle),HiddenOn(Abstr
```

An AttributeControl of" SubPage" type enables inclusion of one page in another. This system is particularly useful in the case of an AttributeControl controlled by a Map, since it elegantly manages the fact that objects of different types can be displayed differently in the same page. Content of the SubPage is dynamically calculated according to the object associated with the element. This calculation can also be executed in deferred mode, and therefore this system can facilitate form display (particularly in Web Front-End).

Typically, the identifier associated with the element corresponds to a MetaPropertyPage which is instanced on the element of the page. This identifier can however be of another type, depending on the options defined on the element, particularly the **Computed** element, which indicates that the page to be displayed is derived from a calculation. This option should be systematically present if you are not in the typical case.

If the MetaPropertyPage indicated in the element is not a standard property page of the object, you should include the **External** option.

The first calculation mode can be defined when the MetaClass of the object associated with the element is not fixed; in this case it is possible to define a specific MetaPropertyPage for each of the MetaClasses possible for the element, using its content map. Elements of this map should be of form:

```
SubPageItem=Item(<MetaClassID>),From(SubPageMap),Param(<PageID>,Default)
```

If the MetaClass of the associated object corresponds to <MetaClassID>, the SubPage displays <PageID>.

The **Default** option should be used when <MetaClassID> is an abstract MetaClass; in this case, if no sub-element corresponds to the class of the element, a second pass will enable association of the element with a <PageID> if the MetaClass of the element is a sub-class of <MetaClassID>.

Another calculation uses the _Type associated with a standard page of the object associated with the element. For this include the **Type** option, and associate with the element an identifier of _Type. In this case the object page corresponding to this _Type will be displayed (if several pages correspond to this _Type, the first one found will be displayed).

To request display of a standard page of the object:

- ➔ Specify the CLSID of the macro of this page (corresponding to `MacroCLSID` mentioned above), using the **CLSID={*clsid of the macro*}** option.

To request display of a virtual page including all object attributes:

- ➔ Use the **CompleteDescription** option.

When no page can be determined, or when no object is associated with the element, the SubPage displays nothing.

If size of the element has not been specified, the Control is resized to contain at least the page at its minimum size.

Options:

Computed, External, Type, CompleteDescription, CLSID={*clsid of the macro*}: options enabling definition of page to be displayed

Owner=off, **Name=off**: these options can be used when you display the object characteristic page and enable hide of Name and Owner fields of this page.

SetMinMax=1: indicates that the size specified for the element should be used to define size of the SubPage.

Maximized=0: deactivates BOTTOMALL style of control (this style is present by default)

Abstract: when this option is present, the object of the page corresponds exactly to the object associated with the element, even if instanced on an abstract class (for example when this object is controlled by the browsing path of a generic MetaAssociation. In the opposite case and therefore by default, the page is initialized with the concrete object.

Associative: enables display of an associative page. The object associated with the page corresponds to the object derived from the association browsed by the object of the element.

Component methods:

Specific methods of the AttributeControl can be called from the component object (AttributeControl.Component)

.SubPage As MegaPropertyPageComponent: returns the object corresponding to the displayed sub-page. It can be **Nothing** if no page was instanced for this control.

APPENDIX: COMPATIBILITY

This documentation is valid for HOPEX 1.0 SP 1 version.

In version HOPEX 1.0 the following points are not functional:

It is not possible to use viewPort to overload standard behavior of forms (see 2.3.4)

Keyword **Refresh(Always)** (see 3.3.1.4**Error! Reference source not found.**) is not systematically recognized and does not operate.

Notification **BarClick** of ListView (see 3.5.3.1.5): the notification is not named in HOPEX 1.0, its internal number 64826 should be used

The Value property of a viewer is not managed

Options **direction** and **AutoExpand** of the **SelPropagate** parameter of MetaTree

Options **DefaultFolderBitmap**, **UnfoldedAtStart=n** and **SelectFolder** of Treeview.

Options **Maximized** and **ValueFrom** of HelpComment

Options **SetMinMax** and **Maximized** of SubPage

Option **MultiLine** of RadioButton

Option **DirectMacro** of Viewer

Option **ValueFrom:<IdProp>** of Label

Option **AssocColumn** on ListView, and option **ColFilter** on alternative selections.

In updatesTools, options **ManageReadOnlyMenu**, **ManageValueID**, **SingleOnly** and management of specific sub-menus commanded by **SPECIFICCHILDMENU**.

The global MegaMacroData used in MetaCommands of UpdateTools is not available in Java.

Forms - Property Pages - Tutorial

Objective

The purpose of this document is to familiarize you with the customization of Properties pages of a MEGA object.

Initializing the courseware

In order to be independent of evolution of the **MEGA** metamodel, this courseware is based on a specifically designed metamodel extension.

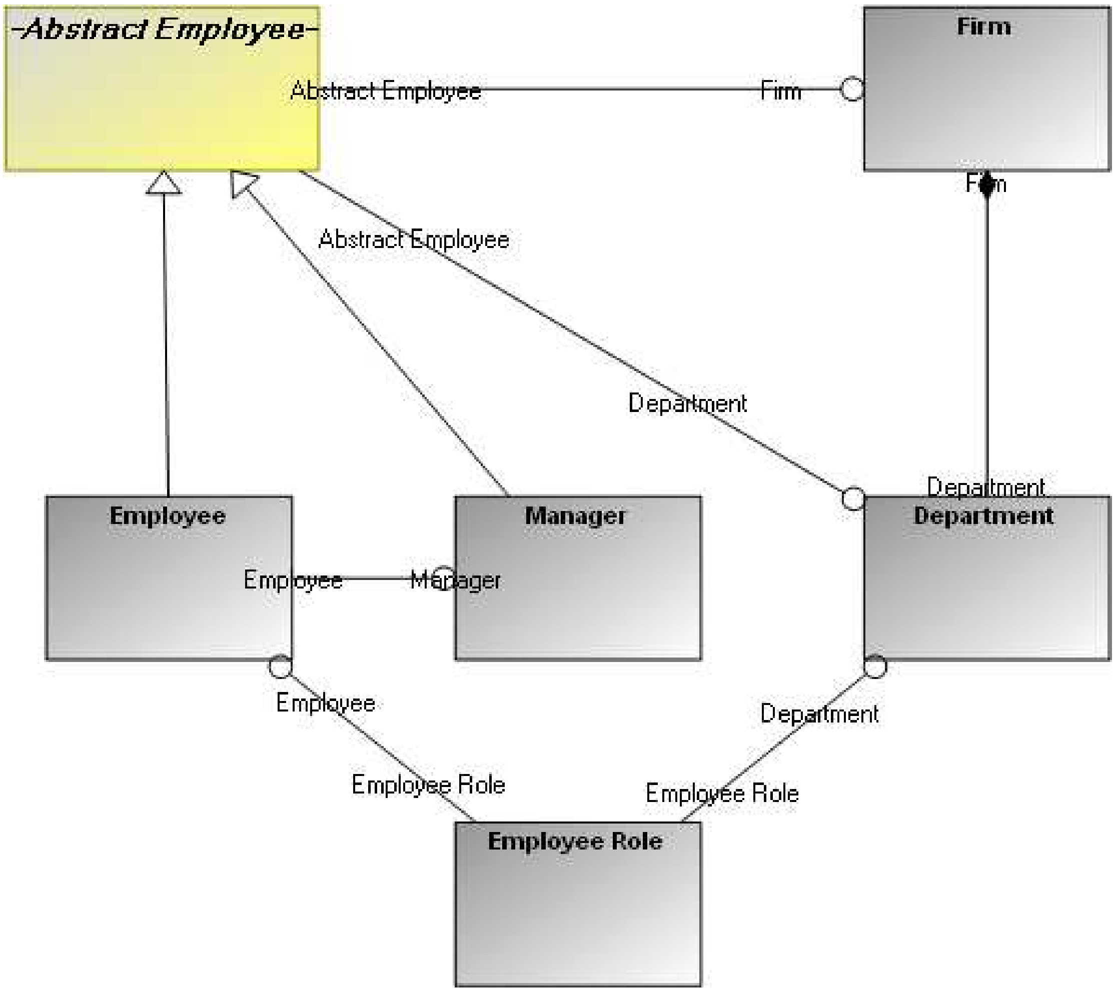
Before starting work, you must initialize your environment:

1. Download the following command file:

   MetaModel Extensions for training courses.mgr

   available here: MetaModel Extensions for training courses.zip
2. From HOPEX, import into the System repository the MetaModel Extensions for training courses.mgr command file.
3. From the HOPEX Administration application (Administration.exe), recompile the metamodel.

Content of this specific metamodel is the following:



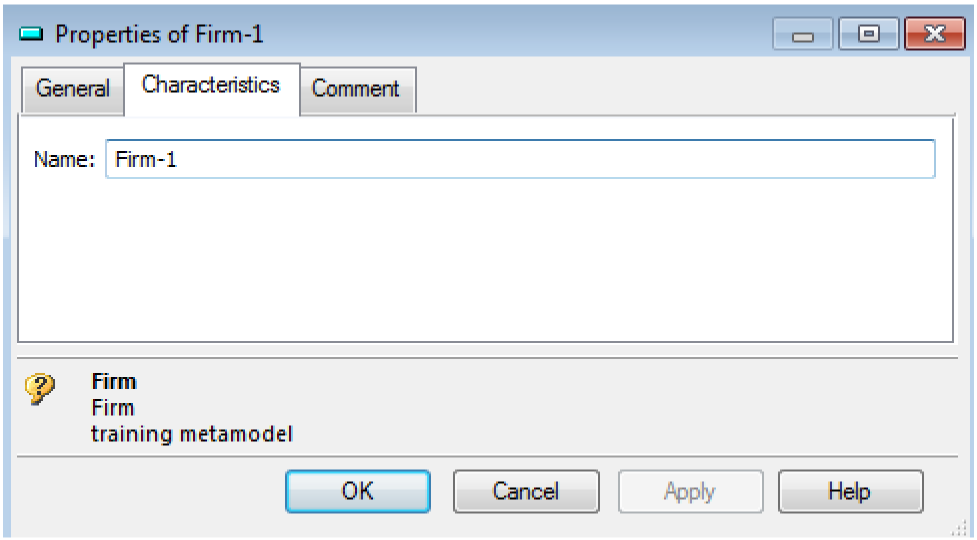
Object standard Properties pages

In the extension provided below, there is no customization relating to Properties pages of defined MetaClasses.

However, when you explore an object of this metamodel, you will find that it has a Properties window already structured.

Example with MetaClass *Firm*:

1. Login to HOPEX with the HOPEX customizer profile.
2. Run the explorer from the desktop .
3. If no *Firm* has previously been created, select **Explore > Create** and select the Firm MetaClass.
4. Display Properties of this object.



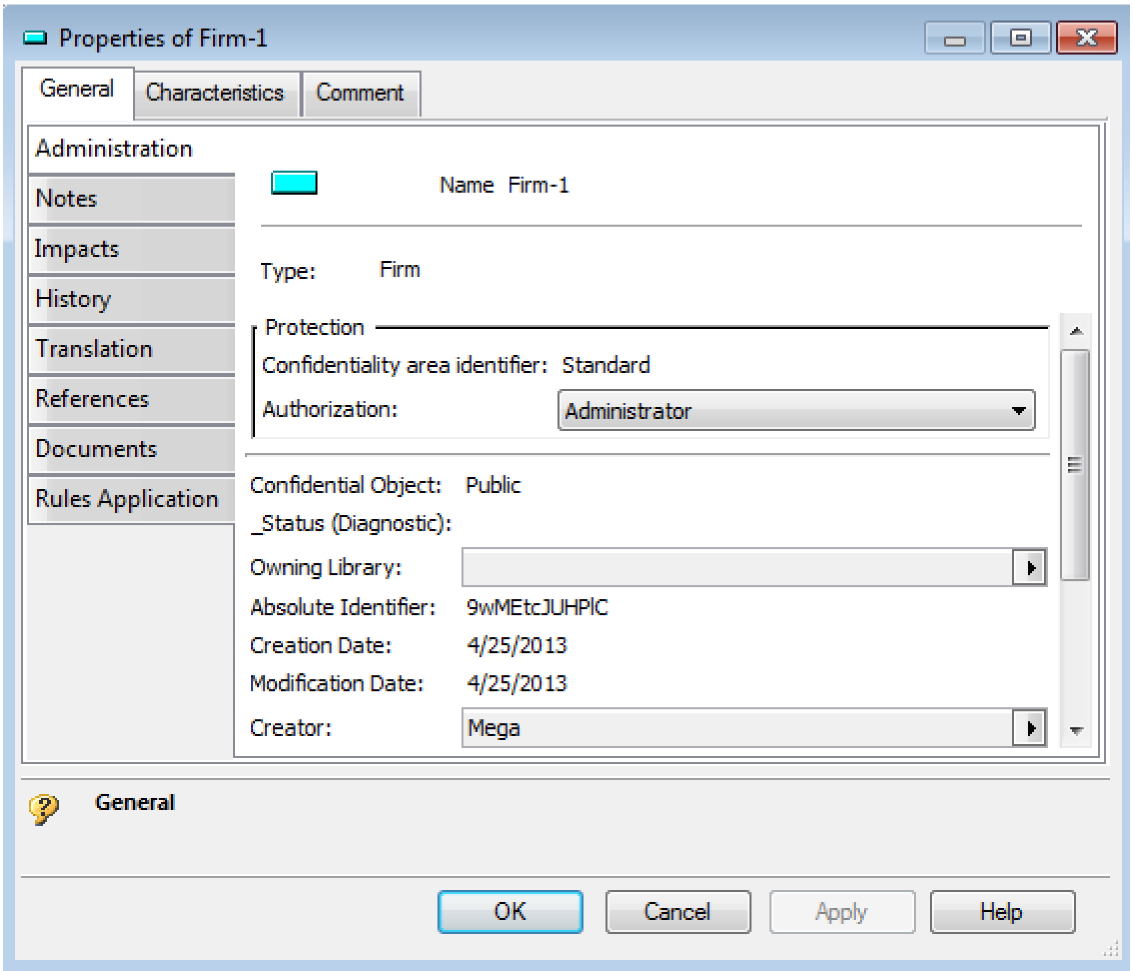
Name: Firm-1

Firm
Firm
training metamodel

This MetaClass contains only two "business" MetaAttributes: Name and Comment.

- The name has been automatically entered in the **Characteristics** tab, which is the main tab of the object.
- The comment is located in the **Comment** tab, designed to host all MetaAttributes of Text type (varchar).

General tab already includes subtabs which have been automatically inserted by **HOPEX**.



Name Firm-1

Type: Firm

Confidentiality area identifier: Standard

Authorization: Administrator

Confidential Object: Public

_Status (Diagnostic):

Owning Library:

Absolute Identifier: 9wMEtcJUHPlC

Creation Date: 4/25/2013

Modification Date: 4/25/2013

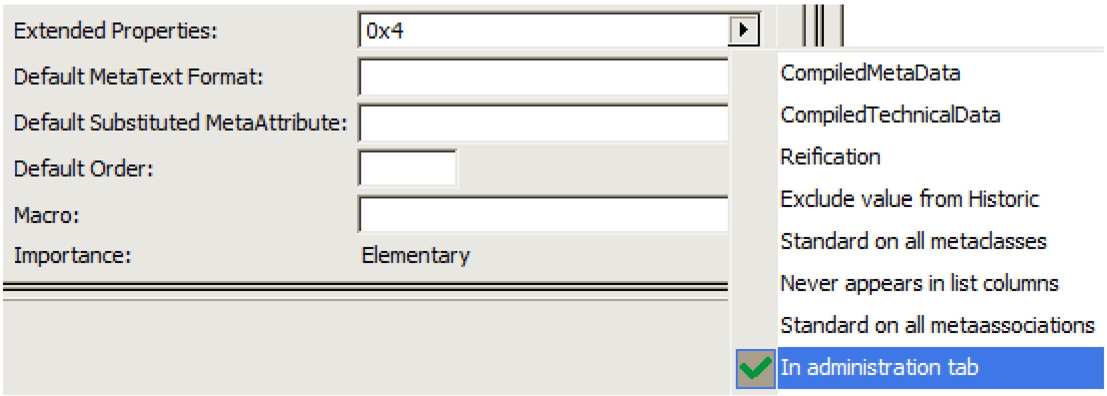
Creator: Mega

The **Administration** subtab groups non-business MetaAttributes of the MetaClass, which are automatically associated with the latter.

The **Translation** subtab handles the issue of translatable attributes.

Other subtabs are added automatically according to standard roles played by the MetaClass.

> Note: administration MetaAttributes are marked with a flag in their extended properties.



Extended Properties:	0x4
Default MetaText Format:	
Default Substituted MetaAttribute:	
Default Order:	
Macro:	
Importance:	Elementary

- CompiledMetaData
- CompiledTechnicalData
- Reification
- Exclude value from Historic
- Standard on all metaclasses
- Never appears in list columns
- Standard on all metaassociations
- ✓ In administration tab

You can see that as standard:

- MetaAttributes of the MetaClass are grouped according to their type.
- Tabs are associated with attribute groups.
- Standard roles of the class (for example inherited from abstract classes) can produce to tabs.

  Note for example that the **Impacts** page appears as a function of behavior of MetaAssociations of the MetaClass relating to the "Impact" operator.
- Automatic tabs are inserted (for example **History** and **Rules Application** tabs).

For more information, see "*HOPEX Forms*" document.

Object Properties pages customization principles

Customization of an object Properties pages consists of being able to redefine all or part of this standard. To do this, you must be able to:

- add specific pages
- modify or substitute generic pages, either to add elements, or to modify their appearance.

Two concepts have been introduced in the MetaModel to meet these demands:

- MetaAttributeGroup
- MetaPropertyPage.

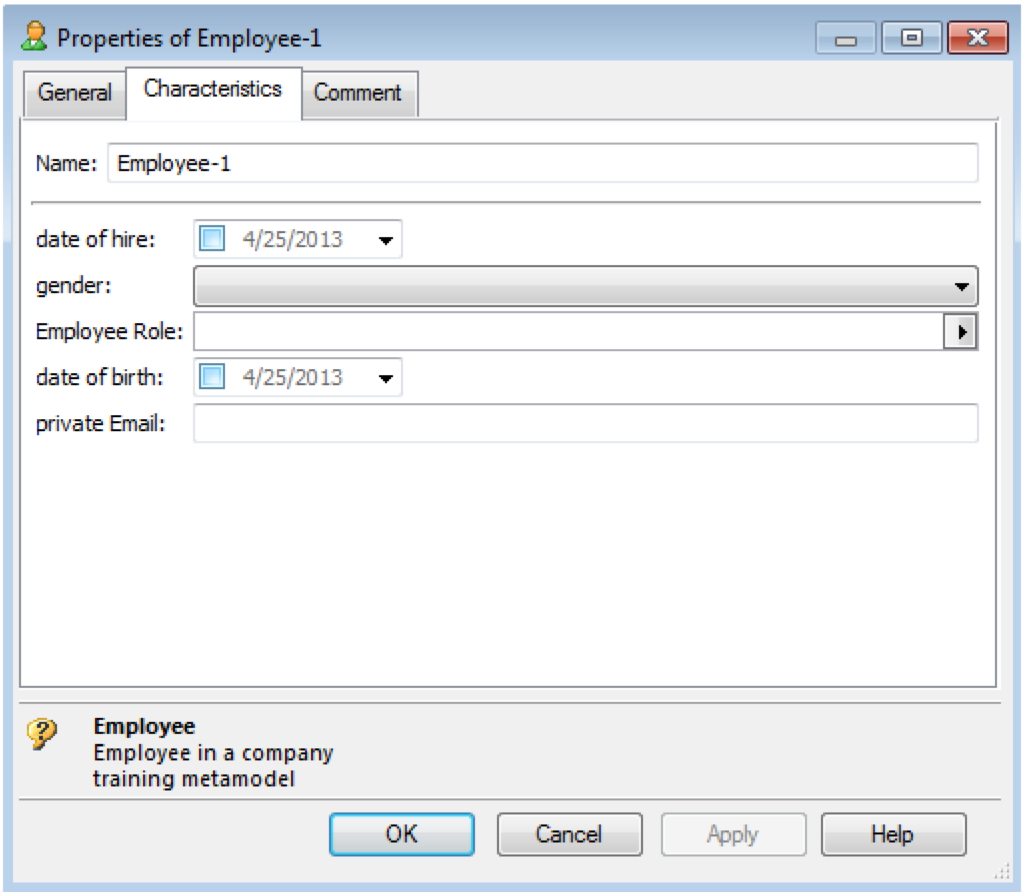
MetaAttributeGroups enable redefinition of the grouping of the MetaClass attributes. Attribute is interpreted here in its broadest sense, including the concepts of MetaAttribute, TaggedValue and LegAttribute:

- a taggedValue is a named value which you can associate with a MEGA object (on condition that it inherits from the "element with tagged value" MetaClass) without creating a MetaModel extension;
- a LegAttribute enables inclusion of a role (ie. MetaAssociationEnd), of maximum cardinality 1, as an attribute of object type.

Usually, any attribute of the MetaClass associated with a specific MetaAttributeGroup of this same MetaClass is removed from the generic grouping mentioned in the previous chapter.
To insert specific elements in the object Properties page which are not attributes of the MetaClass, or to redefine display or behavior of these attributes, use a MetaPropertyPage.

Moving attributes using a MetaAttributeGoup

The objective of the following customization is to clarify use of a MetaAttributeGoup. To do this, consider the Characteristics page of the **Employee** MetaClass:


Properties of Employee-1

Name: Employee-1

date of hire: □ 4/25/2013

gender:

Employee Role:

date of birth: □ 4/25/2013

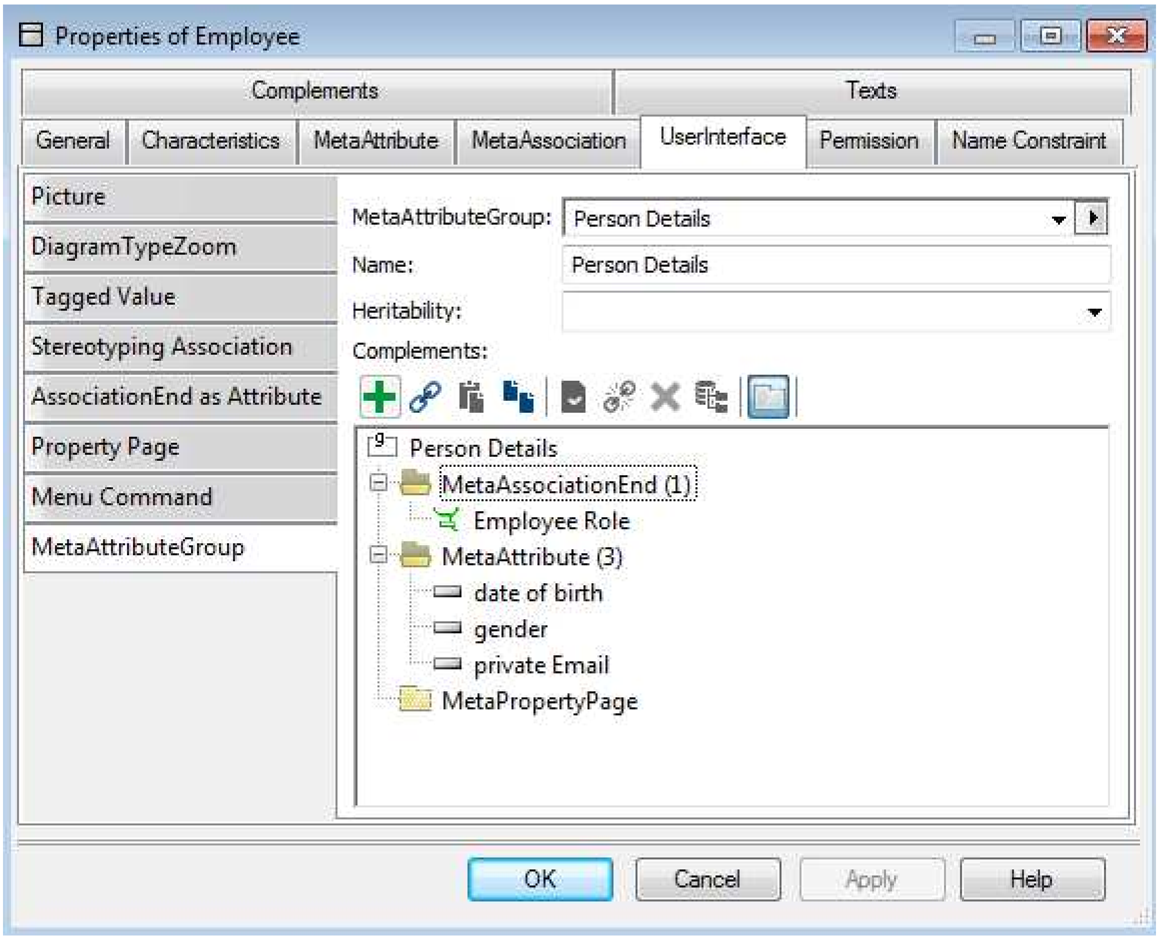
private Email:

Employee
Employee in a company
training metamodel

The objective of customization is to remove attributes *gender*, *date of birth* and *private Email* (which are MetaAttributes) and *Employee Role* (which is a LegAttribute) from the Characteristics page and include these in a specific page.

To do this, create a MetaAttributeGoup with which associate these attributes.

1. Open the Properties window of the **Employee** MetaClass.
2. In the **User Interface** tab, select the **MetaAttributeGroup** subtab.
3. In the **MetaAttributeGroup** field, click the button with the right-facing arrow and select **New** to create a new MetaAttributeGroup.
4. Enter the **Name** of the MetaAttributeGroup (for example "Person Details") and click **OK**.
5. In the **MetaAttributeGroup** field, select the MetaAttributeGroup you just created (for example « Person details »).
6. Associate with this MetaAttributeGroup:
   a. the MetaAttributes (date of birth, gender, private Email): copy/paste from the **MetaAttribute** tab, **SuperMetaAttribute** subtab.
   b. the MetaAssociationEnd (Employee Role), which you copy in the same way from the **AssociationEnd As Attribute** tab.



MetaAttributeGroup: Person Details

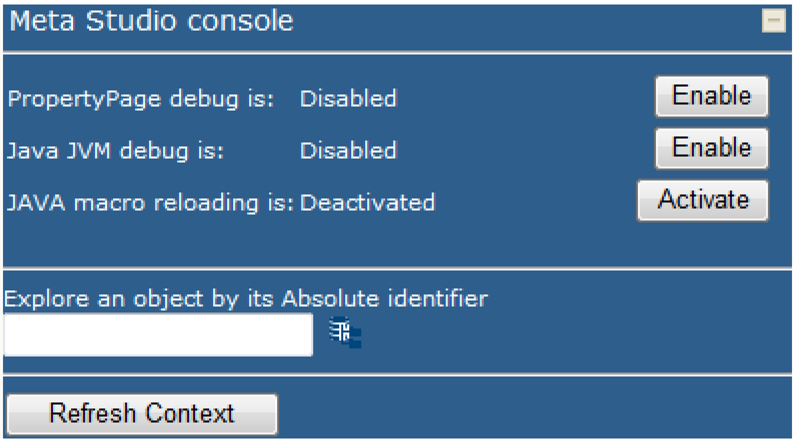
Name: Person Details

Heritability:

Complements:

- Person Details
  - MetaAssociationEnd (1)
    - Employee Role
  - MetaAttribute (3)
    - date of birth
    - gender
    - private Email
  - MetaPropertyPage

7. Refresh the metamodel view so that these modifications are taken into account. To do this:
   a. Display the HOPEX home page.
   b. Display the **Meta Studio Console** element (if not already visible).
   c. Click **Meta Studio Console**.
   d. Click **Refresh Context**.



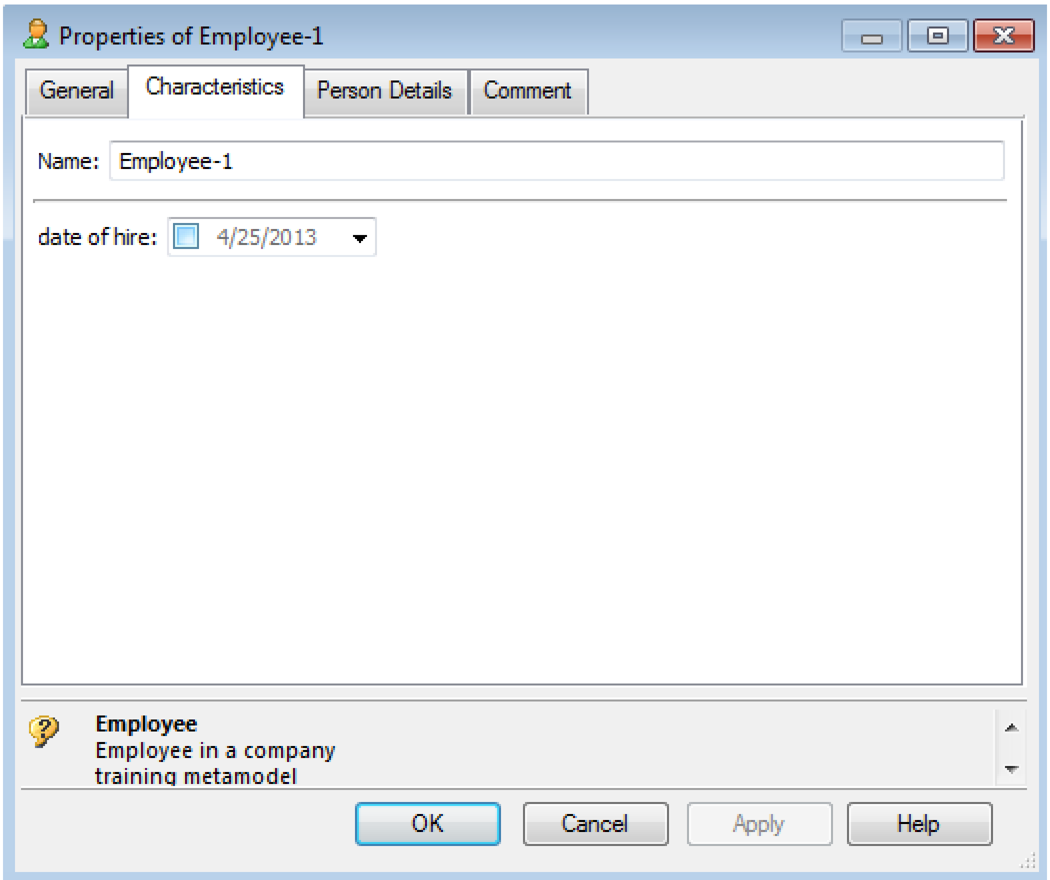
PropertyPage debug is: Disabled

Java JVM debug is: Disabled

JAVA macro reloading is: Deactivated

Explore an object by its Absolute identifier

8. Open the Properties window of an Employee. This shows that:

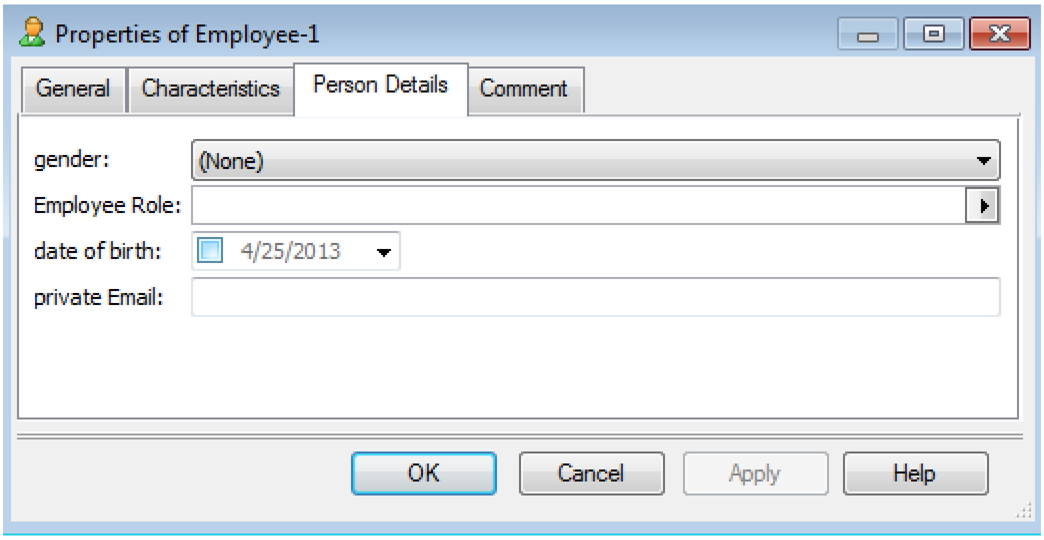

Properties of Employee-1

Name: Employee-1

date of hire: □ 4/25/2013

Employee
Employee in a company
training metamodel

Attributes associated with the MetaAttributeGroup have disappeared from the **Characteristics** tab. They appear in a new tab, created automatically from the MetaAttributeGoup, and carrying the same name "Person Details".


Properties of Employee-1

gender: (None)

Employee Role:

date of birth: □ 4/25/2013

private Email:

Note: When the MetaAttributeGroup has a **GuiName**, it is this attribute that is used to name the tab.

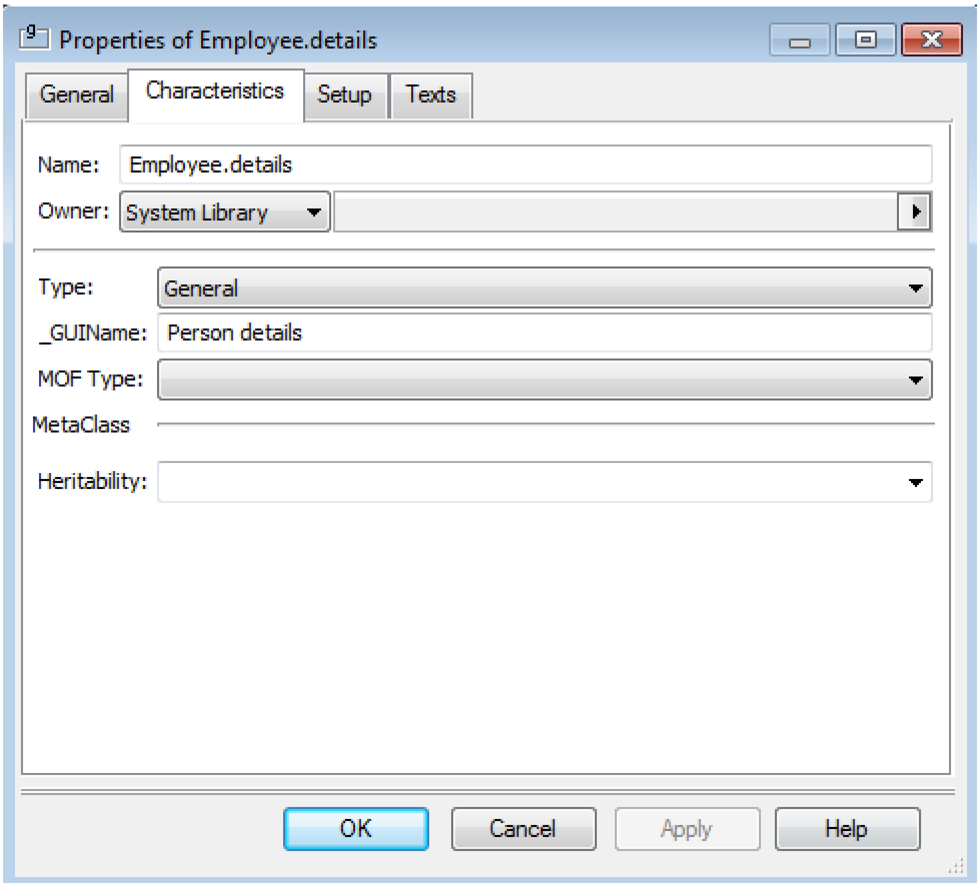
Grouping pages as subtabs

Consider that pages previously defined are satisfactory, but you want to include "Person Details" in the **General** tab.

To do this, be aware that pages are grouped in this way according to their Type. It is the name of the Type which is used to name the main tab. When a page is automatically deduced from a MetaAttributeGoup, it is the _type associated with the MetaAttributeGroup which is considered.

Define your MetaAttributeGroup **Type** as « General ».

You can rename the MetaAttributeGroup and define the name displayed in its _GuiName.


Properties of Employee.details

Name: Employee.details

Owner: System Library

Type: General

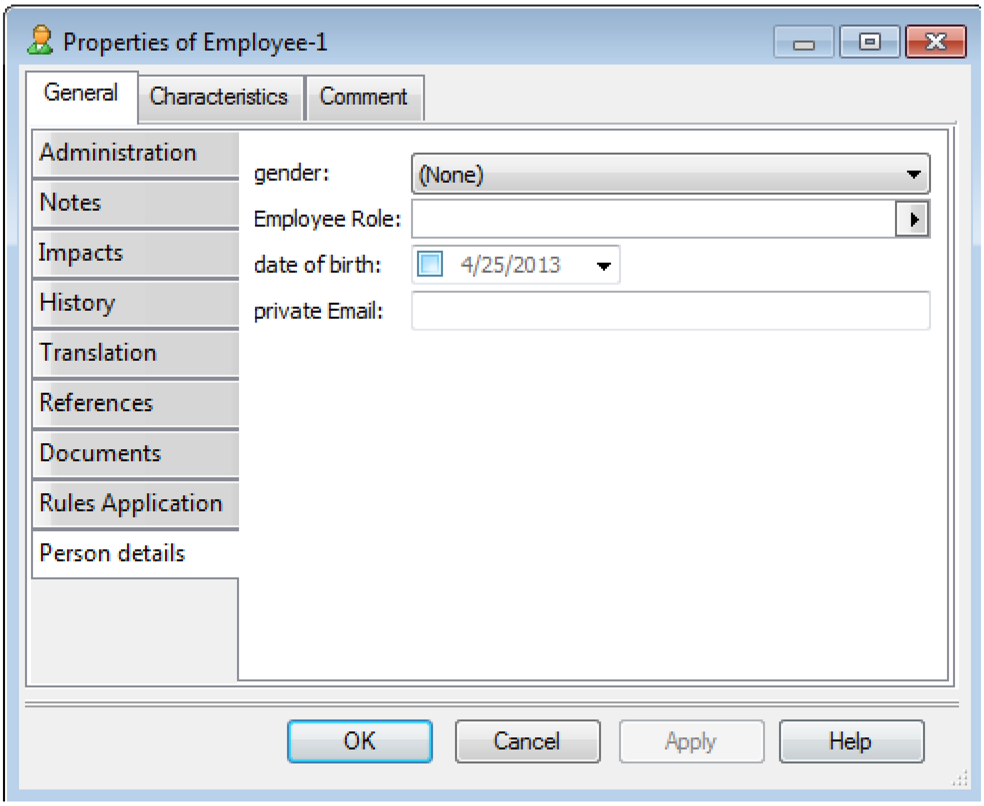
_GUIName: Person details

MOF Type:

MetaClass

Heritability:

After context refresh, the Properties window of an Employee becomes:



gender: (None)
Employee Role:
date of birth: □ 4/25/2013
private Email:

You can add new object types.

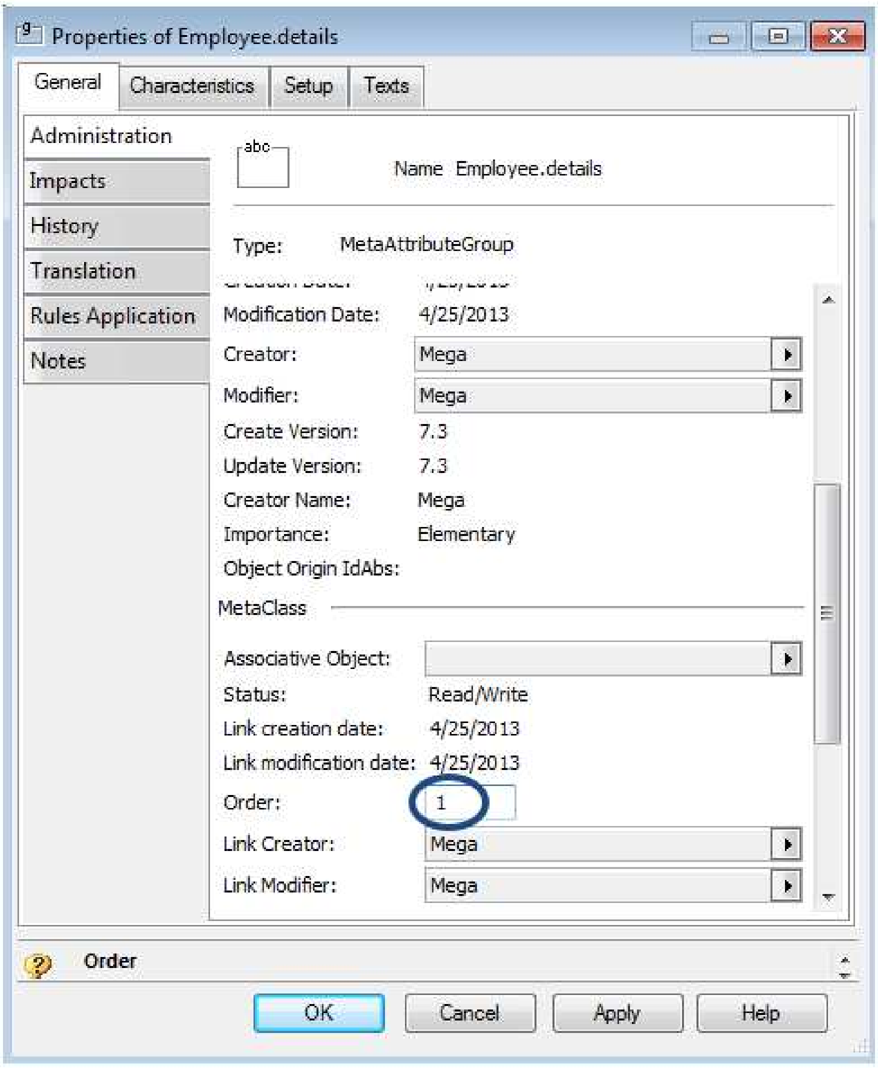
Reordering pages

The above arrangement is almost satisfactory, but you want the 'Person Details' tab to be placed first in the list.

To do that, you must know the following rules:

- Each Properties window page has an associated order number which defines the tab order.
- Depending on page implementation, this order number can be specified in different ways.
- Each generic page has an associated intangible specific order number. For more information, see "*HOPEX Forms*" document.
- When a tab groups several pages, its order number is the smallest order number from among those of the pages it contains.
- When a page is automatically deduced from a specific MetaAttributeGroup, its order number corresponds to the value of the Order' attribute between the MetaClass and the MetaAttributeGoup

In the present case, you can modify the order number between the MetaAttributeGroup and the MetaClass:



Name Employee.details

Type: MetaAttributeGroup

Modification Date: 4/25/2013

Creator: Mega

Modifier: Mega

Create Version: 7.3

Update Version: 7.3

Creator Name: Mega

Importance: Elementary

Object Origin IdAbs:

MetaClass

Associative Object:

Status: Read/Write

Link creation date: 4/25/2013

Link modification date: 4/25/2013

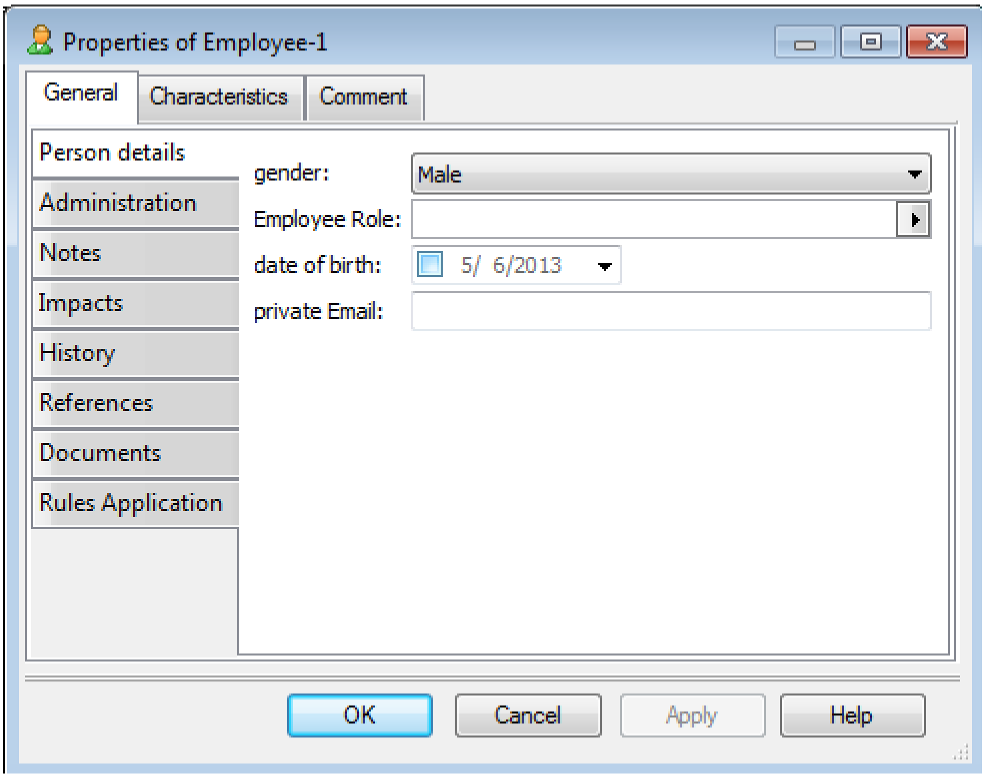
Order: 1

Link Creator: Mega

Link Modifier: Mega

Order

After refresh, the general tab becomes:


Properties of Employee-1

gender: Male

Employee Role:

date of birth: ☐ 5/ 6/2013

private Email:

Defining a new page displaying a non-standard element

You can define a new page in the Properties window of the *Firm* MetaClass, enabling definition of the list of its *Departments*.

This list is obviously not included in the standard attributes of the *Firm* MetaClass. It is not a case of moving standard elements, but of inserting a new element. The MetaAttributeGroup concept is therefore not used in this extension.

To do this, you will use a MetaPropertyPage implemented by the " _PropertyPageExtension" Macro. This macro will build a Properties page, consulting the content of the "parameterization" text of the MetaPropertyPage.

This configuration text is a setting text, that is a text which specifies a list of key/value pairs, grouped in sections:

```
 [Section]
Key1 = Value1
Key2 = Value2
```

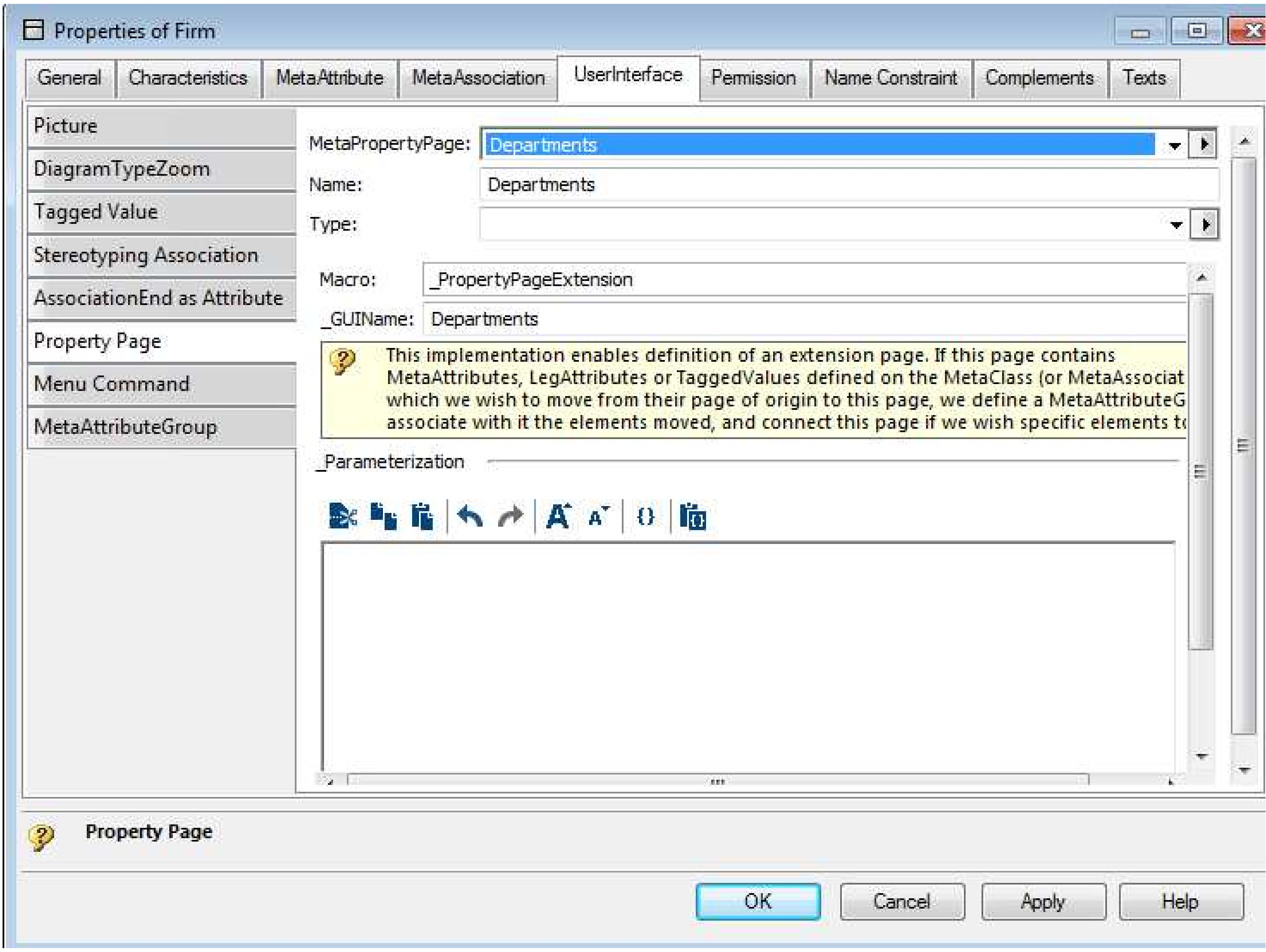
This text is auto-documented and entry help is available. The list of elements you want to include in the page is specified in the [Template] section, and the page order number can be defined in the [Page] section.

The customization envisaged consists of displaying a ListView browsing the *(Firm/Department)* MetaAssociation view from the *Firm*, therefore by the "*Department*" MetaAssociationEnd.

1. Open the Properties window of the **Firm** MetaClass.
2. Select the **User Interface** tab, then the **Property Page** subtab.
3. To create a new Properties page, in the **MetaPropertyPage** field, click the arrow and select **New**.
4. Name the new page « Departments » and click **OK**.
5. In the **MetaPropertyPage** drop-down menu, select the new page « Departments ».

   It is now possible to define the _GuiName of this page, that is the name displayed in the tab: for example "Departments", as well as the Macro which implements this page.
6. In the **_GUIName** field, enter « Departments ».
7. In the **Macro** drop-down menu, select PropertyPage.Kind, then the *_PropertyPageExtension* macro.
8. Click **Apply**.

   The page will then allow you to define configuration of your new page.



MetaPropertyPage: Departments

Name: Departments

Type:

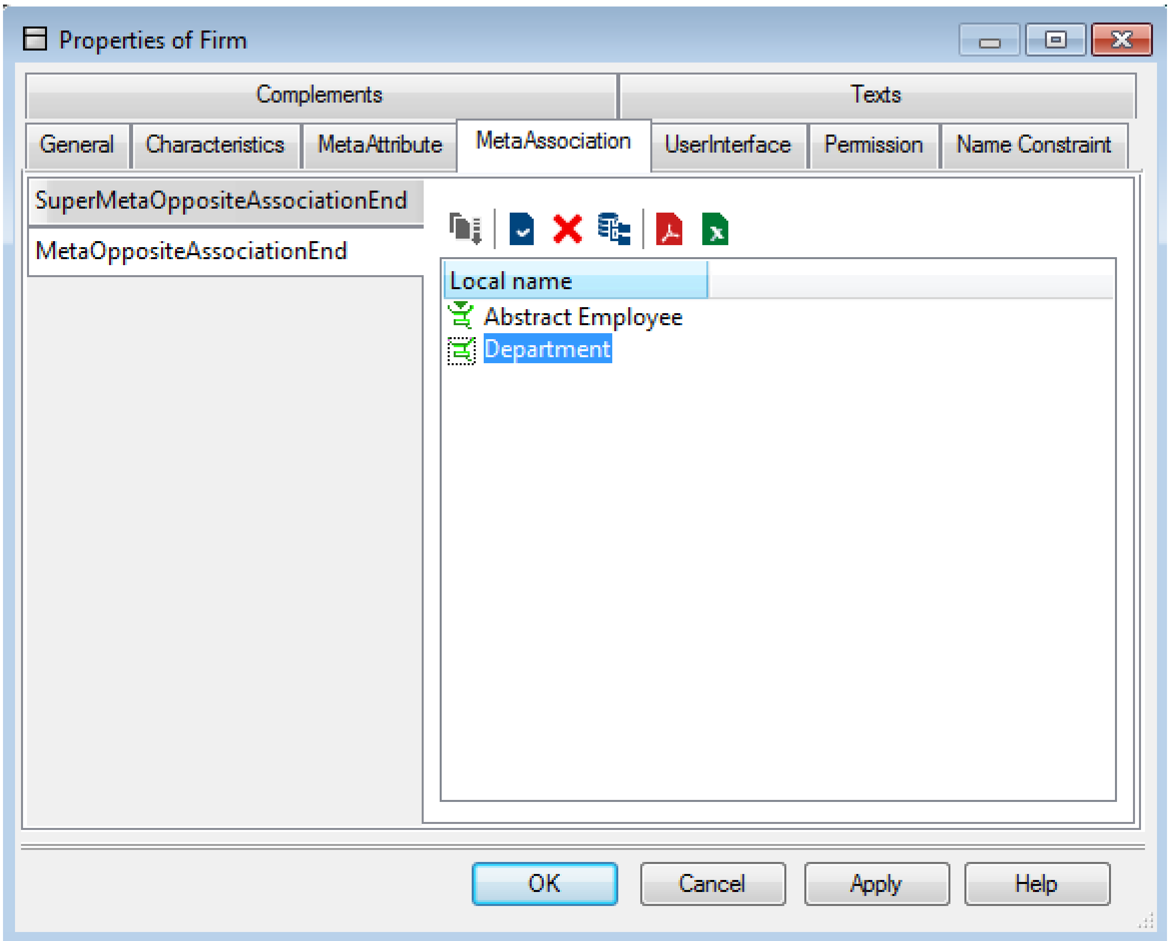
Macro: _PropertyPageExtension

_GUIName: Departments

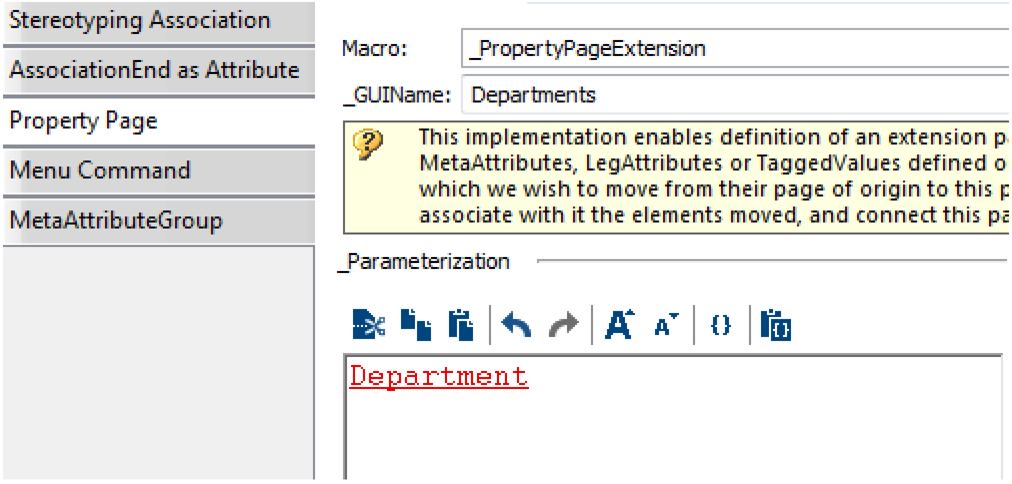
This implementation enables definition of an extension page. If this page contains MetaAttributes, LegAttributes or TaggedValues defined on the MetaClass (or MetaAssociat which we wish to move from their page of origin to this page, we define a MetaAttributeG associate with it the elements moved, and connect this page if we wish specific elements t

Property Page

9. To configure your page, you simply need to know the identifier of the MetaAssociationEnd which must be browsed by the listview. To do this, from the **MetaAssociation** tab, **MetaOppositeAssociationEnd** subtab, copy the "Department" MetaAssociationEnd.



10. Paste the result in the **_Parameterization** frame.



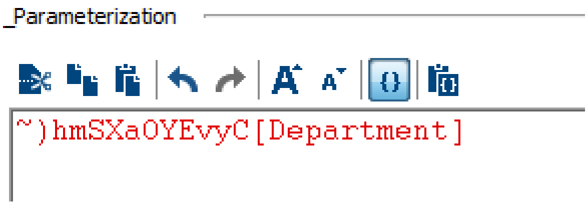
Macro: _PropertyPageExtension

_GUIName: Departments

This implementation enables definition of an extension p
MetaAttributes, LegAttributes or TaggedValues defined o
which we wish to move from their page of origin to this p
associate with it the elements moved, and connect this pa

Department

The pasted text is a "field", in which the absolute identifier of the pasted area is carried. Button {} enables display of this identifier.



_Parameterization

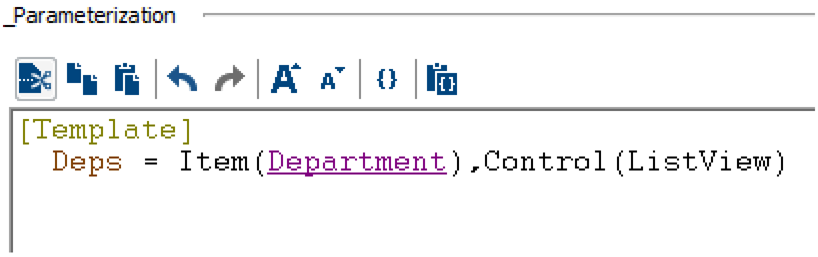
```
~)hmSXaOYEvyC[Department]
```

This field will serve as parameter for the 'ListView' type element which you will define in the page [Template].

11. In the **_Parameterization** frame, paste:

```
[Template]
  Deps = Item(~)hmSXaOYEvyC[Department]),Control(ListView)
```

Text color indicates that your definition complies with syntax.

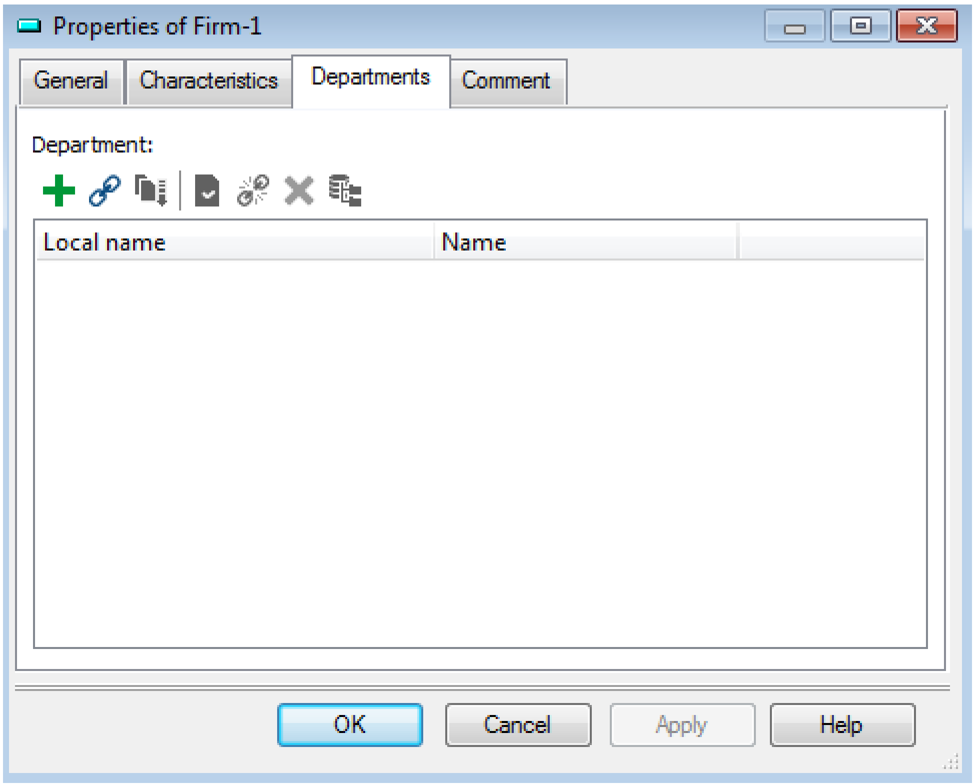


_Parameterization

```
[Template]
  Deps = Item(Department),Control(ListView)
```

12. Click **Apply**.

After refresh, the list appears in the *Firm* Properties window.



Properties of Firm-1

Adding a non-standard element in the Characteristics page

You can include this *Departments* list not in a specific page, but in the object Characteristics page. Ensure that this addition does not interfere with the list of attributes naturally included here.

The standard **Characteristics** page is not an extension page, and not therefore implemented by the same macro; the macro used is _PropertyPageStandard. In addition to the content of the "Characteristics" implicit MetaAttributeGroup, this page displays:

- The name or local name as page header, depending on whether or not the MetaClass has specific name implementation.
- Object owner if applicable.
- MetaAssociation attributes when the object is seen from another object, if applicable.
- Extensions defined on the MetaClass if these have not been stored in specific MetaAttributeGroups, if applicable.

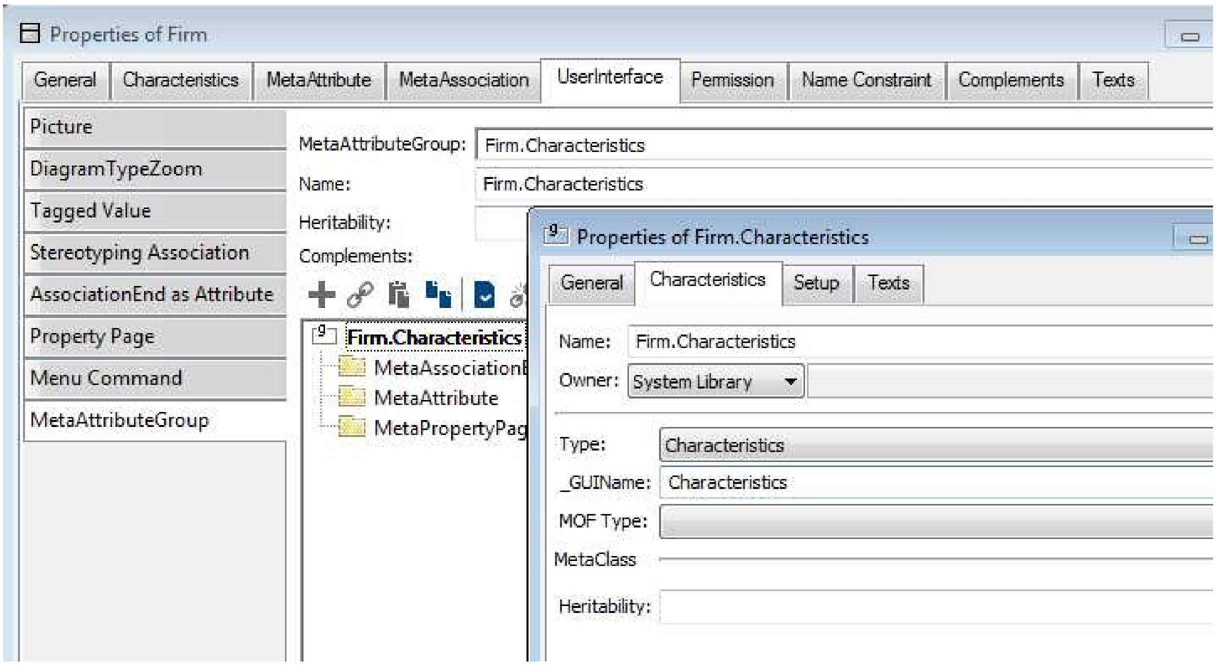
To overload the **Characteristics** page, you must first overload the implicit MetaAttributeGroup with an explicit MetaAttributeGroup: To do this, simply create a MetaAttributeGroup from the MetaClass, and specify "Characteristics" type.

When this MetaAttributeGroup has been created, two cases are possible:

- either you do not want to modify the list of attributes implicitly present in this MetaAttributeGroup: in this case, no element should be defined. This is our example case.
- or you want to explicitly define the list of attributes visible in the **Characteristics** page, and in this case you should populate the MetaAttributeGroup with these attributes.

Note: In this case, standard processing can result in attributes not assigned to a MetaAttributeGroup, and which will not therefore be displayed. These attributes are therefore assigned to the "Extension" standard MetaAttributeGroup. You can detect this phenomenon in noting appearance of an ‘extension’ tab.

Create this MetaAttributeGroup, name it *Firm.Characteristics*, and assign it Characteristics type.



MetaAttributeGroup: Firm.Characteristics

Name: Firm.Characteristics

Heritability:

Complements:

Name: Firm.Characteristics

Owner: System Library

Type: Characteristics

_GUIName: Characteristics

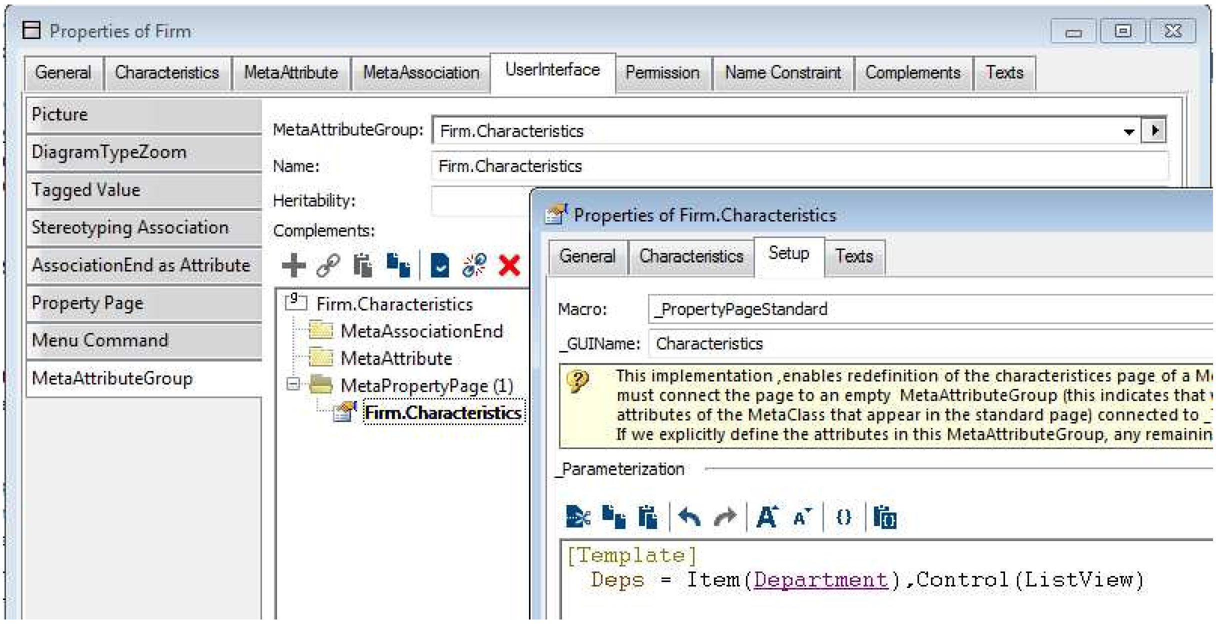
MOF Type:

MetaClass

Heritability:

To achieve your objective, you must customize display of the Properties page derived from the MetaAttributeGroup.

1. To create a MetaProprertyPage dedicated to the MetaAttributeGroup, in the **MetaAttributeGoup** configuration page, click **Create**.
   This MetaPropertyPage will replace the implicit page associated with the MetaAttributeGoup. In addition, if specified:
   - the page Type will be taken into account rather than the MetaAttributeGroup type, allowing you to display the page in another tab.
   - the GuiName of the page will be used as page title rather than the GuiName of the MetaAttributeGroup.
2. As previously indicated, implement this page with the _PropertyPageStandard macro to functionally replace the **Characteristics** page.
3. Add the Departments list in this page, using the same [Template] paragraph as in the previous exercise.



MetaAttributeGroup: Firm.Characteristics

Name: Firm.Characteristics

Heritability:

Complements:

- Firm.Characteristics
  - MetaAssociationEnd
  - MetaAttribute
  - MetaPropertyPage (1)
    - Firm.Characteristics

Macro: _PropertyPageStandard

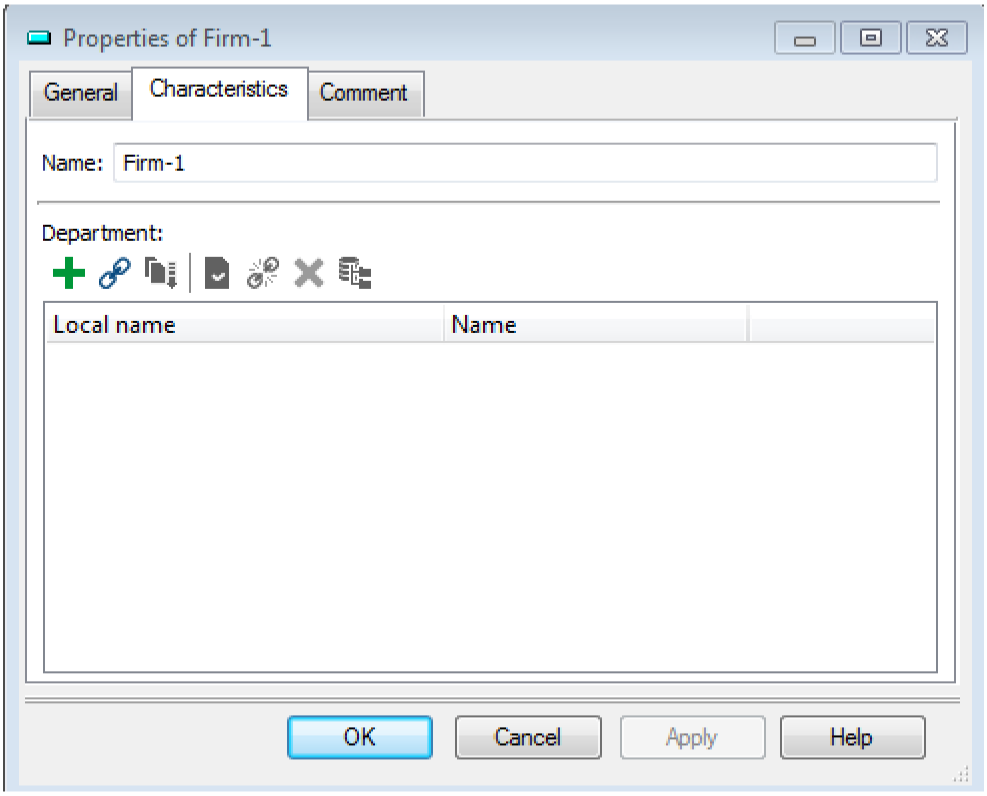
_GUIName: Characteristics

This implementation ,enables redefinition of the characteristices page of a M
must connect the page to an empty MetaAttributeGroup (this indicates that
attributes of the MetaClass that appear in the standard page) connected to _
If we explicitly define the attributes in this MetaAttributeGroup, any remainin

_Parameterization

```
[Template]
  Deps = Item(Department),Control(ListView)
```

Your work is completed and the **Characteristics** page of a firm is now (remember **Refresh Context**):



Name: Firm-1

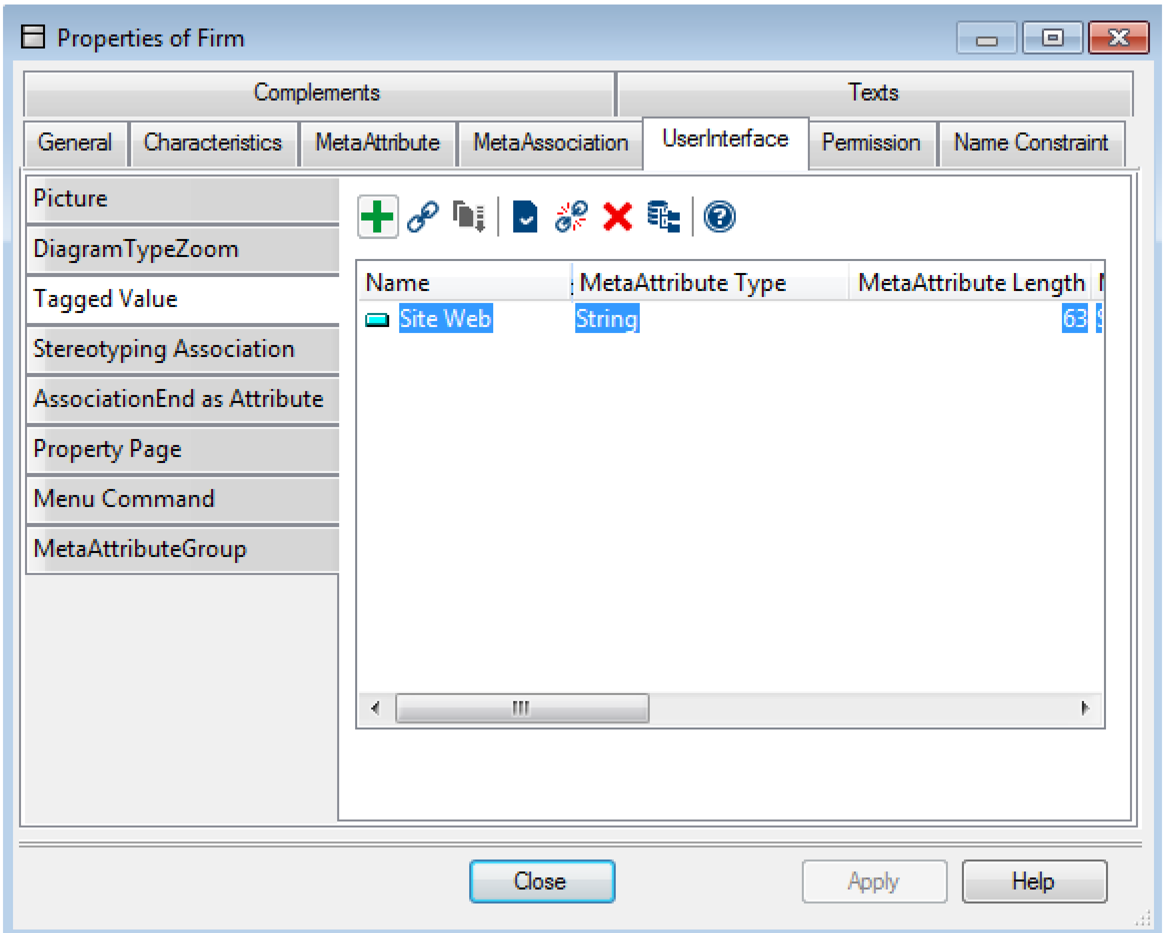
Department:

Local name	Name

Note that the **Name** field appears in this page, though you did not define it in the [Template] paragraph.

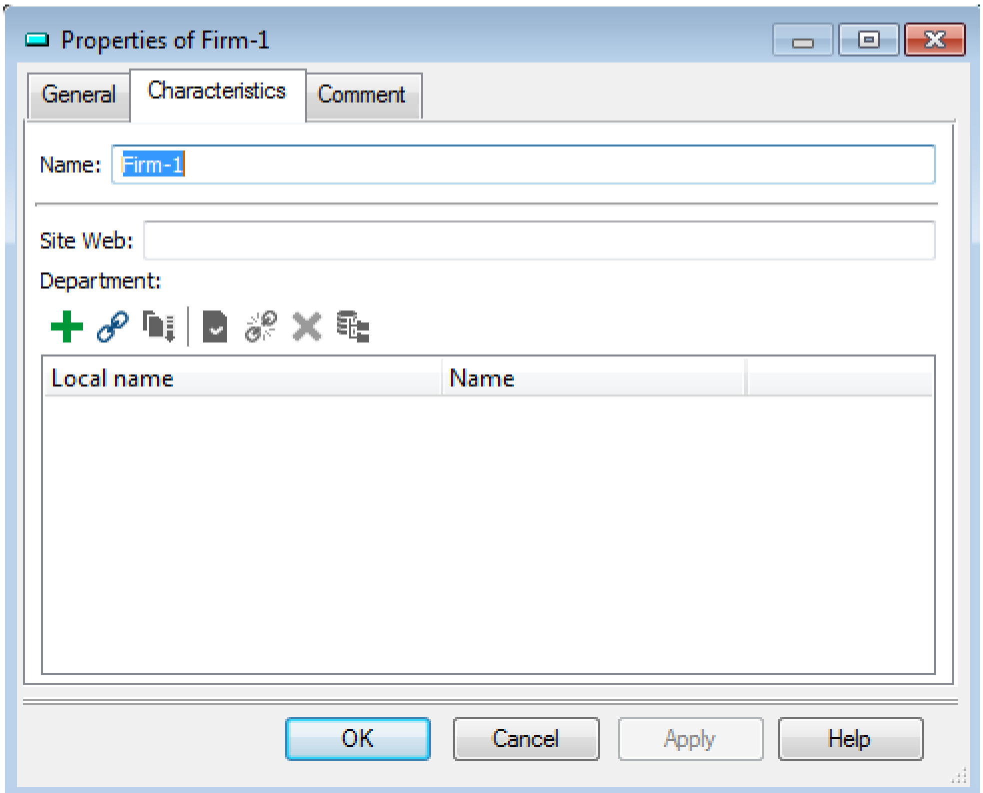
This is because this MetaPropertyPage was associated with a MetaAttributeGoup; this means that all the attributes it contains (in your case the default content of the "Characteristics" group) are automatically inserted in this page, without it being necessary to specify them in the page [Template]. You have therefore correctly defined an extension to the **Characteristics** page, without prejudging its prior content. You will be better able to appreciate the advantage of such a device later in this exercise.

To do this, take into account the fact that your client want to be able to define a *Site Web*' property for each of his *Firms*. This extension can take the form of a TaggedValue which you shall create for this MetaClass:



Name	MetaAttribute Type	MetaAttribute Length
Site Web	String	63

After refresh, this extension appears in the **Characteristics** page, without any other specification being required.



Name: Firm-1

Site Web:

Department:

Local name	Name	

Adding non-standard elements in a page

The previous exercise showed you that you could include a list (ListView) in a page by browsing a MetaAssociationEnd defined on a MetaClass of the object.
The main elements you can feature in a Properties page are listed below.

ListView: view as list

A ListView enables display of a list obtained by browsing a MetaAssociationEnd or a Query from an object in the Properties page. Queries that can be used in this context are either queries without parameters, or queries that have an occurrence as their only parameter.
You can:

- configure the list of columns displayed (which are generally attributes of browsed objects), as well as the list toolbar.
- create a list displaying several distinct collections, by selecting the active collection from the toolbar.

TreeView: view as tree

A treeView enables display of a tree, built from a list of MetaAssociationEnds and Selectors, the root of which is the object in the Properties page. This list can be deduced from an _Operator.
You can:

- define if a path can be of several levels or a single level.
- display folders or not.
- configure the tree toolbar.

Example: the following element displays the default navigation tree of an object. Here, *Navigate* is the _operator used to define the path.

```
Tree=Item(~laCnbKSWt000[Navigate]),Control(TreeView),Param(IsDeep)
```

Viewer: HTML formatter display

You can include an HTML formatter in a Properties page. This formatter will be generated from the object in the Properties page. In this way you can display any content in a Properties page, on condition that it is not intended for update.
You can trigger an action in this HTML document when you execute 'Apply' on the Properties sheet.

HelpComment: help comment display

You can include help text in a Properties page. This text can be text from system data or from modeled data.

SubPage: page in page display

You can display a Properties page in another page, for example to reuse an implementation which can be defined in a generic page.

Displaying elements from another object

So far, you have limited yourselves to display of data obtained directly from the object of the Properties sheet, the display of other objects only being possible by means of:

- TreeView, which allows display of only the name
- ListView, which limits display to columns of attributes, excluding for example the representation of comments.

You can however display in a page elements from other objects, on condition that there is a MetaAssociationEnd or a query allowing you to reach these objects from the main object.

Two possibilities are available:

- The MetaAssociationEnd (or Query) has maximum cardinality 1.
  In this case only one object is visible, and it is possible to include attributes of this object in the same way as for attributes of the main object.
- The MetaAssociationEnd (or Query) accesses a collection of objects. In this case we must have available in the page an element enabling selection of an object from this collection. This object having been selected, elements relating to the attributes of this object can be supplied.

In both cases, you must indicate in page configuration that the element does not relate to the main object, but to another object. To do this, use the Map concept, which will include access to this other object.

Direct display of an associated object

You will first configure a page on the *Department* MetaClass in which you will include the name and comment of the *Firm* of the department.

The *Firm* being unique for a department, you will not require a selector element. To do this, you must:

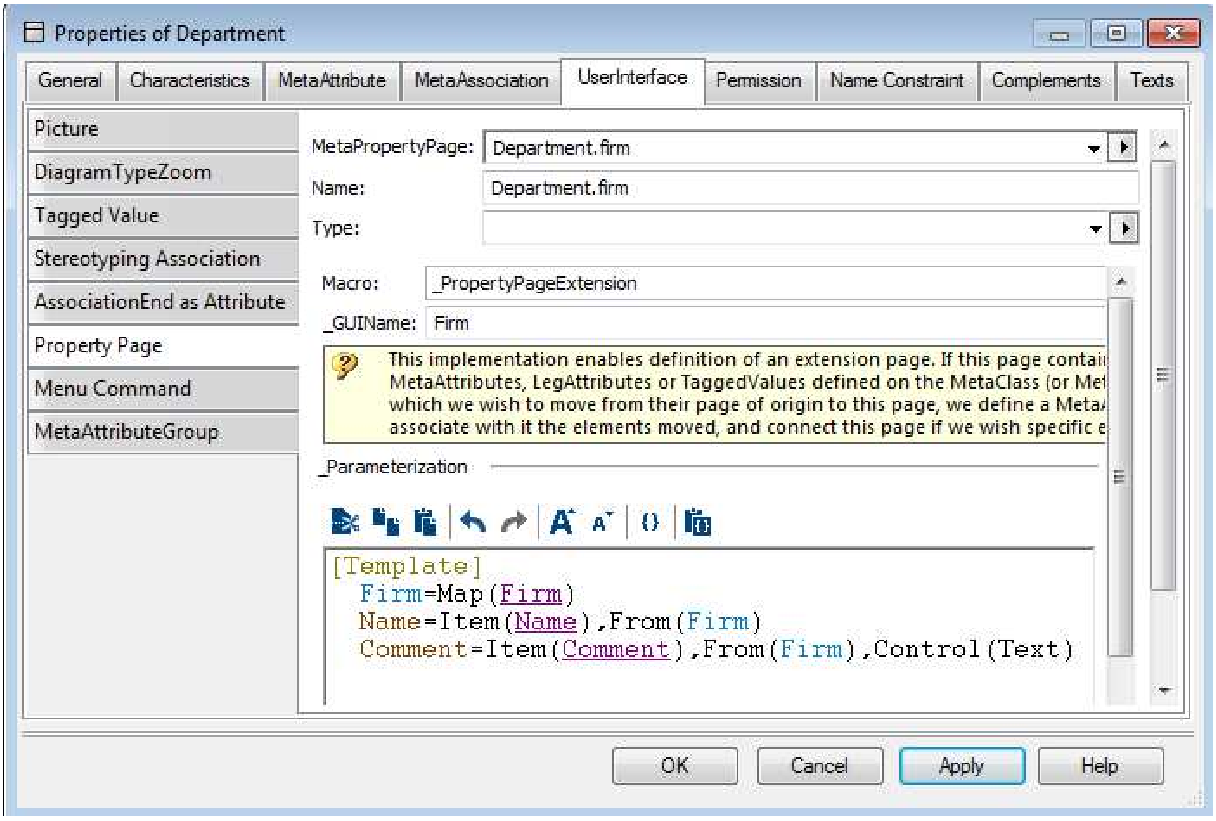
- declare a Map specifying the browsed element, here the *Firm* MetaAssociationEnd

```
Firm=Map(~(hmSXaOYEryC[Firm])
```

- declare the two elements.
  The keyword From(*map*) indicates which object will use the element.

```
Name=Item(~210000000900[Nom]),From(Firm)
Comment=Item(~f10000000b20[Comment]),From(Firm),Control(Text)
```

When the page has been created, you obtain:



MetaPropertyPage: Department.firm

Name: Department.firm

Type:

Macro: _PropertyPageExtension

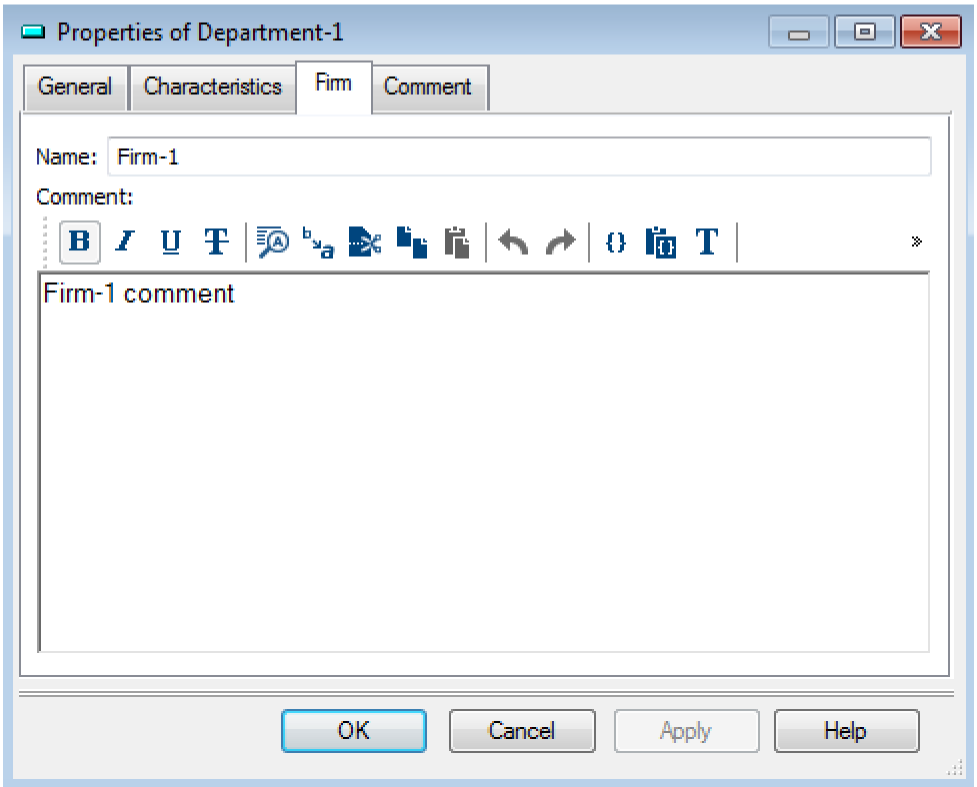
_GUIName: Firm

This implementation enables definition of an extension page. If this page contai
MetaAttributes, LegAttributes or TaggedValues defined on the MetaClass (or Me
which we wish to move from their page of origin to this page, we define a Meta
associate with it the elements moved, and connect this page if we wish specific e

_Parameterization

```
[Template]
  Firm=Map(Firm)
  Name=Item(Name),From(Firm)
  Comment=Item(Comment),From(Firm),Control(Text)
```

When you display the Properties sheet of a *Department*, you obtain:



Name: Firm-1

Comment:

Firm-1 comment

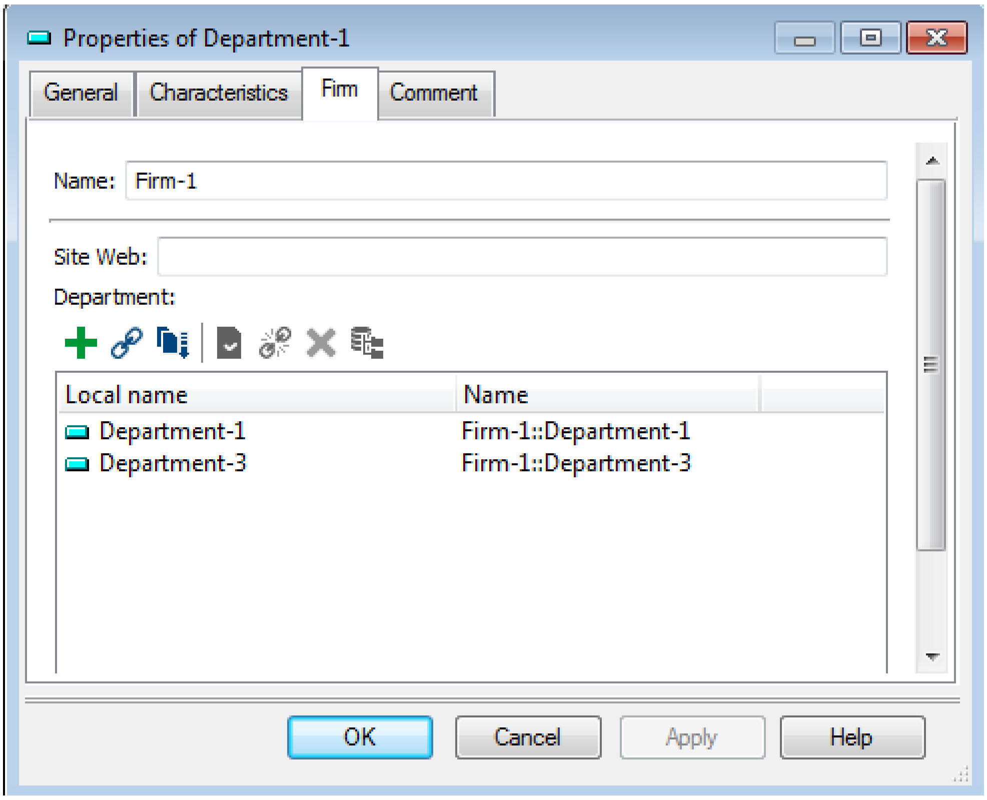
Instead of displaying elements of the *Firm*, you can directly display a Properties page of *Firm*, for example the Characteristics page.
To do this, you must use an element of SubPage type. The Template becomes:

```
[Template]
Firm=Map(~(hmSXaOYEryC[Firm])
Page=Item(~ZPhytJ)cE5k1[Firm.Characteristics]),From(Firm),Control(SubPage)
```

Warning:
In the above example, Firm.Characteristics has an IDabs different from that which you have created. To copy the correct IDAbs:

1. Open the Properties page of the **Firm** MetaClass.
2. Select the **User Interface** tab, then the **MetaAttributeGroup** subtab.
3. Under the **MetaPropertyPage** folder, copy Firm.Characteristics and paste it in the above Template.

The "Firm" tab now displays the **Characteristics** page of the *Firm*.



Name: Firm-1

Site Web:

Department:

Local name	Name
Department-1	Firm-1::Department-1
Department-3	Firm-1::Department-3

Displaying an object from a selection

We will now consider the case of a multiple collection; in this case you must be able to select an element from the list. This selection will be made by means of an element.

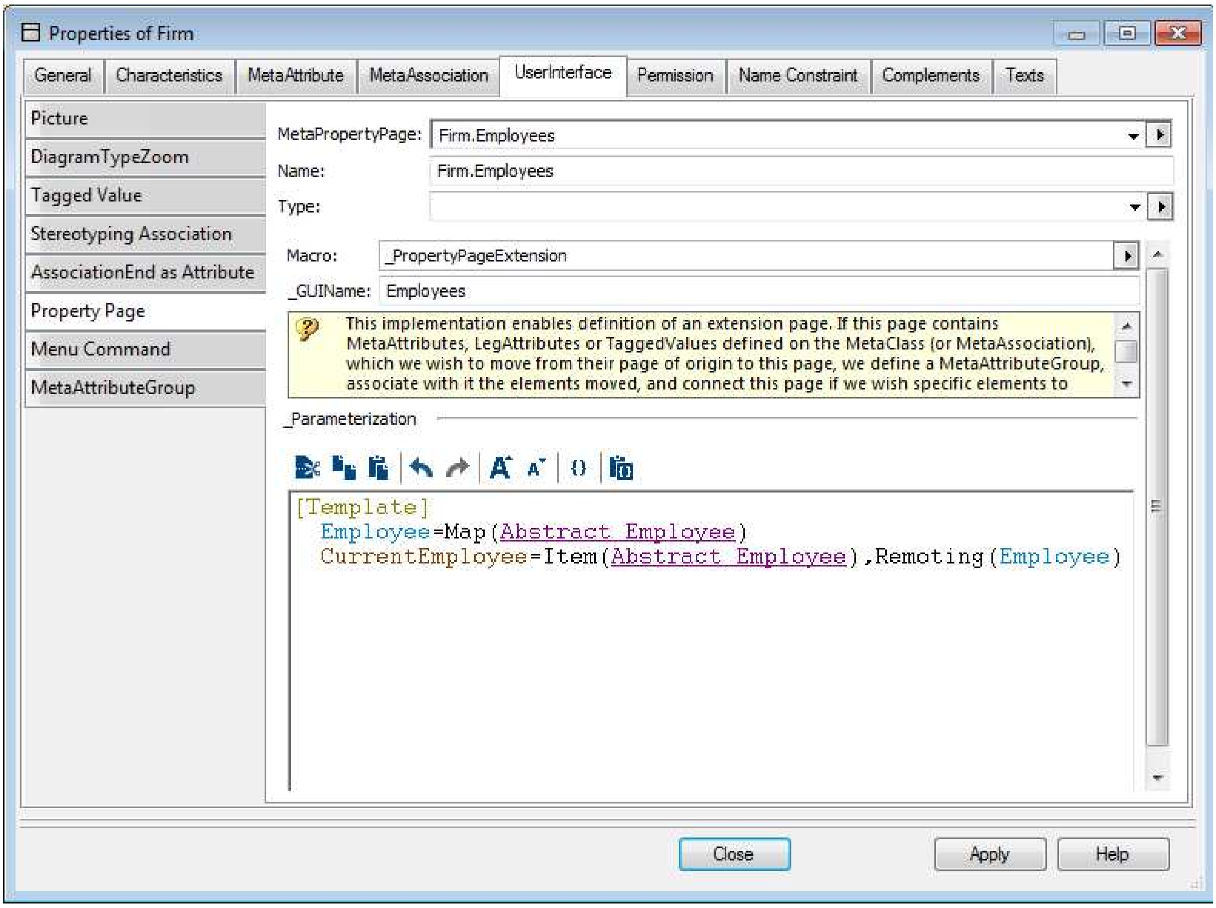
In this context, the identifier of the collection used to define the Map is not used: the Map is ‘passive’ and only knows its current object through the selection. However, by convention we associate with it, if possible, the browsed collection.

A dedicated element enables achievement of this objective: it takes the form of a drop-down list. To declare it, associate it with the browsed collection and the map concerned.

To illustrate this configuration, you will create an **Employee** page on the ***Firm*** MetaClass, enabling display of the Characteristics page of an Employee:

1. You will configure the element which enables selection of the Employee, by means of the Abstract Employee MetaAssociationEnd. In the configuration considered, declare a Map named Employee and its controller:

```
[Template]
  Employee=Map(~f7mZ(UQYEHTI[Abstract Employee])
  CurrentEmployee=Item(~f7mZ(UQYEHTI[Abstract Employee]),Remoting(Employee)
```



MetaPropertyPage: Firm.Employees

Name: Firm.Employees

Type:

Macro: _PropertyPageExtension

_GUIName: Employees

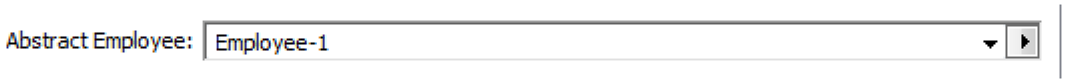
This implementation enables definition of an extension page. If this page contains MetaAttributes, LegAttributes or TaggedValues defined on the MetaClass (or MetaAssociation), which we wish to move from their page of origin to this page, we define a MetaAttributeGroup, associate with it the elements moved, and connect this page if we wish specific elements to

_Parameterization

```
[Template]
  Employee=Map(Abstract Employee)
  CurrentEmployee=Item(Abstract Employee),Remoting(Employee)
```

- The keyword ‘Remoting’ defines the dependency relationship between the ‘Employee’ Map and the "CurrentEmployee" element.
- The browsed collection is defined in the CurrentEmployee item; we define the Map by means of the same collection, but this parameter is not used in the Map.

Thus defined, CurrentEmployee is displayed in the following way:



Abstract Employee: Employee-1

2. You will define the subpage which describes the characteristic page.

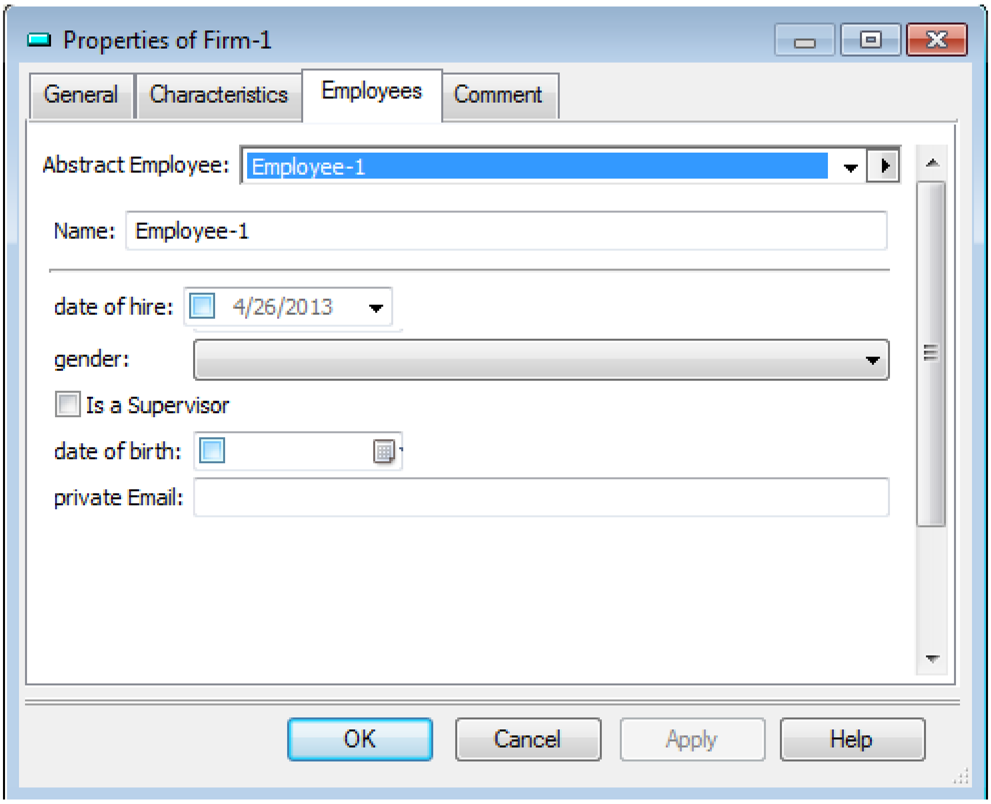
   There is an additional difficulty here, since you cannot directly cite the absolute identifier of the page to be displayed. In fact:

   - the Characteristics page does not physically exist
   - above all, the association used being generic, the objects listed can be either *Employee* or *Manager*, and the page to be displayed therefore differs according to the object selected

   You will use the following specific configuration for the SubPage:

```
EmployeePage=Item(~UZlkzjxjtO00[Characteristics]),From(Employee),Control(Su
bPage),Param(Computed,Type)
```

- The keyword ‘Computed’ indicates that the page must be calculated, and therefore that the identifier of the *Item*() is not a MetaPropertyPage.
- The keyword ‘Type’ indicates the calculation mode. In this case, the identifier of *Item*() is the type to be used: here we request page display of ‘Characteristics’ type of the selected object.


Properties of Firm-1

General | Characteristics | Employees | Comment

Abstract Employee: Employee-1

Name: Employee-1

date of hire: □ 4/26/2013

gender:

□ Is a Supervisor

date of birth: □

private Email:

Modifying customized page appearance

In previous chapters, we covered the problem of populating Properties pages and specification of their content, leaving the responsibility of organization and appearance of elements to be displayed to page implementation. It is possible that the resulting appearance is not satisfactory in terms of:

- element title
- grouping
- order in which elements appear
- size and position
- displayed control not suitable

These parameters can be redefined to achieve the required appearance.

Modifying element title

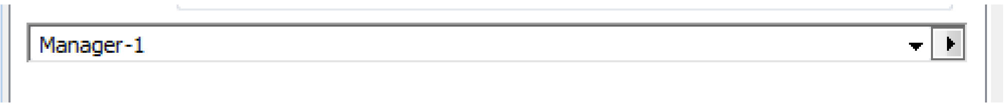
For a given element, you can define the presence, position and name of its title, by means of keywords *Title* and *Name*.

- The keyword *Title* enables definition of the presence or position of an element title using parameters Up, Left or No.
- The keyword *Name* enables definition of the title name. You can directly cite the name to be used, or a system repository occurrence (MetaAttribute, CodeTemplate for example): in this case you can manage multilingualism.

You will apply these configurations to the "Current Employee" element of your previous example, of which the name is not totally satisfactory.

- Deletion of title:

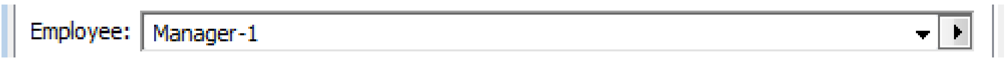
```
CurrentEmployee=Item(~f7mZ(UQYEHTI[Abstract
Employee]),Remoting(Employee),Title(No)
```



Manager-1

- Renaming 'static':

```
CurrentEmployee=Item(~f7mZ(UQYEHTI[Abstract
Employee]),Remoting(Employee),Name(Employee)
```


Employee: Manager-1

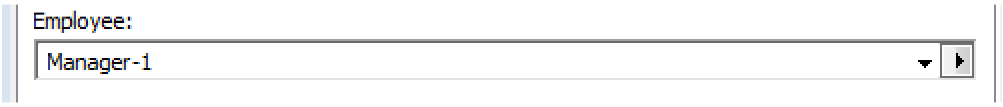
- Renaming by occurrence:

  ```
  CurrentEmployee=Item(~f7mZ(UQYEHTI[Abstract
  Employee]),Remoting(Employee),Name(~ThmSy)NYEzzB[Employee])
  ```

  Here the title 'Employee' is calculated from the name of the `~ThmSy)NYEzzB[Employee]` – system object, in this case the *Employee* MetaClass.

- Title positioned at top (by default for this type of control, the title is positioned at left: Title(Left)

  ```
  CurrentEmployee=Item(~f7mZ(UQYEHTI[Abstract
  Employee]),Remoting(Employee),Name(~ThmSy)NYEzzB[Employee]),Title(Up)
  ```


Employee:
Manager-1

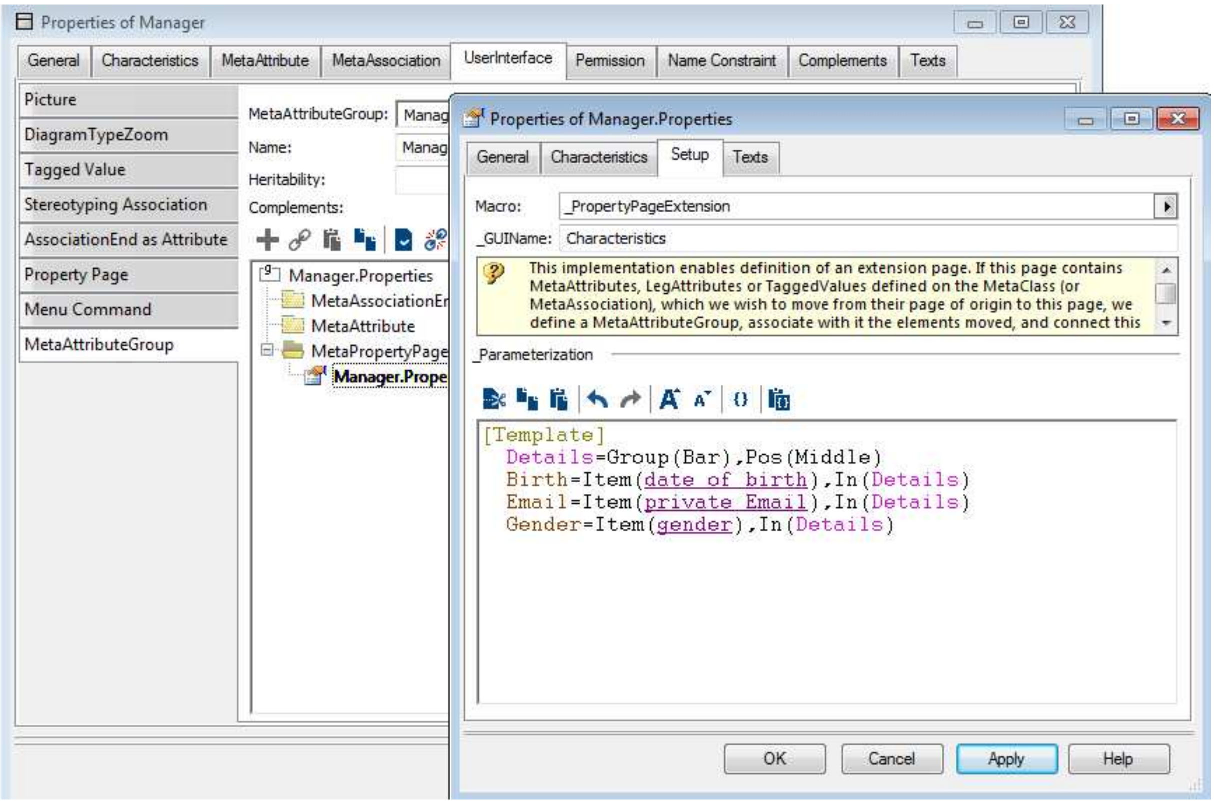
Regrouping elements

To group elements of a Properties page. use the Group concept.

The following exercise will allow you to use this concept in the Characteristics page of a *Manager*, in which you will group personal elements (date of birth, gender, email).

To do this, you will overload the Characteristics page by means of a Characteristics MetaAttributeGoup associated with a MetaPropertyPage.

In the Template paragraph of this page, you will cite the elements you wish to group.



Macro: _PropertyPageExtension

_GUIName: Characteristics

This implementation enables definition of an extension page. If this page contains MetaAttributes, LegAttributes or TaggedValues defined on the MetaClass (or MetaAssociation), which we wish to move from their page of origin to this page, we define a MetaAttributeGroup, associate with it the elements moved, and connect this

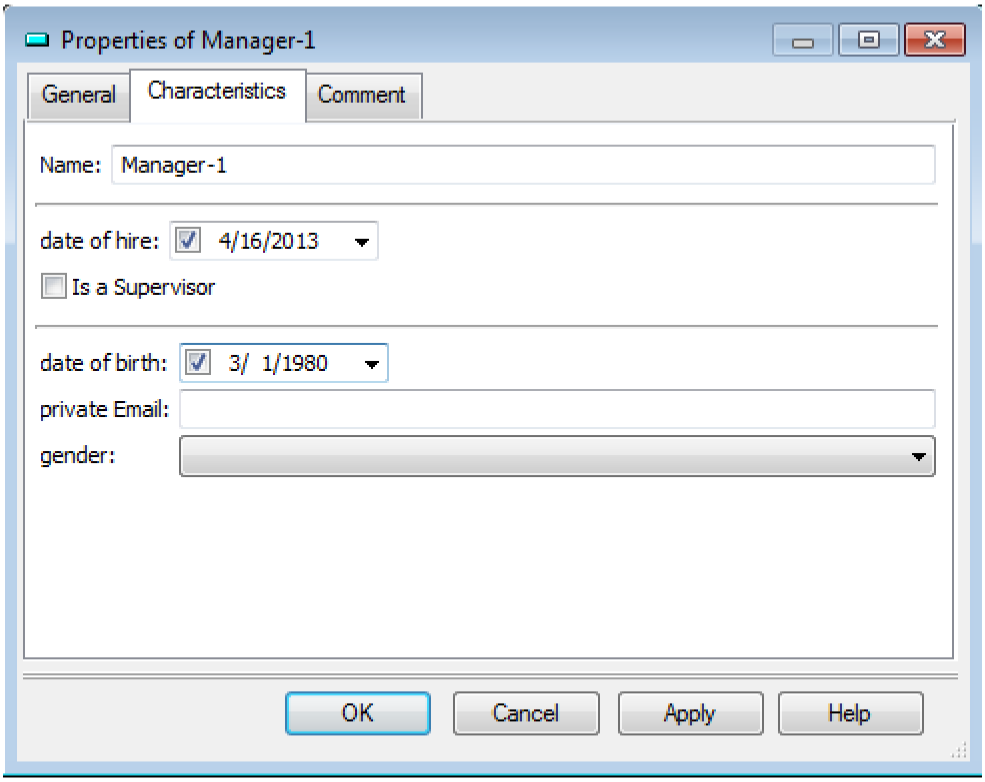
```
[Template]
  Details=Group(Bar),Pos(Middle)
  Birth=Item(date of birth),In(Details)
  Email=Item(private Email),In(Details)
  Gender=Item(gender),In(Details)
```

```
[Template]
  Details=Group(Bar),Pos(Middle)
  Birth=Item(~XFIiLwhZE5PJ[date of birth]),In(Details)
  Email=Item(~8DIipwhZEfTJ[private Email]),In(Details)
  Gender=Item(~6emSkyNYETjB[gender]),In(Details)
```

This template defines a group represented by a bar ('Bar') and positioned in the "middle" of the page.

Note that it is possible in any case to include in the [template] of a page the attributes it contains; in this case the element included in the template simply overloads standard behavior of the attribute.

In the resulting page, elements concerned are grouped and separated from other elements by a bar. They appear at bottom of the page since no other group has been defined (therefore 'middle' is the only available position for 'bottom').



Name: Manager-1

date of hire: ☑ 4/16/2013

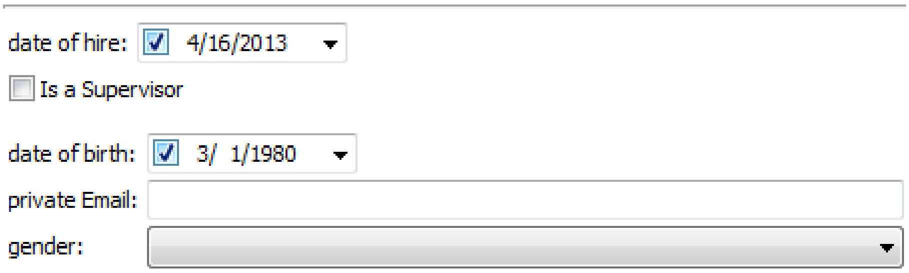
□ Is a Supervisor

date of birth: ☑ 3/ 1/1980

private Email:

gender:

Replace Group(Bar) by Group(Hidden) to regroup the elements, but without separator.



date of hire: ☑ 4/16/2013

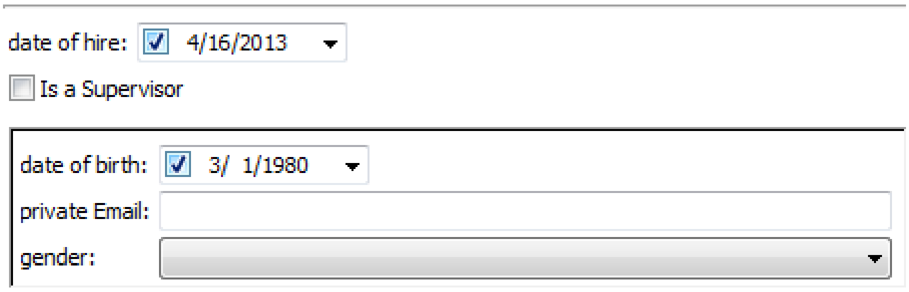
□ Is a Supervisor

date of birth: ☑ 3/ 1/1980

private Email:

gender:

Replace Group(Bar) by Group(Frame) to surround group elements by a frame.



date of hire: ☑ 4/16/2013

□ Is a Supervisor

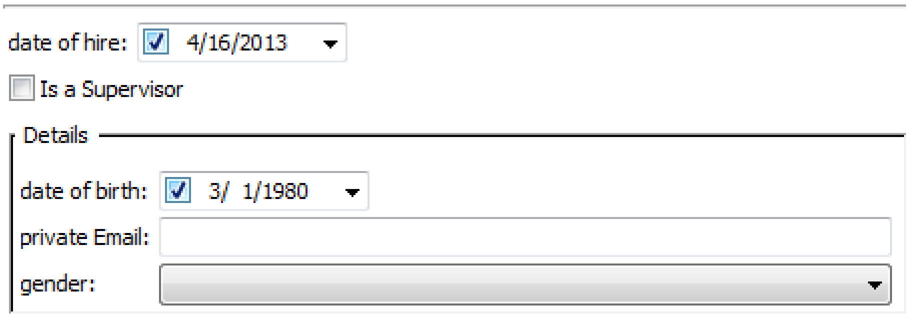
date of birth: ☑ 3/ 1/1980

private Email:

gender:

To name the group similarly as for an element, add Name(<Group Name>):

```
Details=Group(Bar),Pos(Middle),Name(Details)
```



date of hire: ☑ 4/16/2013

□ Is a Supervisor

Details

date of birth: ☑ 3/ 1/1980

private Email:

gender:

Note that a certain number of implicit groups are available, in particular the "Name" group you can see in the Characteristics page, and that you can insert elements in these groups.

Modifying element order

It is possible to modify the order of elements within the same group.

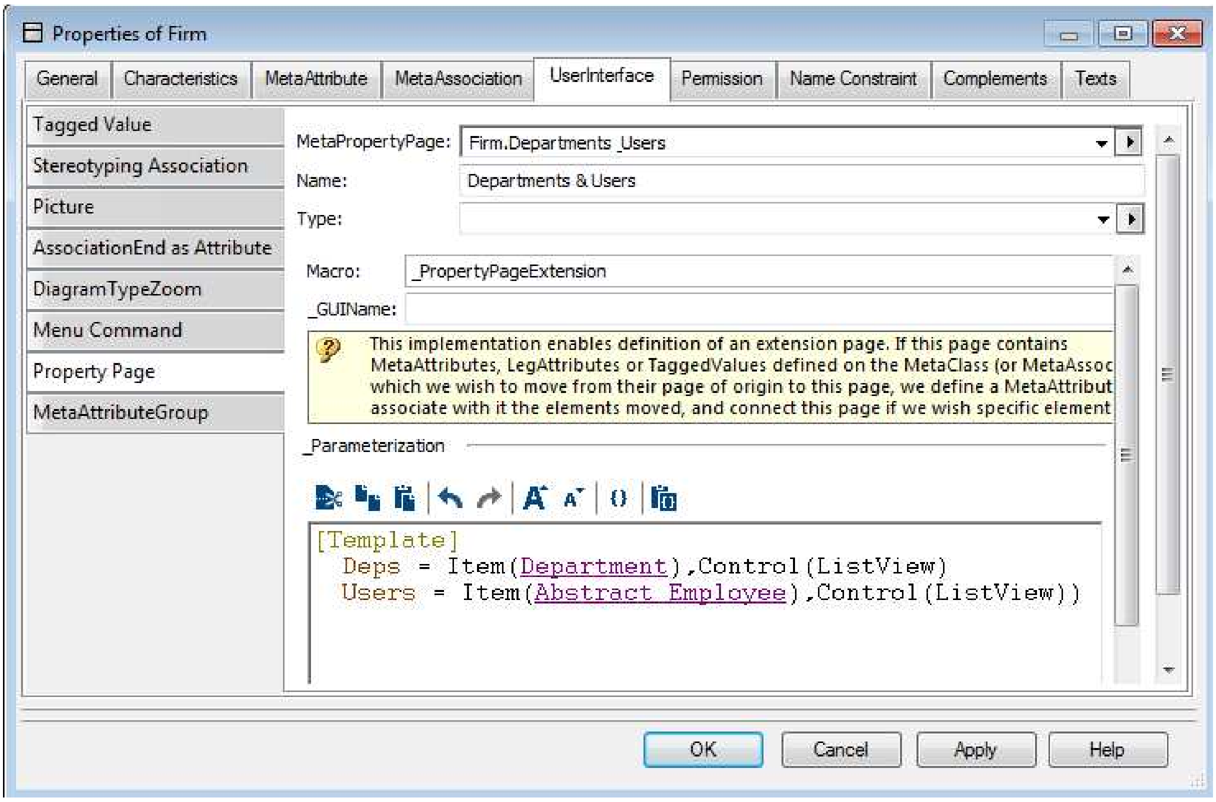
When elements are explicitly defined in a group, the order of their definition in the Template corresponds to display order (from MEGA 2009 SP4). To redefine order of elements, it may only be sufficient to define a Group.

For implicit elements, we use the order of attributes in the link with the MetaClass, that is (MetaClass/MetaAttribute), (MetaClass/TaggedValue), (MetaClass/_LegAttribute) according to the attribute type.
Complete classification of attributes is calculated by consolidating values of the Order attribute on each of these links. For example, a TaggedValue connected to a MetaClass by order number 25 will appear between the MetaAttributes connected to this same MetaClass (or Abstract class if the MetaAttribute is inherited) which have numbers 20 and 30.
To sort elements, it may only be sufficient to reorder them relating to their respective MetaClasses; this sort order will then be valid in all entries.
In no case is sort according to MetaAttributeGroup used.

Modifying element size

You can redefine size of an element; this is particularly useful for elements of ListView or TreeView type. For information, all ‘multiline’ elements (ListView, TreeView, Text, Viewer, HelpComment, SubPage…) use all space remaining in the page when they appear last. Otherwise they have a default height which you can modify.

To illustrate this possibility, you will define on Firm a Properties page displaying two lists.



MetaPropertyPage: Firm.Departments_Users

Name: Departments & Users

Type:

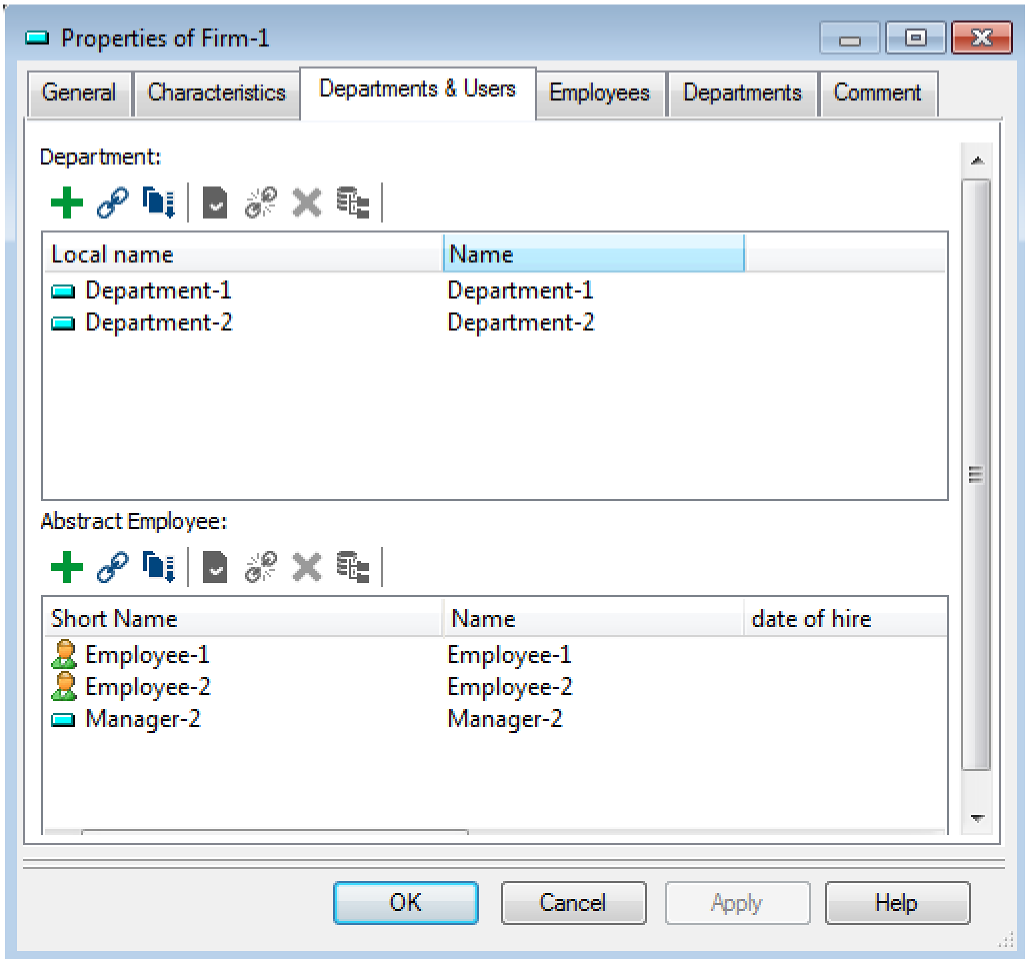
Macro: _PropertyPageExtension

_GUIName:

This implementation enables definition of an extension page. If this page contains MetaAttributes, LegAttributes or TaggedValues defined on the MetaClass (or MetaAssoc which we wish to move from their page of origin to this page, we define a MetaAttribut associate with it the elements moved, and connect this page if we wish specific element

```
[Template]
  Deps  = Item(Department),Control(ListView)
  Users  = Item(Abstract Employee),Control(ListView))
```

A default height is assigned to the "Deps" list, while the "Users" list, positioned at bottom of the page, will occupy the remaining space:



Department:

Local name	Name
Department-1	Department-1
Department-2	Department-2

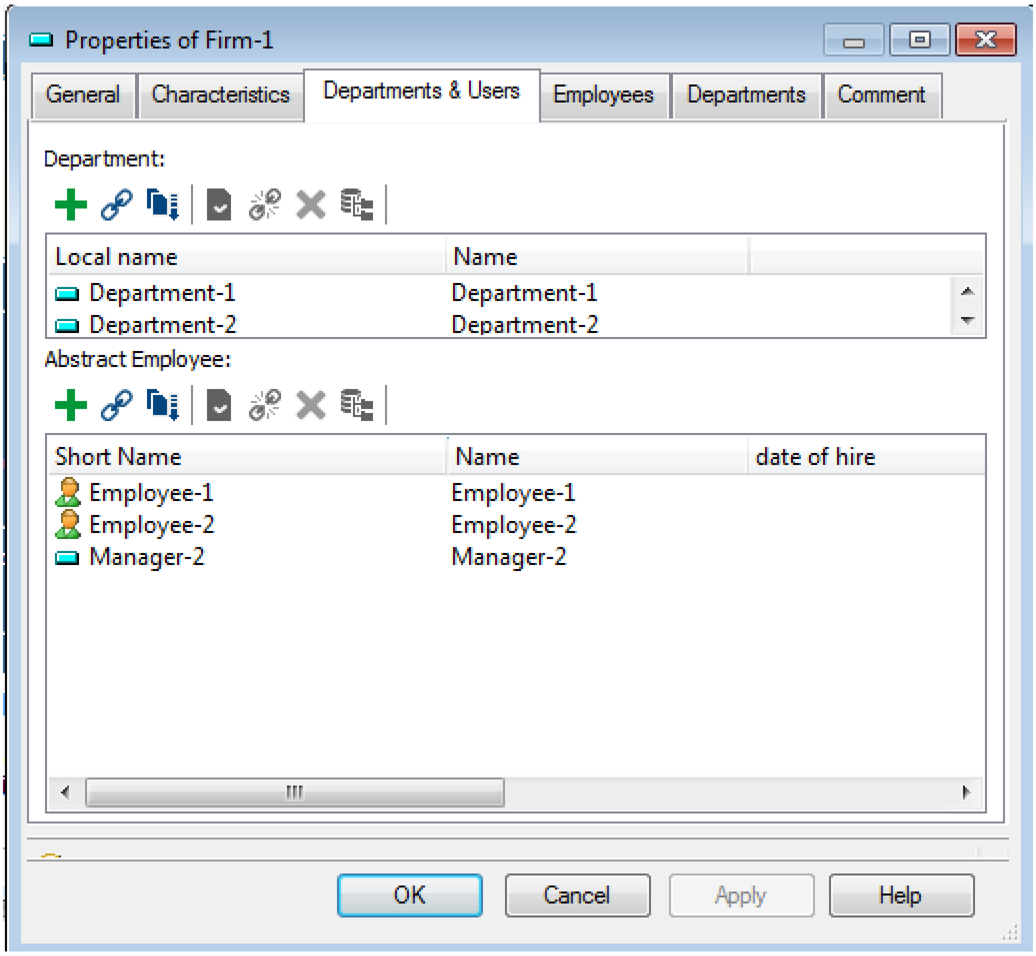
Abstract Employee:

Short Name	Name	date of hire
Employee-1	Employee-1	
Employee-2	Employee-2	
Manager-2	Manager-2	

In redefining a size for the Department area (keyword**Size()**)

```
Deps=Item(~)hmSXaOYEvyC[Department]),Control(ListView),Size(2000,50)
```

You obtain the following result:



Local name	Name
Department-1	Department-1
Department-2	Department-2

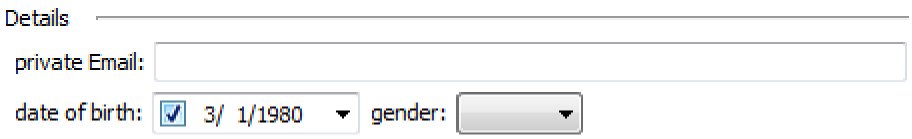
Short Name	Name	date of hire
Employee-1	Employee-1	
Employee-2	Employee-2	
Manager-2	Manager-2	

Width 2000 is deliberately exaggerated to indicate that the ListView should occupy all available horizontal space; height 50 includes the complete listview, that is its title, toolbar, header and the list itself.
Sizes are expressed in "dialog units", a unit in which a medium character of the font used to draw the page measures 5x8. 50 therefore represents approximately 5 lines (a line can equal 10 dialog units including two separation units)

Modifying element positioning and resizing

For even more detailed configurations, you can specify the exact size and position of elements in the same group. Position is expressed in dialog units and is relative to the Group. Elements without coordinates are added at bottom of the group.
Returning to the previous example, you can position elements so that 'gender' is positioned alongside 'date of birth'. The keyword **At()** enables definition of element position.

```
[Template]
  Details=Group(Bar),Pos(Middle),Name(Details)
  Email=Item(~8DIipwhZEfTJ[private Email]),In(Details),At(1,10)
  Birth=Item(~XFIiLwhZE5PJ[date of birth]),In(Details),At(1,25),Size(115,13)
  Gender=Item(~6emSkyNYETjB[gender]),In(Details),At(120,25),Size(70,13)
```



Details

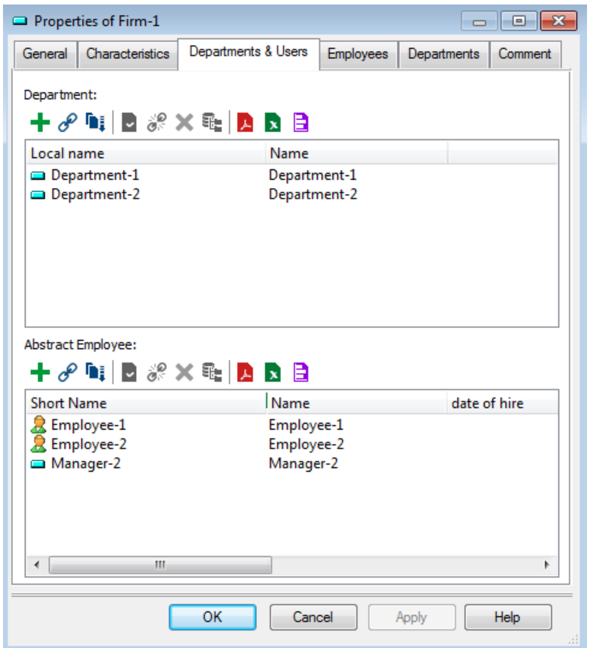
private Email:

date of birth: ☑ 3/ 1/1980 ▼ gender: ▼

Finally you can define resizing laws for a Properties page when its size is modified.
To do this, you must define basic dimensions for the page to define initial docking. It is on this basic dimension that areas must be positioned in the [Template] paragraph.
To apply this principle, you will customize the Department and User page so that both lists can be extended, and not just the second.

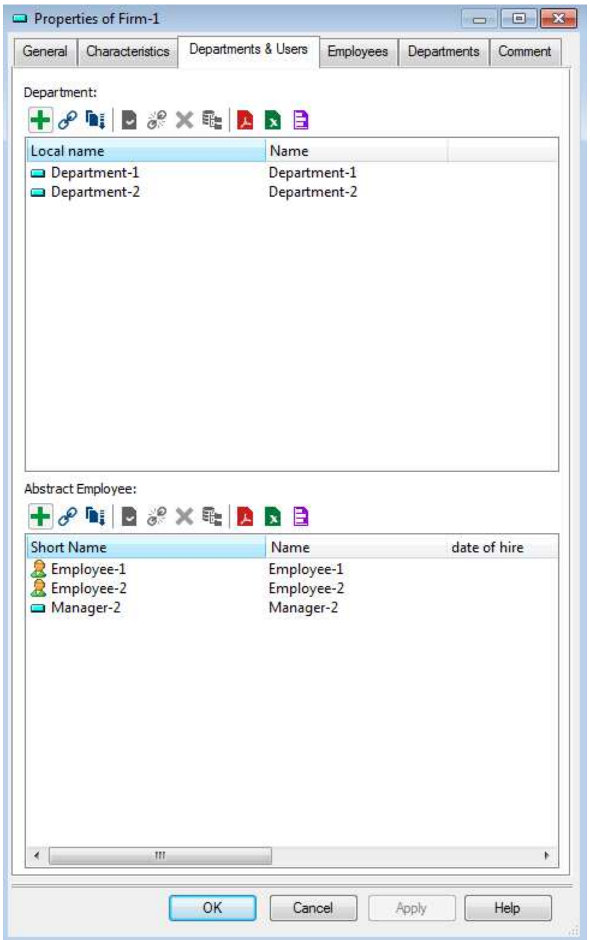
```
[Template]
Deps=Item(~)hmSXaOYEvyC[Department]),Control(ListView),At(1,1),Size(2000,100),VClip
(TopToCenter)
  Users=Item(~f7mZ(UQYEHTI[Abstract
Employee]),Control(ListView),At(1,115),Size(2000,100),VClip(CenterToBottom)
[Page]
MinHeight = 225
```

The keyword **VClip()** enables definition of movement of the element, attaching it at (top), (bottom), or (center) of the page. An element attached at two points (for example TopToCenter) will therefore be distorted. In our example, the two lists will be enlarged each to half of page enlargement.



Local name	Name
Department-1	Department-1
Department-2	Department-2

Short Name	Name	date of hire
Employee-1	Employee-1	
Employee-2	Employee-2	
Manager-2	Manager-2	



Local name	Name
Department-1	Department-1
Department-2	Department-2

Short Name	Name	date of hire
Employee-1	Employee-1	
Employee-2	Employee-2	
Manager-2	Manager-2	

Horizontal clipping uses keyword **HClip(),** and requires definition of page width ([Page] MinWidth)

Modifying element appearance and behavior

Deactivating an element

The following will enable modification of the type, or of the ability to update a control displayed for an attribute.

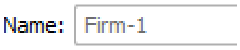
To do this, you will take the first version of the *Department.Firm* MetaPropertyPage on the *Department* MetaClass. In this example, the page Template is:

```
[Template]
  Firm=Map(~(hmSXaOYEryC[Firm])
  Name=Item(~210000000900[Nom]),From(Firm)
  Comment=Item(~f10000000b20[Comment]),From(Firm),Control(Text)
```

In this page, you do not wish the name of the *Firm* to be modifiable.
To do this, deactivate entry of the element:

```
Name=Item(~210000000900[Nom]),From(Firm),DisabledOn(Always)
```

➔ The effect of this is to gray the area and prohibit its modification:



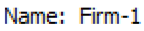
Name: Firm-1

.

You can also replace the 'Edit' control used by default for the name by a 'Static' control:

```
Name=Item(~210000000900[Nom]),From(Firm),Control(Static)
```

➔ The appearance of the area changes:



Name: Firm-1

Forcing an entry

To make it mandatory to enter *gender* of a *Manager* in the Manager.Properties customized Properties page define previously, use the keyword Mandatory.

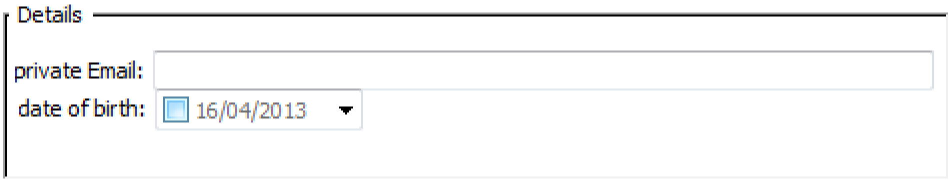
```
Gender=Item(~6emSkyNYETjB[gender]),In(Details),At(120,25),Size(70,13),Manda
tory(Yes)
```

With this configuration, you cannot assign value (None) to *gender*.

Hiding a standard element

In this same page, you now want to hide gender. This attribute being derived from MetaAttributeGroup, we must hide it explicitly. To do this, we use the keyword Visibility.

```
Gender=Item(~6emSkyNYETjB[gender]),In(Details),At(120,25),Size(70,13),Visib
ility(Hidden)
```


Details

private Email:

date of birth: □ 16/04/2013

With:

- Visibility(Hidden): the attribute is never visible.
- Visibility(Admin): the attribute is hidden to users who do not have 'Expert' metamodel access.
- Visibility(Always): the attribute normally restricted to experts or advanced users can be made visible to all users.

Modifying page behavior

Hiding a page

To condition appearance of a MetaPropertyPage, you can define a condition in the [Filter] paragraph of page configuration.

```
[Filter]
Condition = <condition>
```

The following example will hide the "*Firm*" page previously defined on a *Department* when the Department is not associated with a *Firm*.

The condition envisaged will relate to MetaAssociationEnd ~(hmSXaOYEryC[Firm]

You can relate the condition to the number of associated *Firms* by means of test *ItemCount:*

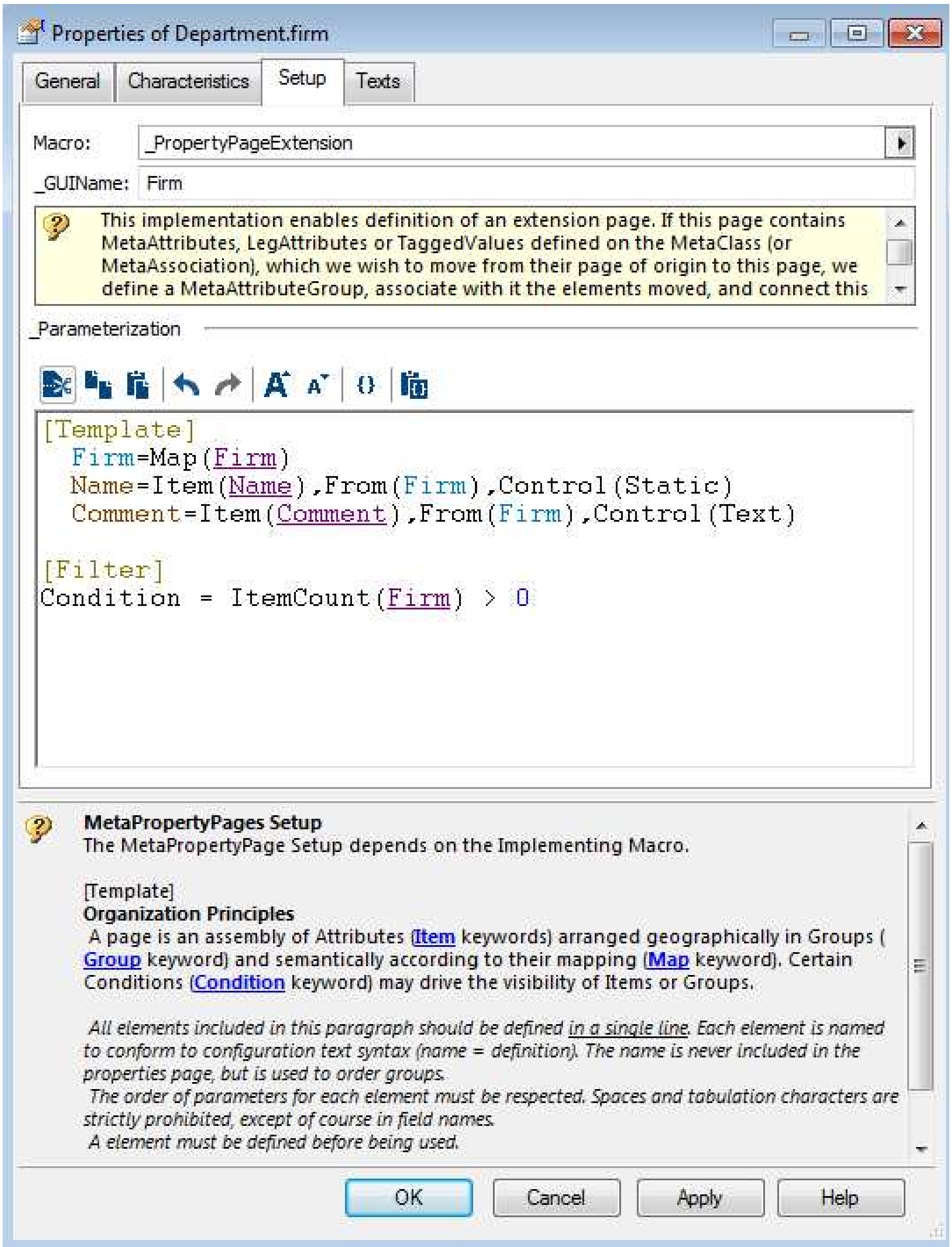
```
ItemCount(~(hmSXaOYEryC[Firm]) > 0
```

In the [Filter] section, you must specify:

```
[Filter]
Condition = ItemCount(~(hmSXaOYEryC[Firm]) > 0
```

Conditions can relate to attributes of the object, to attributes of connected objects, or to context data (for example user metamodel access).

To determine conditioning possibilities, you can be guided by _Parameterization text entry help (called by pressing keys <Ctrl>+<Space>, requesting online help (F1 on condition keyword in _Parameterization text) or by consulting the help section available at bottom of the page. A chapter in "*HOPEX Forms*" document is also dedicated to this subject.



Macro: _PropertyPageExtension

_GUIName: Firm

This implementation enables definition of an extension page. If this page contains MetaAttributes, LegAttributes or TaggedValues defined on the MetaClass (or MetaAssociation), which we wish to move from their page of origin to this page, we define a MetaAttributeGroup, associate with it the elements moved, and connect this

_Parameterization

```
[Template]
  Firm=Map(Firm)
  Name=Item(Name),From(Firm),Control(Static)
  Comment=Item(Comment),From(Firm),Control(Text)

[Filter]
Condition = ItemCount(Firm) > 0
```

MetaPropertyPages Setup
The MetaPropertyPage Setup depends on the Implementing Macro.

[Template]
Organization Principles
A page is an assembly of Attributes (Item keywords) arranged geographically in Groups (Group keyword) and semantically according to their mapping (Map keyword). Certain Conditions (Condition keyword) may drive the visibility of Items or Groups.

All elements included in this paragraph should be defined in a single line. Each element is named to conform to configuration text syntax (name = definition). The name is never included in the properties page, but is used to order groups.
The order of parameters for each element must be respected. Spaces and tabulation characters are strictly prohibited, except of course in field names.
A element must be defined before being used.

Conditions on page elements

In the previous chapter we saw that it was possible to deactivate or hide an element. In particular, the keyword Visibility(Admin) enabled restriction of visibility to expert users: appearance of the page can therefore differ according to context.

This principle can be generalized: you can deactivate or hide elements and groups according to conditions explicitly defined in the page Template.

To do this, use keywords DisabledOn(*condition*) and HiddenOn(*condition*) after having defined a condition.

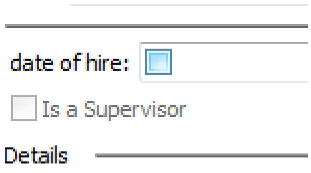
To illustrate conditioning, you shall specify that the "Is a Supervisor" attribute, visible in the Characteristics page of a Manager, will be activated when the Manager has *Employees*. The corresponding condition is declared in the page Template as follows:

```
HasEmployee=Condition(ItemCount(~demSW1OYEDCC[Employee]) > 0)
```

Simply cite this condition when redefining the 'Is a Supervisor' element.

```
Supervisor=Item(~6emSr2OYEnKC[Is a Supervisor]),DisabledOn(HasEmployee)
```

If the condition is false (if the Manager has no employees), the element is deactivated.



Similarly, the element would be hidden if you apply the keyword HiddenOn(HasEmployee).

You can directly deactivate or hide complete content of a group by applying these keywords to a Group in the Template. We can test this possibility on the Details group previously defined in the 'Manager' page.

```
Details=Group(Bar),Pos(Middle),Name(Details),HiddenOn(HasEmployee)
```

Redefining page template content

When content of a Properties page is highly dependent on specific characteristics of the object, definition of conditions can be complicated. To simplify this type of configuration, it is possible to extend Template content by means of an IncludeTemplate directive.

This directive initially enables inclusion in a Template paragraph of lines obtained from a text located on another object.

```
Extra=IncludeProfile(<objectId>),Text(<textid>),Paragraph(prgName)
```

If the keyword Text is not specified, the default text is _Parameterization.

If the keyword Paragraph is not specified, the default paragraph is ‘Template’

In this first use mode, <objectId> represents an explicit system object; this mode enables reuse of a configuration in several templates.

In more highly developed use modes, it is possible to calculate the system object to which the customization will relate.

Configuration depending on a system object connected to the current object

This object can for example be obtained by browsing a MetaAssociationEnd from the current object, which is indicated by the keyword Origin(Service)

```
ExtraParam=IncludeProfile(<AssociationEndId),Origin(Service),Paragraph(prgN
ame:Template)
```

In particular, this device enables extension of an object page according to the _type of the object (_type being a system repository object).

Configuration depending on a MetaAttributeValue

Another possibility is to extend a page according to a particular value of MetaAttributeValue. In this case you must cite the MetaAttribute of the object and the keyword Origin(Value)

You can test this kind of extension on the Manager MetaClass.

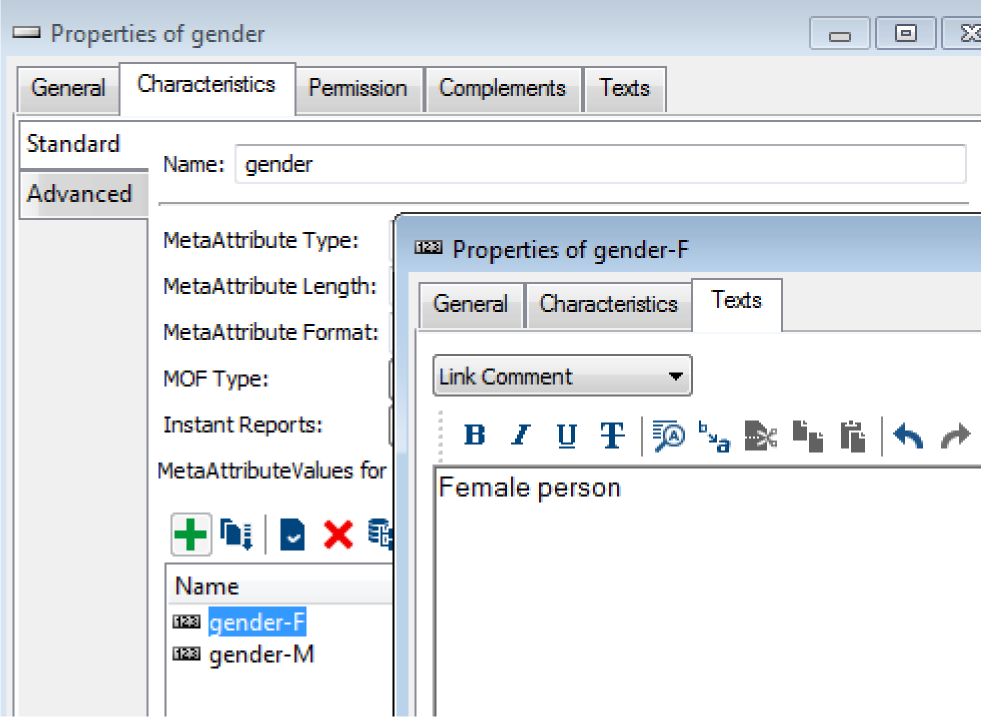
You will first comment the MetaAttributeValues of the gender MetaAttribute, then define a template extension specific to the MetaAttributeValue in the _Parameterization text of the MetaAttributeValue, enabling display of its comment:

- For gender-F

```
[extension:Template]
   helpgrp=Group(Bar),Pos(Bottom)
   help=Item(~yhmSszNYEHuB[gender-F]),In(helpgrp),Control(HelpComment)
```

- For gender-M

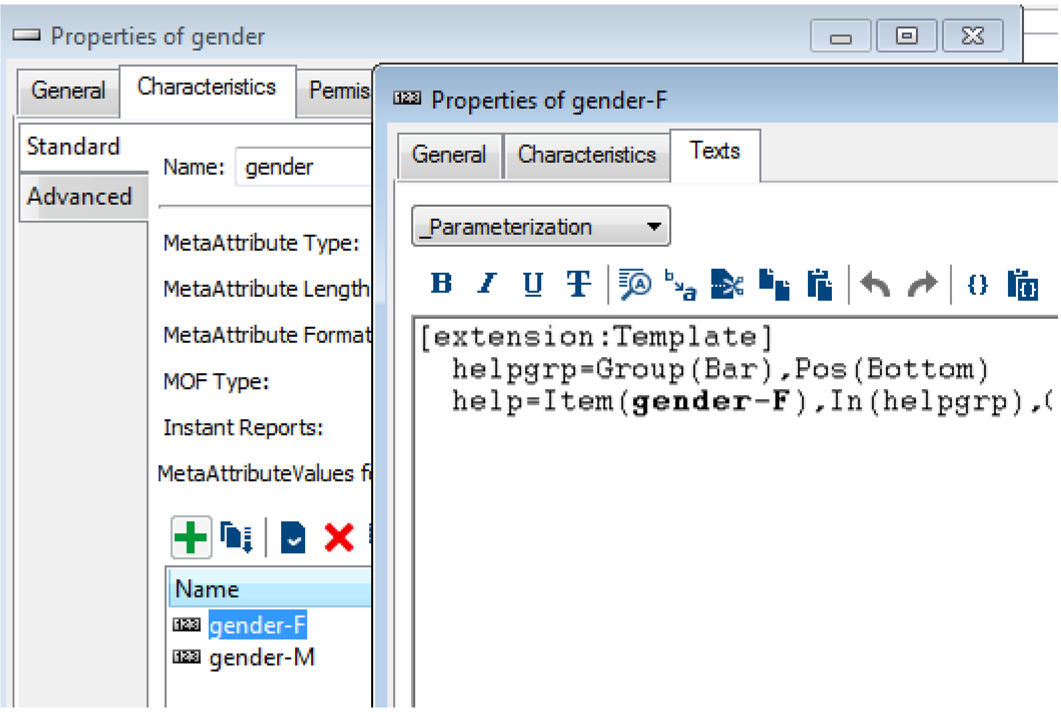
```
[extension:Template]
     helpgrp=Group(Bar),Pos(Bottom)
     help=Item(~IgmSXzNYEzoB[gender-M]),In(helpgrp),Control(HelpComment)
```



Name: gender

gender-F
gender-M

Female person

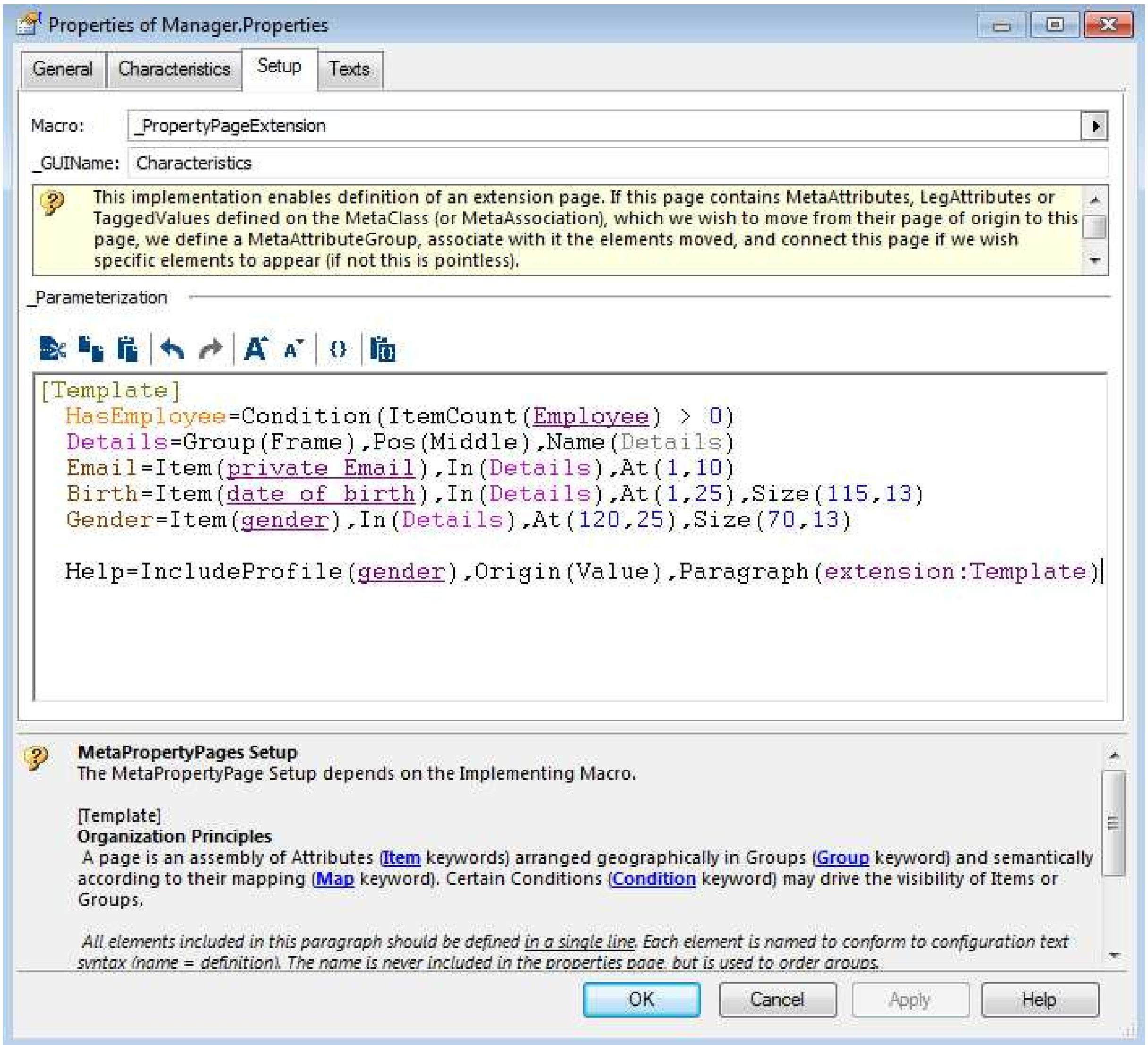


Name: gender

```
[extension:Template]
  helpgrp=Group(Bar),Pos(Bottom)
  help=Item(gender-F),In(helpgrp),(
```

Finally, we add the inclusion directive in the Manager.Properties MetaPropertyPage :

```
Help=IncludeProfile(~6emSkyNYETjB[gender]),Origin(Value),Paragraph(extension:Template)
```



Macro: _PropertyPageExtension

_GUIName: Characteristics

This implementation enables definition of an extension page. If this page contains MetaAttributes, LegAttributes or TaggedValues defined on the MetaClass (or MetaAssociation), which we wish to move from their page of origin to this page, we define a MetaAttributeGroup, associate with it the elements moved, and connect this page if we wish specific elements to appear (if not this is pointless).

_Parameterization

```
[Template]
  HasEmployee=Condition(ItemCount(Employee) > 0)
  Details=Group(Frame),Pos(Middle),Name(Details)
  Email=Item(private Email),In(Details),At(1,10)
  Birth=Item(date of birth),In(Details),At(1,25),Size(115,13)
  Gender=Item(gender),In(Details),At(120,25),Size(70,13)

  Help=IncludeProfile(gender),Origin(Value),Paragraph(extension:Template)
```

MetaPropertyPages Setup
The MetaPropertyPage Setup depends on the Implementing Macro.

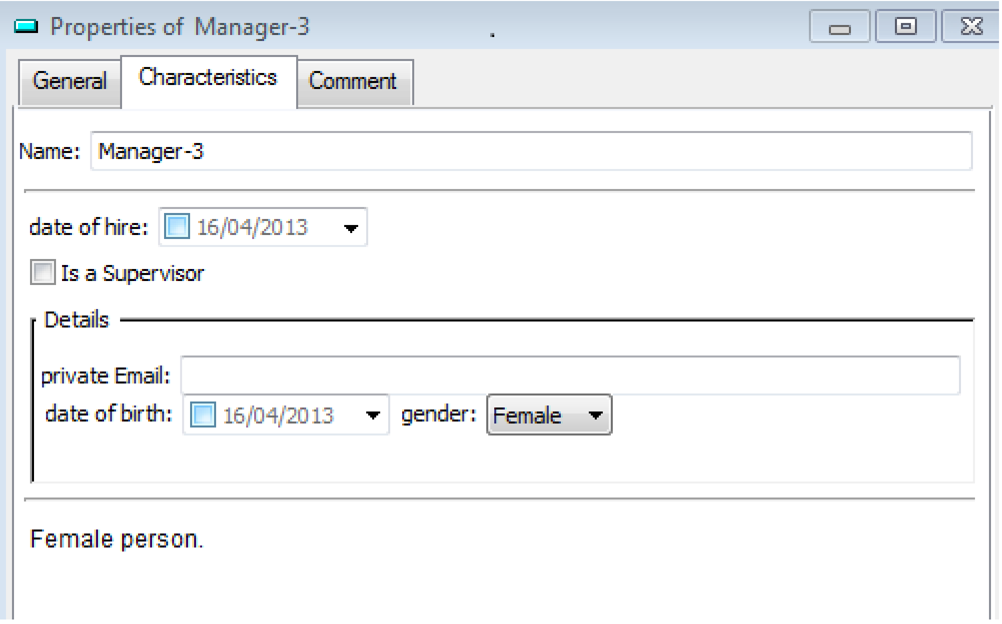
[Template]
Organization Principles
A page is an assembly of Attributes (Item keywords) arranged geographically in Groups (Group keyword) and semantically according to their mapping (Map keyword). Certain Conditions (Condition keyword) may drive the visibility of Items or Groups.

All elements included in this paragraph should be defined in a single line. Each element is named to conform to configuration text syntax (name = definition). The name is never included in the properties page, but is used to order groups.

The help section will appear at bottom of the page (since it is in a Pos(Bottom group)) and will depend on the *gender* MetaAttribute.

If the value is not defined, nothing will be inserted.

Note that this configuration is calculated from an effective value: to note the change, click **Apply** to validate gender modification.


Properties of Manager-3

Name: Manager-3

date of hire: □ 16/04/2013

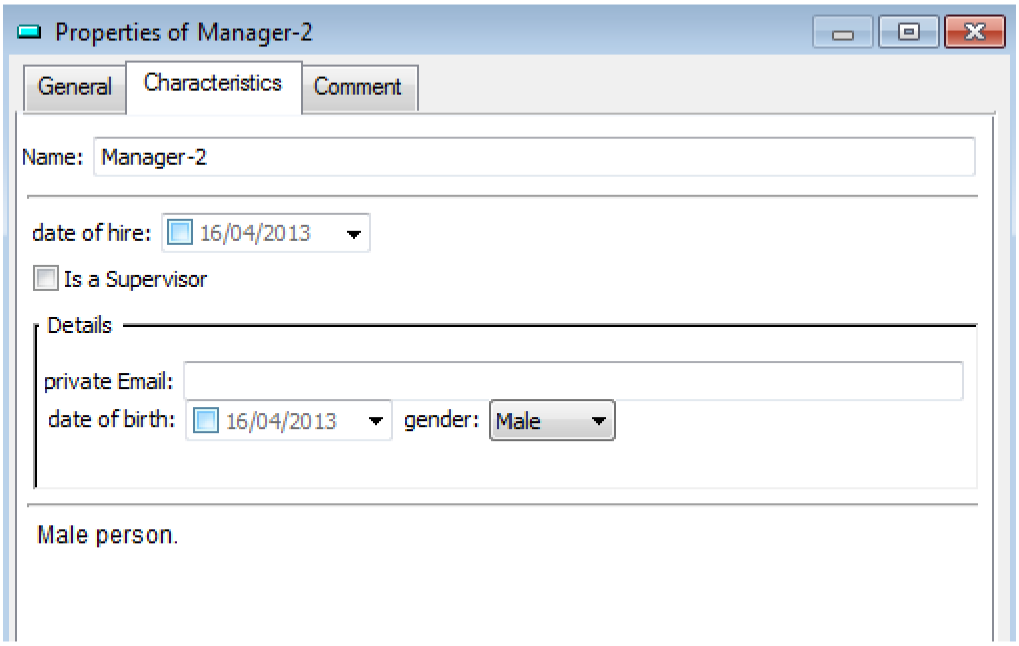
□ Is a Supervisor

Details

private Email:

date of birth: □ 16/04/2013 gender: Female

Female person.


Properties of Manager-2

Name: Manager-2

date of hire: □ 16/04/2013

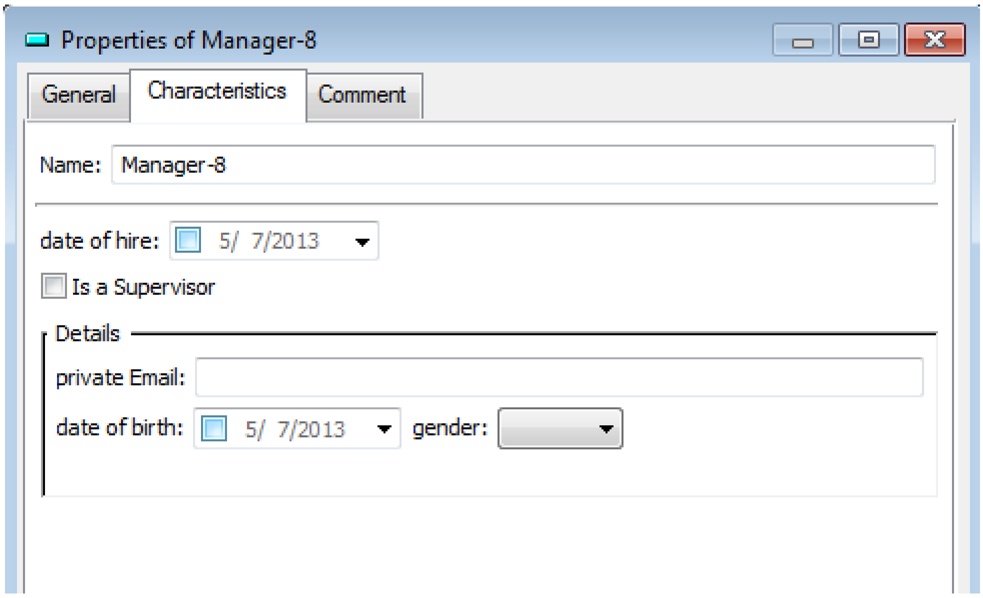
□ Is a Supervisor

Details

private Email:

date of birth: □ 16/04/2013 gender: Male

Male person.



Properties of Manager-8

General | Characteristics | Comment

Name: Manager-8

date of hire: □ 5/ 7/2013

□ Is a Supervisor

Details

private Email:

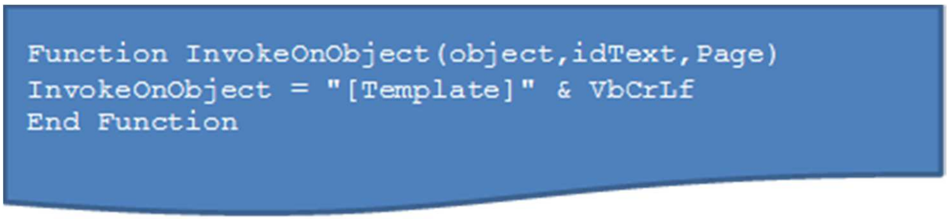
date of birth: □ 5/ 7/2013 gender:

Configuration calculated by macro

You can dynamically generate page content by means of a Macro.

```
ExtraParam=IncludeProfile(<MacroId>),Origin(Macro)
```

The macro should implement the InvokeOnObject function which should return a character string representing a "Template" paragraph.



```
Function InvokeOnObject(object,idText,Page)
InvokeOnObject = "[Template]" & VbCrLf
End Function
```

In this way you can freely customize a Properties page, by making dependent the Template generated from the object supplied at input.

Forms – Wizard Implementation – Tutorial

Objective

HOPEX MetaStudio enables definition of wizards by assembling properties pages and non-interactive code, which can be written in Script.

These wizards have been designed to:

- redefine HOPEX object creation dialog boxes, covered in the first part of this document.
- develop basic user interfaces in the form of dialog boxes or wizards, covered in the second part of the document.

Initializing the courseware

In order to be independent of evolution of the **HOPEX** metamodel, this courseware is based on a specifically designed metamodel extension.

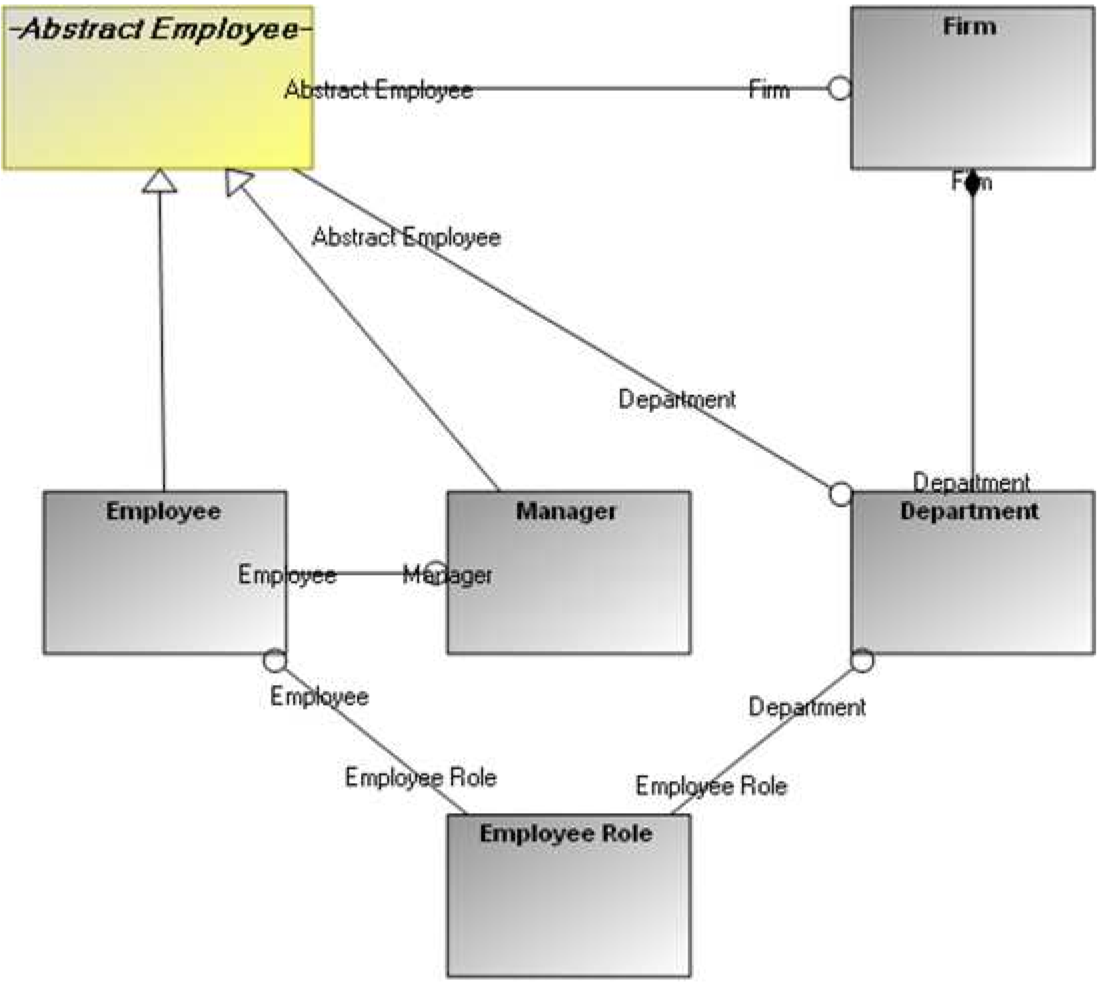
Before starting work, you must initialize your environment:

1. Download the following command file:

   MetaModel Extensions for training courses.mgr

   available here: MetaModel Extensions for training courses.zip.
2. From HOPEX, import into the System repository the MetaModel Extensions for training courses.mgr command file.
3. From the HOPEX Administration application (Administration.exe), recompile the metamodel.

Content of the specific metamodel is the following:



Customizing a creation wizard

This customization can be carried out at three definition levels. You can:

- if this satisfies the objective of the wizard, simply define the MetaAttributes, or Role Attributes that you want to use in the standard page of the creation wizard.
- insert initialization code for the object, or redefine the standard creation page according to context.
- add pages to the wizard, and if necessary modify their sequencing.

➔ The last two levels require creation of a Meta Wizard associated with the MetaClass. The Wizard processing code is implemented in one or in several Macros specified as Wizard Trigger. Pages defined for the wizard are MetaPropertyPages.

Prerequisites

Before customizing a creation wizard, you must define the following **environment options** value:

- **Authorize Dispatched Objects Deletion** : "Authorize"

This customization example is based on the **Employee** MetaClass.

Adding properties in the standard creation page

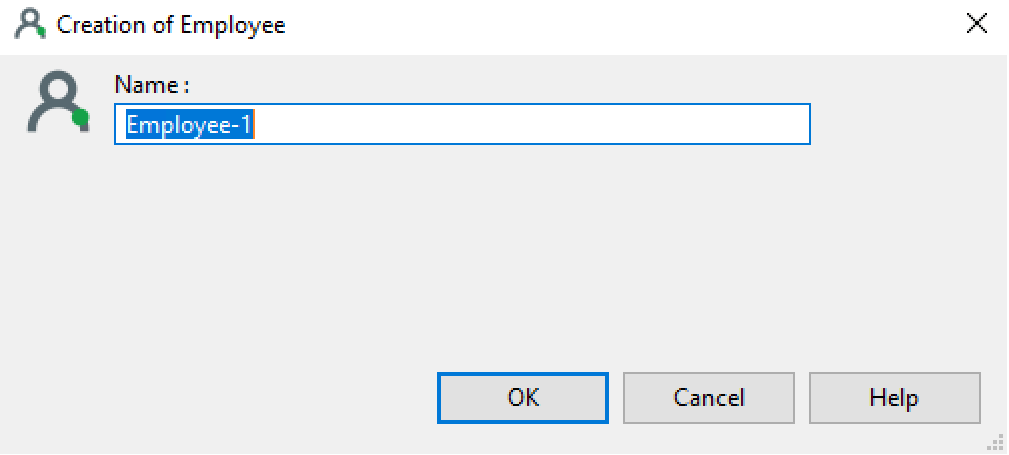
This first customization level does not require creation of a Meta Wizard, since this customization is directly configured on the appropriate concepts of the metamodel by means of the **Detailed Behavior** attribute. This attribute is present on:

- MetaAssociation (MetaClass/MetaAttribute)
- MetaAssociationEndAttribute (via the MetaAssociationEnd _LegAttribute, which indicates that the role wishes to be seen as an attribute).

The **Detailed Behavior** attribute is modified via a multiple choice list:

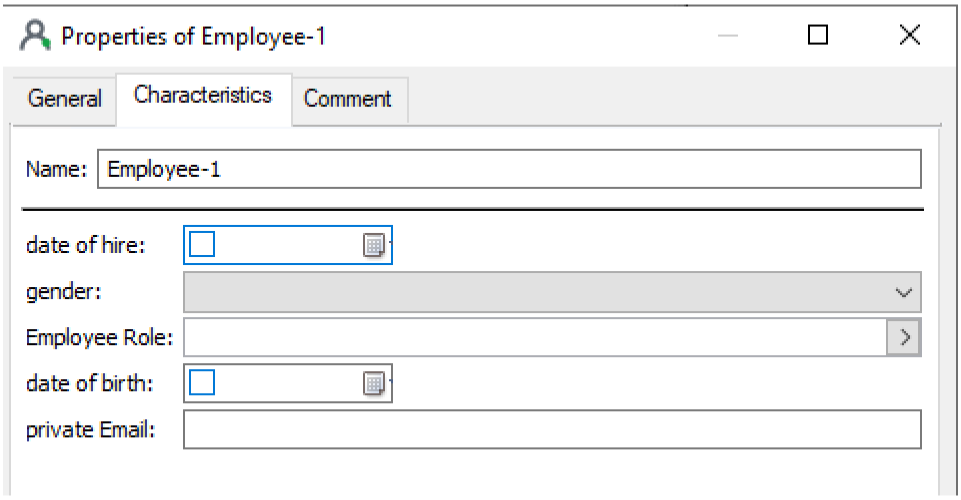
- '**Creation**' indicates that the element must appear in the object creation page
- '**Mandatory**' indicates that the element is required
- '**Immutable**' indicates that the element can only be modified at the time of creation
- '**Category**' indicates that the element defines a category for the object. This category can be used in the creation wizard, which can modify itself according to the category when the latter is initially specified: the name of the category and the associated image replace that of the MetaClass in the wizard.

Example: the creation wizard of the "**Employee**" MetaClass has not been customized. It is presented in the following form:



Name : Employee-1

In the properties page of an **Employee**, there is the "**Employee Role**" which we want to be able to specify at creation of the object.



Name: Employee-1

date of hire:

gender:

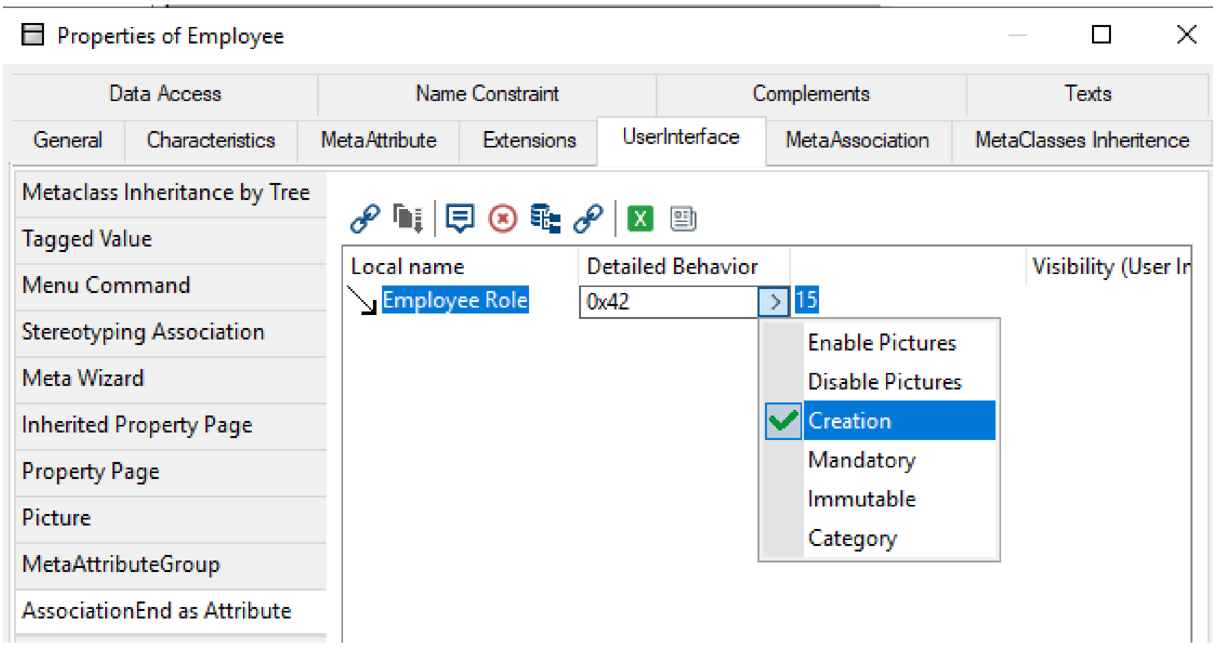
Employee Role:

date of birth:

private Email:

To do this:

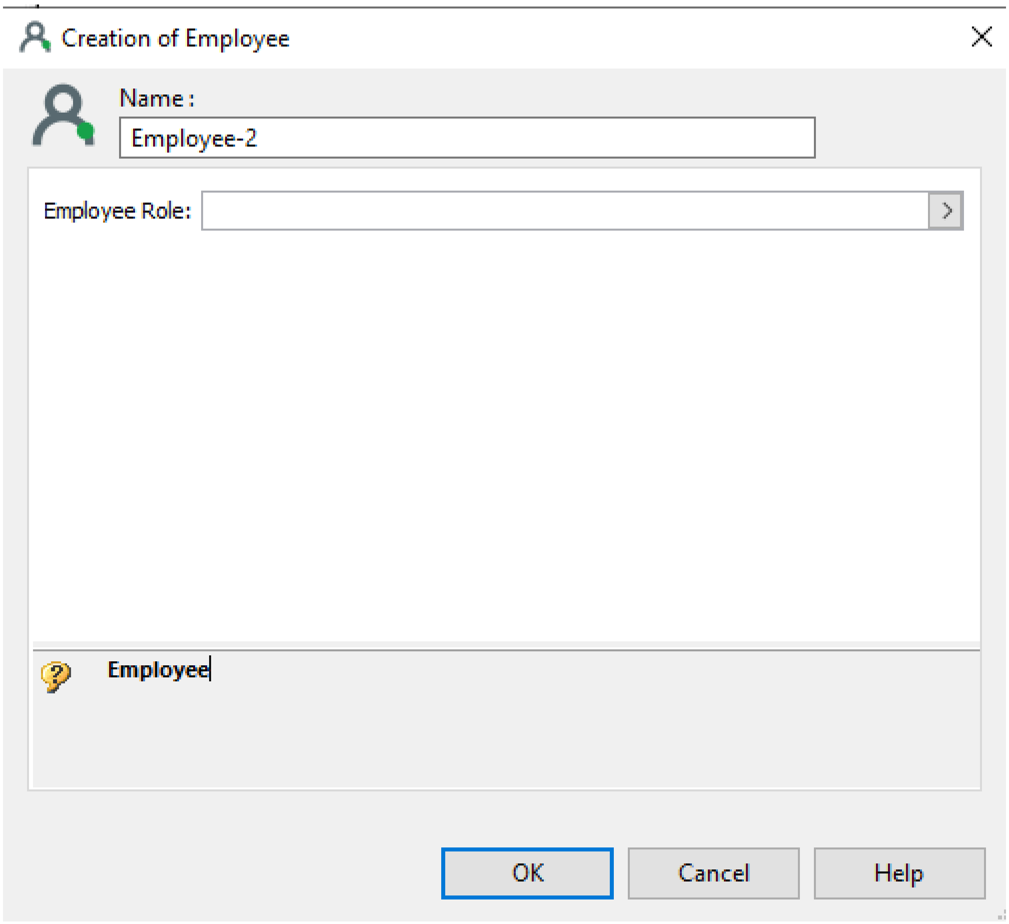
1. From the **Employee** MetaClass Properties page, in the **UserInterface** tab, select the **AssociationEnd as Attribute** subtab.
2. Update the '**Detailed Behavior**' attribute of the link _LegAttribute **Employee Role**.



Local name	Detailed Behavior	Visibility (User Ir
Employee Role	0x42 15	

✓ Creation

On completion of this configuration, the **Employee** MetaClass creation wizard becomes:



Creation of Employee

Name : Employee-2

Employee Role:

Employee

Adding processing code to a creation wizard

This chapter details the case of a configuration of the wizard not requiring page definition, but restricting itself to modifying behavior of the standard wizard by means of a trigger.

To do this, you shall consider the example of the **Employee** creation wizard, modifying its specification:

- The **Employee Role** role should not appear until the **Employee** is created from a **Manager**.
- The comment of the **Employee** should be initialized with the comment of the **Employee Role** when the latter is defined.

To implement the first requirement, it is not possible to configure the MetaClass, since this configuration is not conditional.

- Remove the "Creation" **Detailed Behavior** on the "Employee Role" role of the **Employee** MetaClass.

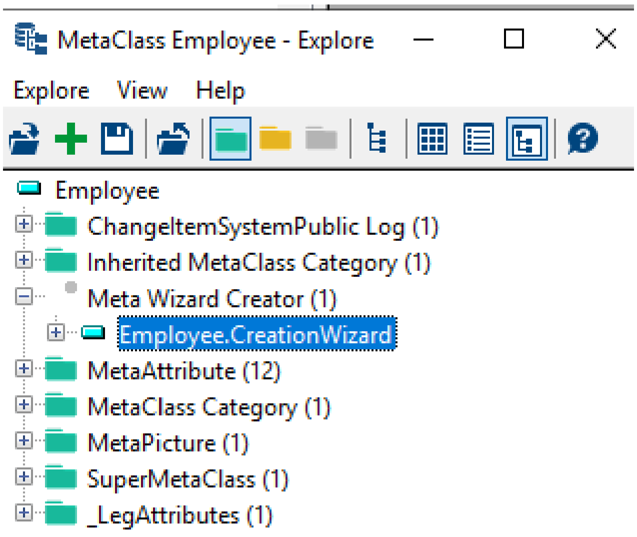
As for the second requirement, you must be able to insert code executed at object creation.

- To carry out this customization, you must implement a Meta Wizard.

Creating a wizard and its trigger

To create this Meta Wizard:

1. In the **MetaStudio** tab, expand the **Employee** MetaClass.
2. Right-click the **Meta Wizard Creator** folder and create a new Meta Wizard.
3. Name this Meta Wizard : Employee.CreationWizard.


MetaClass Employee - Explore

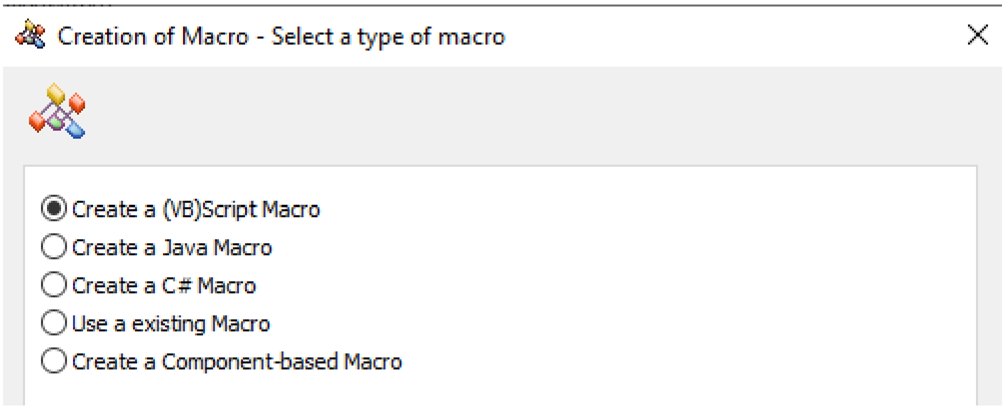
Explore View Help

- Employee
  - ChangeItemSystemPublic Log (1)
  - Inherited MetaClass Category (1)
  - Meta Wizard Creator (1)
    - Employee.CreationWizard
  - MetaAttribute (12)
  - MetaClass Category (1)
  - MetaPicture (1)
  - SuperMetaClass (1)
  - _LegAttributes (1)

To create a trigger for this wizard:

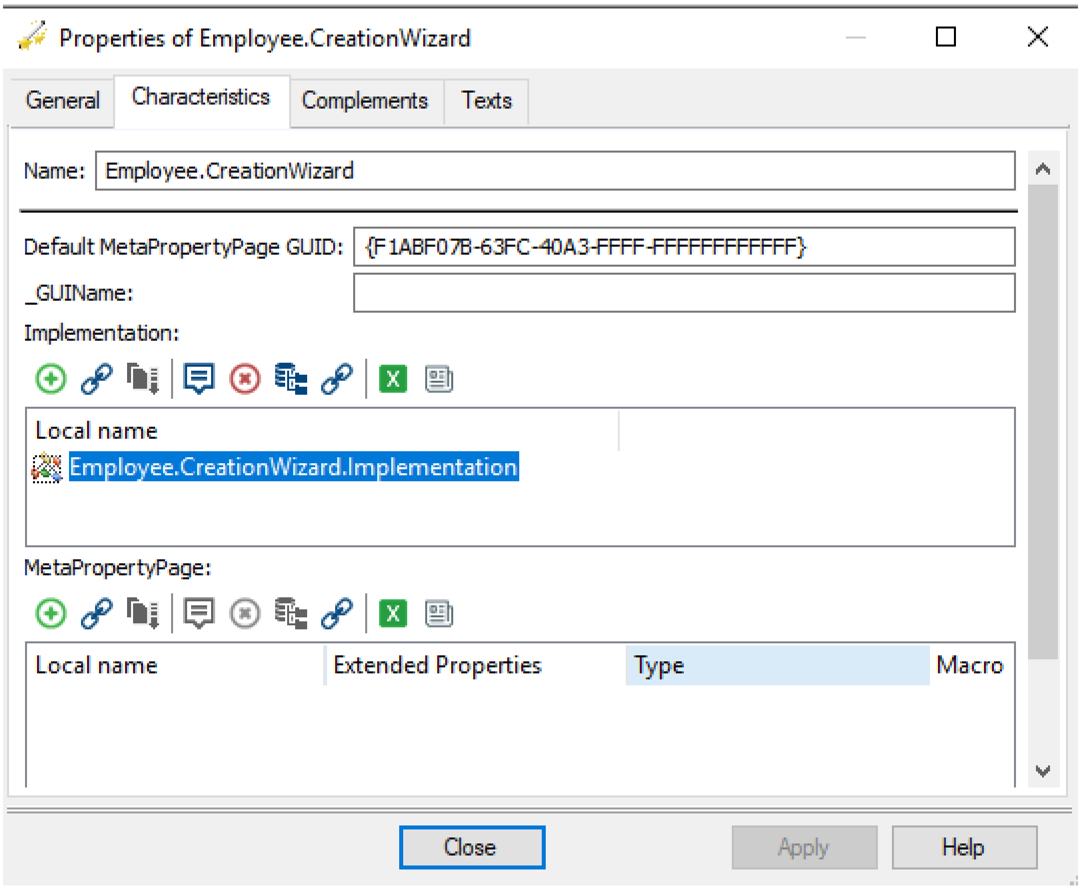
In this example, you can implement the trigger in JAVA or in (VB)Script.

1. From the **Meta Wizard Creator** folder, open the **Properties** of the "Employee.CreationWizard" Meta Wizard.
2. In **Characteristics** > **Implementation** frame, click **New**.
3. Select **Create a (VB)Script Macro** (or **Create a Java Macro**).


Creation of Macro - Select a type of macro

● Create a (VB)Script Macro
○ Create a Java Macro
○ Create a C# Macro
○ Use a existing Macro
○ Create a Component-based Macro

4. Click **Next**, then **Finish**.



Properties of Employee.CreationWizard

Name: Employee.CreationWizard

Default MetaPropertyPage GUID: {F1ABF07B-63FC-40A3-FFFF-FFFFFFFFFFFF}

_GUIName:

Implementation:

Local name
Employee.CreationWizard.Implementation

MetaPropertyPage:

Local name	Extended Properties	Type	Macro

5. Open the **Properties** of Employee.CreationWizard.Implementation.
6. In the **VB Script** tab, copy the code given in Dynamic addition of a property in the default page example p. 17.

Trigger interface description

A wizard trigger enables modification of wizard behavior. In principle, functions implemented in the trigger are called on each transition of the wizard. The list of functions which can be implemented is as follows:

```
void OnWizInitialize(MegaWizardContext mwctx);
void OnWizPageInitialize(MegaWizardContext mwctx,MegaPropertyPage mpage);
void WizPageCheck(MegaWizardContext mwctx, MegaPropertyPage mpage,Integer[]
result);
void OnWizNext(MegaWizardContext mwctx, MegaPropertyPage mFrom,
MegaPropertyPage mTo);
void OnWizPrevious(MegaWizardContext mwctx, MegaPropertyPage mFrom,
MegaPropertyPage mTo);
void OnWizRealize(MegaWizardContext mwctx);
void OnWizBeforeTerminate(MegaWizardContext mwctx, MegaPropertyPage
mpage,Integer[] result);
void OnWizTerminate(MegaWizardContext mwctx);
void OnWizCancel(MegaWizardContext mwctx);
void WizPageNextChange(MegaWizardContext mwctx, MegaPropertyPage
mpage,StringBuffer nextPageId);
void WizPagePreviousChange(MegaWizardContext mwctx, MegaPropertyPage
mpage,StringBuffer PreviousPageId);
```

Function **OnWizInitialize** is called at initialization of the wizard. It is called before creation of the user interface. It is in this function that you can:

- add or remove properties in the default page
- modify wizard display mode (for example force complete mode when called in "in place" mode.
- hide the name and/or the owner fields in the first page of the creation wizard

Function **OnWizPageInitialize** is called at initialization of a wizard page. It is called only once per page, when the wizard needs to collect information about it; it can thus be at any moment in the wizard lifetime, and not necessarily at the moment (but obviously before) the page is shown.

Function **WizPageCheck** is called at least each time the wizard prepares to display a next or previous page; it enables dynamic masking of a page and redefinition of the state of the previous, next and finish buttons. It can be called any number of times.

Functions **OnWizNext** and **OnWizPrevious** are called on each transition between two pages of the wizard.

Function **OnWizRealize** is called at object creation. We shall see later that this is not necessarily at the end of the wizard.

Function **OnWizBeforeTerminate** is called before the termination action; it enables a final check before the code terminating the wizard. It is particularly useful on wizards which have only one page, and therefore no transition.

Function **OnWizTerminate** is called when pressing the finish button (or OK if only one page is displayed).

Function **OnWizCancel** is called at cancellation of the wizard (Cancel button or explicit cancel request during processing). In this case, any HOPEX updates carried out during wizard processing are automatically cancelled.

It is possible to force display of a next or previous page using functions **WizPageNextChange** and **WizPagePreviousChange**.

Wizard default page

Note that for the moment, no specific page has been defined for the wizard we are currently creating. A wizard in this state will continue to use the default page, a page defined for non-customized wizards. It should be noted that this page will remain visible in the wizard as long as the wizard considers it useful, or as long as customization does not explicitly mask it. By default, this page contains, in addition to the name and the owner, the properties which have been declared visible at object creation.

If you want to hide the name and/or the owner fields, add:

Java:

```
public void OnWizInitialize(MegaWizardContext wctx) {
    wctx.propertyFlag("Name", "Visible", false);
    wctx.propertyFlag("Name", "Hidden", true);
    wctx.propertyFlag("~WI(ov(ir7jK0[Owner Container]","Hidden", true);
 }
```

VB(Script):

```
Sub OnWizInitialize(wctx As MegaWizardContext)
  wctx.Property("Name", "Visible") = False
  wctx.Property("Name", "Hidden") = True
  wctx.property("~WI(ov(ir7jK0[Owner Container]","Hidden")=True
End Sub
```

Wizard context: MegaWizardContext interface

Customizations mentioned in our example require knowing the creation wizard call context, since it is specified that behavior differs depending on whether the **Employee** is created from a **Manager** or not.

All information relating to context of the wizard and its state are accessible in the MegaWizardContext object, which is transmitted to each of the trigger functions.

This interface is quite complex. We will cover its different aspects in the course of this document. Note however the presence of functions enabling access to the following properties:

mode

Determines wizard start mode. There are at present 3 main wizard start modes.

- mode(1): complete mode, as used particularly in navigators
- mode(0): 'inPlace' mode. This mode is used in ListViews on the basis that it is not necessary to enter the name of the object to be created, either because this name is already fixed and we do not consider it necessary to offer the possibility of modification, or because it will be modifiable later (in the listview, the name of the newly-created object is modifiable 'inPlace' in the list). It is of course possible to customize the wizard by forcing complete mode.
- mode(2): we wish to call the creation wizard from a non-interactive tool. This mode is required when we consider that only the wizard is capable of creating a 'valid' object, and we wish to ensure that the code executed in the triggers and used to initialize the object will be called. In no event can this mode present a user interface, and creation will fail if all prerequisites of creation are not satisfied; if for example a property is declared as mandatory, and if its value is not known or cannot be deduced in the wizard, the wizard will fail.

It is possible to modify the mode in function **OnWizInitialize**. This is not possible later, the wizard having already started.

Name

Contains the name we wish to give the object during creation. It can be modified in the trigger code.

parentTypeID and parentID

When the object is created from another object (create-connect) these properties indicate the origin object of creation. *ParentID* contains the absolute identifier of this origin object, and *parentTypeID* that of the MetaAssociationEnd (viewed from source) via which we wish to connect the object after its creation.

sourceID and targetID

When the object is created from a link action in a diagram, these two properties contain the source and target absolute identifiers of the action.

kindID

When category of an object is already known, this property contains its absolute identifier. When an image has been associated with the category, this image replaces that of the MetaClass in the wizard; similarly, the name of the category replaces the name of the MetaClass.

template

Represents the object being created, in the form of a MegaObject. In most cases, we can use this as an existing object, but a certain number of functionalities are not accessible to such an object due to its non-existence. In particular, virtual attributes and associations are not suitably managed by this object. Its absolute identifier can however be modified.

property and propertyFlag

These functions enable definition and consultation of properties associated with the context of the wizard. These properties are addressed by an absolute identifier.

These properties represent either:

- Properties of the object being created: this case is automatically detected when the identifier provided corresponds to the absolute identifier of a property (MetaAttribute, TaggedValue or LegAttribute) of the object. In this case the property of the context enables fixing of the initial value of this property (*property*), as well as its display mode in the wizard (*propertyFlag*. See details of these *flags*) below.
- Data specific to the wizard, which we shall call 'Cookies', use of which will be covered later in the document.

Dynamic addition of a property in the default page

Here we shall consider the first customization, which consists of adding the '**Employee Role**' role in the creation page when the **Employee** is created from a **Manager**.

This customization must be implemented in the **OnWizInitialize** function; it is the only function which is always called before creation of the default page, and therefore enables modification of the content of this page.

The following is an example of Java code implementing this customization:

```
import com.mega.modeling.api.*;
import com.mega.modeling.api.util.MegaWizardContext;

public class EmployeeWizardTrigger {

  public void OnWizInitialize(MegaWizardContext wctx) {
    MegaToolkit mToolkit = wctx.template().getRoot().currentEnvironment().toolkit();
    if(mToolkit.sameID(wctx.parentTypeID(),"~demSW1OYEDCC[Employee]")) {
      wctx.propertyFlag("~y4mZK)OYErIH[Employee Role]", "Visible", true);
      wctx.propertyFlag("~y4mZK)OYErIH[Employee Role]", "Updatable", true);
    }
  }
}
```

In (VB)Script this example is as follows:

```
'MegaContext(Fields,Types)
'Uses(Components)
Option Explicit

Sub OnWizInitialize(wctx As MegaWizardContext)
 Dim mToolkit As MegaToolkit
 Set mToolkit = wctx.template.getRoot.currentEnvironment.Toolkit
 If mToolkit.sameID(wctx.parentTypeID, "~demSW1OYEDCC[Employee]") Then
   wctx.property("~y4mZK)OYErIH[Employee Role]", "Visible") = True
   wctx.property("~y4mZK)OYErIH[Employee Role]", "Updatable") = True
  End If
End Sub
```

We first determine whether the object is created as a **Manager** by comparing the type of the parent with the MetaAssociationEnd concerned (to do this, we use the sameID method available in the HOPEX toolkit).

In this case, we inform the wizard that the property concerned (here the '**Employee Role**' role) is visible and can be entered. The propertyFlag function previously evoked enables fixing of flags "Visible" and "Updatable" and therefore make the property visible in the wizard.

The available flags are:

Visible = True makes property visible. It is read-only by default. Visible = False does not mask the property, but only indicates that we wish standard behavior on this property.

Updatable = True authorizes modification of the property (but does not make it visible).

Hidden = True masks the property.

Mandatory = True makes property mandatory.

In Script, the propertyFlag function does not exist; we use the property function with two parameters to access the flag, the second being the name of the flag.

Adding a processing in the wizard

The second customization in the proposed example consists of adding a processing on a newly-created object: the comment of the **Employee** must be initialized with the comment of the **Employee Role** when the latter is defined.

A processing of this type can be inserted in any transition; however in the case of a wizard without a specific page, the only possible transitions are in the function:

- OnWizRealize, immediately after object creation,
- OnWizTerminate at the end of execution of the wizard.

The proposed initialization must take place as late as possible, so as to allow the wizard to specify an **Employee Role** or a comment for the **Employee**.

- it is therefore preferable to include the processing in the final function, that is in **OnWizTerminate**.

The following is an example of code implementing this customization:

In Java:

```
public void OnWizTerminate(MegaWizardContext wctx) {
  MegaObject mEmployee = wctx.template();
  if(mEmployee.getProp("~f10000000b20[Comment]").compareTo("") == 0) {
   if(mEmployee.getCollection("~y4mZK)OYErIH[Employee Role]").count() > 0) {
    MegaObject mRole = mEmployee.getCollection("~y4mZK)OYErIH[Employee Role]").item(1);
    String sComment = mRole.getProp("~f10000000b20[Comment]");
    mEmployee.setProp("~f10000000b20[Comment]", sComment);
   }
  }
```

```
}

En (VB)Script:
      Sub OnWizTerminate(wctx As MegaWizardContext)
        Dim mEmployee As MegaObject
   Set mEmployee = wctx.template
  If mEmployee.getProp("~f10000000b20[Comment]") = "" Then
   If mEmployee.getCollection("~y4mZK)OYErIH[Employee Role]").count > 0 Then
     Dim mRole As MegaObject
         Set     mRole     =     mEmployee.getCollection("~y4mZK)OYErIH[Employee
Role]").item(1)
     Dim sComment As String
     sComment = mRole.getProp("~f10000000b20[Comment]")
     mEmployee.setProp "~f10000000b20[Comment]", sComment
 End If
 End If
End Sub
```

The object in course of creation, which is necessarily created at the OnWizTerminate transition is always accessible via the *template* function. The above code updates the comment from that of the **Employee role**. Note here that update only occurs if the object comment is empty. This is not insignificant:

You should try to ensure that a customization disturbs later customizations of the wizard as little as possible. Suppose in the present case that after this customization or at least independently, we decide to make the comment of an **Employee** visible in the creation wizard by modifying the 'Detailed Behavior' attribute. In this case, the comment of the object appears on the creation page of the wizard, and the creator may be led to initialize it. If the customization above did not take account of the fact that the Comment might already be initialized and overwrites the user entry, behavior of the wizard would be difficult to understand.

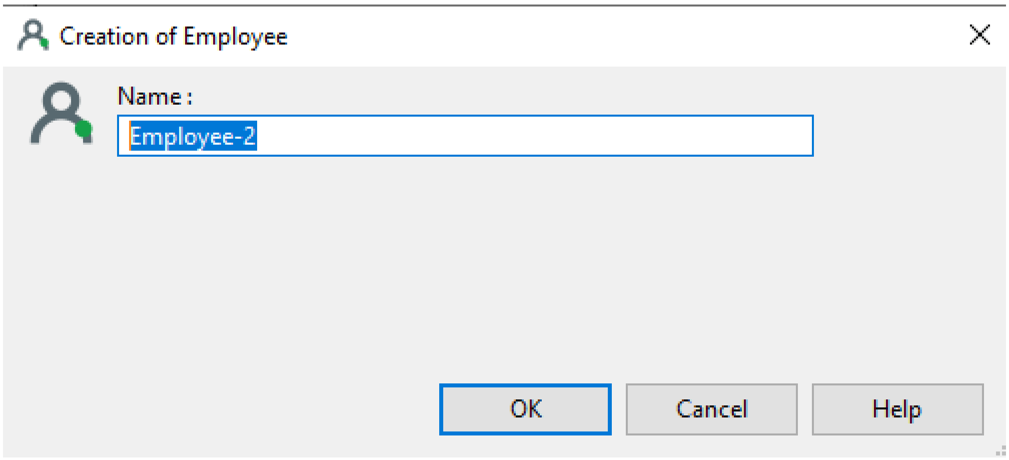
Testing the wizard

To test the wizard, you can:

- Create an **Employee** from the explorer
- Create a **Manager** from the explorer
- Create an **Employee** from a **Manager**
- Connect an **Employee**, already created, to a **Manager**.

To create an **Employee**:

1. From the desktop, run the explorer.
2. Select **Explore > Create**.
3. In the **Choose MetaClass** dialog box, select **Employee**.



Name : Employee-2

4. Click **OK**.

   Employee-2 is created.

   Note that **Employee Role** information does not appear.

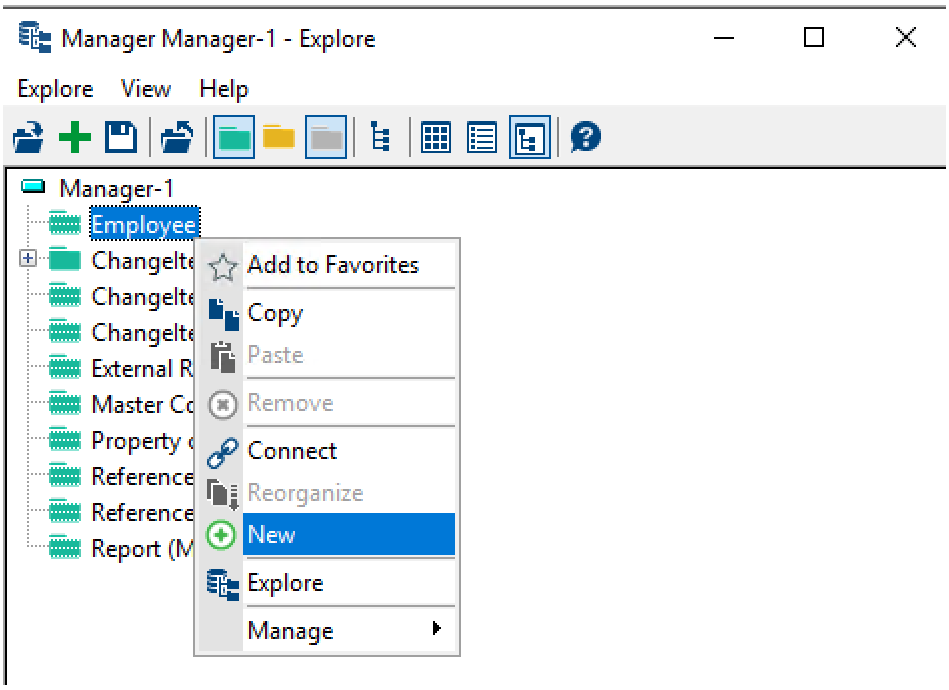
To create a **Manager**:

1. From the desktop, run the explorer.
2. Select **Explore > Create**.
3. In the **Choose MetaClass** dialog box, select **Manager**.
4. Click **OK**.

   Manager-1 is created. Note that conforming to the metamodel diagram, the **Employee** MetaClass is under "Manager-1".

To create an **Employee** from "Manager-1":

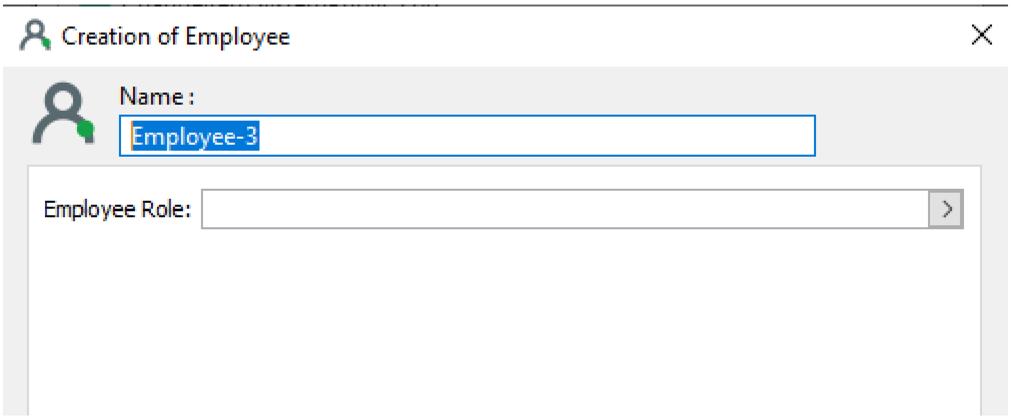
1. From the explorer of "Manager-1", right-click **Employee** and select **New**.



2. In the **Creation of Employee** dialog box, click **OK**.

   "Employee-3" is created.

   Note this time that the **Employee Role** field appears in the creation page.



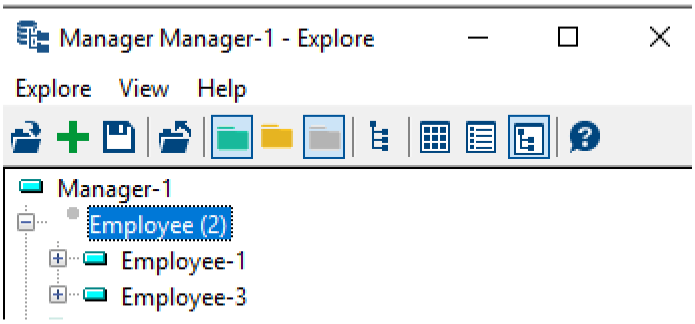
Name : Employee-3

Employee Role:

To connect an **Employee** already created, from "Manager-1":

1. From the explorer of "Manager-1", right-click **Employee** and select **Connect**.
2. In the query tool, select **Employee-1** and click **Connect**.

   "Employee-1" is connected to "Manager-1".



Adding independent triggers

You can associate several independent triggers with a wizard. The order of the link between the *MetaWizard* and the associated macros determines trigger call order.

Defining new pages in a creation wizard

Another type of creation wizard customization has been envisaged. It consists of adding steps in the creation process, in the form of pages.
In a wizard of this type, the 'OK' button is replaced by 'Previous', 'Next' and 'Finish' buttons, enabling navigation between pages defined in this way.
You can define wizards which have several steps without implementing a trigger, the simple logic of succession of pages defined in the wizard being sufficient to achieve the customization objective.

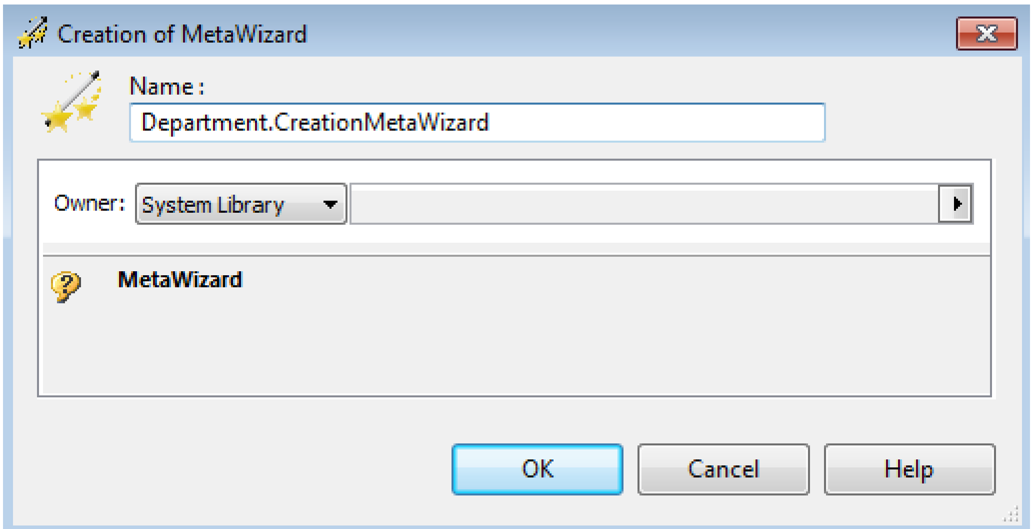
This system also enables presence in the creation wizard of elements that are not properties of the object to be created.

Here you will implement the following customization:

> "The list of **Abstract Employees** (comprising **Employees** and **Managers**) associated with a **Department** must be accessible in the creation wizard".

To do this:

1. Create the creation wizard of the **Department** MetaClass.



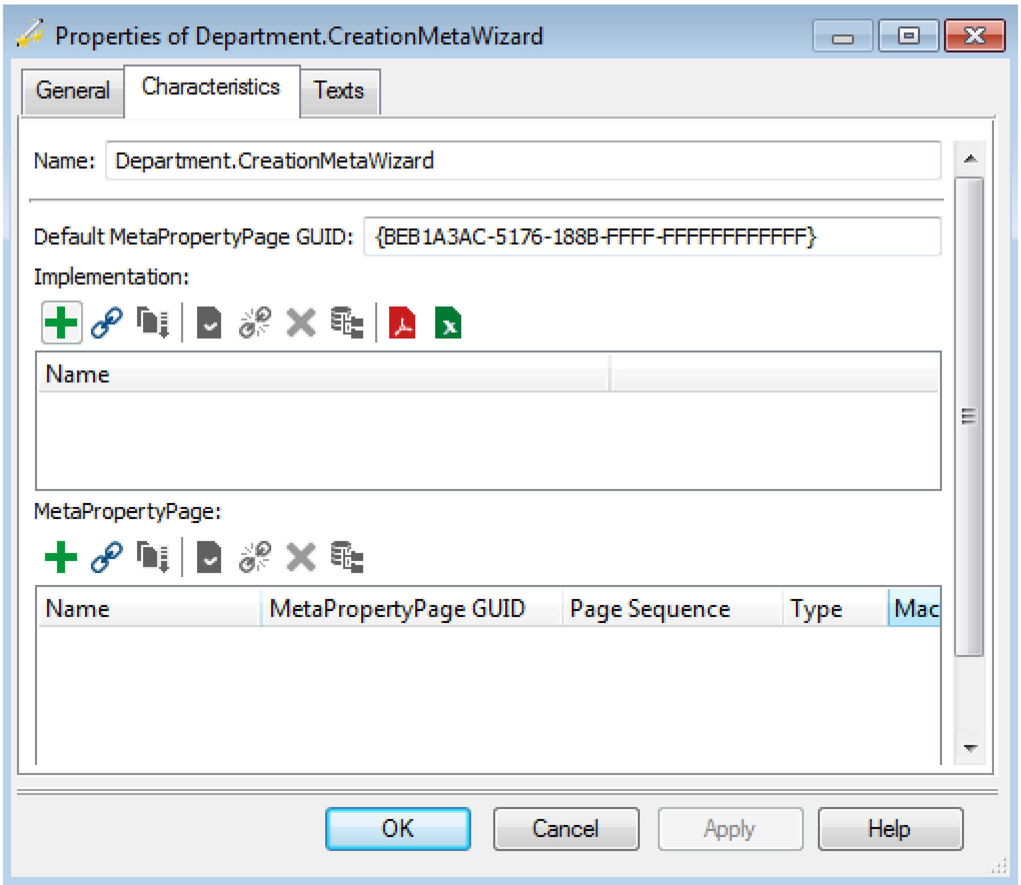
Creation of MetaWizard

Name: Department.CreationMetaWizard

Owner: System Library

MetaWizard

2. Open the properties page of the MetaWizard you have just created.


Properties of Department.CreationMetaWizard

Name: Department.CreationMetaWizard

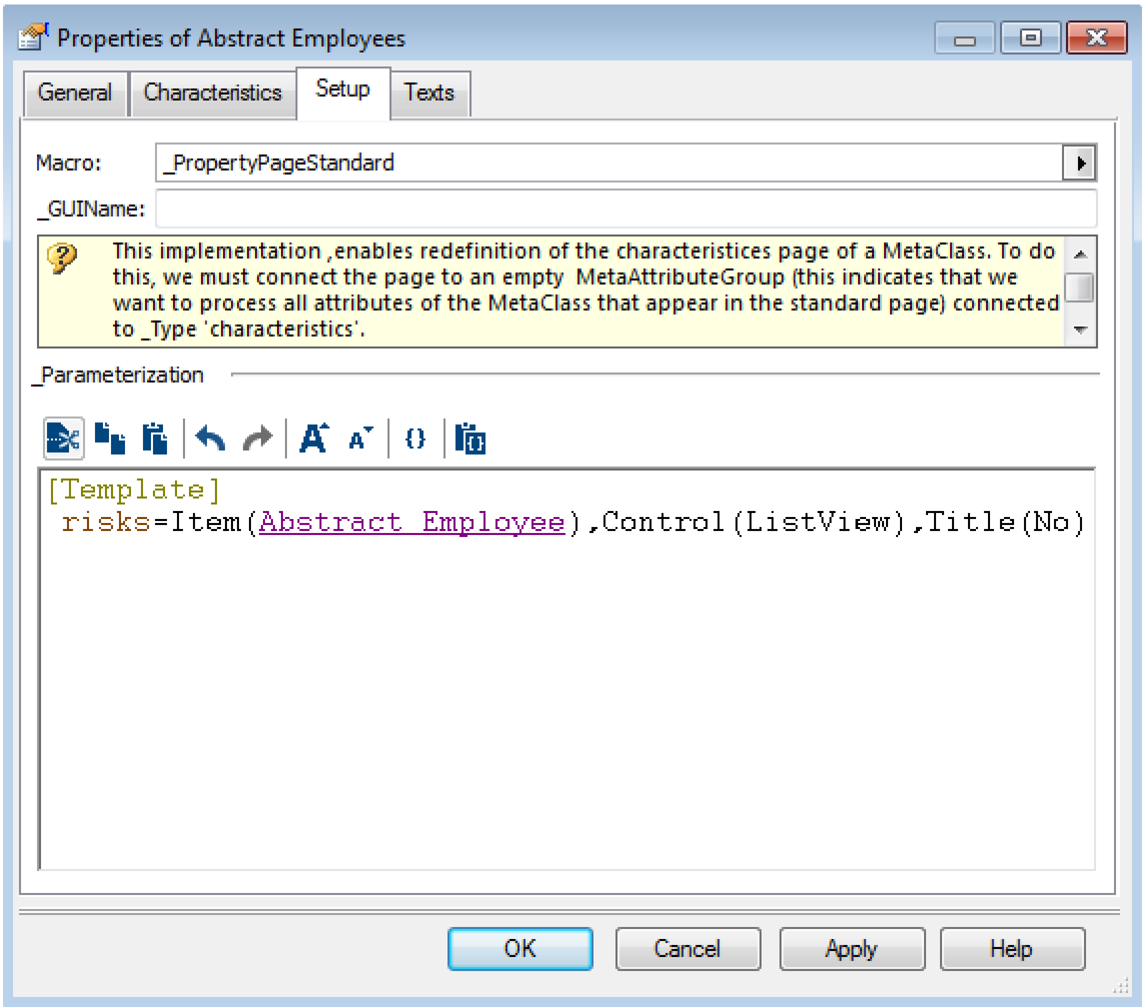
Default MetaPropertyPage GUID: {BEB1A3AC-5176-188B-FFFF-FFFFFFFFFFFF}

The pages you can integrate in a wizard are similar to object properties pages, and are modeled by **MetaPropertyPage**.

In the above wizard, you will create a page "Abstract Employees", display its properties dialog box and define its configuration:

1. In the **MetaPropertyPage** frame, click **Create**.
2. Name the page "Abstract Employees".
3. In the **Macro** field, click the PropertyPage.Kind menu and select macro **_PropertyPageStandard**.
4. Click **OK** to take this macro into account.
5. To insert the list of Abstract Employees, in the same way as for a properties page (for more information see the document on properties pages configuration), in the « Abstract Employees » Properties window select the **Setup** tab and in the **_Parameterization** frame, enter:

```
[Template]
risks=Item(~i5mZlVQYEHoI[Abstract Employee]),Control(ListView),Title(No)
```



Macro: _PropertyPageStandard

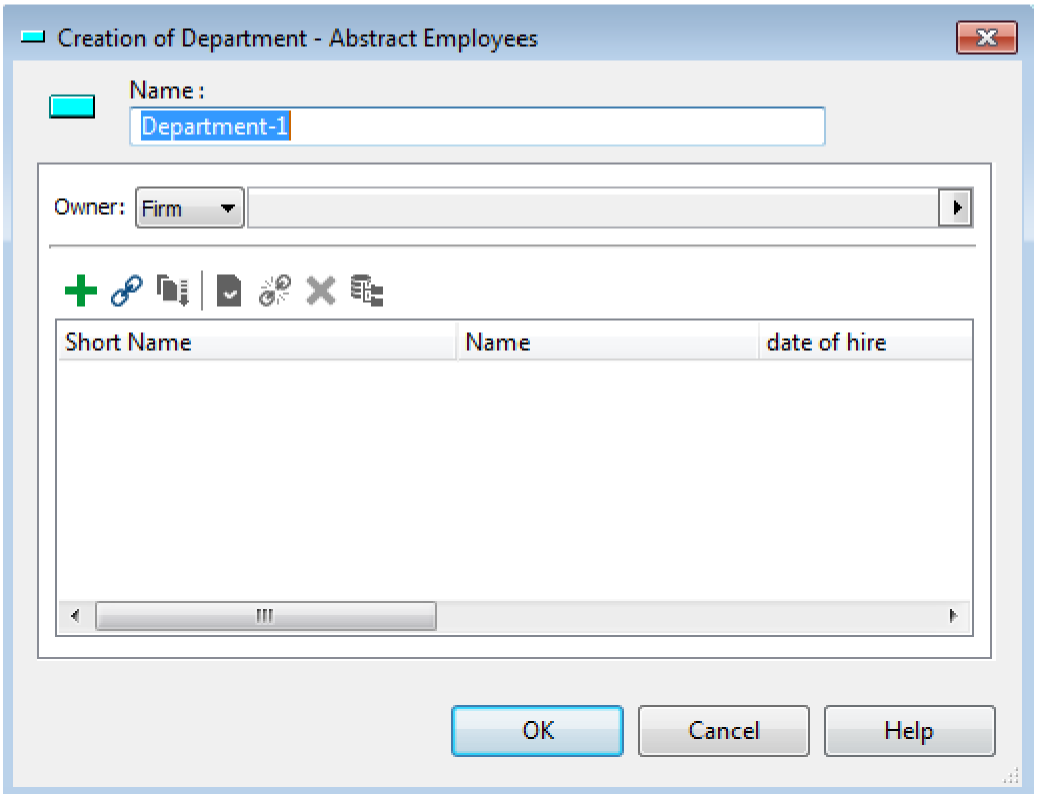
_GUIName:

This implementation ,enables redefinition of the characteristics page of a MetaClass. To do this, we must connect the page to an empty MetaAttributeGroup (this indicates that we want to process all attributes of the MetaClass that appear in the standard page) connected to _Type 'characteristics'.

_Parameterization

```
[Template]
 risks=Item(Abstract Employee),Control(ListView),Title(No)
```

6. Click **Apply**.
7. Start the creation wizard.



The wizard generates only a single page: each non-preparatory page includes name entry; from this the wizard deduces that this page displays all data necessary for execution.

The implementation '_PropertyPageStandard' corresponds to implementation of the default page, and therefore displays entry of the potential owner of the object.

The only significant difference compared with the default page: the help area is not present. This is an unfortunate anomaly which will be corrected…

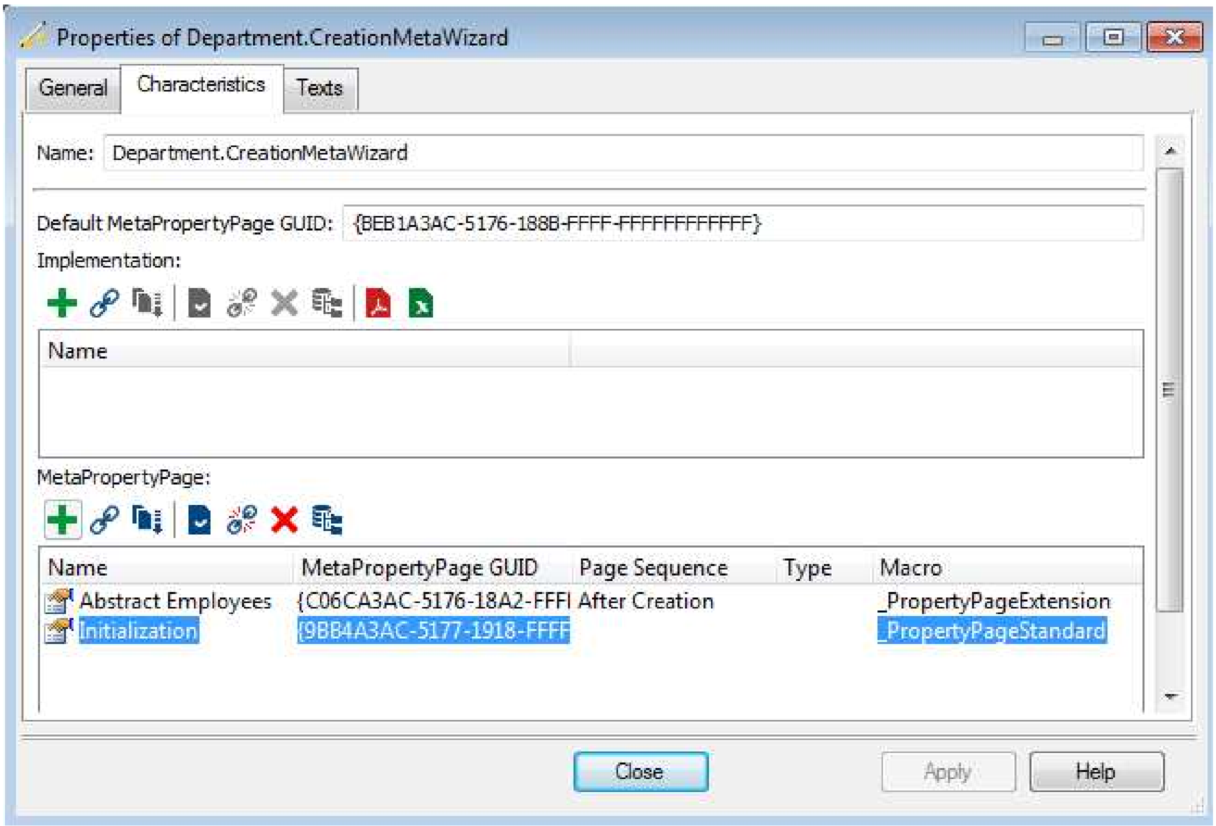
The result does not perhaps conform to what we had hoped, and to achieve this we shall specify our customization request. The wizard should:

1. Present a page displaying the name (and the owner).
2. Create the occurrence of a Department.
3. Propose the "Abstract Employees" entry page, without the owner.

To do this:

1. Create a new page "Initialization" in the wizard.
2. In the **Macro** field, select "_PropertyPageStandard" to propose the entry of the owner.
3. For the "Abstract Employees" page, in the **Page Sequence** field, select "After Creation" and in the **Macro** field, select "_PropertyPageExtension" so that owner entry is not proposed.

The result should be as follows:



Properties of Department.CreationMetaWizard

Name: Department.CreationMetaWizard

Default MetaPropertyPage GUID: {BEB1A3AC-5176-188B-FFFF-FFFFFFFFFFFF}

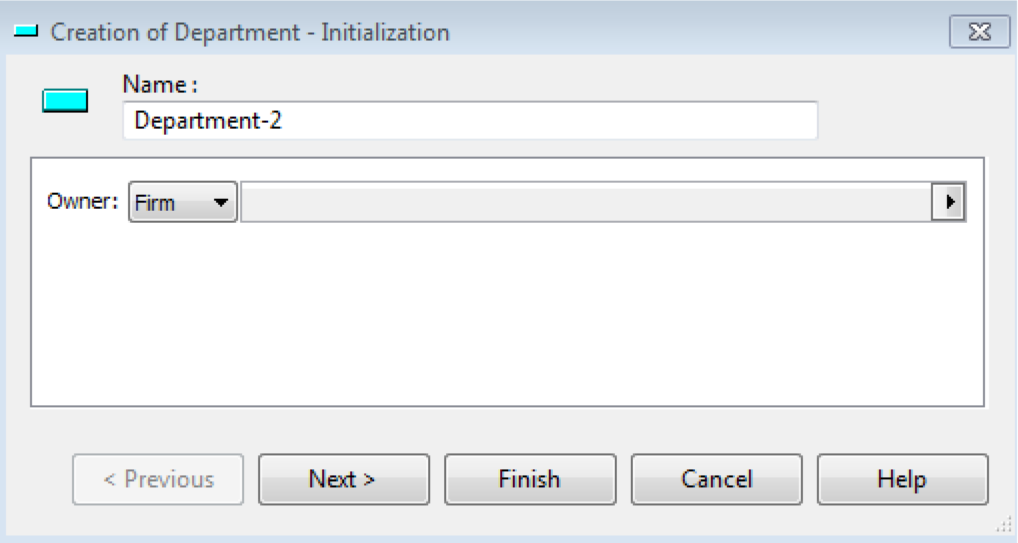
Implementation:

Name	

MetaPropertyPage:

Name	MetaPropertyPage GUID	Page Sequence	Type	Macro
Abstract Employees	{C06CA3AC-5176-18A2-FFFI	After Creation		_PropertyPageExtension
Initialization	{9BB4A3AC-5177-1918-FFFF			_PropertyPageStandard

4. Start the the wizard: from the explorer, create a **Department**.



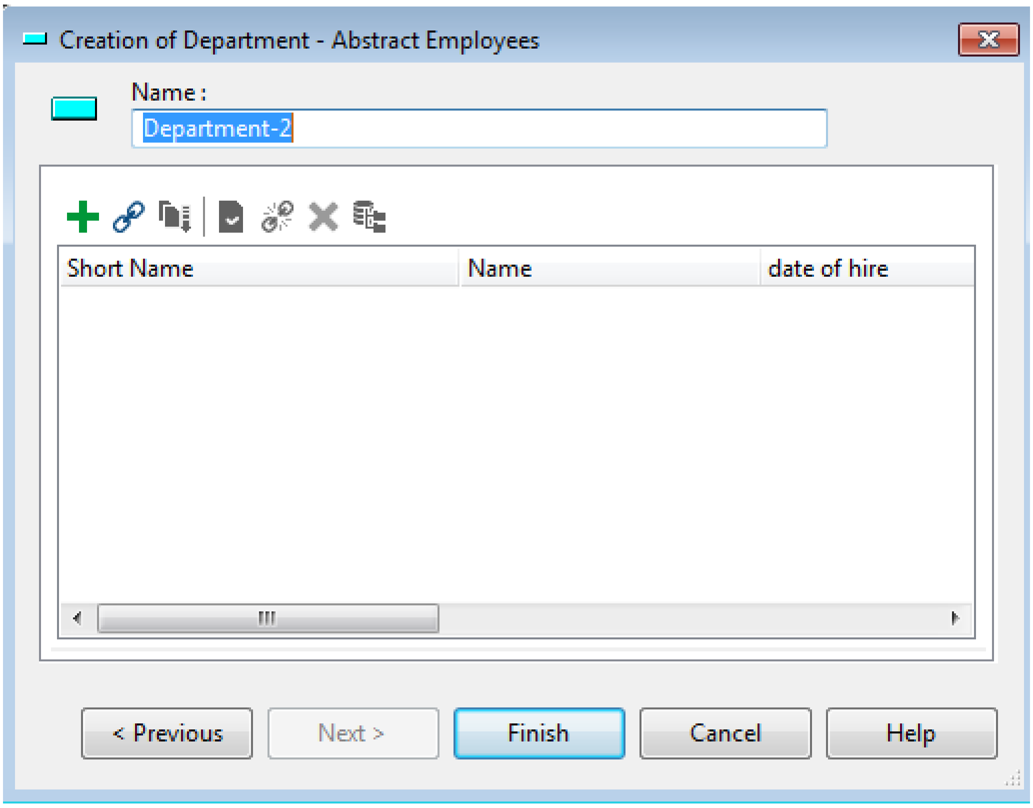
Creation of Department - Initialization

Name : Department-2

Owner: Firm

Note that it now presents two pages (the **Next** button is active).

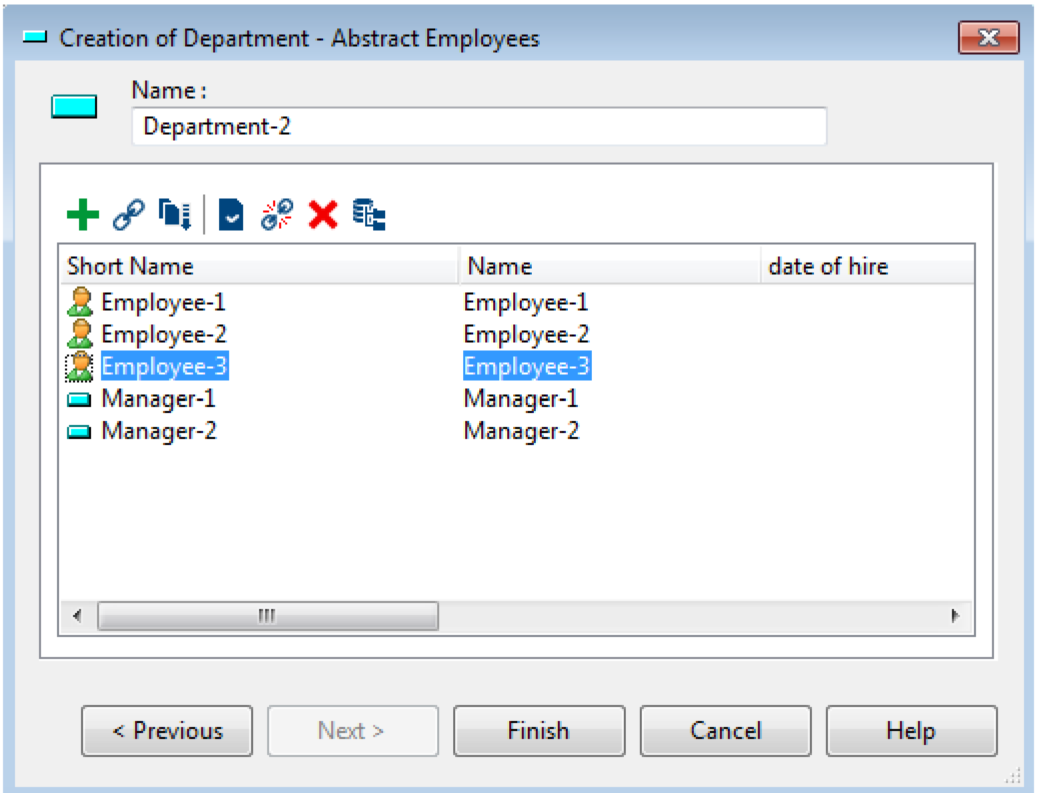
5. Click **Next.**



Name : Department-2

Short Name	Name	date of hire

6. Click **New** or **Connect** to add Employees and/or Managers to “Department-2”.



Name : Department-2

Short Name	Name	date of hire
Employee-1	Employee-1	
Employee-2	Employee-2	
Employee-3	Employee-3	
Manager-1	Manager-1	
Manager-2	Manager-2	

- "Department-2" is created in the transition between the first and second pages.
- Pages appear automatically in the correct order, although you did not order them in the list. The "Abstract Employees" page defined as "After Creation" appears after the "Initialization" page.

Two pages appearing in the same sequence must necessarily be ordered.

This result can be obtained more simply, by configuring the wizard itself, see Calling a wizard by specifying a context p. 30.

Calling a wizard by specifying a context

The context of the wizard, mentioned above, enables extremely powerful external configuration of a wizard. This enables:

- modification of appearance and behavior of a wizard without modifying the wizard.
- design of specific context modes enabling wizards to adapt to use cases.

To do this, we shall consider the functionality of configuring and starting a wizard.

Calling a wizard - configuring properties of the object to be created

Calling a wizard is by means of the *instanceCreator* function accessible from a *MegaCollection*.

The *MegaCollection* used should enable creation of a new object within itself; it therefore consists of collections based on a *MetaClass* or on a *MetaAssociationEnd* seen from a *MetaClass*:

- in the first case, the wizard creates an isolated object,
- in the second, the wizard creates the object seen from the cited association.

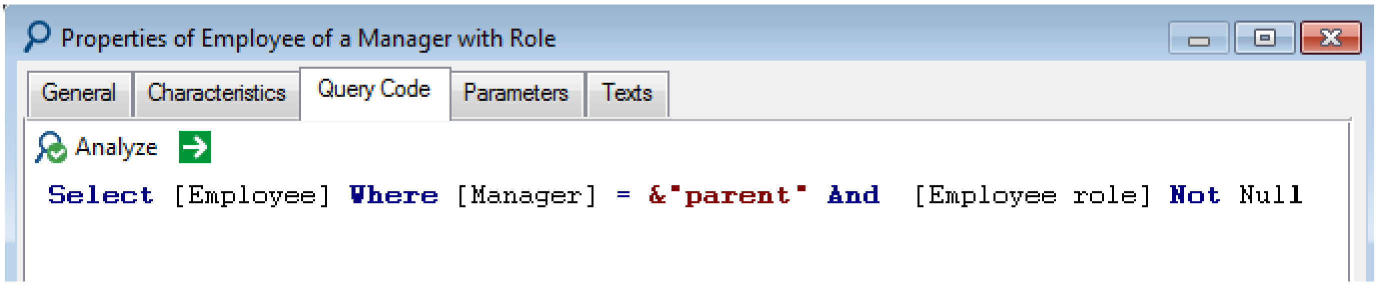
The *instanceCreator* function makes available a *MegaInstanceCreator* component which enables more specific configuration of the wizard and its starting.

The MegaInstanceCreator component implements the functions of MegaWizardContext, and therefore authorizes configuration of the wizard. In addition it implements the *create* function, which enables starting of the wizard.

Consider again the example of creation of a **Manager**, now with the following objective:

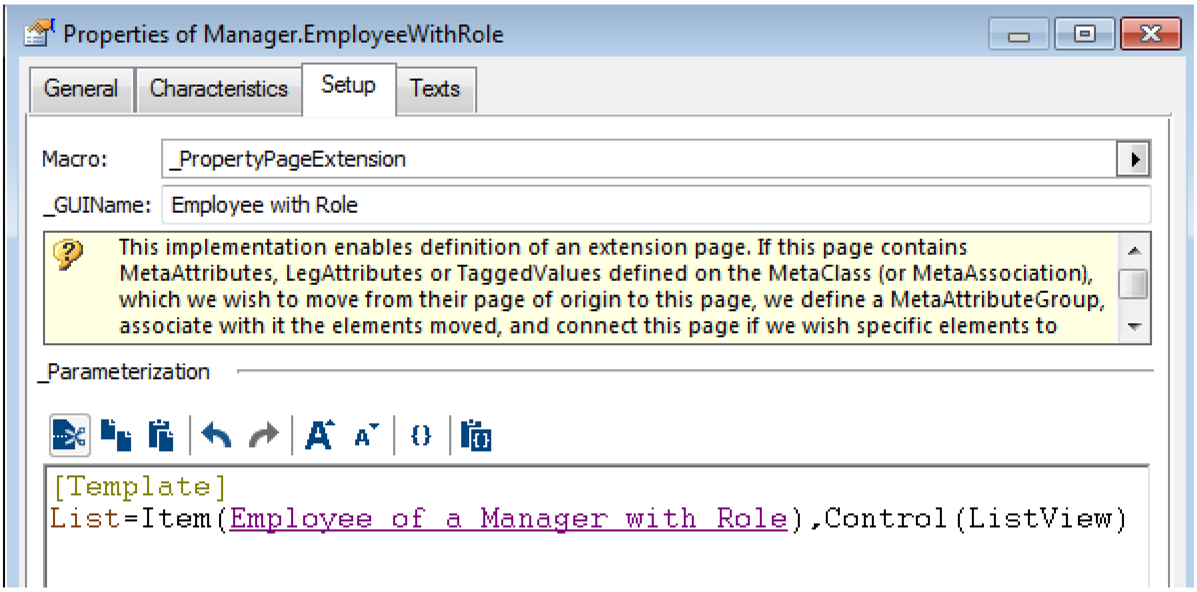
1. Create a query "Employee of a Manager with Role" defined on the **Manager** MetaClass , listing all **Employees** for which an **Employee Role** has been defined.

« **Select** [Employee] **Where** [Manager] **=** **&"parent"** **And** [Employee role] **Not** **Null** »



```
Select [Employee] Where [Manager] = &"parent" And  [Employee role] Not Null
```

2. From the **Manager** MetaClass, create the PropertyPage "Manager.EmployeeWithRole".



Macro: _PropertyPageExtension

_GUIName: Employee with Role

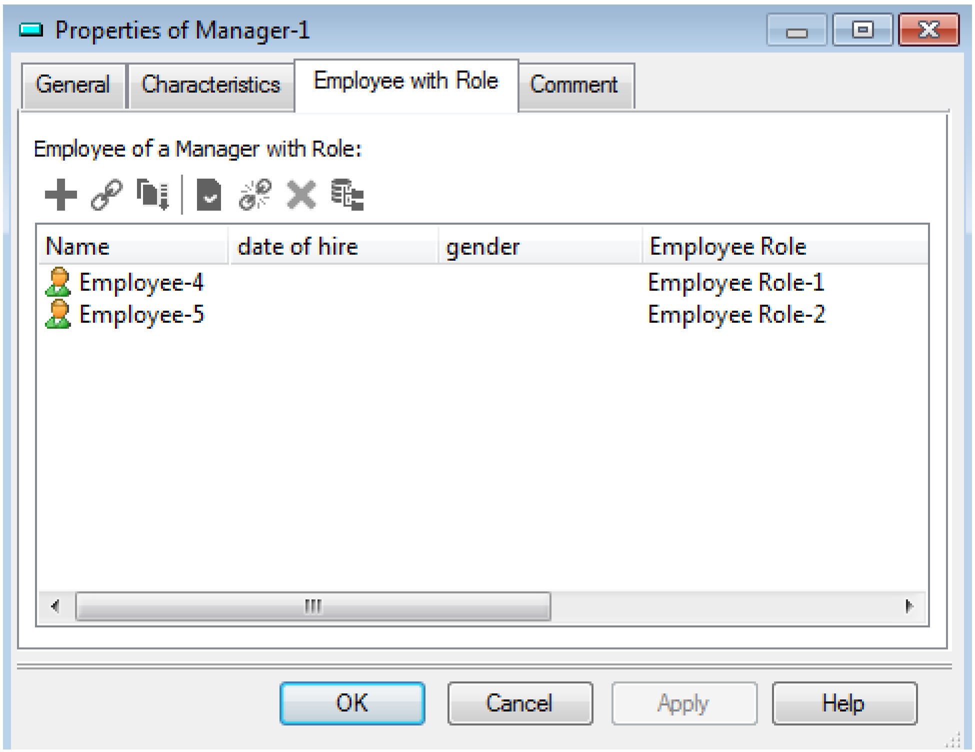
This implementation enables definition of an extension page. If this page contains MetaAttributes, LegAttributes or TaggedValues defined on the MetaClass (or MetaAssociation), which we wish to move from their page of origin to this page, we define a MetaAttributeGroup, associate with it the elements moved, and connect this page if we wish specific elements to

_Parameterization

```
[Template]
List=Item(Employee of a Manager with Role),Control(ListView)
```

From the properties pages of a Manager, in the "Employee with Role" tab, note that:

- only the **Employees** for which an **Employee Role** has been defined are listed.
- from this tab, you cannot create an Employee.



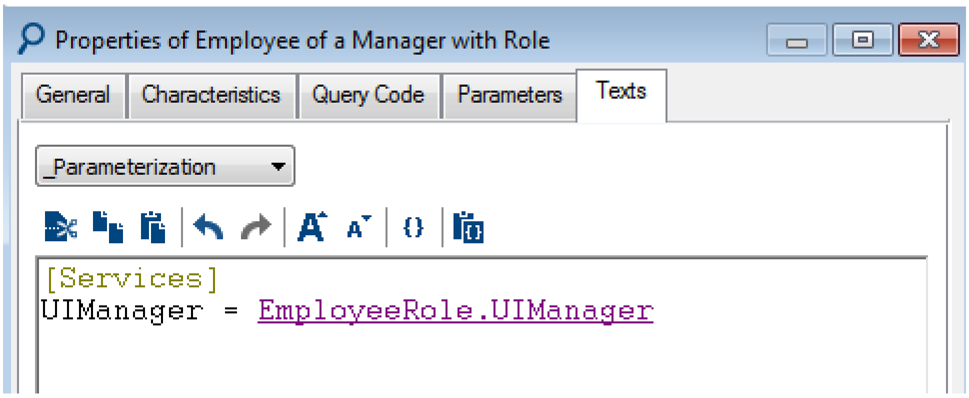
Employee of a Manager with Role:

Name	date of hire	gender	Employee Role
Employee-4			Employee Role-1
Employee-5			Employee Role-2

- You want to be able to create an **Employee** conforming to this query, by implementing a user interface manager implementing the 'create' method accessible from the query, when for example it is presented in a tree or in a list.

3. The user interface manager is configured in the **Texts** *_Parameterization* tab of the "Employee of a Manager with Role" query in the following form:

```
[Services]
UIManager = EmployeeRole.UIManager
```



```
[Services]
UIManager = EmployeeRole.UIManager
```

Where EmployeeRole.UIManager is the Macro – here in VBScript – implementing the DoCreation function which calls the required creator wizard.

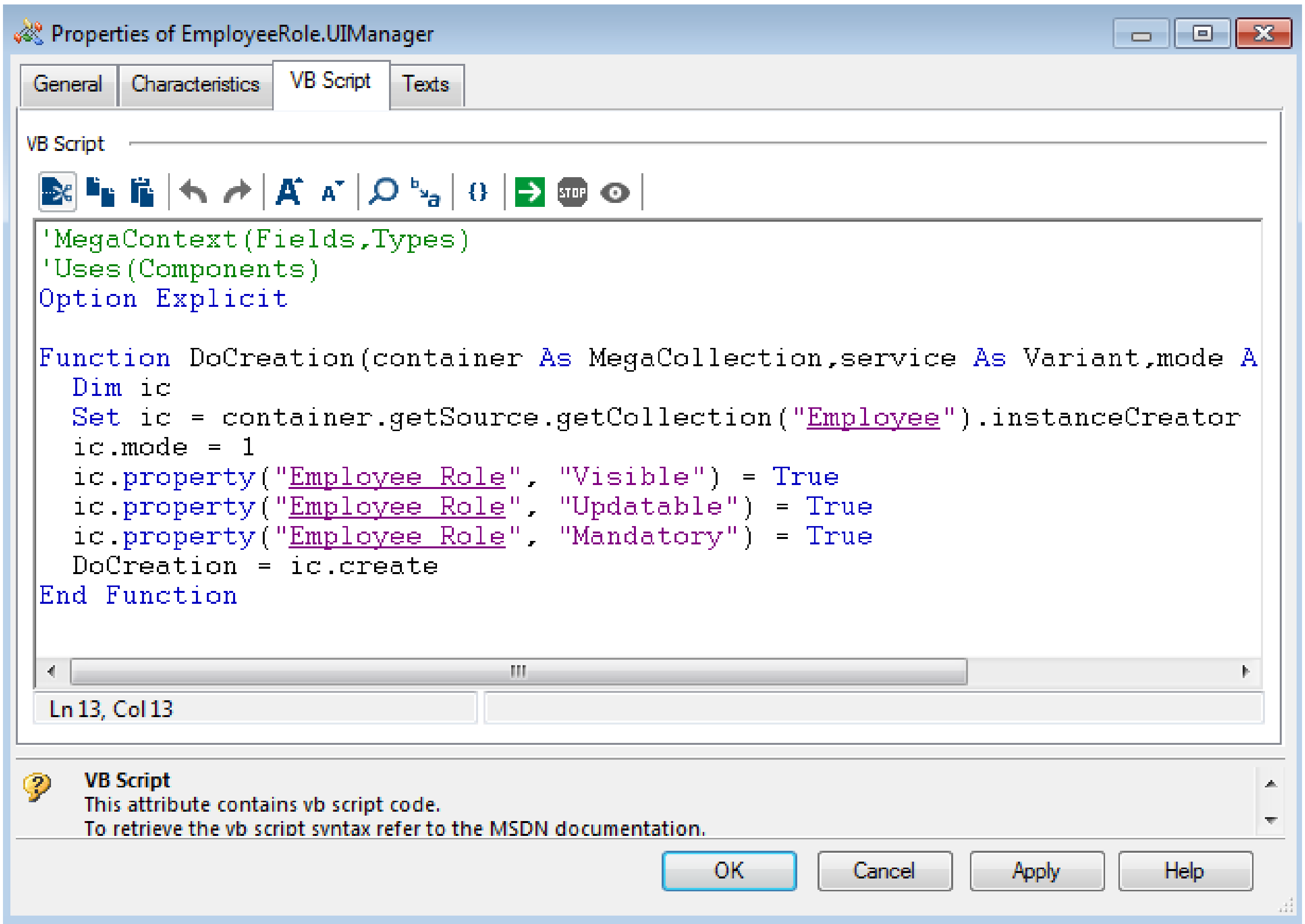
Create the "EmployeeRole.UIManager" macro:

So that the **Employee** created conforms to the query, it must be created from a **Manager** source of the query, and must be associated with an **Employee Role**.
To do this, you have to:

- create an instanceCreator from the collection of **Employees** created from a **Manager** source of the query.
- impose start of the wizard in complete mode by ic.mode = 1
- then configure that the **Employee Role** must be visible and updated, in exactly the same way as in the above customization. The only difference is that here the role is Mandatory.

```
'MegaContext(Fields,Types)
'Uses(Components)
Option Explicit

Function DoCreation(container As MegaCollection,service As Variant,mode As Integer) As Variant
  Dim ic
  Set ic = container.getSource.getCollection("~demSW1OYEDCC[Employee]").instanceCreator
  ic.mode = 1
  ic.property("~y4mZK)OYErIH[Employee Role]", "Visible") = True
  ic.property("~y4mZK)OYErIH[Employee Role]", "Updatable") = True
  ic.property("~y4mZK)OYErIH[Employee Role]", "Mandatory") = True
  DoCreation = ic.create
End Function
```

The function create, and the function DoCreation, return if necessary the identifier of the newly-created object.

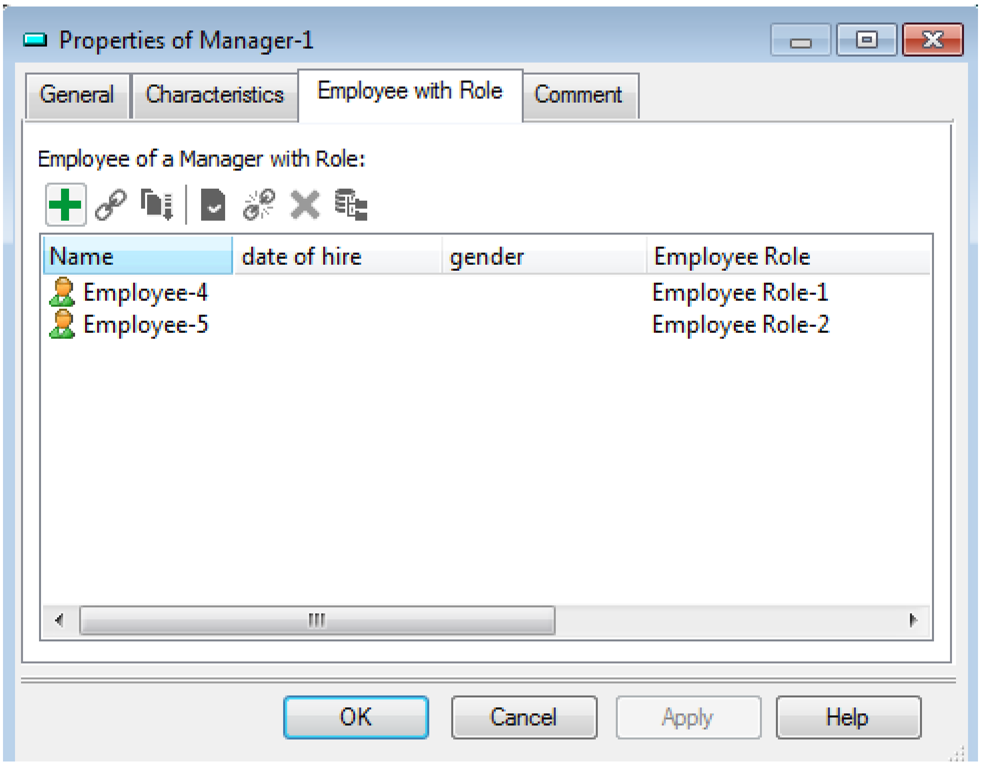


```
'MegaContext(Fields,Types)
'Uses(Components)
Option Explicit

Function DoCreation(container As MegaCollection,service As Variant,mode A
  Dim ic
  Set ic = container.getSource.getCollection("Employee").instanceCreator
  ic.mode = 1
  ic.property("Employee Role", "Visible") = True
  ic.property("Employee Role", "Updatable") = True
  ic.property("Employee Role", "Mandatory") = True
  DoCreation = ic.create
End Function
```

Ln 13, Col 13

VB Script
This attribute contains vb script code.
To retrieve the vb script syntax refer to the MSDN documentation.

From the Properties page of a **Manager**, in the "Employee with Role" tab, note that the **New** button is now accessible.



Name	date of hire	gender	Employee Role
Employee-4			Employee Role-1
Employee-5			Employee Role-2

- You can now from a **Manager** create an **Employee** for which you must specify an **Employee Role**.

This example highlights the fact that it is possible to configure a wizard before calling it. This fact should be noted by the wizard customization designer, who must limit impact of customization to only those elements effectively customized.
It also enables elimination of a possible ambiguity between use of the *Property* function of the context of the wizard, and use of the *GetProp* function of the *template* object of this same wizard. If we wish to initialize a property value for the object to be created, we can consider that the following two functions are possible and therefore redundant:

```
Context.property(propID) = value
Context.template.getProp(propID) = value
```

The *template* object is not available before calling the *create* function: the *property* function is therefore the only one to use if we wish to initialize a property value before starting the wizard. To be more specific:

- Before starting the wizard, the *template* object is not yet created, and use of the *property* function is therefore essential if we wish to consult or update configuration values of the wizard.
- After starting the wizard, and therefore after the *OnWizInitialize* trigger, the *template* object is available; following its creation, property values which may be initialized in the context are assigned to it.
- From the moment the *template* object is available, any modification made from the *property* function is automatically transferred to the template object.
- This system is not however reversible; any direct modifications of the *template* object, for example entries carried out in pages, are not reflected in the corresponding *property* value. During progress of the wizard, this enables determination of whether the current value of *template* has been modified from its initial value.

Specific context elements: cookies

Elements enabling configuration of wizard call context are not necessarily properties of the object.

Certain are sufficiently generic to have been defined directly in the context of creation; in particular this is the case for *sourceID* and *targetID*, which enable indication to the wizard that object creation has been motivated by an action in a diagram (in this case, data contains absolute identifiers of objects to be connected).

However, others that are more specific or less detailed are not anticipated. It is nevertheless possible to carry out configuration of the wizard with such parameters, using cookies.

Like object properties, a configuration cookie is associated with an absolute identifier, and can be consulted or updated by means of the *property* function. A cookie can have any absolute identifier, as long as it does not correspond to the absolute identifier of a property accessible from the object to be created, in which case there is ambiguity.

You can add a cookie to the context of a wizard; to do this, you must define the absolute identifier of this cookie (*cookieID* below), and initialize the property as follows:

```
        context.property(cookieID, « Cookie ») = True ’VBScript
        context.propertyFlag(cookieID, “Cookie”, true);  // Java
```

The “Cookie” flag indicates that the property of the context of the wizard defined does not correspond to a property of the object to be created, but is specific to the context itself.

The context of a wizard can manage two types of cookies; either character strings, or objects. The default type being character strings, we specify that a cookie is of object type by means of the "Object" flag.

The cookies system is used by MEGA tools to enable contextual customization of creation wizards, in particular in diagrams and navigators.

Context cookies of diagrams

When a wizard is called from a diagram, it has the following cookies:

- `~XN(cdzBR8PO0[CookieDiagramNature]` contains the identifier of the *MetaAttributeValue* representing the diagram type.
- `~jL(cb)BR8HP0[CookieDiagramDescribedObject]` contains the identifier of the described object.
- `~bM(cG0CR81Q0[CookieDiagramLegToDescribedObject]` contains the identifier of the *MetaAssociationEnd* to the described object.

Context cookies of navigators

When a wizard is called from a navigator element, it has the following cookies:

- `~aodH6MLoy800[MetaTree]` identifier of *MetaTree* (in the form of a field)
- `~lAhSwEN5)e00[MetaTreeBranch]` identifier of *MetaTreeBranch* (in the form of a field)
- `~u250i2WT(W00[MetaTreeNode]` identifier of *MetaTreeNode* (in the form of a field)

External addition of pages and triggers

You can externally add triggers (in the form of macros) and pages (in the form of *MetaPropertyPages*) to a creation wizard. You can therefore transform a creation wizard into an addition wizard by inserting a suitable preliminary page. These additions should be made before initialization of the wizard, and are therefore available only from the *MegaInstanceCreator* component. This component has the following functions:

- **`addTrigger(TriggerId As Variant,Optional Options As String)` :**
  This method enables insertion of a trigger, of which we define the identifier here. This trigger enables modification of wizard behavior, and in particular the addition of an initialization or processing code. By default, this trigger is called last (that is after the triggers defined for the wizard). However, with the `OnHead` option, it is possible to arrange that this trigger be called first.
- `addPage(PageId As Variant,Optional Options As String)`:
  this method enables insertion of an additional page in wizard. PageId is the identifier of the *MetaPropertyPage* that we wish to insert. The option enables specification of the position of the page, and it can take the following values:

  `Preliminary`: the page is presented in first position. It does not propose entry of name.

  `Preparatory`: the page is presented after the preliminary pages, but before creation of the object.

  `Standard`: This is the default; the page is presented after the preliminary and preparatory pages, but before creation of the object. It displays the name of the object if necessary.

  `AfterCreation`: the page is presented after creation of the object.

  `Conclusive`: the page is presented in last position.

You can add several pages. If they are of the same type, they will be proposed in the order in which they are added.

Creation wizard static configuration

Certain customization elements of a creation wizard can be specified independently of the triggers and pages which have been associated with it. These elements appear in the *_Parameterization* text of *Meta Wizard*.

Default page configuration

Rather than inserting a specific page, it is often wiser and preferable to configure the wizard default page; in fact this page is likely to appear as soon as properties are added (via the metamodel or external customization), and it is therefore difficult to imagine that it will never appear.
The following configurations have been defined for this:

```
[CreatorWizard]
ForceDefaultPage = 1
```

This configuration forces display of the default page, even if not necessary. In this way, you can for example simplify the activity creation wizard customization example mentioned above, by avoiding creation of the "Initialization" page.

`[Template]`, `[Page]`: these two sections, defined for configuration of a properties page, can appear in wizard configuration and enable definition in complete freedom of the default page of the creation wizard, knowing that properties defined externally will be naturally inserted.

Wizard execution configuration

The following configuration:

```
[Wizard]
CloseAfterTerminate = 1
```

arranges that the wizard window closes after execution of *OnTerminate* triggers; it can be used when particularly lengthy processing is being carried out, associating this with a gauge device.

Uncancelable wizard

The following configuration:

```
[Wizard]
Uncancelable = 1
```

removes the wizard from the management of the undo/redo list. This is useful when the wizard execution implies the creation of a great number of objects (several tens or hundreds of thousands). In that case, the standard management of the undo/redo list may slow down the process.

Deactivating the undo/redo management speeds up the process, but object creation cannot be cancelled.

Wizard execution check

In addition to the possibility of dynamically configuring a wizard and carrying out specific processing at transitions, you can modify behavior of a wizard by means of triggers

- by redefining conditions and if necessary the display order of wizard pages.
- by causing its stop, either by discard or by reuse.

Filtering and conditioning of wizard page sequencing

Wizard pages, like *MetaPropertyPages* of properties pages, can be statically filtered. Page display conditions, and in particular filtering carried out in the *[Filter]* section of its configuration, must be respected for the page to appear in the wizard.

You can however dynamically modify this filtering by implementing the *WizPageCheck* function in a trigger, and by acting particularly on the *result* input/output argument.

This function can be used if you wish to modify appearance of wizard buttons and conditions of page display. It can be called several times according to requirements induced by wizard processing logic; in particular it is called at each transition leading to a page.

The *result* parameter is a bit field enabling definition of button appearance and page visibility; bits used are the following:

- bit 1 (value 1): activated when the 'Finish' button is active
- bit 2 (value 2): activated when the 'Previous' button is active
- bit 3 (value 4): activated when the 'Next' button is active
- bit 5 (value 16): activated when the page must be hidden.

When calling this function, button bits (1, 2, and 3) have a value calculated by wizard logic, which in particular deduces that if we are on the first page, the Previous button is inactive, and if we are on the last page it is the Next button which is grayed. If a page declared as mandatory has not yet been displayed, the Finish button cannot be selected.

The *WizPageCheck* function enables overload of these values and also indication that the page should not be displayed on activating bit 5. Consistency of modifications is not checked, and it is therefore possible to activate the Previous button, even if we are on the first page (in this case the press action will have no effect); it is also possible, and here the consequences are more serious, to mask a mandatory page; in this case wizard logic cannot validate the 'Finish' button (since a mandatory page has not been displayed), and the trigger must itself manage this button if we wish to allow the wizard to terminate satisfactorily.

When several triggers implement the *WizPageCheck* function, each trigger receives as input value the result value of the previous trigger.

When a page has been masked, wizard logic carries out the same processing on the next page (if it exists).

Identifying wizard pages

In functions of a trigger referencing properties pages of the wizard, these are represented by objects of *MegaPropertyPage* type. In this component, pages can be identified by using the *GetID* function, which returns a string corresponding to a GUID. In the case of standard pages, this GUID (Global Unique IDentifier) is calculated from the absolute identifier for specific pages, and from that of the *Meta Wizard* for the standard page.la *MetaPropertyPage*

The following script code obtains the GUIDs of pages of a *Meta Wizard*:

```
Function GUIDFromMegaID(mObject)
  Dim xID
  xID = mObject.GetProp("_hexaidabs")
  GUID =  & mid(xID,5,4) & left(xID,4) & "-" & mid(xID,9,4) & "-" & right(xID,4)
  GUIDFromMegaID = "{" & GUID & "-FFFF-FFFFFFFFFFFF}"
End Function

Dim myCol
Set myCol = GetCollection("Meta Wizard").SelectQuery

Dim wiz
For each wiz in myCol
  print "Identifier of pages of wizard " & wiz.Name
  print "     " & GUIDFromMegaID(wiz) & " : Page Standard"
  Dim page
  for each page in wiz.MetaPropertyPage
    print "     " &  GUIDFromMegaID(page) & " : Page " & page.Name & "(" & page.MegaField & ")"
  next
Next
```

Modifying page sequencing

The page display order of a wizard is statically defined in the link between the *MetaWizard* and the *MetaPropertyPage*, using the *Order* and *Sequence* attributes. We can however imagine a wizard presenting pages in a different order depending on execution context, or one which requires looping on pages. In this case, it is possible to redefine succession of pages using functions **WizPageNextChange** and **WizPagePreviousChange**

Check before termination

A trigger can implement a specific check on pressing the 'Finish' button, in particular to prevent the action if a condition has not been satisfied. This check must be implemented in the *OnWizBeforeTerminate* function; the value of *result* must be set to "1" to prevent execution of the code terminating the wizard.

Reuse mode: implementing an addition wizard

You can transform a creation wizard into an addition wizard. In this case, the object returned by the wizard may already exist. When the occurrence to be reused has been defined, wizard logic should not try to create an object, and the wizard must be able to terminate without an additional transition.
This behavior is permitted by means of the *reusedID* function of the *MegaWizardContext* component. In assigning the identifier of an object by means of this function, we cause exit from the wizard after the current transition, and return by the wizard of the identifier specified.

Dynamic customization of the wizard user interface

You can dynamically customize the wizard user interface by means of the following functions of *MegaWizardContext*:

wizardCaption and addingCaption

This function enables redefinition of the creation wizard header; when you wish to transform a creation wizard into an addition wizard , we use the `addingCaption` function.

mainHelpTitle, mainHelpBody and hideMainHelp

These functions enable redefinition of the title and content of the help area displayed in the wizard default page. This help area can also be hidden by `hideMainHelp`.

Wizard specific use and display data

In examples up to this point, pages display data relating to HOPEX objects which exist or will exist; this data is 'justified’ or 'mapped’ by elements of the metamodel.

During creation of a wizard however, you are quickly led to display and enter elements that do not correspond to HOPEX data, but to data local to the wizard. This data can be assimilated in variables.

Wizards can handle such variables by means of cookies. The possibility of displaying cookies in wizard pages is not however envisaged for the moment.

Defining a collaboration between the wizard and its pages

As long as data displayed in wizard pages corresponds to objects described in the metamodel, an implicit collaboration can be included via the *template* object; this object can be handled in the properties pages like a standard HOPEX object, and can be accessed and handled in the code of the wizard and its triggers. However, to display data specific to the wizard, we must define a collaboration.
From the viewpoint of the properties page, this collaboration is necessary to indicate that the data it will display does not come from the *template* object but from another data source.

This other data source is defined by means of an informal query, which allows definition of the description of the collaboration of the wizard.
This collaboration must be declared in the wizard configuration text as follows:

```
[Wizard]
  Description = <ID of the query>
```

On completion of this declaration, the properties and collections declared in the collaboration are accessible from the context of the wizard via the MegaObject "CookieObject". Note that these properties and collections are also accessible as a cookie.

To initialize or modify a property of the collaboration, you can therefore choose from:

- `context.property(cookieID) = <value>`
- `context.cookieObject.getProp(cookieID) = <value>`

To initialize a collection, you must use:

- `context.property(cookieID) = <Collection>`

  When the collection is not initialized in this way, it cannot be used, and the call to GetCollection produces an error "No collection associated to this cookie". In addition, you cannot modify this collection after initialization.

Declaration of this collaboration produces automatic injection of the "Context" map in all wizard properties pages.

- You can therefore include a property or collection of the collaboration in a properties page of the wizard by declaring the element in this map by means of keyword `From(Context)`.

{compatibility} Declaration of the description in configuration of the wizard avoids:

- use of function context.cookieDescription, which is therefore obsolete,
- explicit declaration of the collaboration map in the [template] of the properties page

  ```
  Cookies=Map(<Description>,Cookie(~DutIllKV5X40[Meta Wizard])
  ```

  this map **Cookies** being replaced by the implicit map **Context**.

Although possible technically, the list of cookies does not naturally correspond to the list of properties defined for a *MetaClass*, *MetaAssociation* or *MetaAssociationEnd*. Nor is it reasonable to create such concepts in the metamodel with the sole aim of defining collaboration of a wizard.

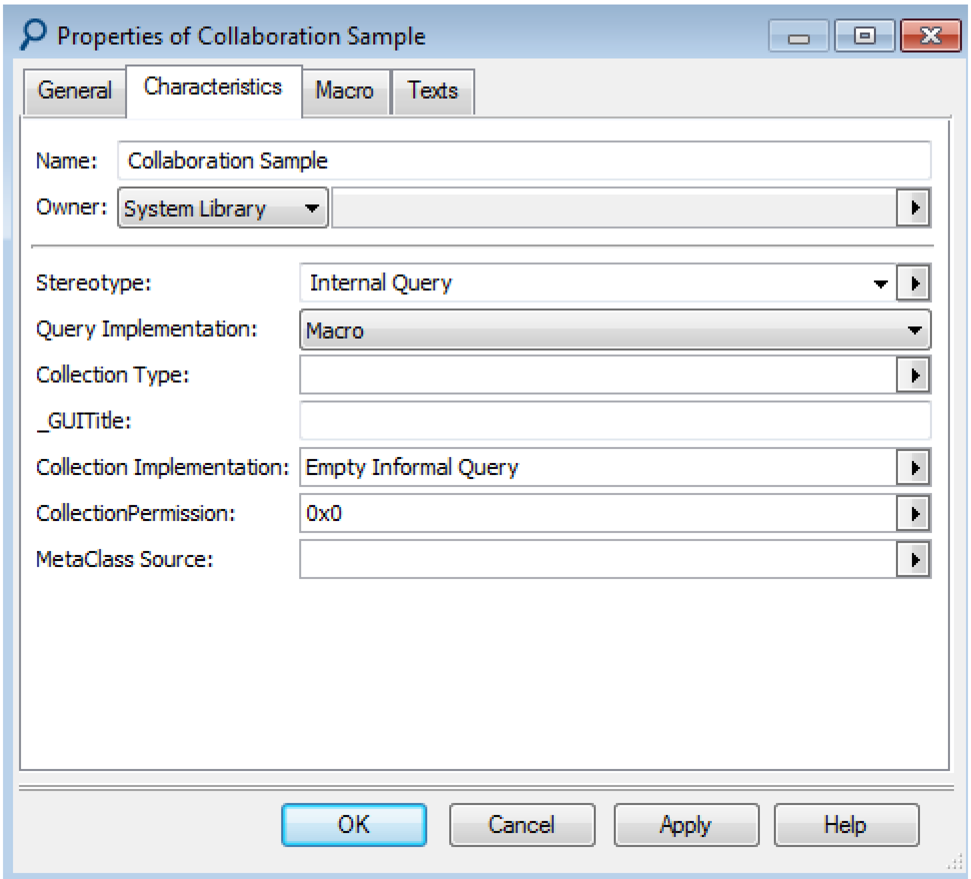
To do this, we favor use of an informal query as collaboration of a wizard.

Defining an informal query

Informal queries are queries not associated with a target MetaClass; they cannot therefore be used in the standard query tool. They can however be associated with a macro which implements their supply.

These queries are mainly used by script tools, and enable definition of collections of informal objects used as *MegaObjects* within a *MegaCollection*.

To create an informal query, you must create a query of 'Macro' type and associate it with the "Empty Informal Query" macro.


Properties of Collaboration Sample

Name: Collaboration Sample

Owner: System Library

Stereotype: Internal Query

Query Implementation: Macro

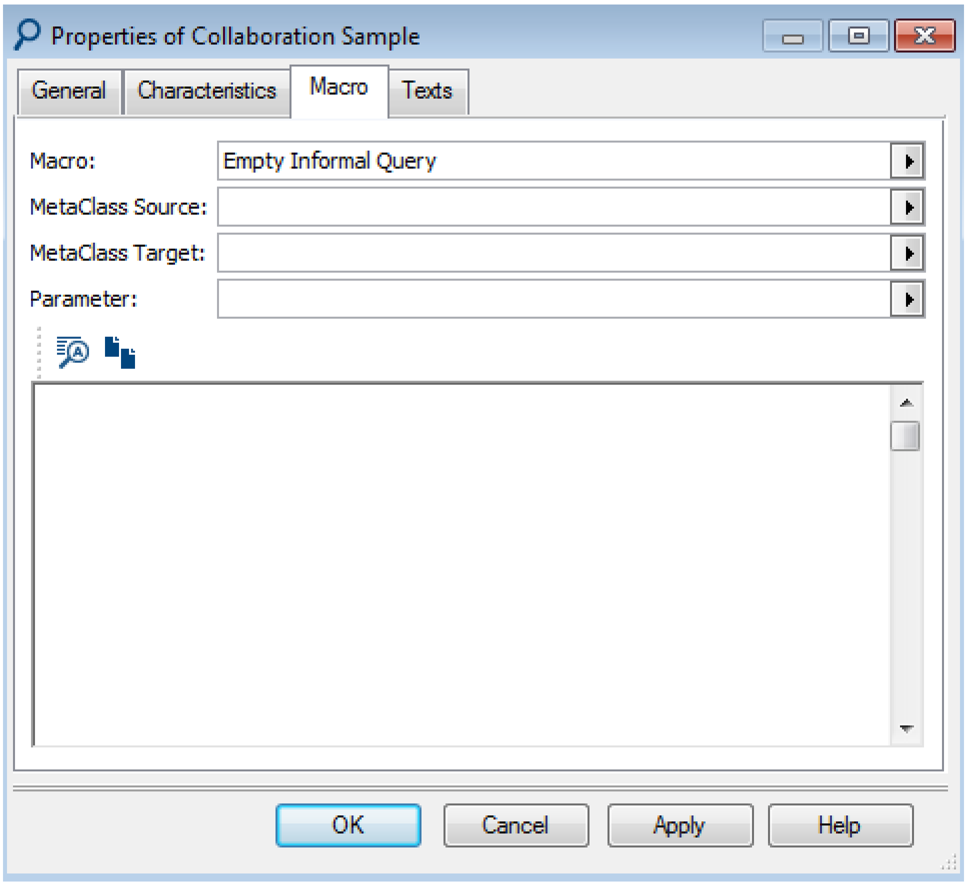
Collection Type:

_GUITitle:

Collection Implementation: Empty Informal Query

CollectionPermission: 0x0

MetaClass Source:


Properties of Collaboration Sample

Macro: Empty Informal Query

MetaClass Source:

MetaClass Target:

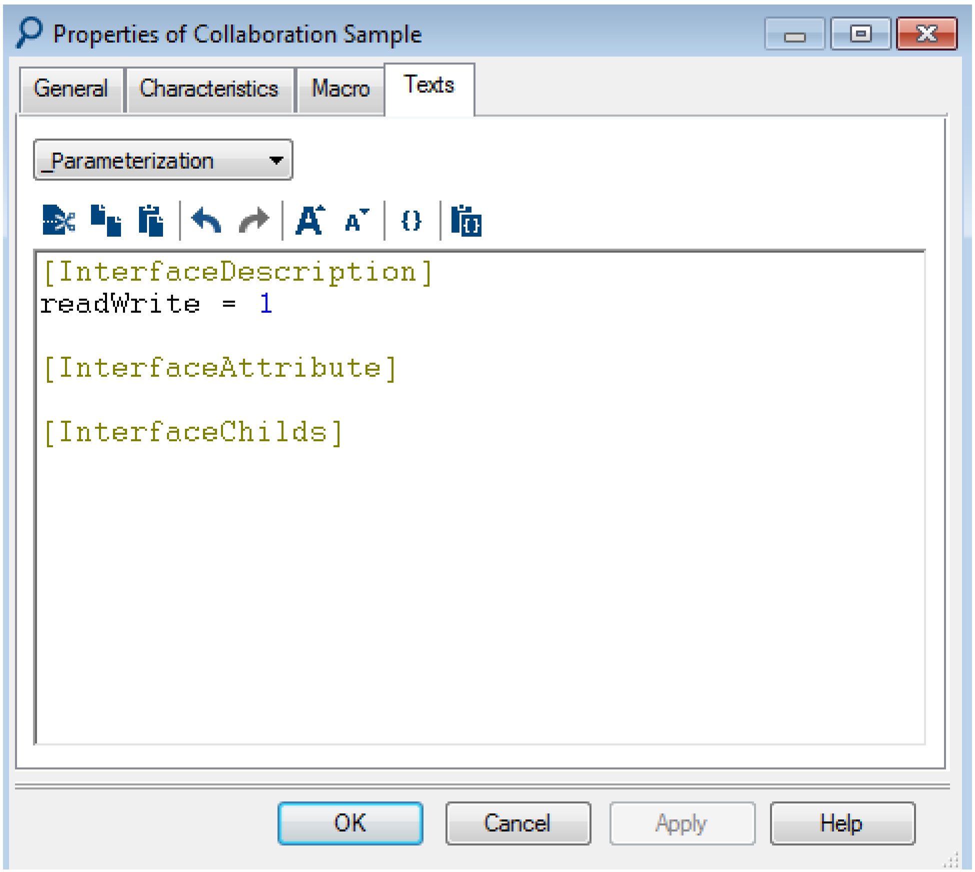
Parameter:

Description of an informal query not being based on a MetaClass, you must define it specifically; no metamodel extension is used for this definition, which uses the _parameterization text of the query.

- The [Interface Description] paragraph enables definition of general data of the informal query, and in particular if it is possible to freely create objects (key ReadWrite = 1). You can in this paragraph redefine the attribute which plays the role of identifier (by default this is the 'Order' Mega attribute). Attributes defined by default for these roles are generally satisfactory and it is not necessary to redefine these.

To add:

- properties to informal queries, insert keys in the [InterfaceAttribute] section.
- collections accessible from objects corresponding to this description, insert keys in the [InterfaceChilds] section.



```
[InterfaceDescription]
readWrite = 1

[InterfaceAttribute]

[InterfaceChilds]
```

In your « Collaboration sample » informal query example, you shall define:

- a property of list type enabling definition of an action to be selected from 'Option 1' 'Option 2' and 'Option 3'
- a collection of Keywords

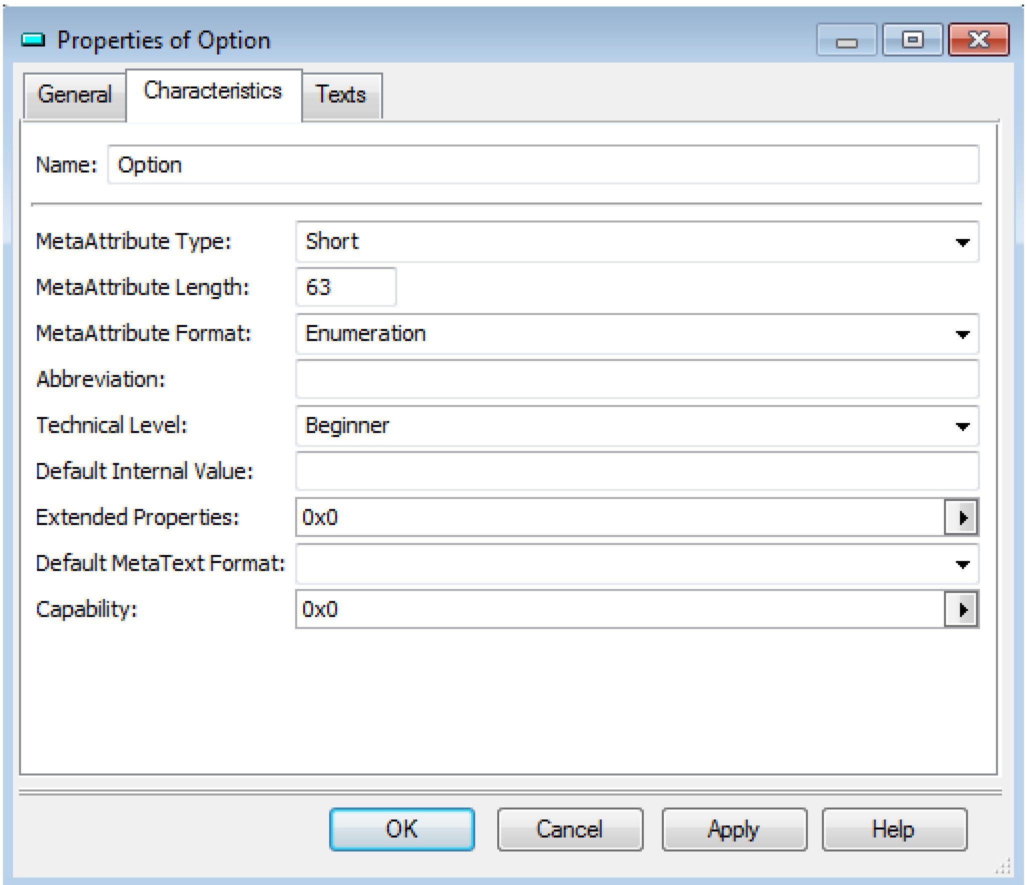
The list of *[InterfaceChilds]* can accept *MetaClasses*, *MetaAssociationEnds* or Queries (which can also be informal). The collection of Keywords can appear, either via a query with Keyword target, or via the Keyword MetaClass itself. We choose the latter solution, the first only being useful if we wish to define several collections based on Keyword.

1. Insert:

   ```
   [InterfaceChilds]
   ~VrUiN9B5iSN0[Keyword] = XREF
   ```

   XREF indicates that we define a reference to a collection definition.

2. To define the requested list, create an "Option" TaggedValue of 'Short' type and 'Enumeration' format.



Properties of Option

Name: Option

MetaAttribute Type: Short

MetaAttribute Length: 63

MetaAttribute Format: Enumeration

Abbreviation:

Technical Level: Beginner

Default Internal Value:

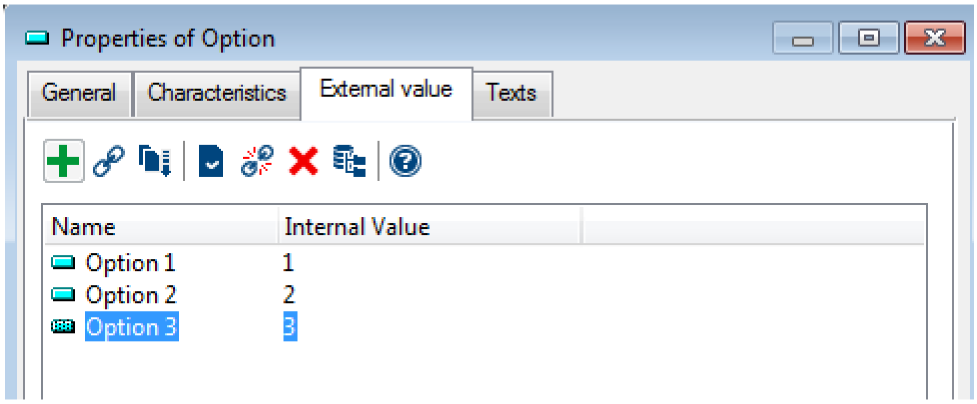
Extended Properties: 0x0

Default MetaText Format:

Capability: 0x0

3. Close the properties dialog box of the TaggedValue and reopen it.
   The **External Values** tab appears.

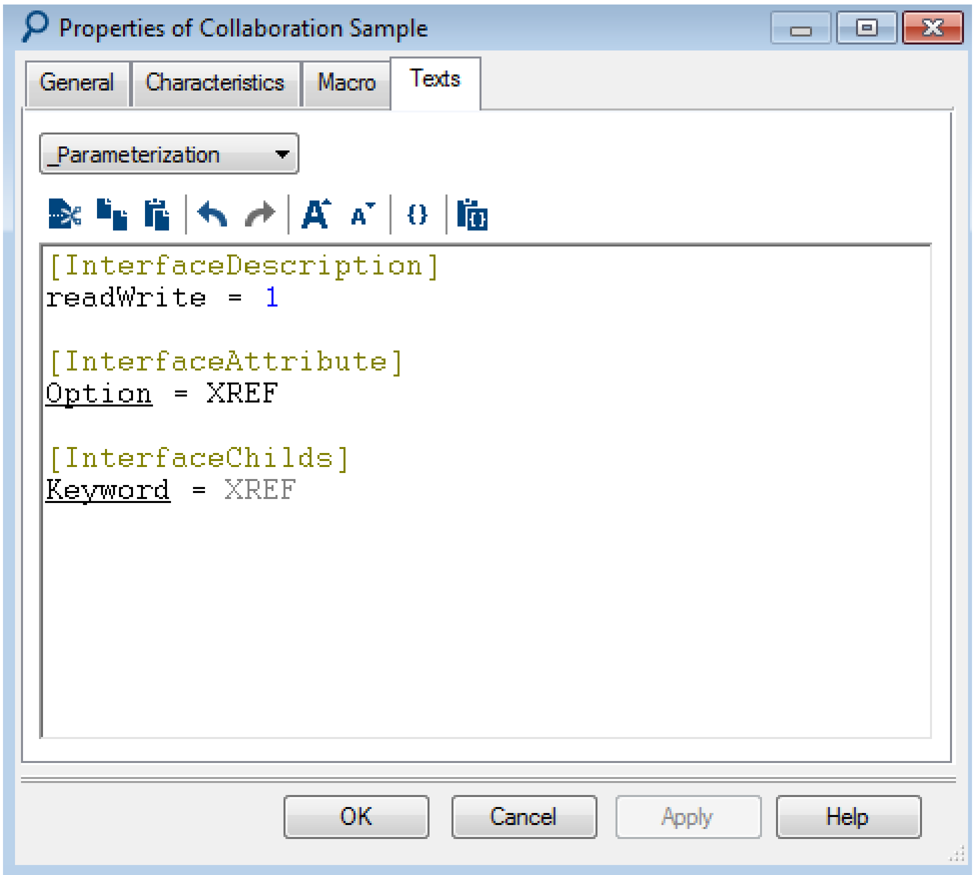
4. On this taggedValue, define 3 tabulated values with internal values 1, 2 and 3 corresponding to the 3 desired options.



Name	Internal Value
Option 1	1
Option 2	2
Option 3	3

5. Insert this *taggedValue* in the list of properties of the "Collaboration Sample" query:

```
[InterfaceAttribute]
Option = XREF
```



```
[InterfaceDescription]
readWrite = 1

[InterfaceAttribute]
Option = XREF

[InterfaceChilds]
Keyword = XREF
```

In the above trigger you can use the following collaboration cookies:

- Property("<Option>") which will contain a numerical value
- Property("<Keyword>") which will contain a list of keywords

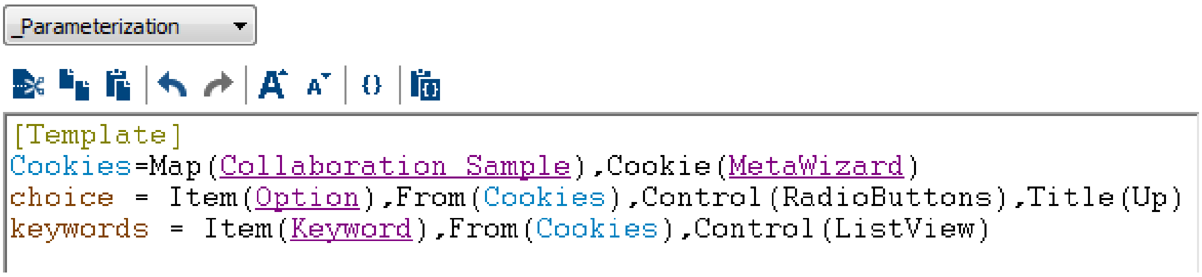
Data between square brackets indicates fields corresponding to identifiers of the objects being handled.

6. You will define a page preparatory to the Employee creation wizard by displaying the data mentioned above.

   To do this, insert the following lines in the [Template] paragraph of the "Employee.CreationWizard.Implementation" macro. Remember to associate the trigger macro to the wizard.

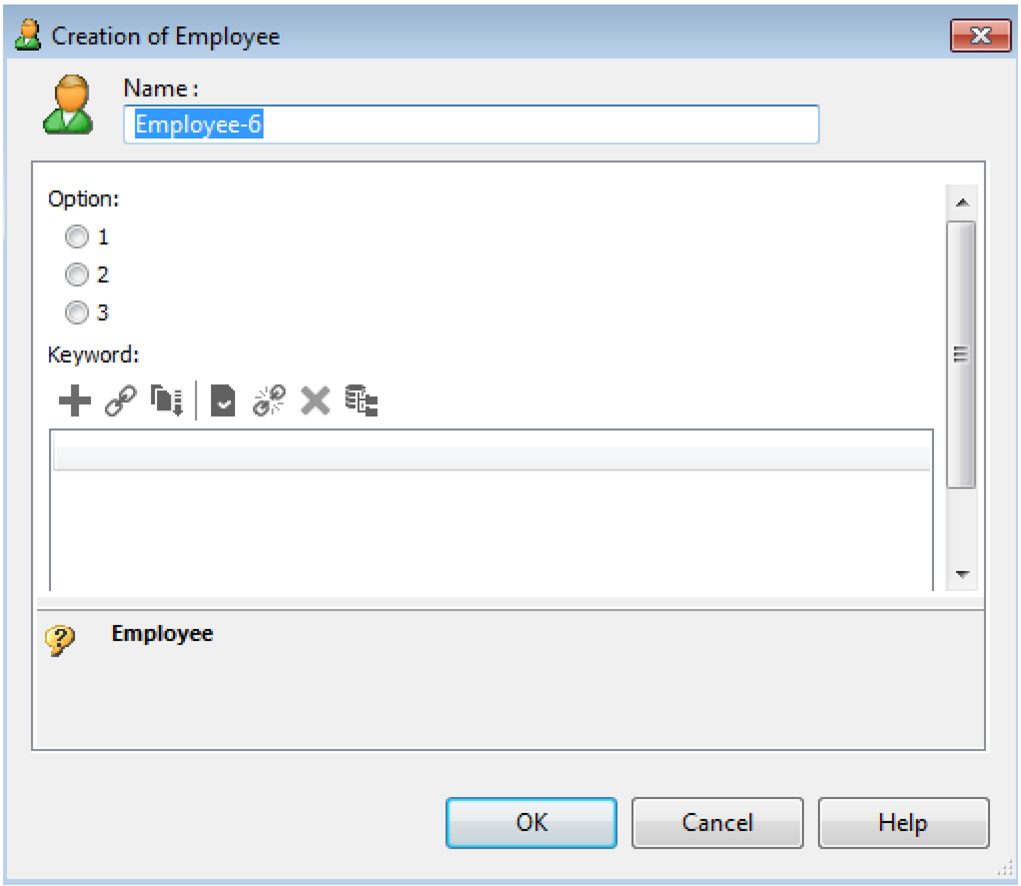
```
 [Template]
Cookies=Map(<Collaboration Sample>),Cookie(~DutIllKV5X40[Meta Wizard])
choice = Item(<Option>),From(Cookies),Control(RadioButtons),Title(Up)
keywords = Item(<Keyword>),From(Cookies),Control(ListView)
```

Note : <objet> corresponds to IdAbs of Object[Object Name]



```
[Template]
Cookies=Map(Collaboration Sample),Cookie(MetaWizard)
choice = Item(Option),From(Cookies),Control(RadioButtons),Title(Up)
keywords = Item(Keyword),From(Cookies),Control(ListView)
```

7. Create an Employee.



Initializing collaboration cookies

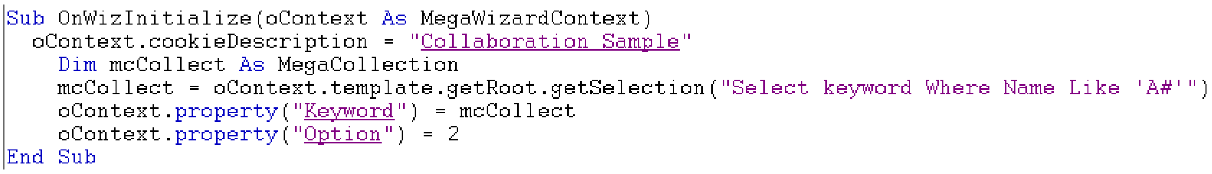
It is essential to initialize collaboration cookies producing collections, as is the case in the previous example for the Keyword collection. When displayed, this page will request the context for the collection of keywords via the <Keyword> property. When this collection has been requested, it will be displayed in the list and no longer requested from the wizard context.

If the context of the wizard wishes to check the content of this cookie, it should be initialized before display of the page, without change of collection; otherwise there will be inconsistency between cookie content and the page list.

In the example above, you can initialize this collection with the following keyword selection:

VBScript :

```
Sub OnWizInitialize(oContext As MegaWizardContext)
  oContext.cookieDescription = "<DescriptionId>"
  Dim mcCollect As MegaCollection
  mcCollect  =  oContext.template.getRoot.getSelection("Select  Keyword  Where  Name
Like 'A#'")
  oContext.property("~VrUiN9B5iSN0[Keyword]") = mcCollect
  oContext.property("<OptionId>") = 2
End Sub
```

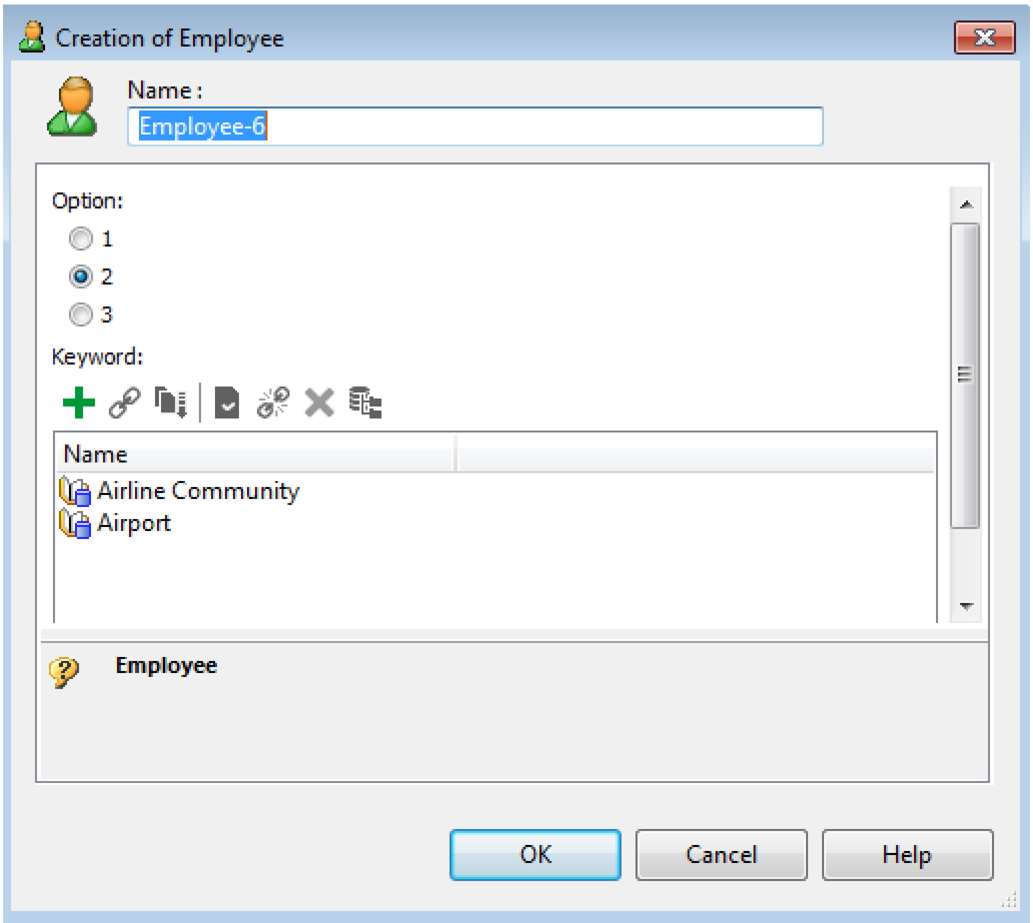


```
Sub OnWizInitialize(oContext As MegaWizardContext)
  oContext.cookieDescription = "Collaboration Sample"
    Dim mcCollect As MegaCollection
    mcCollect = oContext.template.getRoot.getSelection("Select keyword Where Name Like 'A#'")
    oContext.property("Keyword") = mcCollect
    oContext.property("Option") = 2
End Sub
```

Java :

```
public void OnWizInitialize(MegaWizardContext oContext) {
  oContext.cookieDescription("<DescriptionId>");
  MegaCollection mcCollect;
  mcCollect = oContext.template().getRoot().getSelection("Select Keyword Where Name
Like 'A#'");
  oContext.property("~VrUiN9B5iSN0[Keyword]", mcCollect);
  oContext.property("<OptionId>", 2);}
```

In the above example, you also initialize the value of the option (<OptionID>) cookie. The page now appears as follows:



Name : Employee-6

Option:

○ 1
● 2
○ 3

Keyword:

Name
Airline Community
Airport

Employee

In processing of the trigger, if you wish to modify content of this list, we should not change the collection assigned to this cookie; this change will not be taken into account by the properties page, except in the case of global update. It is preferable to modify content of the collection by means of the *insert* and *remove* functions of the *MegaCollection*.

Now having a preliminary page, the creation wizard can condition display of the following pages to the value of the 'option' cookie. Let us consider that the 'Option' induces as the next page a page chosen from three pages with different content. We define a filtering code in the WizPageCheck function, and mask the page if it does not correspond to the value selected in 'Option'.

Wizards on abstract MetaClasses

You cannot create an occurrence of abstract class; you can however define a wizard on an abstract class:

These wizards enable definition of pages and triggers common to all concrete MetaClasses derived from this abstract class.

Pages and triggers defined on the wizard of the abstract class are inserted in respective sequences by consolidation of order numbers to connected wizards (for example, the trigger order number 10 on the wizard of the abstract class will be called before the trigger order number 20 on the wizard of the concrete class).

Where order numbers are equal, the trigger of the concrete class is called last.

Interactive tool based on a MEGA wizard

We shall now broaden our field of study; use of a MEGA wizard is not restricted to customization of object creation, but can assist in implementation of all kinds of interactive tools. The limitation of this use is the contour of what can be managed by a HOPEX standard properties page.

Such wizards can be for example tools for modification or transformation of HOPEX occurrences, or tools for import or export to third party tools.

Differences from a creation wizard

A MEGA wizard not being a creation wizard, its operation and use are different:

- the *template* available in the context does not represent an object in creation phase.
- there is no default page; configuration relating to this page defined in the *Meta Wizard*, or specified in the context of the wizard, is not taken into account.
- page sequences are not used, in particular those which reference a creation phase.
- the creation phase itself does not exist, and the *OnWizRealize* function of triggers is not called.
- pages do not display by default the name of the object.

Concerning use, calling a wizard is not by means of the *InstanceCreator* function.

In addition it is possible to redefine in greater detail the appearance and *frame* of such a wizard.

system process call

MEGA wizards can be called by means of the *MegaWizard* component.

This component can be created by the *WizardRun* function. This function can be called on any *MegaObject*, providing as parameter the absolute identifier of the *Meta Wizard* to be called. In Java, builders of the Wrapper MegaWizard class calling this function are made available.

The *MegaObject* to which the *WizardRun* function relates, which can be a *MegaRoot*, will be accessible from the context of the wizard via the *template* member, which in this case represents a real object.

The MegaWizard component has the following functions:

- **context**: makes available the *MegaWizardContext* of the wizard. It is therefore possible to configure the wizard before calling it.
- **picture(<PictureID>)**: enables definition of the image displayed in the wizard; this image can be defined in wizard configuration.
- **run**: this function calls the wizard.

Configuring a wizard

A generic wizard is modeled in HOPEX by a *Meta Wizard*; however, this *Meta Wizard* is not associated with a *MetaClass*, as a creation wizard could be.

Static configuration of these wizards (defined in the *_Parameterization* text of the *Meta Wizard*) has been enhanced; in particular, it is possible for these wizards to define an image displayed in their *frame*, while the creation wizard, for reasons of consistency, simply displayed the icon of the MetaClass to be created.

The parameters introduced are:

```
[WizardFrame]
      BitmapStyle = 1 or 2
```

Indicates that the image is displayed at top (1) or on left (2).

```
      BitmapHorizontalMargin = n
```

Indicates in pixels the horizontal margin of the bitmap.

```
      BitmapVerticalMargin = n
```

Indicates in pixels the vertical margin of the bitmap.

```
      BitmapWidth = n
```

Indicates in pixels the width of the bitmap.

```
      BitmapHeight = n
```

Indicates in pixels the height of the bitmap.

```
      BitmapName = name
```

Specifies the name of the file for the bitmap. This file should be located in one of the HOPEX configuration files (Mega_Std). The .bmp file extension should not appear.

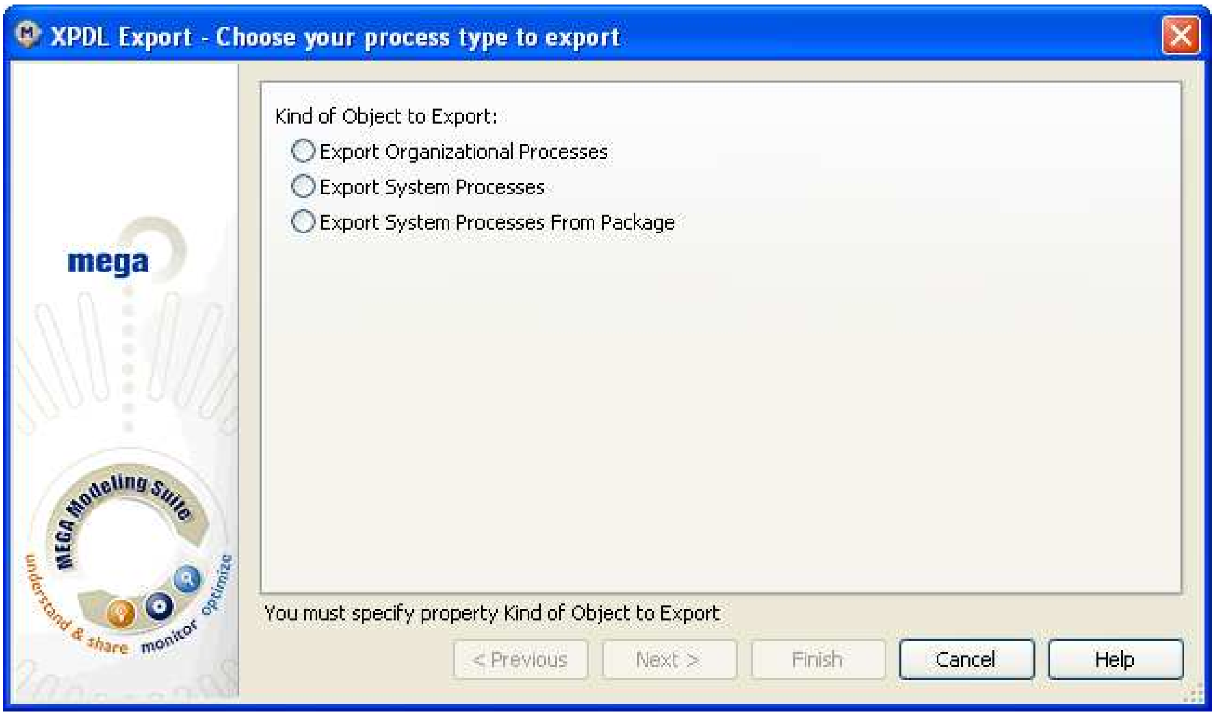
It is possible to redefine the icon of the wizard by specifying the picture in:

```
[Wizard]
Picture=<identifier>
```

Configuration example: export wizard XPDL BPMN

```
[WizardFrame]
BitmapStyle = 2
BitmapName = processWizard
BitmapWidth = 121
BitmapHorizontalMargin = 0
BitmapVerticalMargin = 0

[Wizard]
Picture = ~sj4c8ZPf2n40[BaseExport]
```

The configured frame becomes:



XPDL Export - Choose your process type to export

Kind of Object to Export:

○ Export Organizational Processes

○ Export System Processes

○ Export System Processes From Package

You must specify property Kind of Object to Export

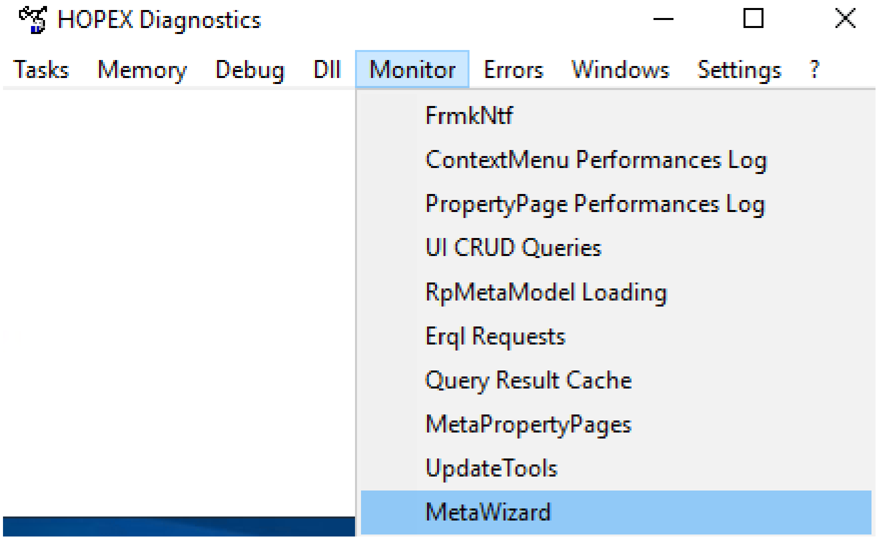
Appendices

Wizard execution tracking

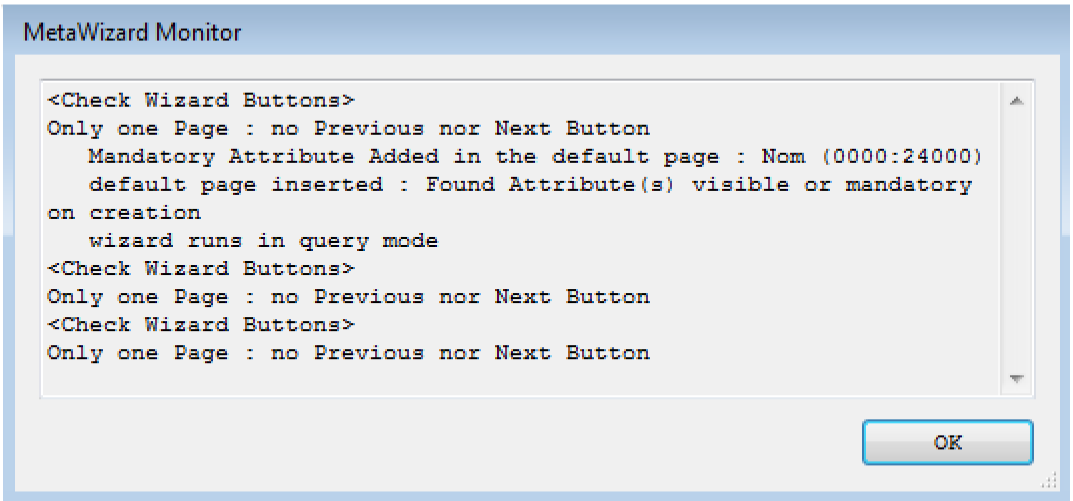
A utility enabling tracking of execution logic of a wizard is available. To activate this, the **HOPEX diagnostics** window should be opened. This can be done using the following line of script:

```
currentEnvironment.site.toolkit.createMegaObject("Mapp.Diagnostics").open
```

In this window, the **MetaWizard** menu is available in the **Monitor** tab when a wizard is started in HOPEX. We must therefore start a wizard before selecting the menu.



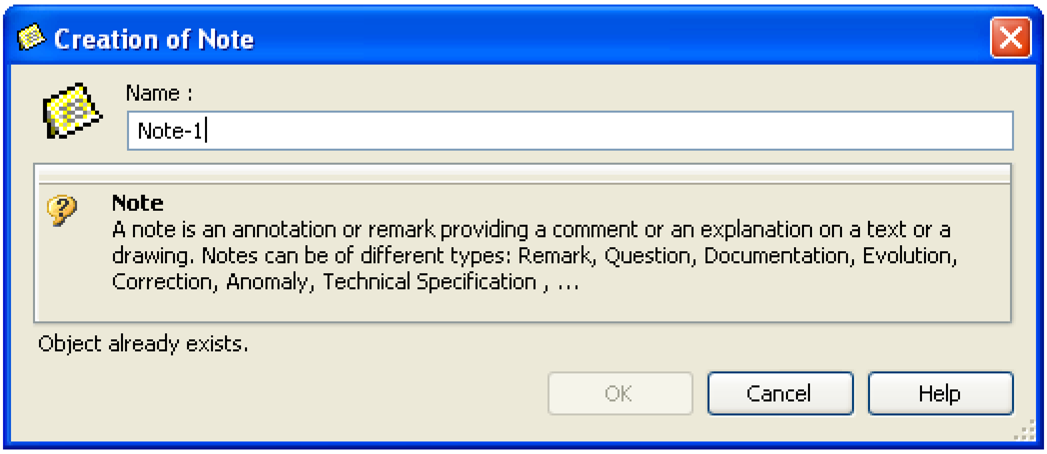
This command opens a window displaying elements relating to execution of wizards.



```
<Check Wizard Buttons>
Only one Page : no Previous nor Next Button
   Mandatory Attribute Added in the default page : Nom (0000:24000)
   default page inserted : Found Attribute(s) visible or mandatory
on creation
   wizard runs in query mode
<Check Wizard Buttons>
Only one Page : no Previous nor Next Button
<Check Wizard Buttons>
Only one Page : no Previous nor Next Button
```

Wizard button dynamic activation

In creation wizards, the name edit area is treated in a specific way: besides its positioning, outside the page area, the content of the area acts dynamically on activation of buttons **OK** or **Finish**, in particular when the name is not specified, or when the name is already assigned to another object; in this case, a message of explanation is displayed: "This object already exists".


Creation of Note

Name : Note-1

Note
A note is an annotation or remark providing a comment or an explanation on a text or a drawing. Notes can be of different types: Remark, Question, Documentation, Evolution, Correction, Anomaly, Technical Specification , ...

Object already exists.

However, default behavior of the wizard is such that other areas are only checked at a transition; the transition is refused if all areas have not been validated:

- all mandatory areas must be specified.
- all specified values must be valid.

Additional checks can be defined:

- by specifying a mandatory element in the properties page:

  ```
  myItem = Item(...)...,Mandatory(True)
  ```

- or by implementing these in a trigger, in the *OnWizBeforeTerminate* function. This function being called when pressing the "Finish" button (or "OK" if there is only one page in the wizard), it enables prevention of validation of the wizard. It is compulsory in this case to inform the user, either by displaying a message box, or preferably by explicitly specifying the message area as follows:

In VBScript:

```
Page.Component.StatusMessage = "Message"
```

In Java :

```
page.component().toMegaPropertyPageStandard().setStatusMessage("Message");
```

This check should not however be implemented in the *WizPageCheck* function when 'Previous' and 'Next' buttons are not active; this function being called at display of the page, the button cannot be reactivated and it will not be possible to terminate the wizard.

This behavior is functionally satisfactory, but you can arrange that behavior similar to that of the name area can be applied to other areas of the wizard, that is activation of buttons can evolve dynamically, according to modifications carried out in the page.

To do this, you must configure the page so as to activate instantaneous check; this can be done at initialization of the page, in the *OnWizPageInitialize* function.

VBScript :

```
Sub OnWizPageInitialize(Manager As MegaWizardContext,Page As MegaPropertyPage)
 Page.Component.ActiveValidation = True
End Sub
```

Java :

```
public void OnWizPageInitialize(MegaWizardContext wctx,MegaPropertyPage mpp){
  MegaPropertyPageStandard mpps = mpp.component().toMegaPropertyPageStandard();
  mpps.setActiveValidation(true);
}
```

You can also activate customized checks by establishing a connection in the properties page; this connection enables a specific code to be activated on each user action in the page. This connection can also be activated at page initialization; in Java, the specific code is implemented via the interface

```
MegaPropertyPageStandard.PageConnectionPoint
```

```
mpp.connect(this); // the wizard class implements the PageConnectionPoint interface
```

The *PageConnectionPoint* interface comprises the function:

```
public void OnCommand(MegaPropertyPageStandard page, int notif, int control)
```

Notifications received depend on the elements contained in the page and are not the subject of a detailed description here. The example below enables tracking of events received in the connected pages:

```
public void OnCommand(MegaPropertyPageStandard page, int notif, int control) {
  String itemName = page.currentControl().getProp("ItemName");
  page.getRoot().print("OnCommand:" + itemName + "  (" + notif + ")");
}
```

(To view displays controlled by the 'print’ function activate tracking of macros by menu "Monitor>Mega Scripts Output" of the **HOPEX diagnostics** window mentioned above).

Versatile Desktop

1 VERSATILE DESKTOP OVERVIEW

The **Versatile Desktop** functionality enables creation and customization of the work environment of HOPEX users. A user can access different desktops adapted to his/her requirements and to the actions he/she must execute as a function of a given role.

1.1 Public concerned

Desktop creation and definition is intended for:

- Functional developers, for HOPEX desktops supplied as standard.
- Product Engineers or Administrators, for customized desktops for client accounts.

1.2 Elements of a work environment

To customize the work environment of users, the following elements are available:

- **a desktop**, which can contain one or several containers
- **containers**, which can contain other containers or desktop components
- **desktop components**, which are:
  - tools associated with a configuration (**MEGA Parameterized Tools**)

    E.g.: tree, list, menu, HTML formatters.
  - tools (**MEGA Tool**)

    E.g.: Query, Properties pages, Wizard, Widget, diagram editors and HTML.

⚠ **A desktop cannot directly contain a MEGA Tool.**

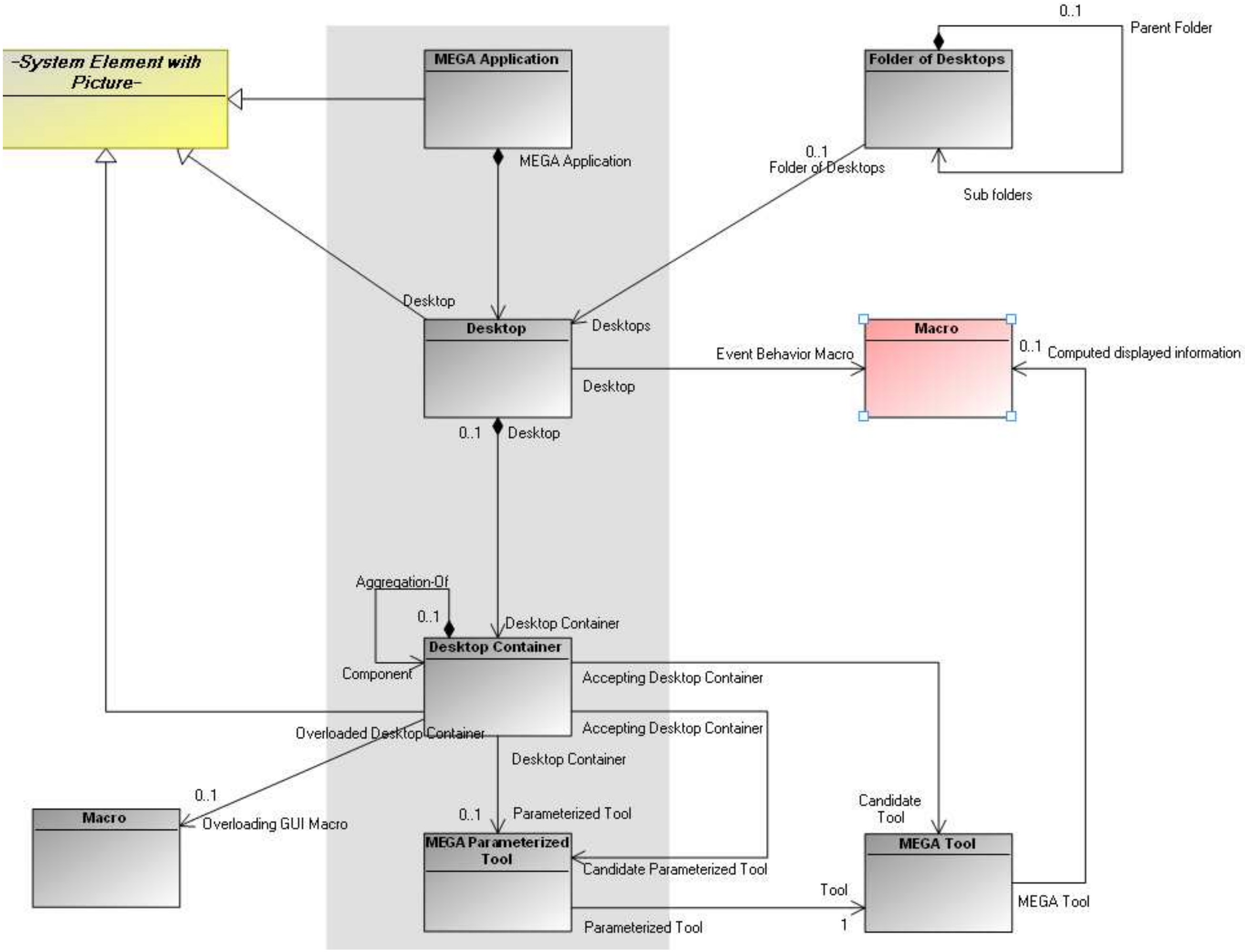
The desktop contains a MEGA Parameterized Tool which contains the MEGA Tool.

1.3 Metamodel

The following Metamodel schema shows architecture and links between:

- **Application**
- **Desktop**
- **Container**
- **Desktop components**
- **Tools**

➢ To see another Metamodel using a Working Environment Template (WET) combined with Desktops, see WET metamodel.



Principle:

A **user** can connect to one or several desktops depending on his/her profile and authorizations. The user passes from one desktop to another by logoff/login or by using the MEGA Tool **Desktop Switcher**.

A **desktop** comprises one or several containers, in which are defined tools that will be displayed.

⚠ A **container** can contain only a single tool.

A **tool** must be hosted by a container.

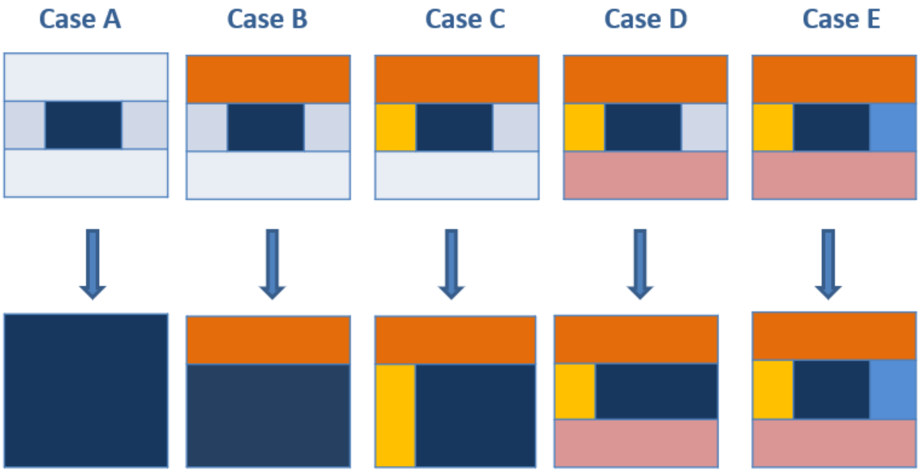
To create and customize the desktop of an application, you should follow these steps:

Step	Action	See chapter
1.	Define desktop configuration	2
2.	Define desktop component elements	3
3.	Create desktop structure	4
4.	Create desktop Containers	5
5.	Define characteristics of Containers in work environment	6
6.	Configure desktop (pop-up menus, toolbars)	7

2 DESKTOP CONFIGURATION

Before configuring a desktop, you must define:

- the elements the desktop will contain and how these elements will be arranged.
- the number of containers the desktop will contain.
  The desktop must contain at least one Container (the Center Container).



	Number of elements	Actions
Case A	One	- Define Center Container
Case B	Two	- Define Center Container - Define Top Container
Case C	Three	- Define Center Container - Define Top Container - Define Left Container
Case D	Four	- Define Center Container - Define Top Container - Define Left Container - Define Bottom Container
Case E	Five	- Define Center Container - Define Top Container - Define Left Container - Define Bottom Container - Define Right Container

3 Desktop component elements

A desktop comprises Containers and desktop components (Parameterized Tools).

3.1 Container types

In your desktop you can insert the following **Container** types:

- **Container of Tool**: this Container is a tool recipient.
- **Container of Sub-Containers**: this Container is a recipient designed to receive other Containers of **Container of Sub-Containers** or **Container of Tool** type.

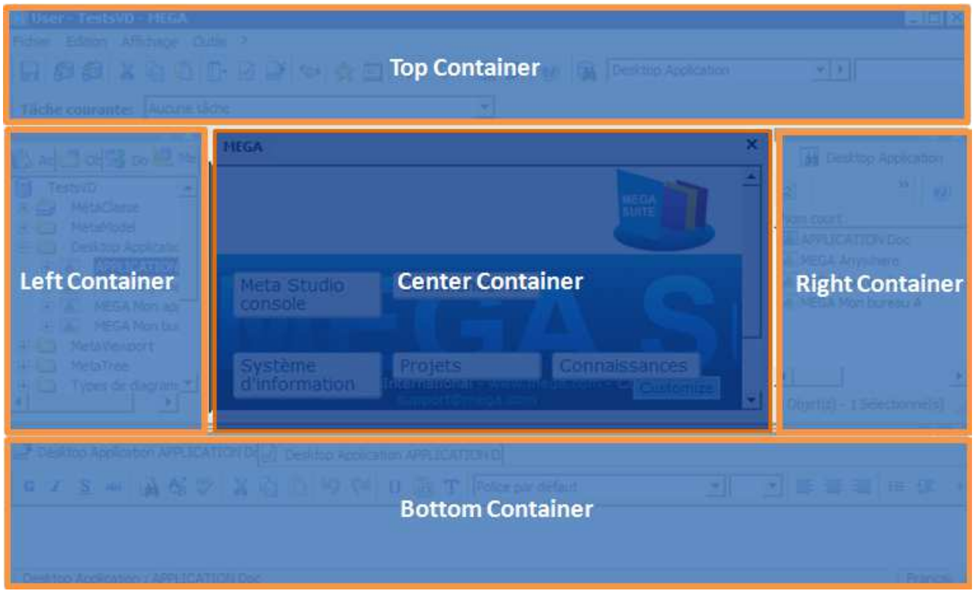
3.2 Container of Sub-Containers

A **Container of Sub-Containers** can be of type:

- **Border Layout**
- **Accordion**
- **TabPanel**
- **Portal**
- **Hbox**
- **Vbox**

3.2.1 Border Layout

The **Container of Sub-Containers** of **Border Layout** type can comprise one to five Containers. It is represented as here:

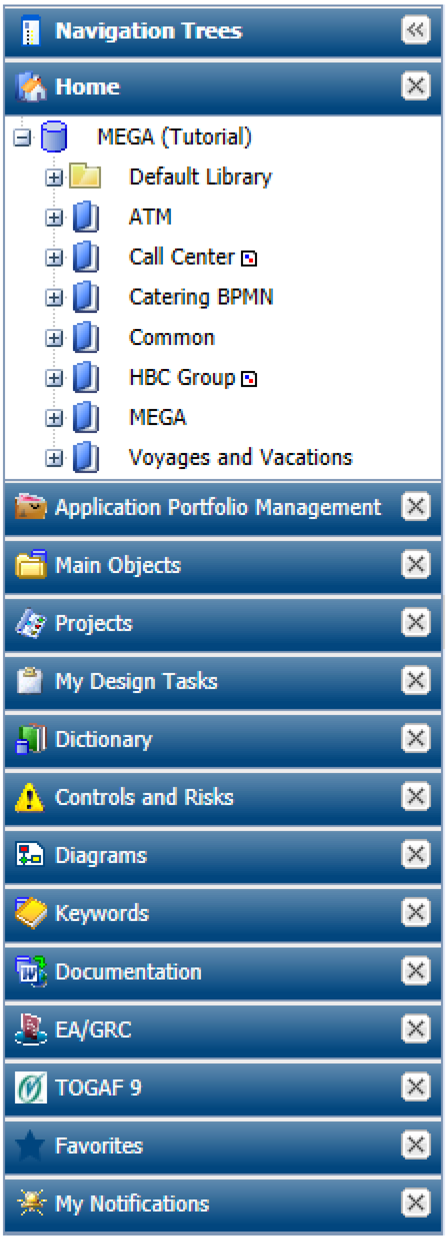


The **Center Container** is mandatory.

The **Top**, **Left**, **Right**, and **Bottom Containers** are optional.

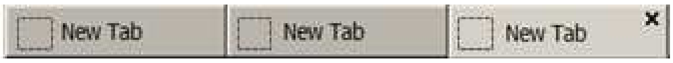
3.2.2 Accordion

The **Container of Sub-Containers** of **Accordion** type comprises a stack of pages containing desktop components. It can be used for example in a **Left Container**. It is represented as here:


Navigation Trees
Home
MEGA (Tutorial)
Default Library
ATM
Call Center
Catering BPMN
Common
HBC Group
MEGA
Voyages and Vacations
Application Portfolio Management
Main Objects
Projects
My Design Tasks
Dictionary
Controls and Risks
Diagrams
Keywords
Documentation
EA/GRC
TOGAF 9
Favorites
My Notifications

3.2.3 TabPanel

The **Container of Sub-Containers** of **TabPanel** type comprises tabs. It can be used for example in a **Left Container**. It is represented as here:



3.2.4 Portal

The **Container of Sub-Containers** of **Portal** type comprises widgets and is defined by a number of columns for the widget list. It can be used for example in a **Center Container** for the MegaParameterized tool used for widgets.

3.2.5 Vbox and Hbox

The **Container of Sub-Containers** of **Hbox** (or **Vbox**) type are simple containers used for example for a left to right display (or a top to down display).

3.2.6 Container attributes

Each Container can be customized by means of its attributes. Attributes of a **Container** are defined in its **Properties** page, **Characteristics** tab. Some attributes vary according to the Container type:

- **General attributes**:
  - name in user interface
    _GUIName

    ⚠ The name of the Container in the user interface is specified only in the case of a Container of Containers. If the Container contains a tool, it is the tool name that is displayed.

  - icon in user interface
    MetaPicture

    ⚠ The icon in the user interface is specified only in the case of a Container of Containers. If the Container contains a tool, it is the tool icon that is displayed.

  - display or not the title and image of the container in the user interface
    Display Mode
    Default value: "Name and image" for a container of tools and "none" for a container of containers.
  - position of title (top, bottom, left, right)
    Title Position: Top (default value), Bottom, Left, Right
  - type of Container (frame, accordion, tab, portal, vertical box, horizontal box, none)

Container Layout: Border Layout, Accordion, TabPanel, Portal, Vbox, Hbox, (none)

- position relative to parent (desktop or Container in the case of a Container of Border Layout type) (top, bottom, left, right, center)
  Position: Top, Bottom, Left, Right, Center

- style sheet
  CSS:
    - Body CSS Class: additional CSS class for the Container body
    - Css sensitive to language: to make the Container CSS sensitive to the current language, default: false
    - Container CSS Class

- current overloading enables to define a macro that overloads the emitted current.

  This is particularly interesting when you use the component pattern, and you want to display the type information instead of the component itself.

  Current Overloading: macro connection

- **Dimension** (width, height, minimum height, minimum width)

  Dimension attributes depend on Container type:
  Left Container: Width, minimum Width
  Right Container: Width, minimum Width
  Top Container: Height, Minimum Height
  Bottom Container: Height, Minimum Height

  The center Container has no Dimension attribute, it adapts to desktop dimensions depending on other Containers.

- **Portal settings** (for a Portal type Container)

  - Number of columns for the widget list
    Portal – Number of columns (default value: 2)

  - Initialize with default widgets: indicates if the panel must be initialized with the default widgets defined on the Profile. Otherwise, tools must be explicitly added through desktop parameterization, macros or the end-user UI.
    Initialize with default widgets (default value: true)

  - persistent state: indicates if the panel tools list must be saved and restored when the user logs in again.
    Enable persistent state (default value: true)

  - end-user widget selection UI: indicates if the panel should show a UI allowing the end-user to choose widgets to add to the container.
    Enable end-user widget selection UI (default value: true)

  - editable mode
    Disable Editable Mode (default value: false)

  - add a toolbar at the top
    Toolbar at top (default value: false)

- **Behavior attributes**:

  These attributes simplify desktop readability.

  - closable, collapsible, collapsed, hidden, maximizable
    Is Closable, is Collapsible, Is Collapsed, Is Hidden, Is Maximizable (default values: false)
  - default container, container that receives tools that do not have candidate container to receive them
    Is Default Container (default value: false)
  - drop for objects authorized or not in its component(s)
    Accepts Dropped Items (default value: false)
  - current sensitive
    Is not current sensitive (default value: false)
  - resizable
    Is Resizable (default value: false)
  - current emission
    No current Emission (default value: false)
  - refresh on activate
    Refresh On Activate (default value: false)
  - collapse mode, for a collapsible container which is a direct child item of a border layout container
    Collapse Mode:
    - No value (default): when collapsed, a placeholder header is injected into the layout to represent the Container and provide a UI to allow the user to re-expand the Container.
    - Header: the Container collapses to leave its header visible as when not inside a border layout
    - Mini: the Container collapses without a visible header
  - behavior on close: defines the behavior when the container is closed by the cross
    Behavior On Close (hidden, collapsed, closed, default value: none)

- **Available Desktop Container contexts**

  Context Display Mode attribute defines the way the container is displayed in a context (for example at initialization). The context applies on tools and macros: EmitCurrent in a context (floating toolbar, Activity feed)
  Context Display Mode (Opened, Closed)

  ➢ See Defining a context-specific display for a Desktop Container.

- **Desktop components (MEGA Parameterized Tool):**

  MEGA Parameterized Tool attributes enable insertion of already-parameterized tools.

Examples:

- Administration tree
  Administration MetaTree Component
- Collaboration tree
  Collaboration MetaTree Component
- Controls and Risks tree
  Controls and Risks MetaTree Component
- Diagrams tree
  Diagrams MetaTree Component
- Properties page
  Docked PropertyPage Component
- Documentation tree
  Documentation MetaTree Component
- Main toolbar undo/redo tool
  Undo/Redo

For example, desktop components of tree type are implemented by the **MEGA Tool** "MetaTree Tool":

Example: the **MEGA Parameterized Tool** "Documentation MetaTree Component" is implemented by the **MEGA Tool** "MetaTree Tool".

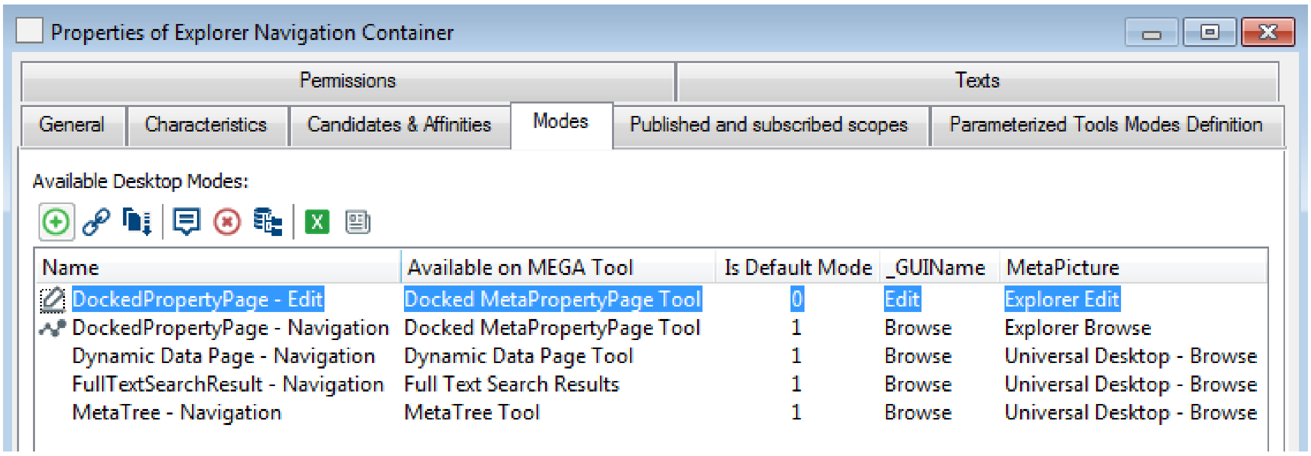
3.2.7 Container specific behaviors (Mode)

For each **Mega Parameterized Tool** included in a container you can define either:

- a specific behavior according to a mode.

  The **Mega Parameterized Tool** specific behavior is defined in the container **Properties** page, **Modes** tab. For each tool for which you want to define a specific behavior in the Container, the Mode parameter enables to overload the default behavior.
  The mode contains specific settings dedicated to specific tools that enable to define specific behaviors.

  For example, in the HOPEX Explorer desktop, where the browse mode is the default the "Explorer Navigation Container" includes the "Docked MetaPropertyPage Tool" which is editable in Edit mode and not editable in Browse mode.


Properties of Explorer Navigation Container
Permissions
Texts
General
Characteristics
Candidates & Affinities
Modes
Published and subscribed scopes
Parameterized Tools Modes Definition
Available Desktop Modes:

Name	Available on MEGA Tool	Is Default Mode	_GUIName	MetaPicture
DockedPropertyPage - Edit	Docked MetaPropertyPage Tool	0	Edit	Explorer Edit
DockedPropertyPage - Navigation	Docked MetaPropertyPage Tool	1	Browse	Explorer Browse
Dynamic Data Page - Navigation	Dynamic Data Page Tool	1	Browse	Universal Desktop - Browse
FullTextSearchResult - Navigation	Full Text Search Results	1	Browse	Universal Desktop - Browse
MetaTree - Navigation	MetaTree Tool	1	Browse	Universal Desktop - Browse

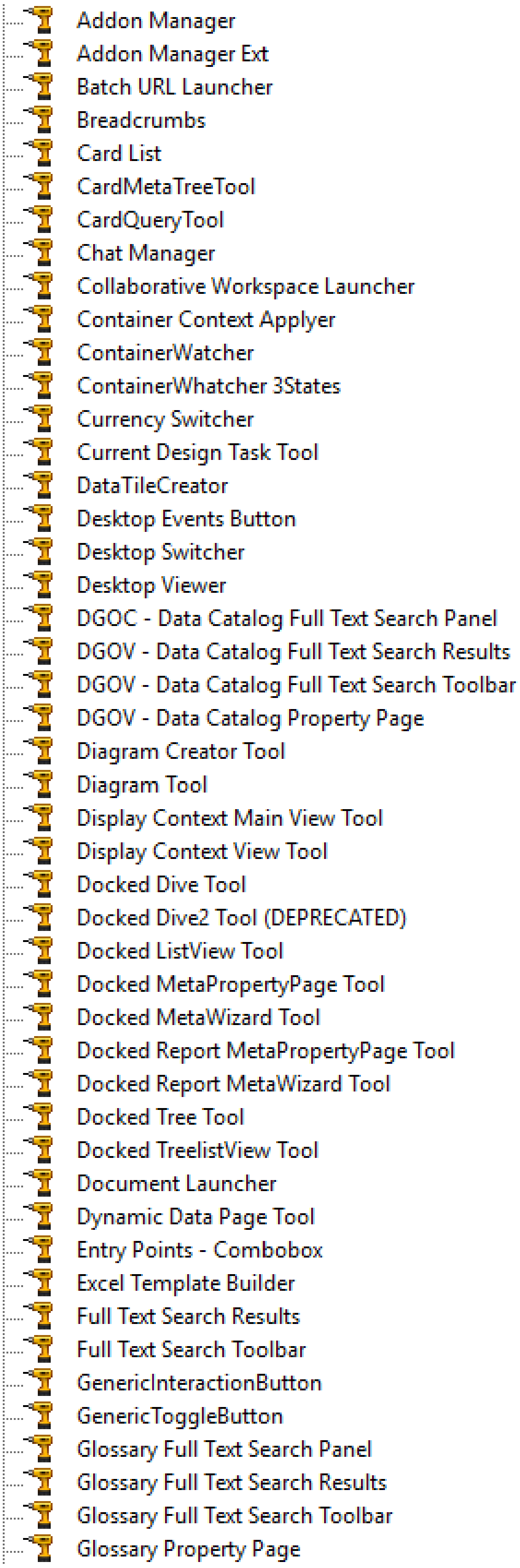
- a standard behavior available for any mode

  The other tools for which you do not want a specific behavior according to a mode.

3.3 Tools

Tools (**MEGA Tools**) enable viewing and/or integration with repository data. A MEGA Tool cannot be used directly in a desktop, but only via a desktop component (**MEGA Parameterized Tool**).

3.3.1 Examples of tools



- Addon Manager
- Addon Manager Ext
- Batch URL Launcher
- Breadcrumbs
- Card List
- CardMetaTreeTool
- CardQueryTool
- Chat Manager
- Collaborative Workspace Launcher
- Container Context Applyer
- ContainerWatcher
- ContainerWhatcher 3States
- Currency Switcher
- Current Design Task Tool
- DataTileCreator
- Desktop Events Button
- Desktop Switcher
- Desktop Viewer
- DGOC - Data Catalog Full Text Search Panel
- DGOV - Data Catalog Full Text Search Results
- DGOV - Data Catalog Full Text Search Toolbar
- DGOV - Data Catalog Property Page
- Diagram Creator Tool
- Diagram Tool
- Display Context Main View Tool
- Display Context View Tool
- Docked Dive Tool
- Docked Dive2 Tool (DEPRECATED)
- Docked ListView Tool
- Docked MetaPropertyPage Tool
- Docked MetaWizard Tool
- Docked Report MetaPropertyPage Tool
- Docked Report MetaWizard Tool
- Docked Tree Tool
- Docked TreelistView Tool
- Document Launcher
- Dynamic Data Page Tool
- Entry Points - Combobox
- Excel Template Builder
- Full Text Search Results
- Full Text Search Toolbar
- GenericInteractionButton
- GenericToggleButton
- Glossary Full Text Search Panel
- Glossary Full Text Search Results
- Glossary Full Text Search Toolbar
- Glossary Property Page

3.3.2 Characteristic attributes of tools

- **Asynchronous**: the tool can be loaded in parallel on the desktop. There is no need to wait for the desktop to be loaded to load the tool (example: ToolbarTool)

  Is Asynchronous

- **Batch Tool**: the tool executes without user interface, indicating whether display should be managed or not.

  Is Batch Tool

3.3.3 Use examples

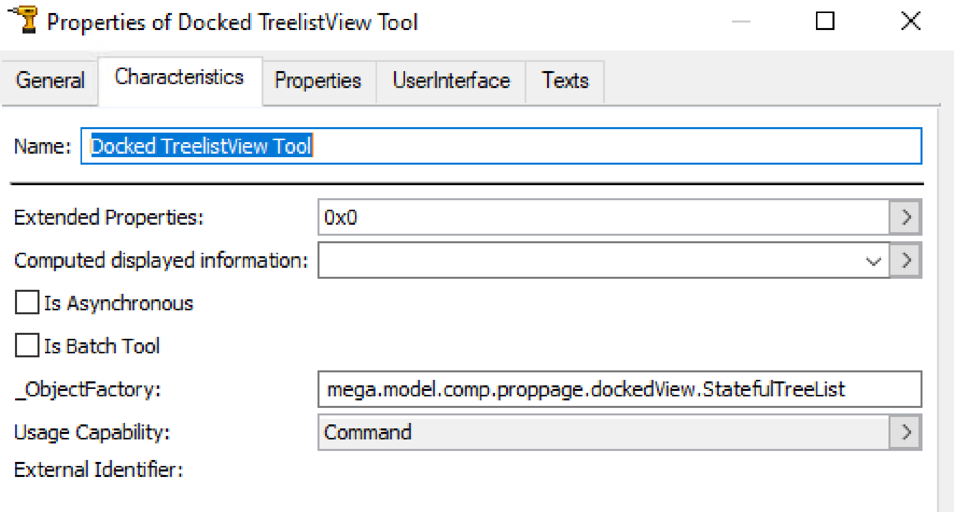
For examples of use of the **MetaTree Tool** tool, see Giving a title and/or icon dynamically to a Container.

3.3.4 Navigation dedicated tools

Navigation dedicated tools are available (to replace **Docked MetaPropertyPage Tool** and Docked Tree Tool). A navigation tool receives, via a property page, a unique component: either a navigation tree or a navigation list (including alternated list):

- **Docked ListView Tool**: displays a list or a set of lists via an alternated list

- **Docked TreeListView Tool:** displays a tree in a tree list format


Properties of Docked TreelistView Tool

Name: Docked TreelistView Tool

Extended Properties: 0x0

Computed displayed information:

□ Is Asynchronous

□ Is Batch Tool

_ObjectFactory: mega.model.comp.proppage.dockedView.StatefulTreeList

Usage Capability: Command

External Identifier:

These tools include the same parameters as the docked Properties panel, but their context is kept in the breadcrumb, and they also benefit from the floating toolbar.

Use **Docked ListView Tool / Docked TreeListView Tool** instead of **Docked MetaPropertyPage Tool** so that:

- after filtering a simple list, when navigating back to the list, the list is still filtered.

- after expanding a tree, when navigating back to the tree, the tree is already loaded.

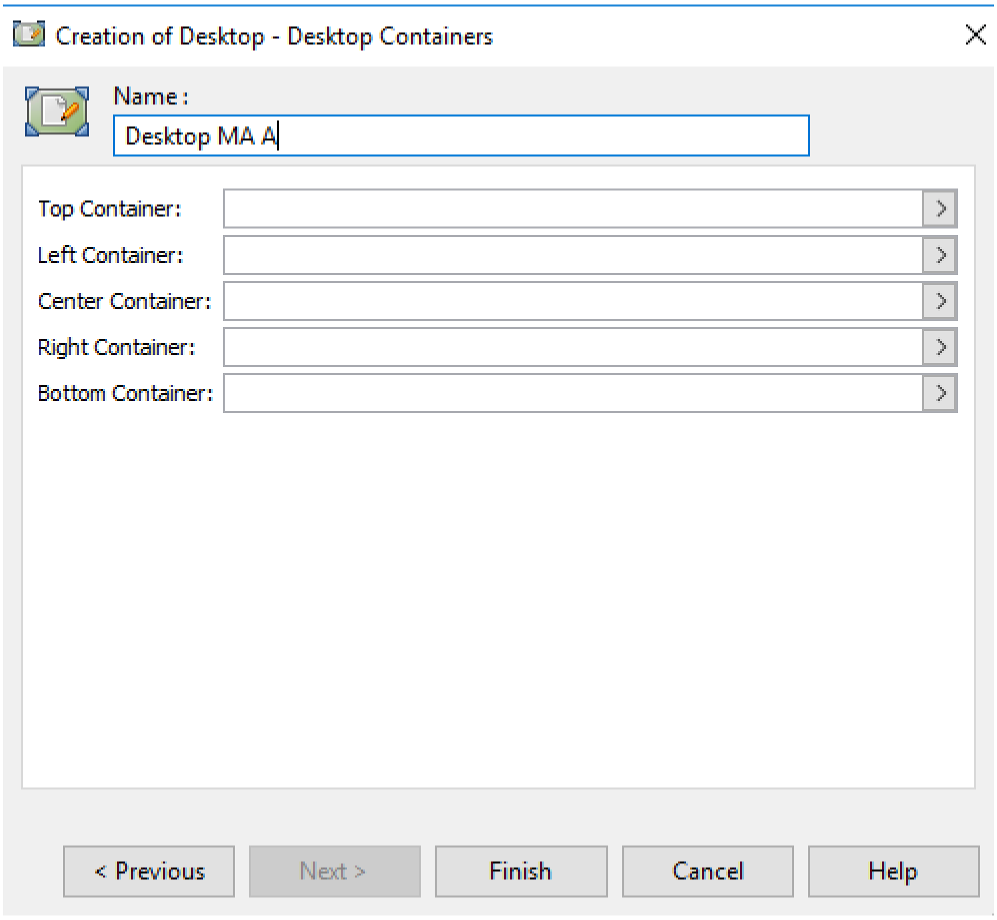
4 Creating a desktop

You can create a desktop from scratch or from a model already created.

4.1 Creating a desktop from scratch

To create a desktop from scratch:

1. Connect to **HOPEX** with the **HOPEX Customizer** profile.
2. Display the **MetaStudio** Navigation window.
3. Expand **Desktops > Other Desktops** folders.
4. Right-click the required folder, then select **New > Desktop**.
   - ➢ The **Creation of Desktop** dialog box appears.
5. Enter the **Name** of the desktop.
   `Example: "Desktop MA A".`
6. Click **Next**.



Creation of Desktop - Desktop Containers

Name : Desktop MA A

7. Depending on the desktop configuration you want to create (see Desktop configuration) you must specify the required fields:

   - **Center Container** (Mandatory)
   - **Top Container**
   - **Left Container**
   - **Right Container**
   - **Bottom Container**

   To specify these fields, see the corresponding procedures, for example see:

   - Creating a Desktop Container of Border Layout type
   - Creating a Desktop Container of Accordion type
   - Creating a Desktop Container of TabPanel type
   - Creating a Desktop Container of Tool type

8. Click **Finish**.

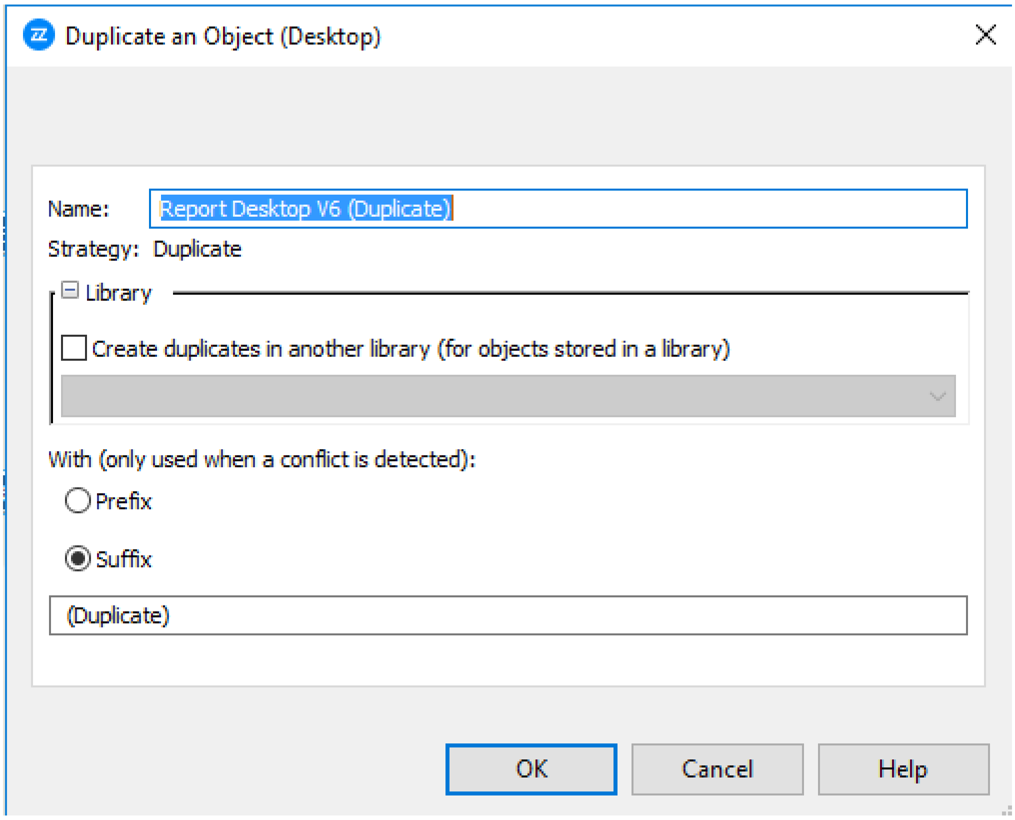
4.2 Creating a desktop from a model

You can create your desktop starting from a model. This method allows you to start from a structure already set up and simplifies your work. To do this you must duplicate an existing model.

To create a desktop starting from a model:

1. Connect to **HOPEX** with the **HOPEX Customizer** profile.
2. Display the **MetaStudio** Navigation window.
3. Expand the Desktop folder that contains the desktop you want to use as a model.
4. Right-click the desktop name (example: Report Desktop V6) and select **Manage > Duplicate**.

   ➢ The **Duplicate an Object (Desktop)** dialog box appears.



Duplicate an Object (Desktop)

Name: Report Desktop V6 (Duplicate)

Strategy: Duplicate

⊟ Library

□ Create duplicates in another library (for objects stored in a library)

With (only used when a conflict is detected):

○ Prefix

● Suffix

(Duplicate)

5. By default, the name of the duplicated desktop is: <Name of desktop> (Duplicate). If you want to change the format of the duplicated desktop name to: Duplication of <Name of desktop>, select **Prefix**.

> **Warning**: If you then modify the name of the desktop in the **Name** field, the trace of duplication is no longer visible.

6. Click **OK**.
   ➢ The duplicated desktop is displayed in the application of the model desktop.

7. Select the duplicated desktop (example: Report Desktop V6 (Duplicate)) and drag-and-drop it to your folder.

8. (Optional) You can rename the desktop. To do this, select the name of the desktop and press key F2, or from the desktop pages select the **Characteristics** tab.

9. You can rename, move, modify and/or delete Containers from your desktop. This does not modify the model desktop.

> **Warning**: If you modify the name of a tool or a command, the name of this tool or command is also modified in the model desktop.

➢ To modify the desktop, see Modifying a desktop.

5 CREATING DESKTOP CONTAINERS

Prerequisites	Before creating Containers and desktop components, you must: • have already defined desktop components, see Desktop component elements • have already created desktop structure, see Creating a desktop

A **Container** can be represented in the following forms:

- borders (**Border Layout**), see Creating a Desktop Container of Border Layout type.
- accordion (**Accordion**), see Creating a Desktop Container of Accordion type.
- tabs (**TabPanel**), see Creating a Desktop Container of TabPanel type.
- tool (**Container of Tool**), see Creating a Desktop Container of Tool type.

A **Container** can contain:

- a group of tools, see Creating a toolbar.
- a group of Commands, see Creating pop-up menus.

5.1 Creating a Desktop Container of Border Layout type

The **Desktop Container**, is a container of **Border Layout** type for the desktop. It comprises:

- a **Center Container** (mandatory container)
- **Top**, **Left**, **Right**, and **Bottom Containers** (optional).

For example, the Desktop Container of **Border Layout** type can contain:

- a **Center Container**, which can contain the home page, see Creating the Center Container of a Border Layout Container.
- a **Bottom Container**, which can contain the Properties page, see Creating the Bottom Container of a Border Layout Container.

Creation of this Desktop Container of Border Layout type calls three creation wizards:

- **Desktop Container** creation wizard ("Edit Area")
- **Center Container** creation wizard ("Main Page") of **Desktop Container** ("Edit Area")

- **Bottom Container** creation wizard ("Properties Page") of **Desktop Container** ("Edit Area").

5.1.1 Creating the Center Container of a Border Layout Container

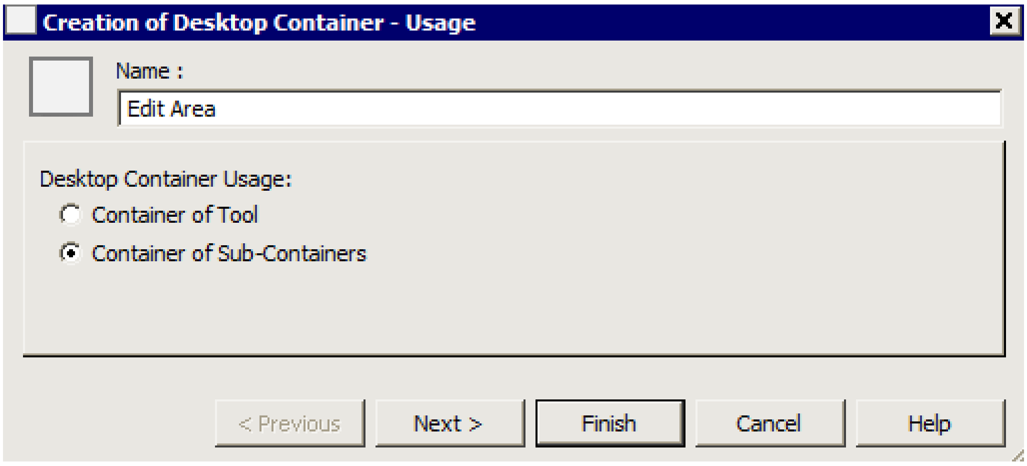
To create the Center Container of a Desktop Container of Border Layout type:

1. From the **Creation of Desktop – Desktop Containers** dialog box (see Creating a desktop), click the **Center Container** field arrow and select **New**.

   ➢ The first step of the **Creation of Desktop Container** wizard appears: **Usage**.

2. Enter the **Name** of the Container of Border Layout type (example: «Edit Area").

3. Select **Container of Sub-Containers**.



Creation of Desktop Container - Usage

Name : Edit Area

Desktop Container Usage:

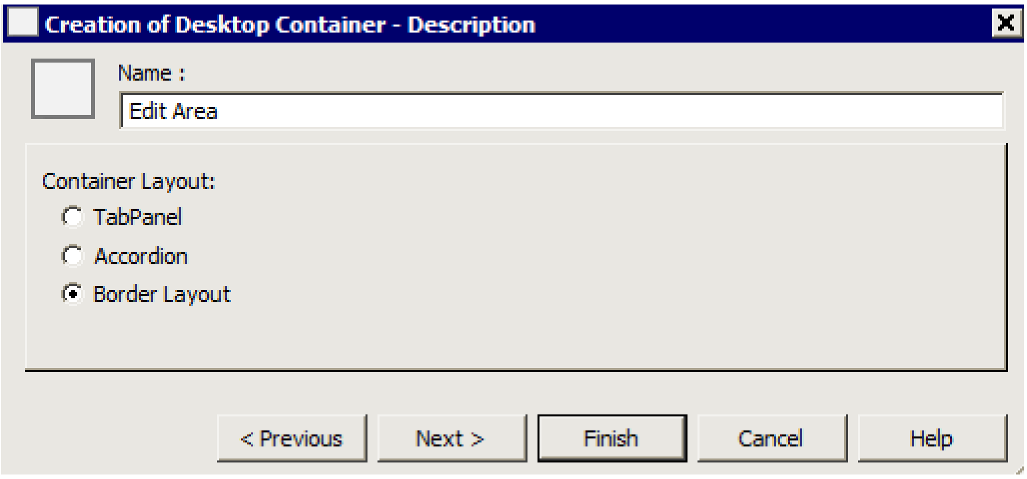
○ Container of Tool

● Container of Sub-Containers

4. Click **Next**.

   ➢ The second step of the **Creation of Desktop Container** wizard appears: **Description**.

5. In the **Container Layout** frame, select the Container layout type, example: **Border Layout**.



Creation of Desktop Container - Description

Name : Edit Area

Container Layout:

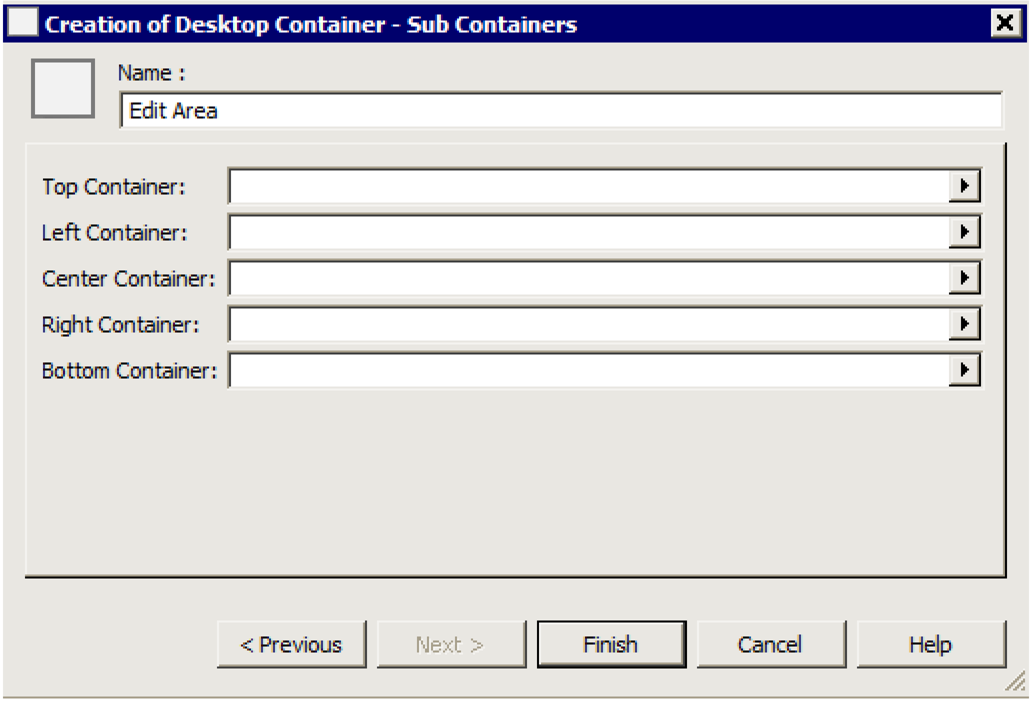
○ TabPanel

○ Accordion

● Border Layout

6. Click **Next**.

   ➢ The third step of the **Creation of Desktop Container** wizard appears: **Sub Containers**.



Creation of Desktop Container - Sub Containers

Name : Edit Area

Top Container:

Left Container:

Center Container:

Right Container:

Bottom Container:

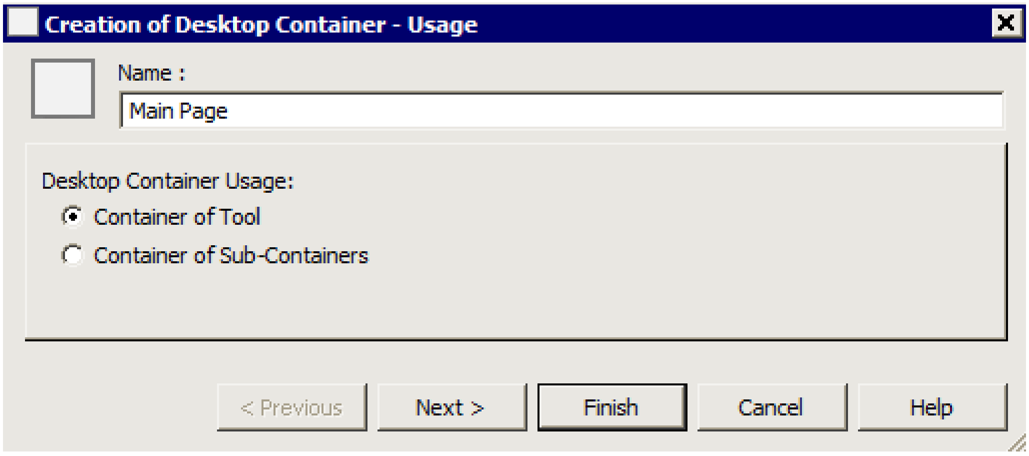
7. Click the **Center Container** field arrow and select **New**.

   ➢ The first step of a new **Creation of Desktop Container** wizard appears: **Usage**. This enables definition of use of the Center Container of the "Edit Area" Container.

Depending on the layout of the Container you want to create, click arrows of the **Top**, **Left**, **Right** and/or **Bottom Container** fields and select **New**.

8. Enter the **Name** of the container (example: "Main Page").

9. In the **Desktop Container Usage** frame, select **Container of Tool**.



Creation of Desktop Container - Usage

Name : Main Page

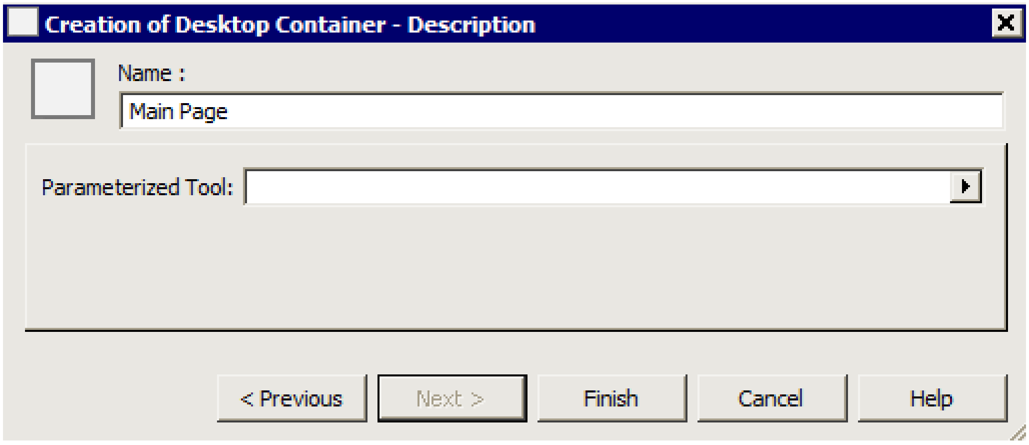
Desktop Container Usage:

● Container of Tool

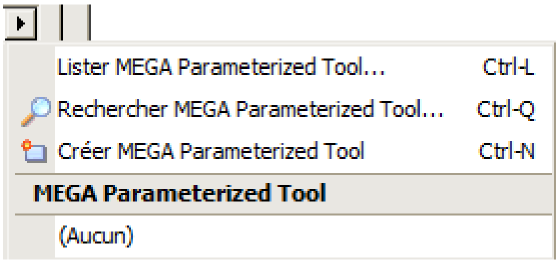
○ Container of Sub-Containers

10. Click **Next**.

- ➢ The second step of the new **Creation of Desktop Container** wizard appears: **Description**. This enables description of the Center Container of the "Edit Area" Container.



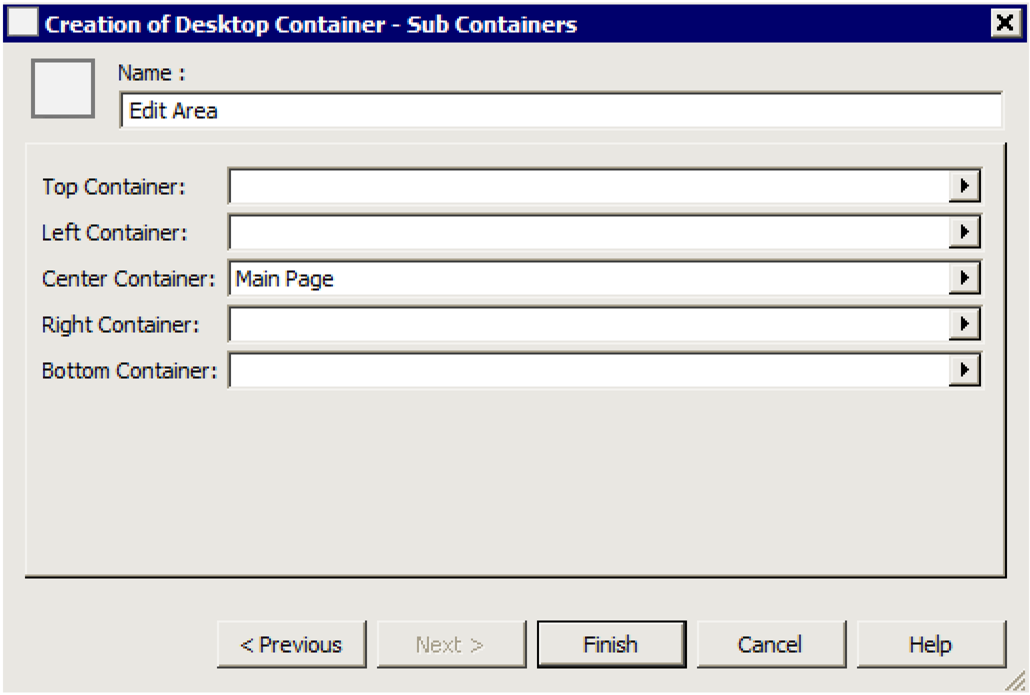
11. In the **Parameterized Tool** field, click the arrow and select or create the tool you require (example: "Widget Component").



Lister MEGA Parameterized Tool... Ctrl-L
Rechercher MEGA Parameterized Tool... Ctrl-Q
Créer MEGA Parameterized Tool Ctrl-N
MEGA Parameterized Tool
(Aucun)

12. Click **Finish**.

- ➢ "Main Page" (which contains the "Widget Component" tool) appears in the **Center Container** field.


Creation of Desktop Container - Sub Containers

Name : Edit Area

Top Container:

Left Container:

Center Container: Main Page

Right Container:

Bottom Container:

13. See Creating the Bottom Container of a Border Layout Container before clicking **Finish**.

   ➢ The Container of the home page is created.

5.1.2 Creating the Bottom Container of a Border Layout Container

To create the Bottom Container of a Desktop Container of Border Layout type:

1. From the **Creation of Desktop Container – Sub Containers** dialog box, click the **Bottom Container** field arrow and select **New**.

   ➢ The first step of a new **Creation of Desktop Container** wizard appears: **Usage**. This enables definition of use of the Bottom Container of the "Edit Area" Container.

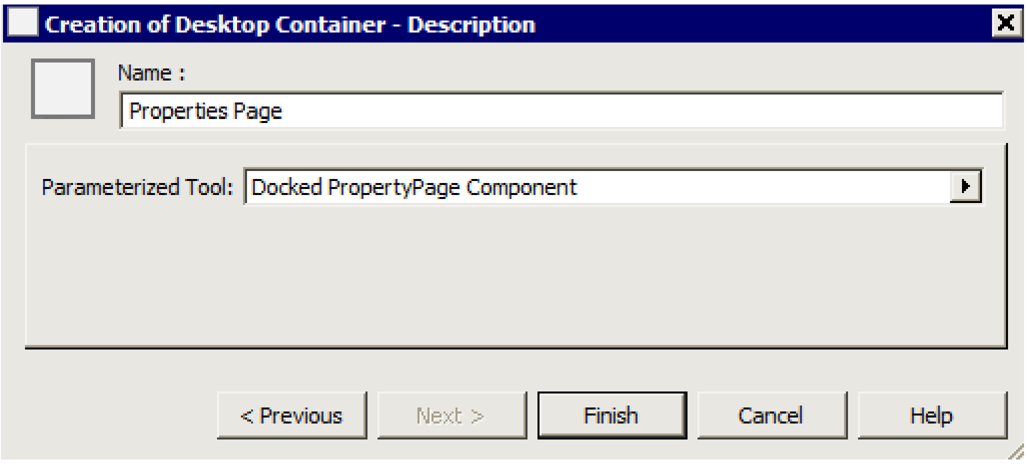
2. Enter the **Name** of the Bottom Container (example: "Properties Page").

3. In the **Creation of Desktop Container** frame, select **Container of Tool**.

4. Click **Next**.

   ➢ The second step of the new **Creation of Desktop Container** wizard appears: **Description**. This enables description of the Bottom Container of the "Edit Area" Container.

5. In the **Parameterized Tool** field, click the arrow and select (or create) the MEGA Parameterized Tool you require (example: **Docked PropertyPage Component**).

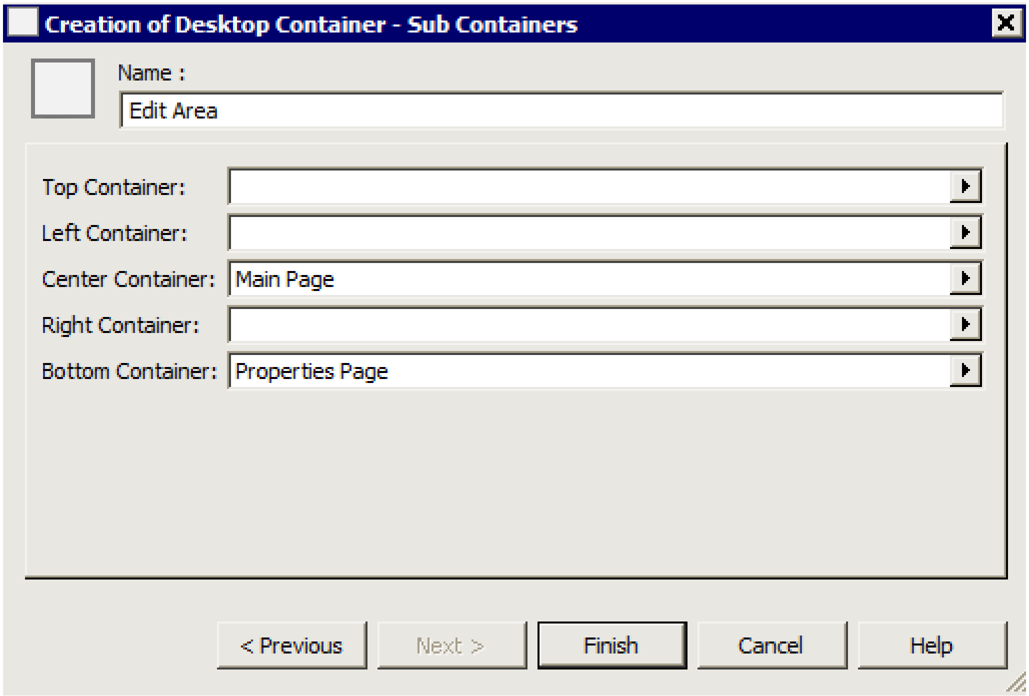

Creation of Desktop Container - Description

Name : Properties Page

Parameterized Tool: Docked PropertyPage Component

6. Click **Finish**.

   ➢ The **Properties of Desktop Container** "Edit Area" dialog box appears.


Creation of Desktop Container - Sub Containers

Name : Edit Area

Top Container:

Left Container:

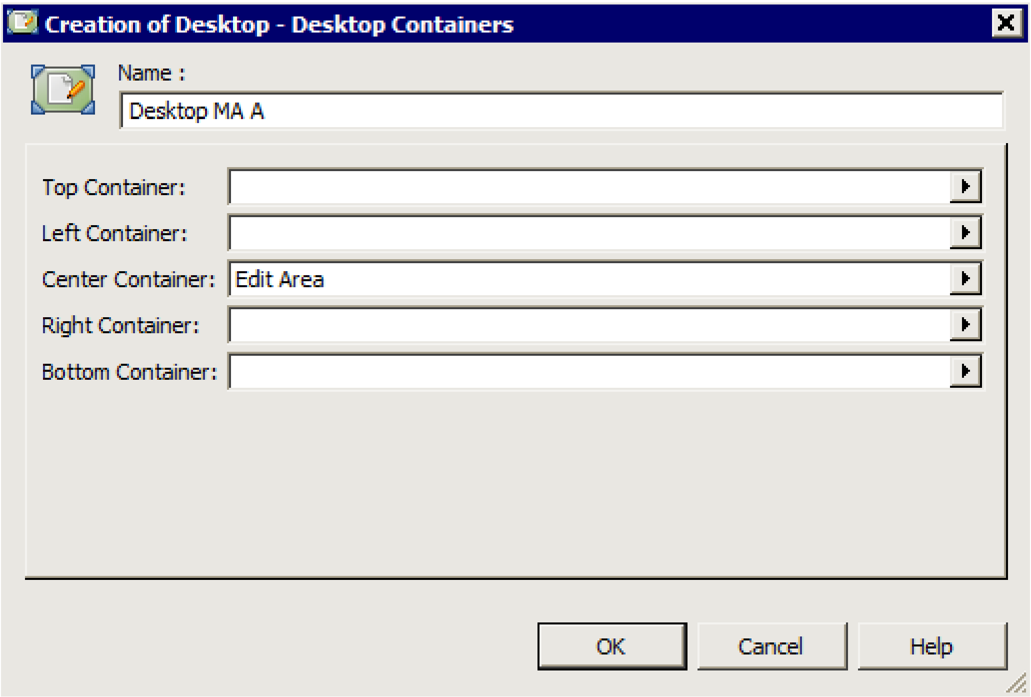
Center Container: Main Page

Right Container:

Bottom Container: Properties Page

   ➢ The two Containers ("Main Page" and "Properties Pages") of the "Edit Area" Center Container are created.

7. Click **Finish**.



Creation of Desktop - Desktop Containers

Name : Desktop MA A

Top Container:

Left Container:

Center Container: Edit Area

Right Container:

Bottom Container:

- The "Edit Area" **Center Container** of Border Layout type of desktop ("Desktop MA A") is created and configured.

When the **Center Container** has been specified, you can interrupt creation of other Containers of your desktop.
To create a new Container later, see Adding a Desktop Container to a desktop.

5.2 Creating a Desktop Container of Accordion type

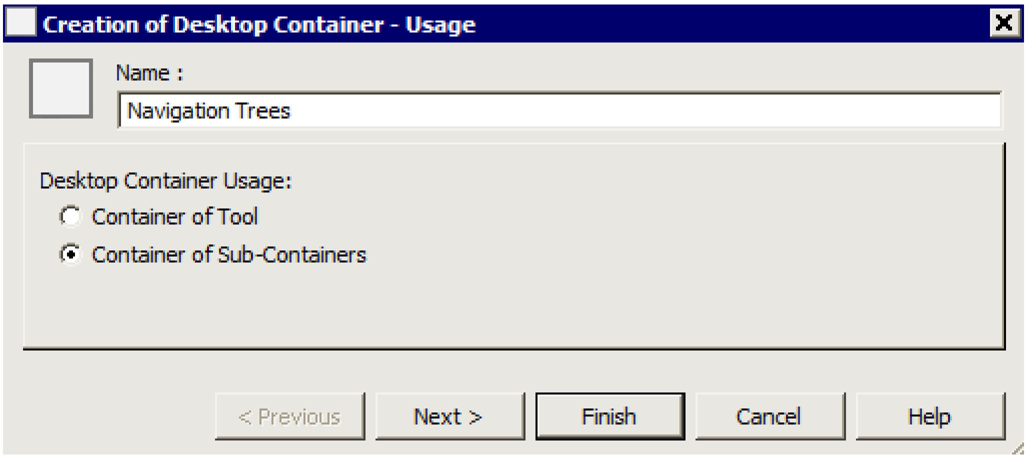
To create a Desktop Container of Accordion type:

1. From the **Creation of Desktop – Desktop Containers** dialog box (see Creating a desktop), click the **Left Container** field arrow and select **New**.

   - The first step of the **Creation of Desktop Container** wizard appears: **Usage**.

2. Enter the **Name** of the Desktop Container of Accordion type (example: "Navigation Trees").

3. Select **Container of Sub-Containers**.


Creation of Desktop Container - Usage

Name : Navigation Trees

Desktop Container Usage:

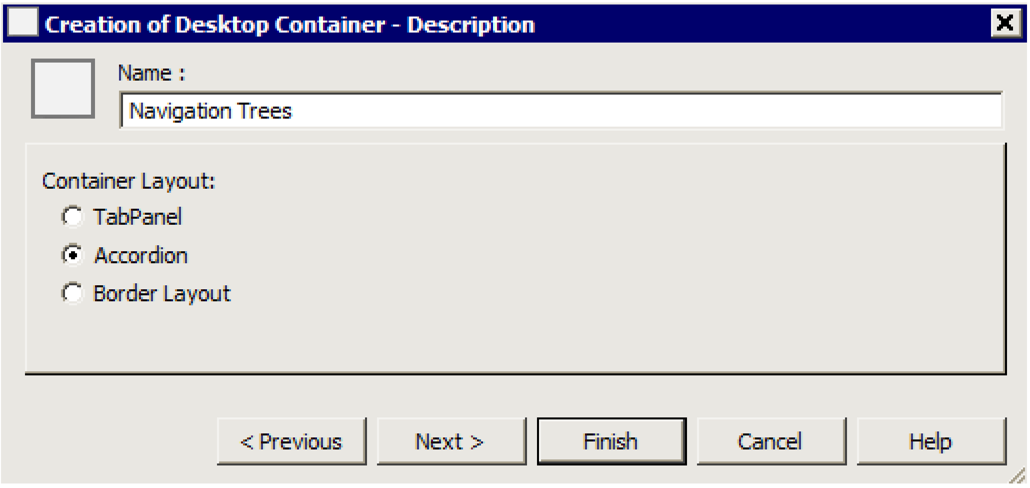
○ Container of Tool

● Container of Sub-Containers

4. Click **Next**.

   ➢ The second step of the **Creation of Desktop Container** wizard appears: **Description**.

5. In the **Container Layout** frame, select the Container layout type: **Accordion**.


Creation of Desktop Container - Description

Name : Navigation Trees

Container Layout:

○ TabPanel

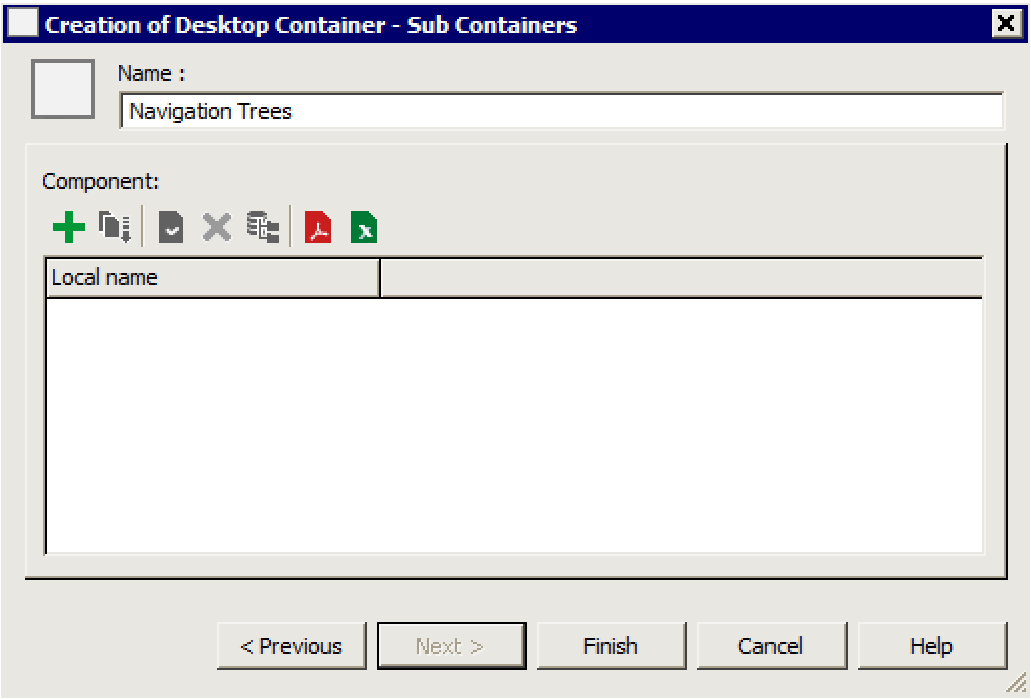
● Accordion

○ Border Layout

6. Click **Next**, to specify content of the **Accordion** Container ("Navigation Trees").

   Click **Finish**, if you want to postpone till later the specification of the content of the **Accordion** Container ("Navigation Trees").

   ➢ The third step of the **Creation of Desktop Container** wizard appears: **Sub Containers**.


Creation of Desktop Container - Sub Containers

Name : Navigation Trees

Component:

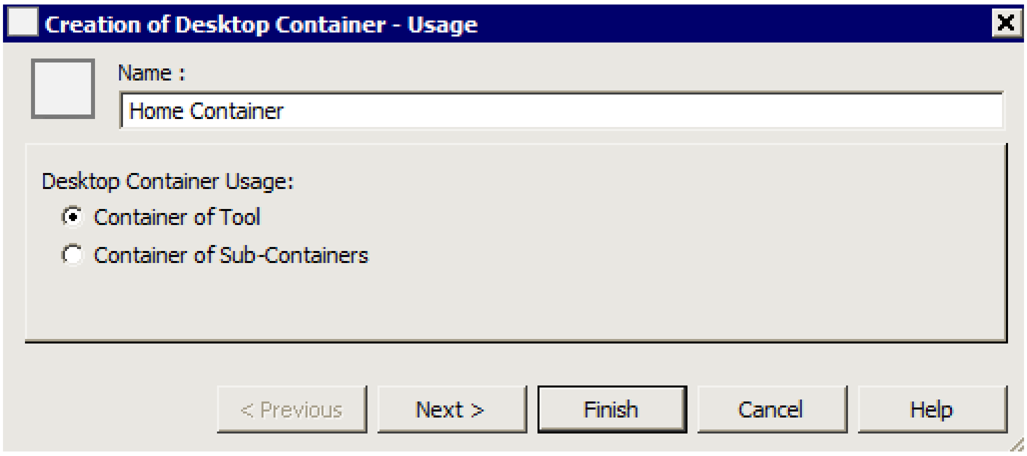
Local name	

7. In the **Component** frame, click **New** ✚**.**

   ➢ The first step of a new **Creation of Desktop Container** wizard appears: **Usage**. This enables definition of use of the first Container of the "Navigation Trees" Container.

8. Enter the **Name** of the Container (example: "Home Container").

9. Select for example **Container of Tool**.


Creation of Desktop Container - Usage

Name : Home Container

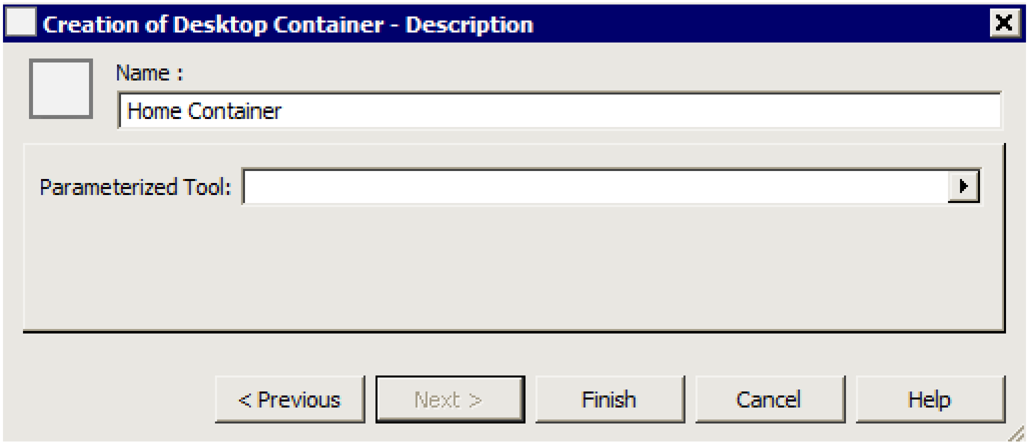
Desktop Container Usage:

● Container of Tool

○ Container of Sub-Containers

10. Click **Next** (or Finish if you want to complete later).

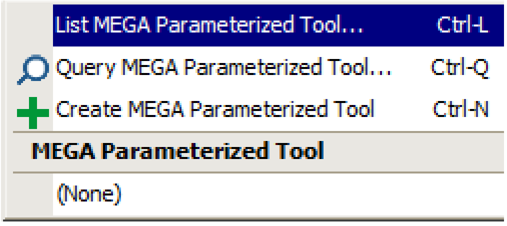
   ➢ The second step of the new **Creation of Desktop Container** wizard appears: **Description**. This enables description of the first Container of the "Navigation Trees" Container.



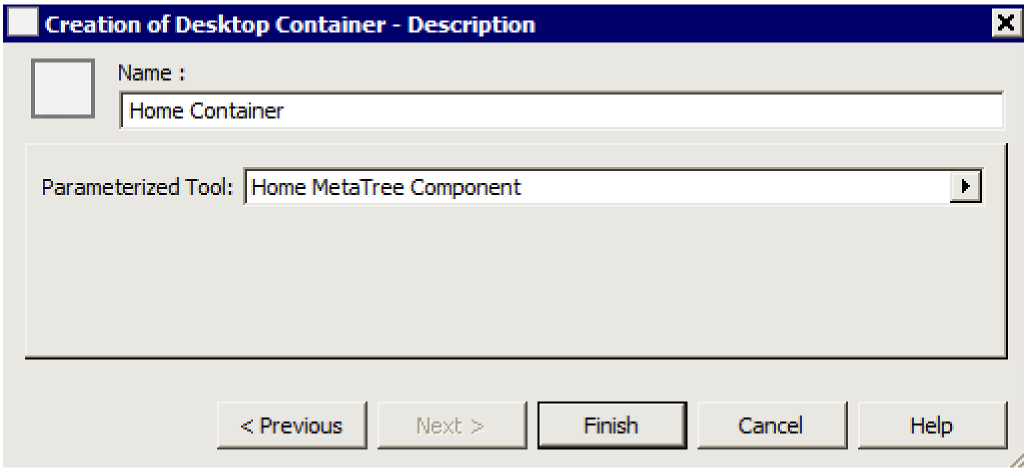
Name :
Home Container

Parameterized Tool:

11. In the **Parameterized Tool** field, click the arrow and select or create the tool you require (example: "Home MetaTree Component").


List MEGA Parameterized Tool... Ctrl-L
Query MEGA Parameterized Tool... Ctrl-Q
Create MEGA Parameterized Tool Ctrl-N
MEGA Parameterized Tool
(None)

12. Click **OK**.

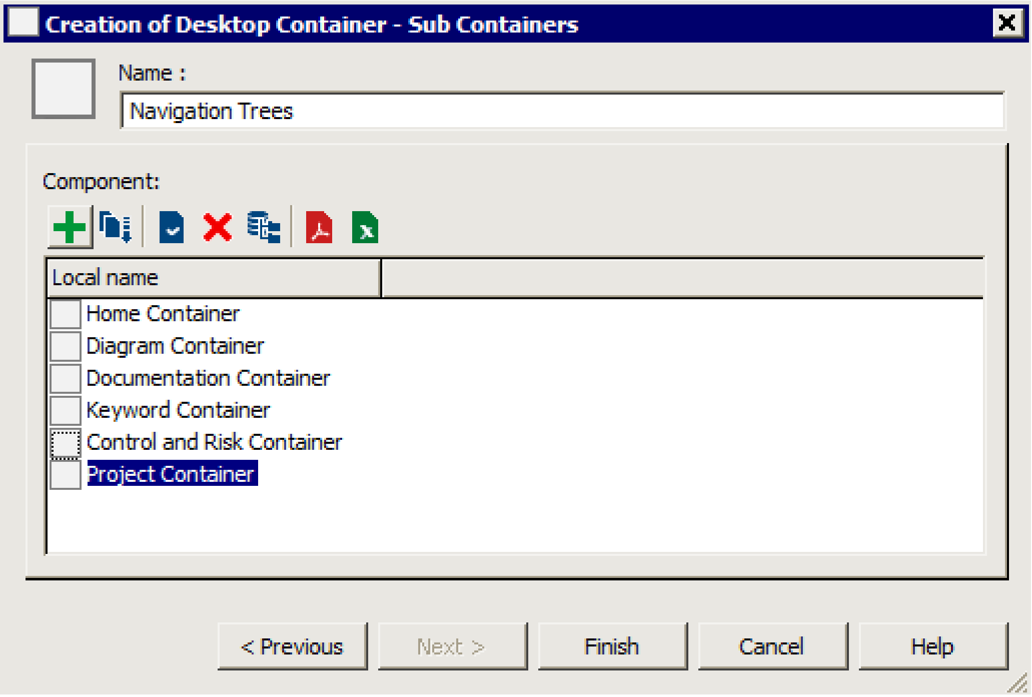


Name :
Home Container

Parameterized Tool: Home MetaTree Component

13. Click **Finish**.

14. Repeat steps 7 to 13 and create other Containers of the "Navigation Trees" Container.


Creation of Desktop Container - Sub Containers

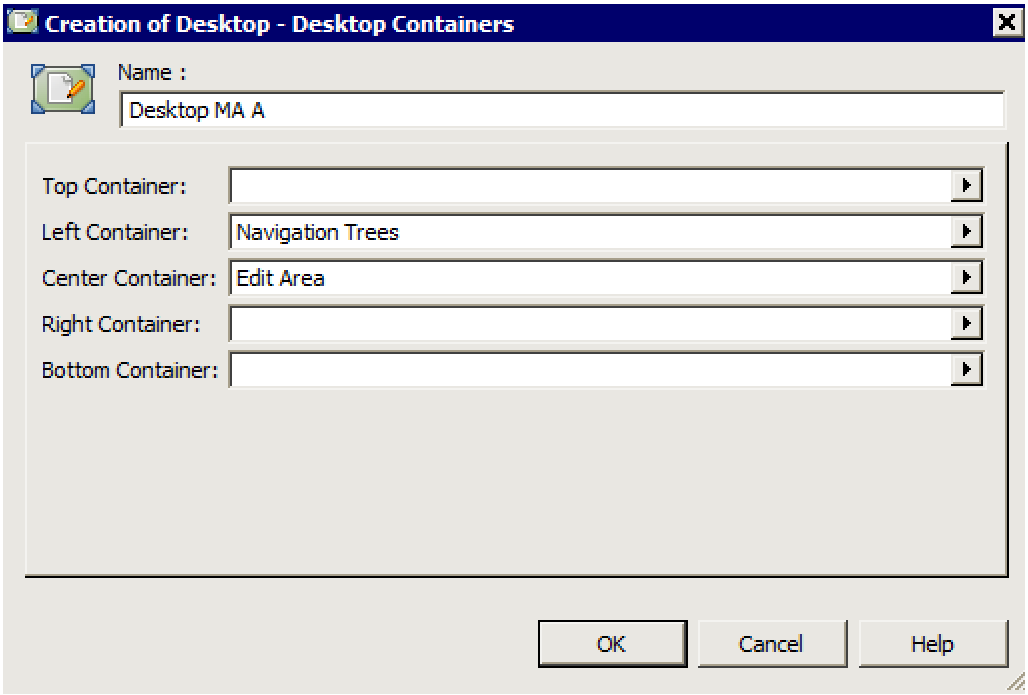
Name : Navigation Trees

Component:

Local name	
□ Home Container	
□ Diagram Container	
□ Documentation Container	
□ Keyword Container	
□ Control and Risk Container	
□ Project Container	

15. Click **Finish**.

➢ "Navigation Trees" (which contains the trees created) appears in the **Left Container** field of the desktop ("Desktop MA A").


Creation of Desktop - Desktop Containers

Name : Desktop MA A

Top Container:

Left Container: Navigation Trees

Center Container: Edit Area

Right Container:

Bottom Container:

5.3 Creating a Desktop Container of TabPanel type

To create a Container that contains tabs that are always visible, you must create a **Desktop Container** of **TabPanel** type.

To create tabs on the fly, see Defining Container candidates.

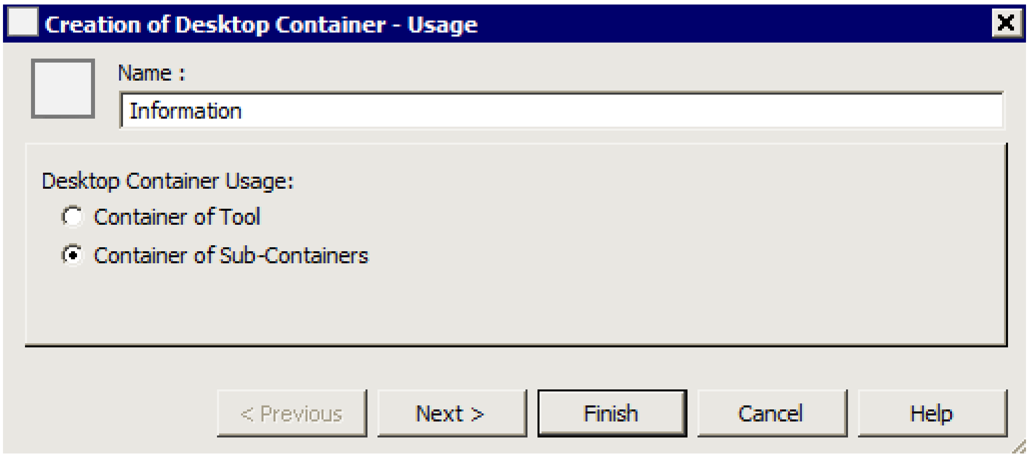
To create a Desktop Container of TabPanel type:

1. From the **Creation of Desktop – Desktop Container** dialog box (see Creating a desktop), in the **Bottom** (**Top**, **Left**, **Center** or **Right**) **Container** field, click the arrow and select **New**.

   ➢ The first step of the **Creation of Desktop Container** wizard appears: **Usage**.

2. Enter the **Name** of the Container (example: "Information").

3. In the **Desktop Container Usage** field, select **Container of Sub-Containers**.



Creation of Desktop Container - Usage

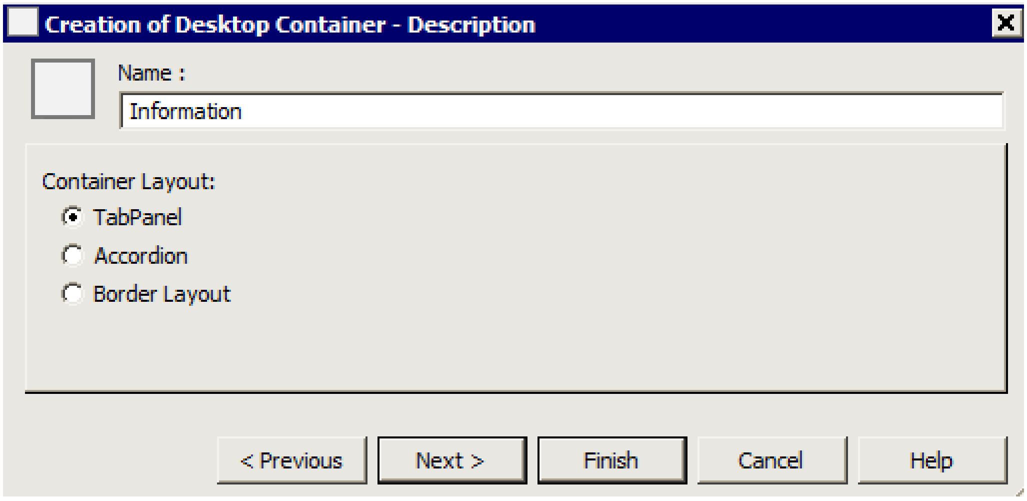
Name : Information

Desktop Container Usage:
○ Container of Tool
● Container of Sub-Containers

4. Click **Next**.

   ➢ The second step of the **Creation of Desktop Container** wizard appears: **Description**.

5. In the **Container Layout** frame, select the Container layout type: **TabPanel**.



Creation of Desktop Container - Description

Name : Information

Container Layout:

● TabPanel

○ Accordion

○ Border Layout

6. Click **Next**.

   ➢ The third step of the **Creation of Desktop Container** wizard appears**: Sub Containers**.

7. In the **Component** frame, click **New** + to insert the tabs you require.

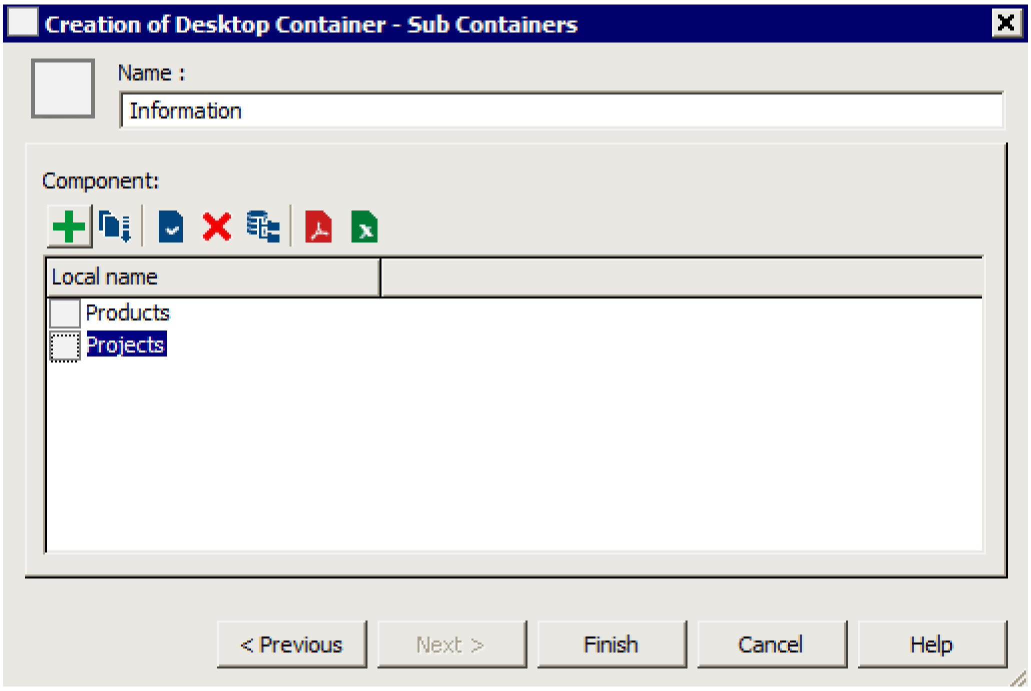
   ➢ The first step of a new **Creation of Desktop Container** wizard appears: **Usage**. This enables definition of use of the first tab of the "Information" Container.

8. Enter the **Name** of the tab (example: Products).

9. In the **Desktop Container Usage** frame, select **Container of Tool**.

10. Click **Finish** (you can create tools later).

11. Repeat steps 7 to 10 as many times as there are tabs to define.



Creation of Desktop Container - Sub Containers

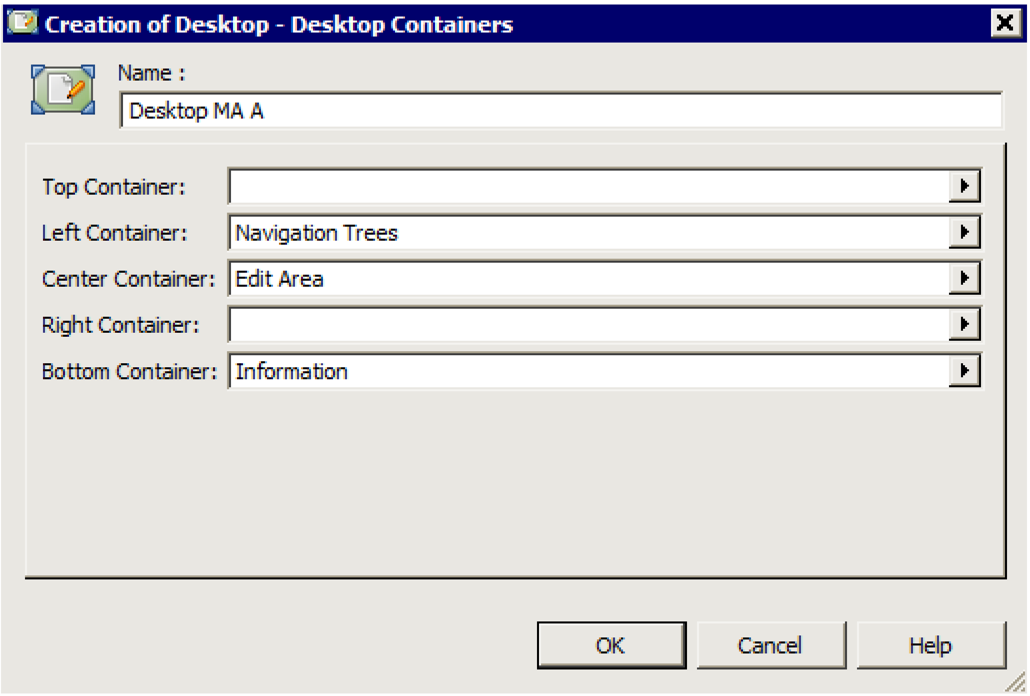
Name : Information

Component:

Local name	
Products	
Projects	

12.Click **Finish**.

- "Information" (which contains tabs "Products" and "Projects") appears in the **Bottom Container** field of the desktop ("Desktop MA A").



Creation of Desktop - Desktop Containers

Name : Desktop MA A

Top Container:

Left Container: Navigation Trees

Center Container: Edit Area

Right Container:

Bottom Container: Information

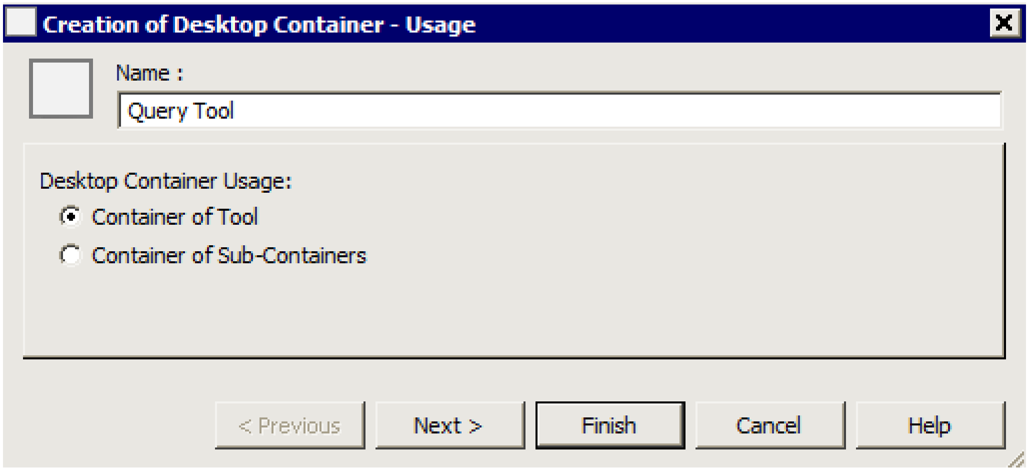
5.4 Creating a Desktop Container of Tool type

To create a Desktop Container of Tool type:

1. From the **Creation of Desktop – Desktop Containers** dialog box (see Creating a desktop), in the **Right Container** (**Top**, **Left**, **Center** or **Bottom**) field, click the arrow and select **New**.

   ➢ The first step of the **Creation of Desktop Container** wizard appears: **Usage**.

2. Enter the **Name** of the Container (example: "Query Tool").



Creation of Desktop Container - Usage

Name : Query Tool

Desktop Container Usage:

● Container of Tool

○ Container of Sub-Containers

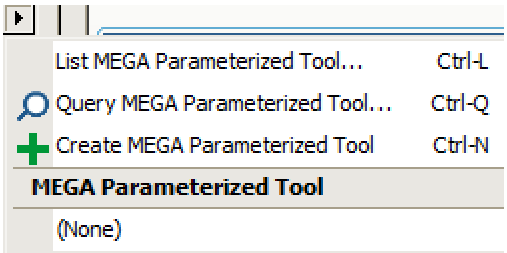
3. In the **Desktop Container Usage** frame, select **Container of Tool**.

4. Click **Next**.

   ➢ The second step of the **Creation of Desktop Container** wizard appears: **Description**.

8. Click the **Parameterized Tool** field arrow, select **List MEGA Parameterized Tool** and select the tool you require (example: "Query Component").

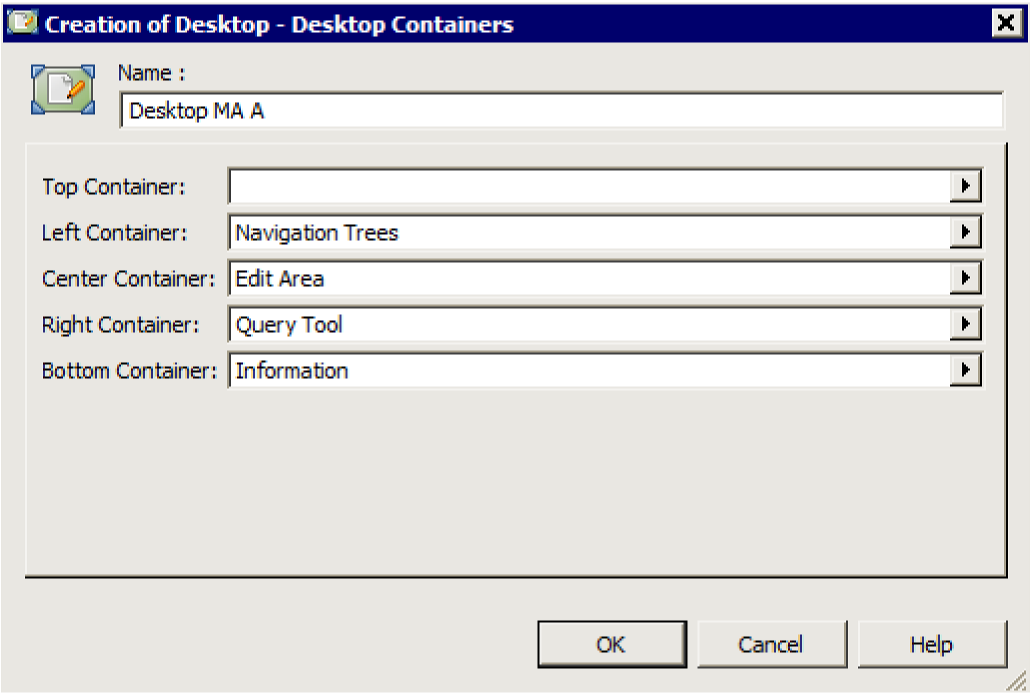
> Alternatively: if you know the name of the tool, select **Query MEGA Parameterized Tool**; if the Tool you want to insert does not exist, select **Create MEGA Parameterized Tool**.



List MEGA Parameterized Tool... Ctrl-L
Query MEGA Parameterized Tool... Ctrl-Q
Create MEGA Parameterized Tool Ctrl-N
MEGA Parameterized Tool
(None)

9. Click **Finish**.

   ➢ The name of the tool (example: "Query Tool") appears in the field concerned (example: **Right Container**).



Creation of Desktop - Desktop Containers

Name : Desktop MA A

Top Container:

Left Container: Navigation Trees

Center Container: Edit Area

Right Container: Query Tool

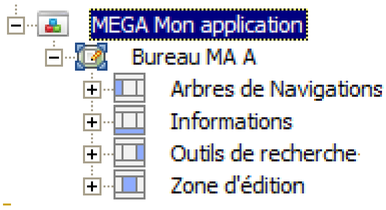
Bottom Container: Information

5.5 Completing Desktop Container creation

When Desktop Containers have been specified, to complete desktop creation:

1. From the **Creation of Desktop – Desktop Containers** dialog box (see Creating a desktop), click **OK** (or **Finish**).

   ➢ The desktop and its Containers appear in the tree of your application.



MEGA Mon application
- Bureau MA A
  - Arbres de Navigations
  - Informations
  - Outils de recherche
  - Zone d'édition

When the **Center Container** has been specified, you can interrupt creation of the other Containers (optional) of your desktop.
To create a new Container, see Adding a Desktop Container to a desktop.

6 DEFINING CONTAINER CHARACTERISTICS

6.1 Customizing Containers

Once you created a Container, you can customize its display and size in the workspace. To do this, you must specify its characteristics (see Container attributes) in its Properties pages:

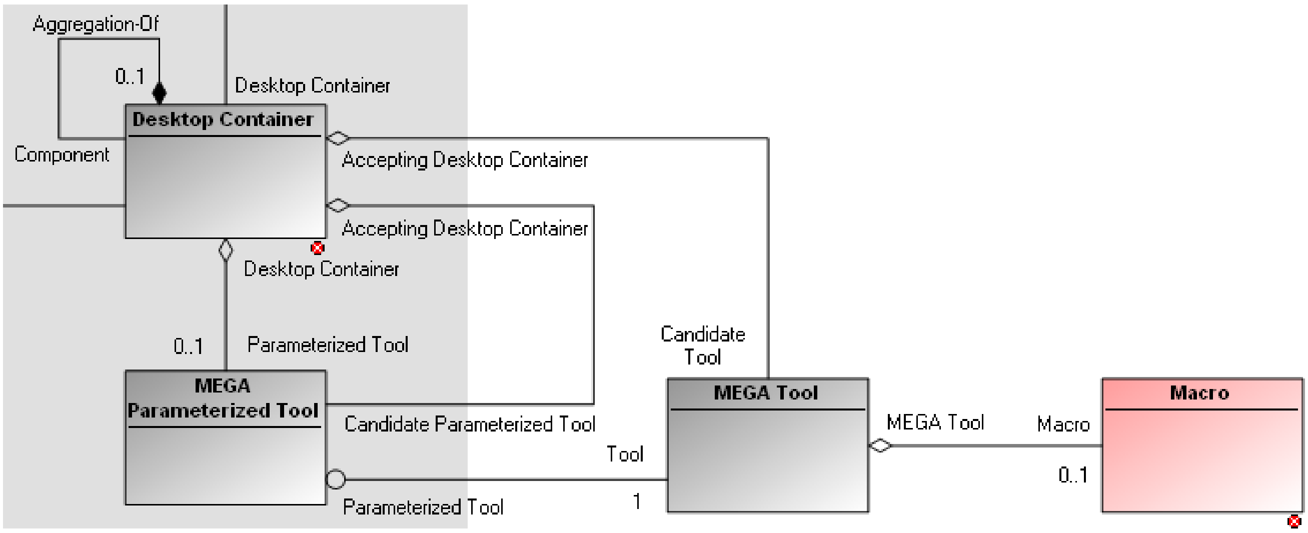
1. Open the Container **Properties**.
2. Select the **Characteristics** tab.
3. In the **Presentation** section, you can for example specify the name of the Container in the user interface (_GUIName) and its corresponding icon (MetaPicture).

⚠ _GUIName and MetaPicture are only specified in the case of a Container of Containers. If the Container contains a tool, then the _GUIName and MetaPicture of the tool are displayed.

Note: Alternatively, the title and icon of the Container can be specified dynamically in the user interface by means of a macro, see Giving a title and/or icon dynamically to a Container.

6.2 Giving a title and/or icon dynamically to a Container

To give a title and/or icon dynamically to a Container, you must create a macro, which you will connect to the **MEGA Tool** of the Container. It is the **MEGA Tool** (example: "MetaTree Tool") that gives the title (GUI name) and icon to the Container.



To give the title and icon of a Container dynamically in the user interface (example: "Home Container"):

1. In the **MEGA Desktop** workspace, execute a **Query** on the **MEGA Tool** concerned (example: "MetaTree Tool").

   Alternatively, if you do not know the name of the **MEGA Tool**:

   a. In the **MEGA Desktop** workspace, display the **MetaStudio** navigation window.

   b. Expand the **MEGA Application** folder, then the application and desktop concerned.

   c. Expand the **Desktop Container** (example: "Navigation Trees"), then the **Desktop Container Component** (example: "Home Container" of) then the **MEGA Parameterized Tool** (example: "Home MetaTree Component") concerned.

2. Right-click the **MEGA Tool** (example: "MetaTree Tool") concerned and select **Explore**.

3. From the exploration tree of the **MEGA Tool** (example: "MetaTreeTool”), right-click **Macro** and select New.

4. Edit the macro and specify in the script:

   - the function that will specify the title:

     Function GetTitle (mgRoot As MegaRoot, ParameterizedToolId as Object)
     …/…
     End Function

   Example of macro implemented for the **MEGA Tool** "MetaTree Tool":

```
Function GetTitle(mgRoot As MegaRoot, ParameterizedToolId as Object)
  Dim mgScanner
  Set mgScanner = New Scanner
  Set mgScanner.mgResource = mgRoot.CurrentEnvironment.Resources
  Set mgScanner.mgToolkit = mgRoot.currentEnvironment().toolkit()

  mgScanner.mgScan = 1
  mgScanner.mgFunction = 1

  mgScanner.mgResource.ScanCollection ParameterizedToolId,
"Abstract Property", mgScanner ,1 , "Reference" & " " &
"MetaAttribute Type" & ":T"
   If mgScanner.mgName = "" Then
      GetTitle = "!!! No GuiName Found"
Else
  GetTitle = mgScanner.mgName
 End If

 End Function
```

   - the function that will call the icon:

```
Function GetPicture(mgRoot As MegaRoot, ParameterizedToolId as Object)
.../...
End Function
```

Example of macro implemented for the **MEGA Tool** "MetaTree Tool":

```
Function  GetPicture(mgRoot  As  MegaRoot,  ParameterizedToolId  as
Object)
Dim mgScanner
  Set mgScanner = New Scanner
  Set mgScanner.mgResource = mgRoot.CurrentEnvironment.Resources
  Set mgScanner.mgToolkit = mgRoot.currentEnvironment().toolkit()

  mgScanner.mgScan = 1
  mgScanner.mgFunction = 2
  mgScanner.mgResource.ScanCollection
ParameterizedToolId,"Abstract Property", mgScanner ,1 ,
"Reference" & " " & "MetaAttribute Type" & ":T"

GetPicture = mgScanner.mgPicture
  If mgScanner.mgPicture = "" Then
      GetPicture = ""
Else
  GetPicture                =                "~"                &
mgRoot.CurrentEnvironment.Toolkit.GetString64FromID(mgScanner.mgPi
cture)
 End If
End Function
```

➢ In the user interface, the title and icon of the Container concerned (example: "Home Container") are specified dynamically. You do not need to specify **_GUIName** and **MetaPicture** in the **Properties** of the Container. When the name of the Container changes, modification is taken into account automatically.

6.3 Defining Container dimensions

You cannot define dimensions of the Center Container, this adapts to the other Containers to fill the space.

By default, the width of a Container (other than the Center Container) is defined as 600 pixels. You can modify this width and/or define a minimum width (**Minimum Width**).

To modify the dimensions of a Container:

1. Open the Container **Properties**.
2. Select the **Characteristics** tab.
3. In the **Dimension** frame, specify the parameters concerned (width, **Minimum Width**).

6.4 Defining Container behavior

Having created a Container, you must define its behavior in the workspace. To do this, you must specify its **Characteristics** (see Container attributes) in its **Properties** pages:

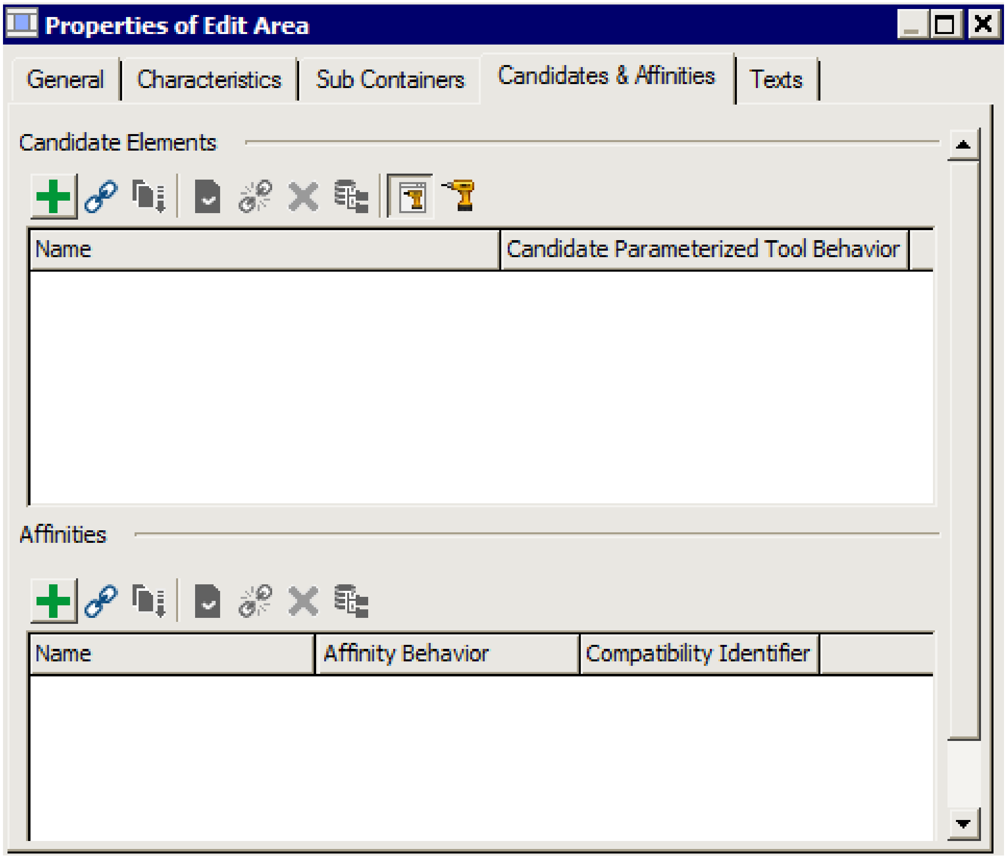
1. Open the Container **Properties**.
2. Select the **Characteristics** tab.
3. In the **Behavior** frame, select the required behaviors.

Note for example that by default a Container is visible and not resizable.

6.5 Defining Container candidates

To define where the **MEGA Tool** or the **MEGA Parameterized Tool** opens when you request its opening (from the pop-up menu of an object for example) you must define this **MEGA Tool** or **MEGA Parameterized Tool** as candidate for a Container.

1. Open the **Properties** of the Container in which you want to see the tool appear (example: "Desktop Container Edit Area").
2. Select the **Candidates & Affinities** tab.



3. In the **Candidate Elements** frame, click the icon of the candidate concerned **Candidate Parameterized Tool** or **Candidate Tool** .

4. Click **Connect** (or **New** if the MEGA Parameterized Tool/Tool concerned is not yet created).

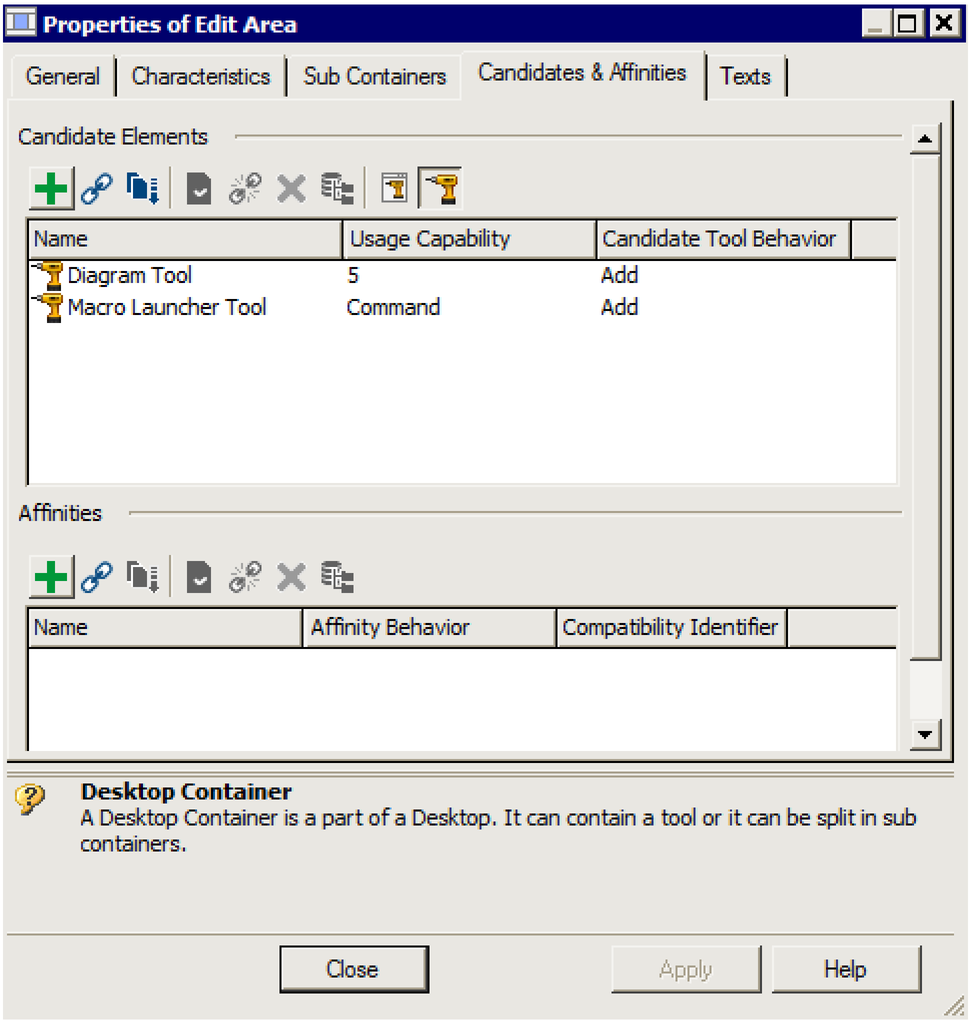
5. Select the MEGA Parameterized Tool/Tool and click **OK**.

➢ The Parameterized Tools/Tools are listed in the Candidate Elements frame of the Desktop Container (example: "Edit Area").

In the **Candidate Tool/Parameterized Tool Behavior** field, the value:

- "Replace" indicates that the existing page will be replaced,
- "Add" indicates that a new page will be added.

At an opening command of these **Parameterized Tools/Tools**, they open in the specified Container (example: "Edit Area").



Name	Usage Capability	Candidate Tool Behavior
Diagram Tool	5	Add
Macro Launcher Tool	Command	Add

Name	Affinity Behavior	Compatibility Identifier

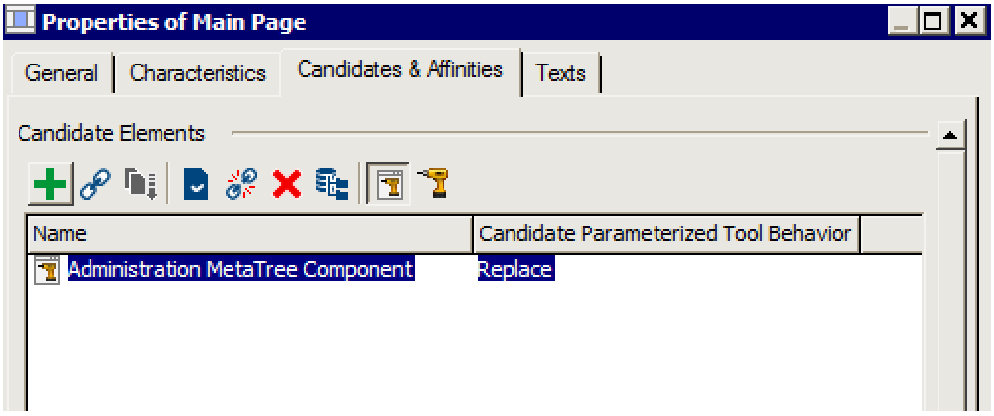
Desktop Container
A Desktop Container is a part of a Desktop. It can contain a tool or it can be split in sub containers.

Example:

In our case, the Left Container of **Accordion** type is candidate to receive all trees. For example, the Favorites tree opens in the Accordion Container.

➢ If you want a particular tree (example: Administration) to open in a Container A (example: "Main Page"), this Container A (example: "Main

Page") must have as **Candidate Parameterized Tool** the Administration tree.


Properties of Main Page

Name	Candidate Parameterized Tool Behavior
Administration MetaTree Component	Replace

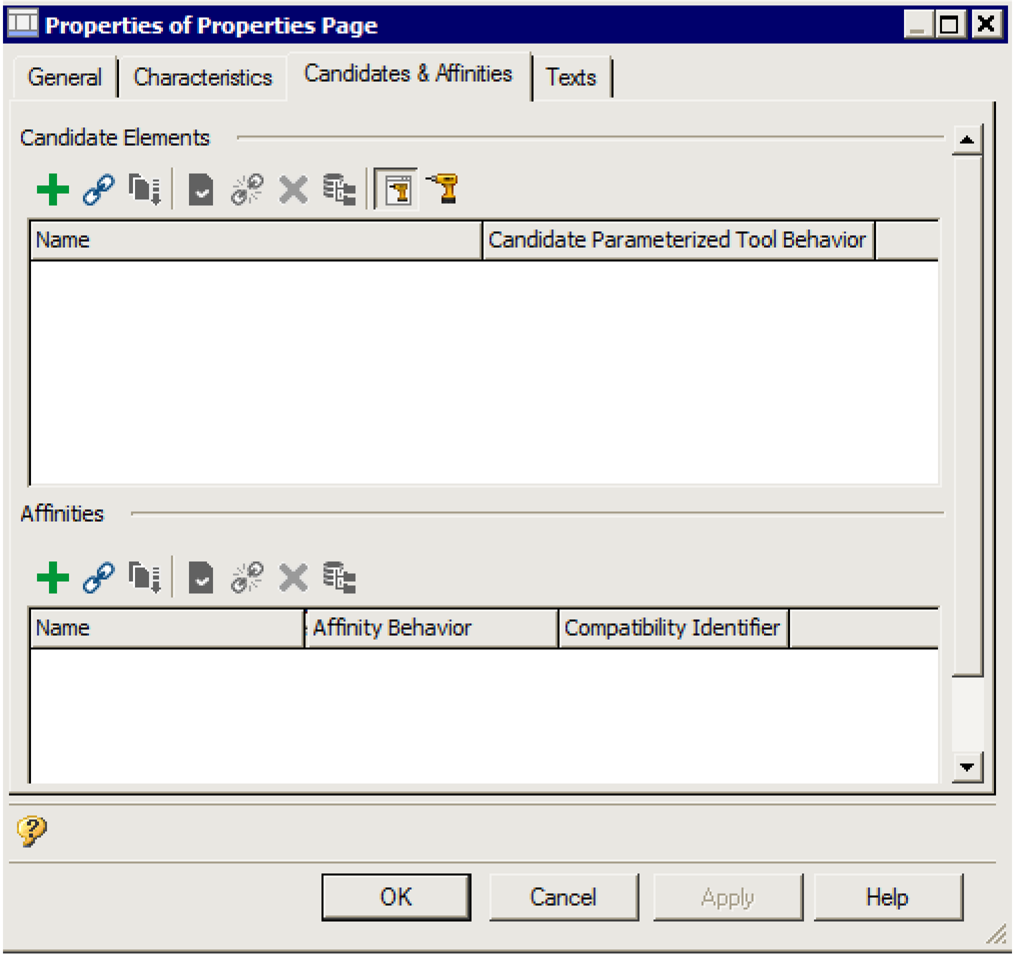
6.6 Defining Container affinities

An affinity enables characterization of a Container. You can for example define that a **Tool** or **MEGA Parameterized Tool** opens in a Container that has an affinity "Affinity Name".

An affinity is only used by code, unlike candidates which are defined in the metamodel. If no code is defined for opening of a tool with affinity, then the configuration defined for candidates is taken into account (see Defining Container candidates).

1. Open the **Properties** of the Container for which you want to define an affinity (example: Desktop Container "Properties Page").

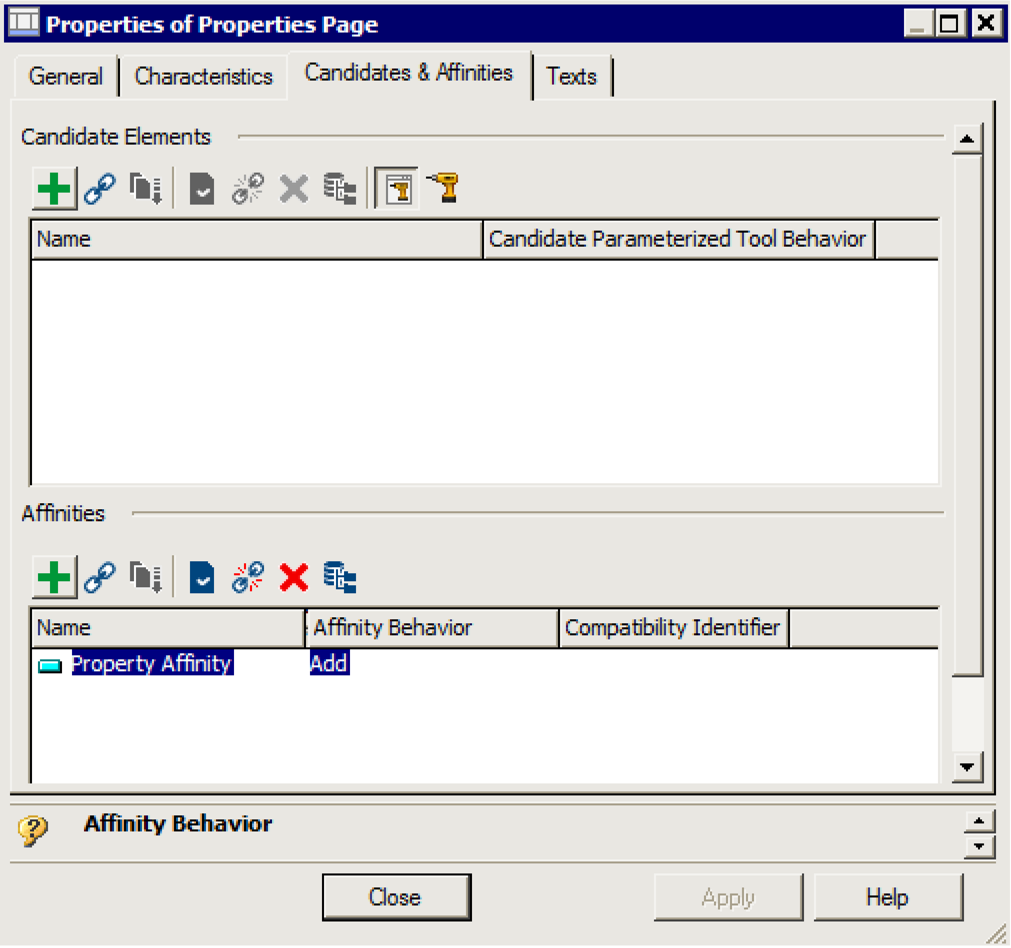
2. Select the **Candidates & Affinities** tab.



3. In the **Affinities** frame, click **New** ✚ (or **Connect** 🔗 if the affinity concerned is already created).

4. Enter the Name of the affinity (example: "Property Affinity").

5. (Optional) In the **Affinity Behavior** field, by default the affinity behavior has value **Replace**, in this case the page will open replacing the previous content of the Container. To add rather than replace a Container tab, modify the affinity behavior parameter to **Add**.



Name	Candidate Parameterized Tool Behavior

Name	Affinity Behavior	Compatibility Identifier
Property Affinity	Add	

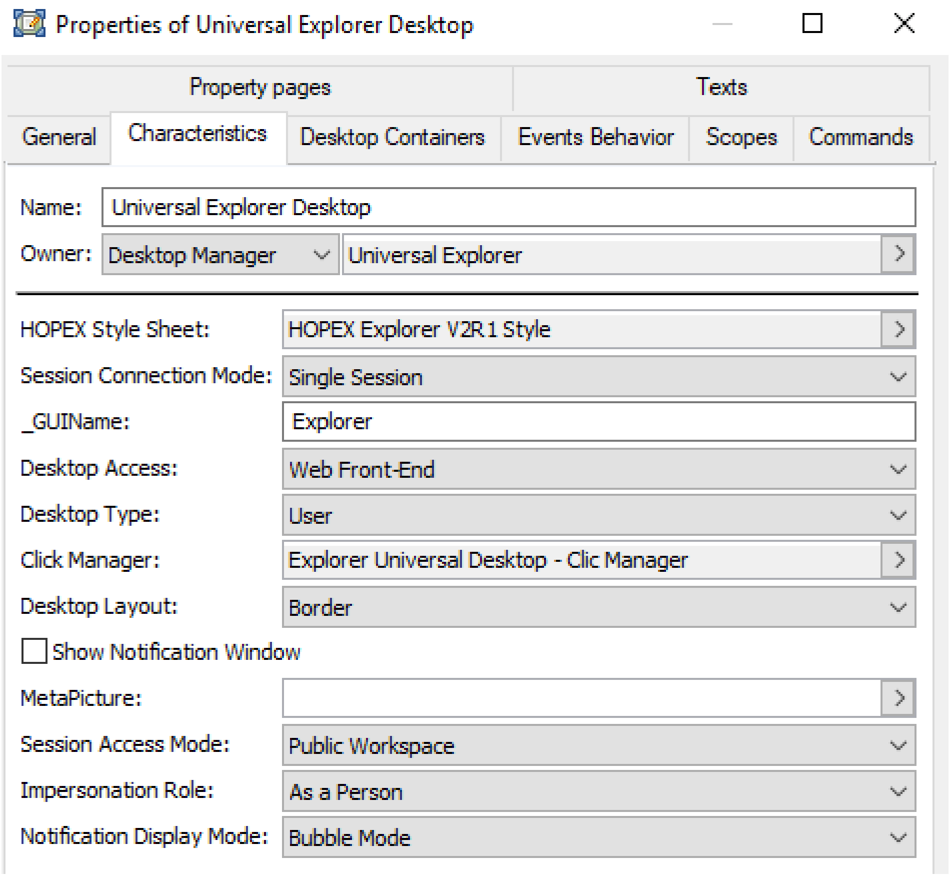
6. (Optional) The **Compatibility Identifier** field enables unique definition of a Container. The tool to be opened will therefore always open in the Container that has this specific identifier.

   Example: This can be useful for example to always open analysis reports in a specific tab, and Properties pages in another specific tab.

7 DEFINING DESKTOP CHARACTERISTICS

From the desktop Properties, you can modify default values and configure:

- the desktop characteristics
- the desktop connection configurations.



Name: Universal Explorer Desktop

Owner: Desktop Manager | Universal Explorer

HOPEX Style Sheet: HOPEX Explorer V2R1 Style

Session Connection Mode: Single Session

_GUIName: Explorer

Desktop Access: Web Front-End

Desktop Type: User

Click Manager: Explorer Universal Desktop - Clic Manager

Desktop Layout: Border

☐ Show Notification Window

MetaPicture:

Session Access Mode: Public Workspace

Impersonation Role: As a Person

Notification Display Mode: Bubble Mode

7.1 Defining the desktop characteristics

To define the desktop characteristics:

1. Access the **Properties** of the desktop concerned (example: "Universal Explorer Desktop").
2. In the **Characteristics** tab, modify or specify the required fields:
   - **HOPEX Style Sheet**: enables to define a specific style sheet for the desktop.

     Default value: empty (it is the value defined in **HOPEX Options > Installation > Theme used in Web Application**).
   - **_GUIName**: defines the desktop name display in the user interface (useful when using **Desktop Switcher** MEGA Tool).
   - **Desktop Access**: defines whether the desktop is accessible via Web Front-End or Windows Front-End.

Default value: "Web Front-End".

- **Desktop Type**: defines whether the desktop is user or administrator type. A desktop of administrator type should have more rights and visibility.

  Default value: "User".

- **Click Manager**: by default, a (left) click manages standard current. You can configure alternative behavior of the (left) click on your desktop.

  See Configuring click (left).

- **Show Notification Window**: select this parameter to activate display of notifications at the bottom of the page when you execute an action.

- **MetaPicture**: defines desktop icon display in the user interface (useful when using **Desktop Switcher** MEGA Tool).

7.2 Modifying the desktop connection configurations

A desktop is linked to a MEGA Application. To define connection modes of the desktop of an application, you must open the Properties of the desktop from the MEGA Application.

You cannot define desktop connection characteristics if you open its Properties from the query tool.

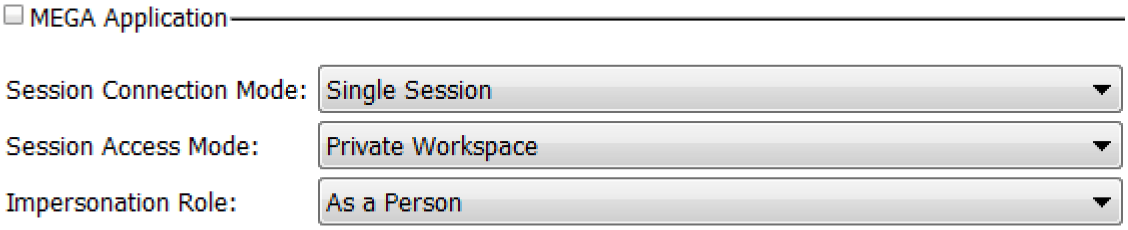
To modify desktop default connection characteristics:

1. In **HOPEX**, display the **MetaStudio** navigation window.

2. Expand the **MEGA Application** folder, then the application concerned (example: "HOPEX Explorer").

3. Right-click the desktop concerned (example: "Desktop MA A") and select **Properties**.

4. Select the **Characteristics** tab.



□ MEGA Application

Session Connection Mode: Single Session

Session Access Mode: Private Workspace

Impersonation Role: As a Person

5. In the **MEGA Application** frame, modify the required fields:

   – **Session Connection Mode**: enables definition of whether users connected to HOPEX share the same process or not. In "Multi-Session" mode, users share the same process and therefore have the same view of the repository. "Multi-Session" mode is more optimized but blocks SystemDb repository updates.

     Default value: "Single Session".

- **Session Access Mode**: enables definition of mode in which application opens.

  "Public Workspace": the application opens in the current state and data can be modified. All updates are visible to all users using the application at the same time.

  "Private Workspace": the application opens in the current state and data can be modified. All updates made by the user are kept in the private Workspace of the user until he/she decides to dispatch them.

  Default value: "Private Workspace".

  Note: Workspace is the new wording for transaction.

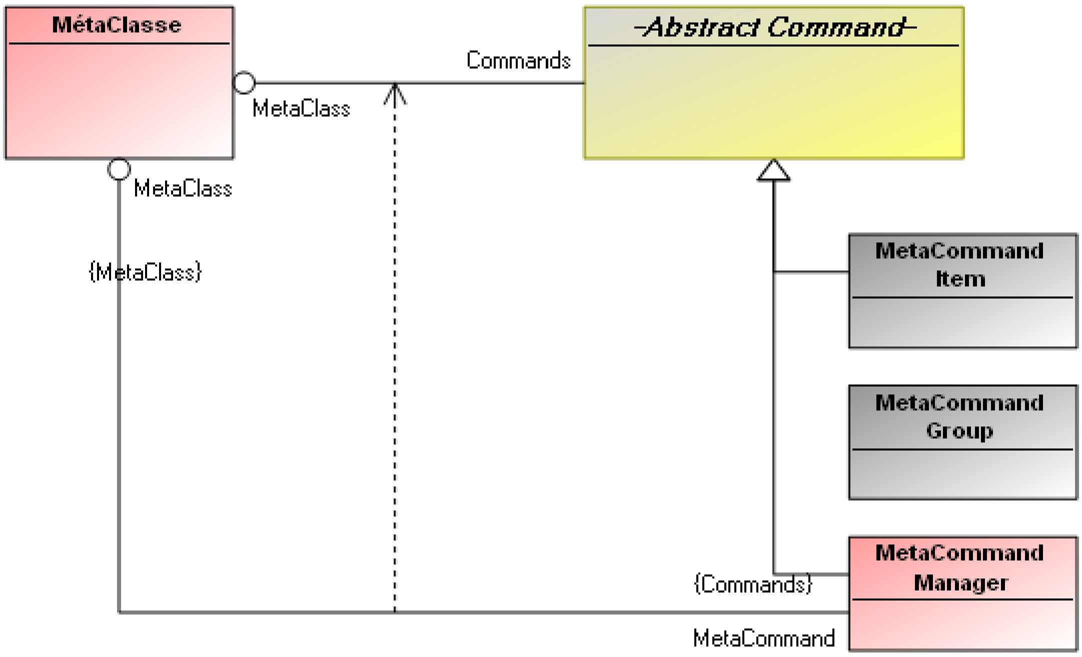
- **Impersonation Role**: enables definition of whether a user connects as a person with his/her rights, or as a person group to which he/she belongs with rights associated with the group.

  Default value: "As a Person".

8 CONFIGURING THE DESKTOP

By default, no pop-up menu (right-click on object) is available in the desktop.

The MetaModel linked to commands is the following:



8.1 Creating pop-up menus on an object

A pop-up menu is a set of commands (**MetaCommand Manager/MetaCommand Item**) accessible by right-click on a desktop object. Each command is characterized by a Category.

Note: if necessary, you can restrict a pop-up menu to a specific desktop, see Restricting a command to a specific desktop

To define pop-up menus accessible (by right-click) from an object (MetaClass) of the desktop, you must define commands accessible from this object (MetaClass).

For each command (of **MetaCommand Item** or **MetaCommand Manager** type) you must define its name (**_GUIName**) and display icon (**MetaPicture**) in the user interface, as well as the **Command Category** to which it belongs. This category is one of the standard categories supplied by MEGA.

The pop-up menu (defined by the command) is classified in a category.

To define a pop-up menu (of MetaCommand Item type) of a desktop object:

1. In **HOPEX**, run a query on **MetaClass**.

2. Right-click the MetaClass you want to configure (example: **Org-Unit**) and select **Properties**.

3. Select the **User Interface** tab and the **Menu Command** subtab.

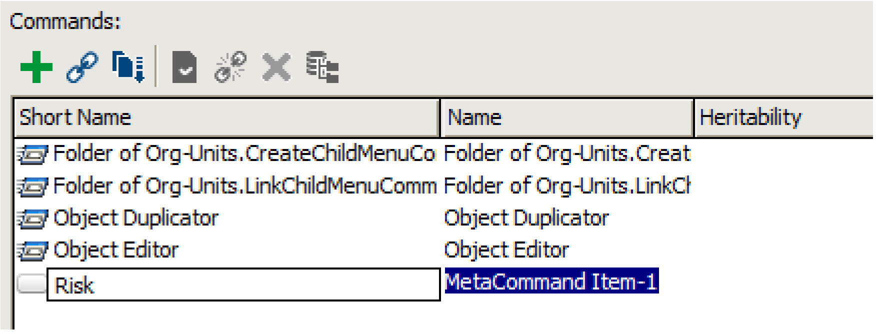
4. Click **New** +.

   ➔ The **Choice of MetaClass Abstract Command** appears.

5. In the **MetaClass** field, select **MetaCommand Item**.

   ➔ The new command appears in the frame listing commands (**MetaCommand Item** and **MetaCommand Manager**).

6. In the **Short Name** field, enter the command name (example: « Risk»).



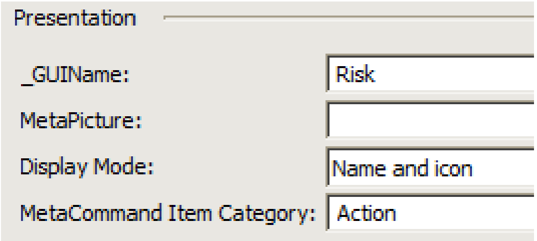
Commands:

Short Name	Name	Heritability
Folder of Org-Units.CreateChildMenuCo	Folder of Org-Units.Creat	
Folder of Org-Units.LinkChildMenuComm	Folder of Org-Units.LinkCl	
Object Duplicator	Object Duplicator	
Object Editor	Object Editor	
Risk	MetaCommand Item-1	

   ➔ Command characteristics appear.

7. In the **Presentation** frame, enter the following fields:

   - in the **_GUIName** field, enter the display name of the command in the user interface.
   - in the **MetaPicture** field, click the arrow and specify the icon of the command.
   - in the **Display Mode** field, select the display mode.
   - in the **MetaCommandItem Category** field, select the command category (example: "Action").



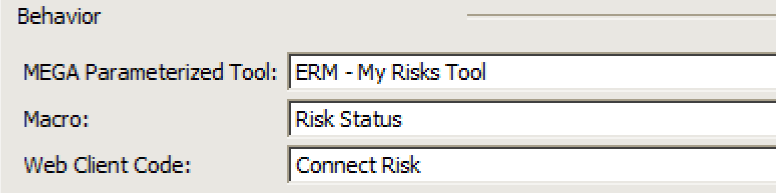
Presentation

_GUIName:	Risk
MetaPicture:	
Display Mode:	Name and icon
MetaCommand Item Category:	Action

8. In the **Behavior** frame, specify one of the following fields. Click the arrow in field:

   - **MEGA Parameterized Tool** and select the **MEGA Parameterized Tool** that will implement the command (example: "ERM – Risk Consequences").
   - **Macro** and select/create the macro that will implement the command (example: "Risk Status").

- **Web Client Code** and select/enter the JavaScript _Code Template (example: « Connect Risk »).



Behavior

MEGA Parameterized Tool: ERM - My Risks Tool

Macro: Risk Status

Web Client Code: Connect Risk

9. Click **OK**.

⚠ If you want a pop-up menu command to be specific to a single desktop, see Restricting a command to a specific desktop.

8.2 Restricting a command to a specific desktop

When a command defined on a MetaClass should be accessible from a particular desktop only, you must define this restriction in the desktop properties.

1. In **HOPEX**, display the **MetaStudio** navigation window.
2. Expand the **MEGA Application** folder, then the application concerned.
3. Right-click the desktop (example: "Desktop MA A") and select **Properties**.
4. Select the **Commands** tab.
5. In the **Specific Command** frame, click **Connect** 🔗 (or **New** ➕ if the command is not yet created) and select the command you want to restrict to this desktop (example: "Desktop MA A").

8.3 Examples of macros

8.3.1 Creating commands of a MetaClass

In the pop-up menu, MetaCommand Manager and MetaCommand Item coexist:

- **MetaCommand Manager** is implemented by a macro with Sub cmdInvoke:

```
Sub cmdInvoke(oBookPart, number)

End Sub
```

➔ For more information, refer to the ***MEGA Studio*** guide.

- **MetaCommand Item** is implemented by a macro with Function invokeOnObject:

```
Function InvokeOnObject(oRoot, sUserData)

End Function
```

8.3.2 Implementing a group of dynamic commands

The following is an example of a group of commands enabling addition of commands dynamically and specification of their behavior when clicked by a user.

- Associate the following macro with MetaCommand Group:

```
'=======================================================================
'Macro : ~j3IwNEVIGTfA[Languages.Implementation]
'-----------------------------------------------------------------------
'MegaContext(Fields,Types)
Option Explicit

Class Language
  Public idabs
  Public guiname
  Public picture
End Class

Class ScannerLanguages
  Public mgResource
  Public mgToolkit
  Public mgbTrouve
  Public mgLanguages()
  public nbLanguages

  Public Sub OnItem(Context, Id)
    mgbTrouve = True
    nbLanguages = nbLanguages + 1
    ReDim Preserve mgLanguages(nbLanguages)

    Dim aLanguage
    Set aLanguage = New Language
    aLanguage.idabs = mgToolkit.getString64FromID(Id)
    aLanguage.guiname = mgResource.name(Context, "GuiName")

    Dim mgScannerMP
    Set mgScannerMP = New ScannerMetaPicture
    mgScannerMP.mgbTrouve = False
    mgResource.ScanCollection Id, "~A1mUbldyx840[MetaPicture]", mgScannerMP, 1,
""
    If mgScannerMP.mgbTrouve Then
      aLanguage.picture = mgToolkit.getString64FromID(mgScannerMP.idPicture)
    End If

    Set mgLanguages(nbLanguages) = aLanguage
    'Context.Abort
  End Sub
End Class

Class ScannerMetaPicture
  Public mgbTrouve
  Public idPicture

  Public Sub OnItem(Context, Id)
    mgbTrouve = True
    idPicture = Id
    Context.Abort
  End Sub
End Class

Function GetLanguages(mgRoot As MegaRoot)
  Dim mgScanner
  Set mgScanner = New ScannerLanguages
  Set mgScanner.mgResource = mgRoot.CurrentEnvironment.Resources
  Set mgScanner.mgToolkit = mgRoot.currentEnvironment.toolkit

  Dim idDepart
  Dim idLink
  Dim listAttributes

  idDepart = "~I9o3by0knG00" '~I9o3by0knG00[Neutral]
  idLink = "~41000000CW30[_Lower]"
  listAttributes = ""
  mgScanner.mgbTrouve = False
  mgScanner.nbLanguages = 0
  mgScanner.mgResource.ScanCollection idDepart, idLink, mgScanner, 1,
listAttributes
  if (mgScanner.mgbTrouve) Then
    GetLanguages = mgScanner.mgLanguages
  else
    GetLanguages = null
  end if
end function

Function count (mo, CommandGroupID, strUserData )
  Dim mgLanguages
  mgLanguages = GetLanguages(mo.GetRoot)
  If (Not IsNull(mgLanguages)) Then
    Dim i
    count = 0
    For i = 1 To UBound(mgLanguages)
      If
mo.GetRoot.CurrentEnvironment.Toolkit.IsAvailable(mgLanguages(i).idabs) Then
        count = count + 1
      End If
    Next
  End If
End Function

Sub CommandEnum(mo, CommandGroupIDParent, vCommandGroupID, oGenContext,
oMenuContext, strUserData)
  dim idabsMaCommande
  dim strTitle
  dim idabsImageMoniker
  dim strTooltip
  dim dwCategory
  dim dwStyle
  dim idabsMacro
  dim idabsParameterizedTool
  dim strToolbarPosition
  dim idabsCodeTemplate

  dim mgLanguages
  mgLanguages = GetLanguages(mo.GetRoot)

  dim i
  dim res
  If (Not IsNull(mgLanguages)) Then
    For i = 1 To Ubound(mgLanguages)
      If
mo.GetRoot.CurrentEnvironment.Toolkit.IsAvailable(mgLanguages(i).idabs) Then
        idabsMaCommande = mgLanguages(i).idabs
        dwCategory =10
        strTitle = mgLanguages(i).guiname
        idabsImageMoniker = "~" & mgLanguages(i).picture
        idabsMacro = "~m3IwTHVIGDjA" '~m3IwTHVIGDjA[Set Current Language.Macro]
        res = oMenuContext.CommandAdd(mo, "~3uw6D72eErUS[Command]",
idabsMaCommande, strTitle, idabsImageMoniker, strTooltip, dwCategory , dwStyle,
strUserData , idabsMacro, idabsParameterizedTool ,  strToolbarPosition,
idabsCodeTemplate)
      End If
    Next
  End If
end sub
```

This macro uses scanners to browse available languages. But commands essential to MetaCommand Group are CommandEnum and Count, which enable addition of commands dynamically.

- The macro called when a user clicks any button dynamically creates:

```
'========================================================================
'Macro : ~m3IwTHVIGDjA[Set Current Language.Macro]
'------------------------------------------------------------------------
'MegaContext(Fields,Types)
Option Explicit

'------------------------------------------------------------------------------
' FUNCTION : InvokeOnObject
' Function called when the command is triggered from a MegaObject.
' Example : click on a menu.
'
' @param mgobjSource
'           MegaObject on which the command is applied.
'------------------------------------------------------------------------------
Function InvokeOnObject(mgobjSource as MegaObject, sUserData As Variant)
End Function


'------------------------------------------------------------------------------
' FUNCTION : InvokeOnRoot
' Function called when the command is triggered from a mgRoot.
' Example : click on a menu.
'
' @param mgRoot
'           mgRoot on which the command is applied.
'------------------------------------------------------------------------------
Function InvokeOnRoot(mgRoot as MegaObject, sUserData As Variant)
  Dim JSON
  Set JSON =
mgRoot.GetRoot.CurrentEnvironment.GetMacro("~j0Iw0fWIGLnA[GetCommandIdFromJSON]
")
  Dim languageId
  languageId = JSON.getCommandId(sUserData)
  If languageId <> "" Then
    languageId = "~" & languageId
    mgRoot.GetRoot.CurrentEnvironment.SetCurrentLanguage languageId
    InvokeOnRoot = "{refresh :true}"
  End If
  ' The desktop will be refreshed. To display the current language, you can use
the Macro ~52IwefUIGXMA[Get Current Language] that can be displayed in a static
zone tool
End Function
```

This macro operates the change of language based on the command clicked.

- To recover the command id, create a JavaScript macro, which will enable recovery of this information in the form json in the sUserData variable. Its code is the following:

```
'=========================================================================
'Macro : ~j0Iw0fWIGLnA[GetCommandIdFromJSON]
'-------------------------------------------------------------------------
//Language:JavaScript

function getCommandId(szJSON) {
  var jsonObject = eval('(' + szJSON + ')');
  var szId = jsonObject.commandId;
  return szId;
}
```

A macro enables addition of commands.

⚠ You cannot have groups of commands in groups implemented by a macro.

Example: the macro **Languages.Implementation**

In the example below:

- the **count** function should return >0 for CommandEnum to be called.
- **count** indicates the number of commands that will be added to CommandGroup CommandGroupID.
- in CommandEnum you can add commands to CommandGroupID by calling function CommandAdd of oMenuContext.

```
Idabs = j3IwNEVIGTfA

Sub CommandEnum(mo, CommandGroupIDParent, CommandGroupID,
oGenContext, oMenuContext, strUserData)
  dim idabsMyCommand
  dim strTitle
  dim idabsImageMoniker
  dim strTooltip
  dim dwCategory
  dim dwStyle
  dim idabsMacro
  dim idabsParameterizedTool
  dim strToolbarPosition
  dim idabsCodeTemplate
Dim res
res = oMenuContext.CommandAdd(mo, "~3uw6D72eErUS[Command]",
idabsMyCommand, strTitle, idabsImageMoniker, strTooltip, dwCategory ,
dwStyle, strUserData , idabsMacro, idabsParameterizedTool ,
strToolbarPosition, idabsCodeTemplate)
End sub

Function count (mo, CommandGroupID, strUserData )

End function
```

8.4 Configuring click (left)

By default, a (left) click manages change of standard current. It applies to MEGA occurrences.

You can configure alternative behavior of the (left) click on:

- a desktop, or
- a MEGA Parameterized Tool

To configure a (left) click:

1. Open the **Properties** of the desktop (example: "Desktop MA A") or of MEGA Parameterized Tool.
2. Select the **Characteristics** tab.
3. In the **Click Manager** field, click the arrow and select **Create** (or **List** if the implementation macro is already created).
4. (Optional) Enter the Name of the click manager.
5. In the **Implementation Macro** field, click the arrow and select **Create Macro** (or **List** if the macro is already created).
    a. In the **Creation of Macro** dialog box, select **Create (VB)Script Macro**.
    b. Click **Next**.
    c. Enter the **Name** of the macro.
    d. (Optional) Select **Reusable** if you want to be able to reuse the macro.

e. Click **Finish**.

➔ The name of the implementation macro appears in the **Implementation Macro** field.

6. In the **Implementation Macro** field, click the arrow, select the macro you have just defined and select **Edit**.

7. In the script edit dialog box, enter the macro script:

```
Sub GenerateStream(oObject, oContext, sUserData, oStream)
End Sub.
```

- oObject: object to which command is applied
- oContext: defines execution context
- sUserIn: defines any additional information that may be required to define action on object click
- oTextStream returns:
    - the JSON corresponding to the command to be executed, or
    - the JSON after which simple current management is required:

      {setCurrent: true).

Example of click on a tree element:

```
Sub GenerateStream(mgobjSource , oContext , sUserData , oStream)

oStream.Write("{setCurrent:true}")

End Sub
```

- mgobjSource: Mega object to which command is applied
- oContext: defines generation context
- sUserData: defines any additional parameter that may be required to decide to run the action.
- oStream contains the result of JSON to be sent in return to the client
    - a standard JSON is returned for the setCurrent action:

      {setCurrent:true}
    - a dedicated JSON is returned if a command must be executed

The JSON awaited in return of the function is the JSON of a command. To do this, MEGA provides a standard function enabling recovery of the JSON associated with a MetaCommand Item: GetJSONCommand

```
Sub Generate(mgobjSource , oContext , sUserData , sResult )
'Dim oRoot
'Set oRoot = mgobjSource.GetRoot
```

```
'sResult = oRoot.GetJSONCommand("CommandIdabs")
End Sub
```

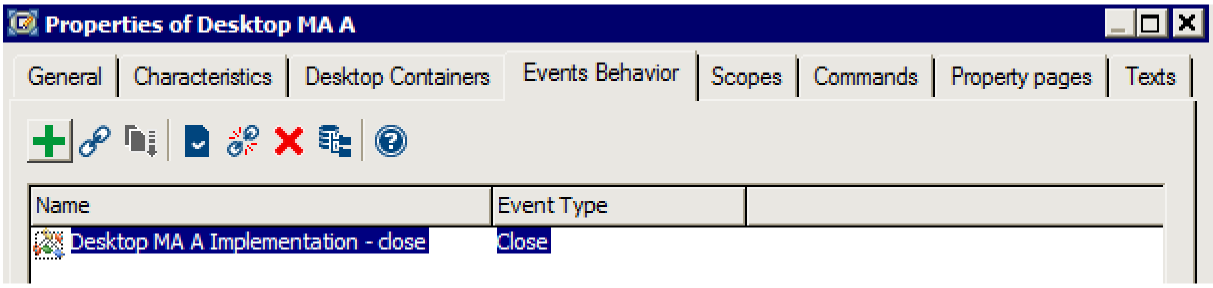
8.5 Configuring behavior on an event

You can configure your desktop to generate a specific behavior at opening, at closing and/or at backup of your desktop.

Example:

- when first opening the desktop, you can run a questionnaire HTML formatter.
- when closing the desktop**,** you can run a dispatch wizard.

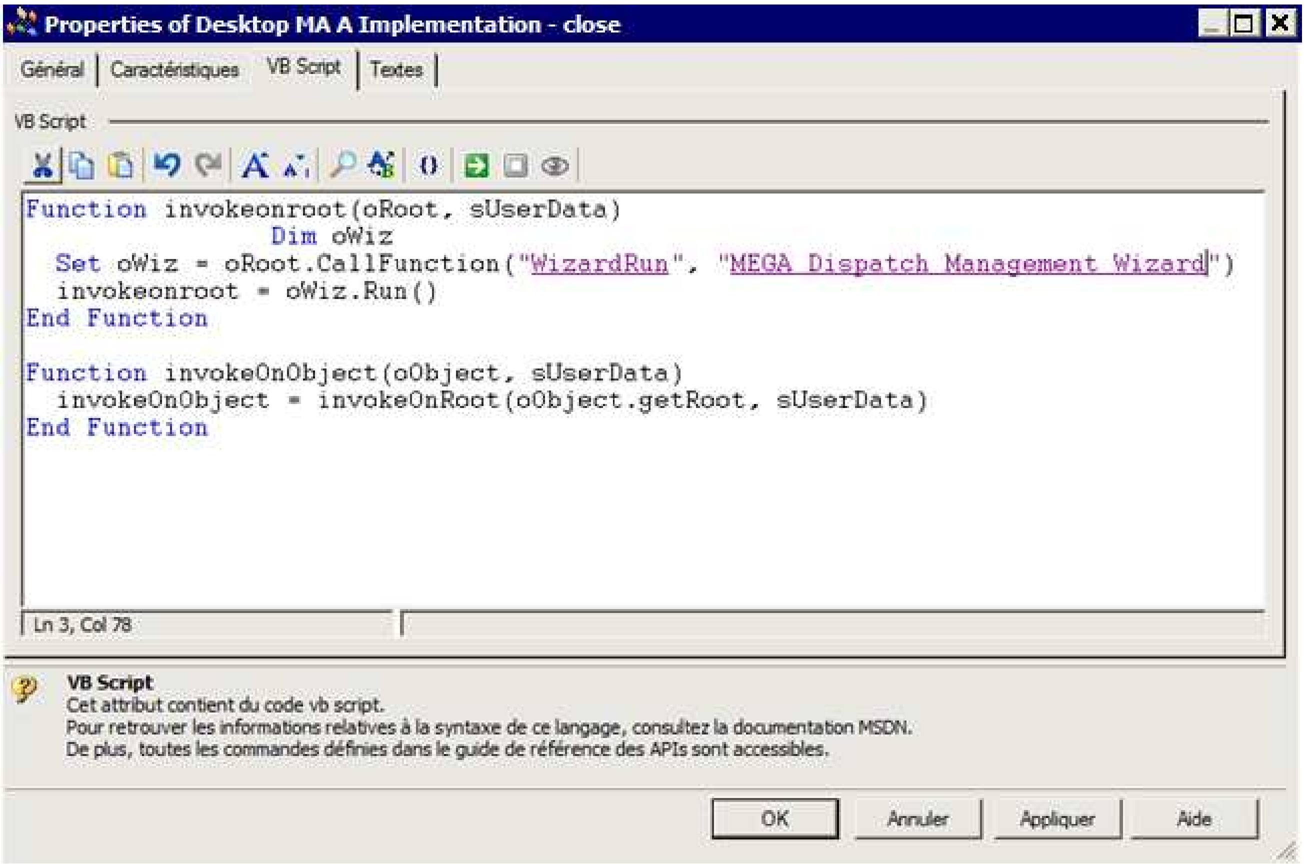
1. Open the **Properties** of the desktop (example: "Desktop MA A") or of MEGA Parameterized Tool.
2. Select the **Events Behavior** tab.
3. Click **New** +.
4. Select **Manage Macro (VB)Script**.
5. Click **Next**.
6. Enter the **Name** of the macro (example: "Desktop MA A Implementation – Close").
7. Click **Finish**.
8. In the **Event Type** field, select **Close**.



Name	Event Type
Desktop MA A Implementation - close	Close

9. Open the **Properties** of the macro you have just created (example: "Desktop MA A Implementation – Close").
10. Select the **VB Script** tab and enter the code of the macro.

For example:



```
Function invokeonroot(oRoot, sUserData)
                Dim oWiz
  Set oWiz = oRoot.CallFunction("WizardRun", "MEGA Dispatch Management Wizard")
  invokeonroot = oWiz.Run()
End Function

Function invokeOnObject(oObject, sUserData)
  invokeOnObject = invokeOnRoot(oObject.getRoot, sUserData)
End Function
```

VB Script
Cet attribut contient du code vb script.
Pour retrouver les informations relatives à la syntaxe de ce langage, consultez la documentation MSDN.
De plus, toutes les commandes définies dans le guide de référence des APIs sont accessibles.

"MEGA Dispatch Management Wizard" is the dispatch wizard, of which the identifier is:

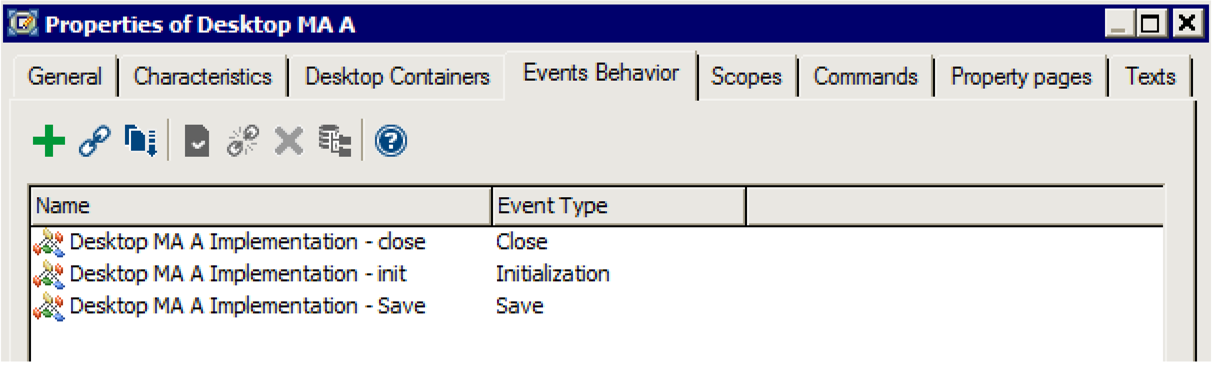
~)pVZ5wu47b00[**MEGA Dispatch Management Wizard**]

11. Repeat steps 3 to 10 to create specific behaviors.

For **Save** Event Types and the initialization macro, the prototype of functions to be implemented is the same:

```
Function invokeonroot(oRoot, sUserData)

End Function

Function invokeonobject(oRoot, sUserData)
End Function
```

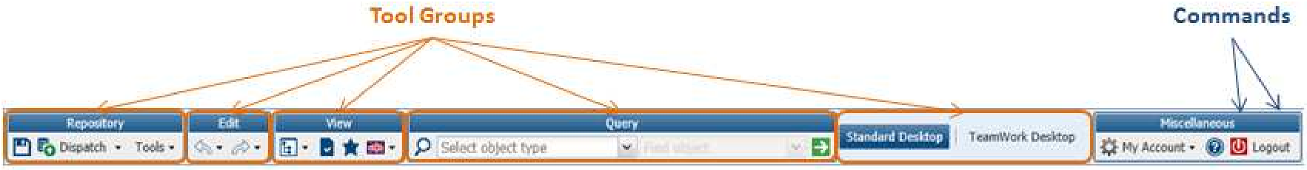
Do not omit Function invokeonobject; although empty, it is necessary for operation.



Name	Event Type
Desktop MA A Implementation - close	Close
Desktop MA A Implementation - init	Initialization
Desktop MA A Implementation - Save	Save

8.6 Creating a toolbar

A toolbar comprises commands, groups of commands and/or MEGA Parameterized Tools. These components can be grouped by themes (tool groups).



Example of a toolbar

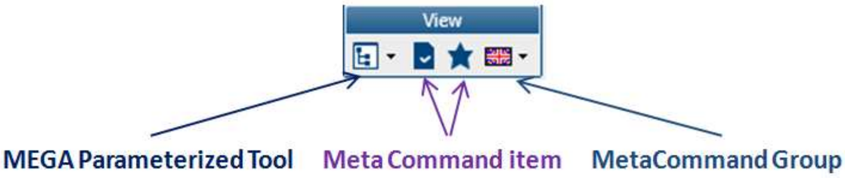
A tool group is a **MetaCommand Group** (example: "View") which can contain:

- **MetaCommand Groups**
  Example: "Language"

- **MetaCommand Items**
  Example: "Favorites", "Properties"

- **MEGA Parameterized Tools**

  Example: "Navigation Trees Manager"



Example of a toolbar tool group ("View")

Creating a toolbar consists of creating a **Desktop Container** of **Container of Tool** type. This **Container of Tool** will contain the **Toolbar** desktop component (MEGA Parameterized Tool).

To create a toolbar, you should proceed as follows:

Step	Action	See
1.	Creating your toolbar structure	8.6.1
2.	Creating toolbar tool groups	8.6.2
3.	Configuring toolbar tool group display	8.6.3
4.	Defining toolbar tool group commands	8.6.4
5.	Configuring toolbar commands display	8.6.5

Note: See Command group configuration examples.

8.6.1 Creating your toolbar structure

To create your toolbar, you must create a **Desktop Container** of **Container of Tool** type.

You will define this **Container of Tool** by a desktop component (**MEGA Parameterized Tool**) of **Toolbar** type, which has a group of commands as its parameters.

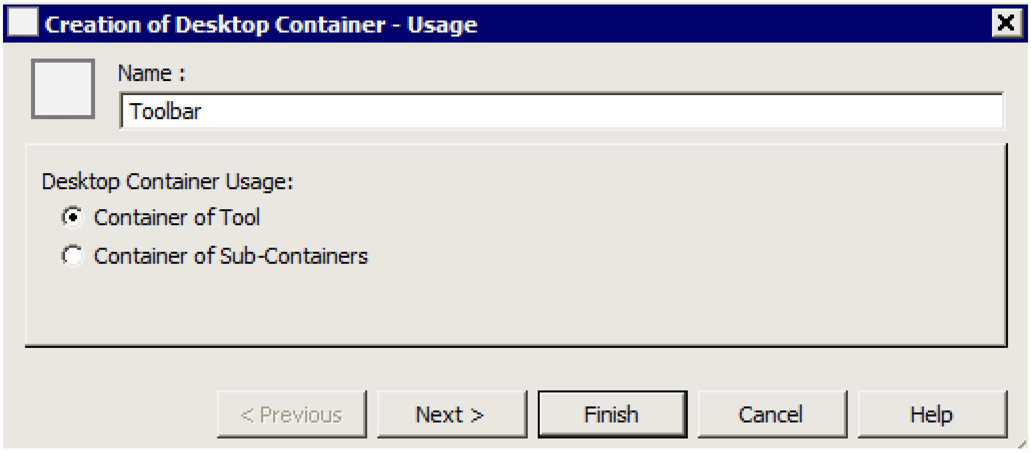
You will create the group of commands (**MetaCommand Group**) representing your toolbar, and define representation of your toolbar (ribbon, ribbon without frame or drop-down menus):

1. From the **Creation of Desktop – Desktop Containers** dialog box (see Creating a desktop), in the **Top Container** (**Left**, **Right**, **Center** or **Bottom**) field, click the arrow and select **New**.

   ➢ The first step of the **Creation of Desktop Container** wizard appears: **Usage**.

2. Enter the **Name** of the Container (example: "Toolbar").

3. Select the type of **Container of Tool** Desktop Container.



Creation of Desktop Container - Usage

Name : Toolbar

Desktop Container Usage:
● Container of Tool
○ Container of Sub-Containers

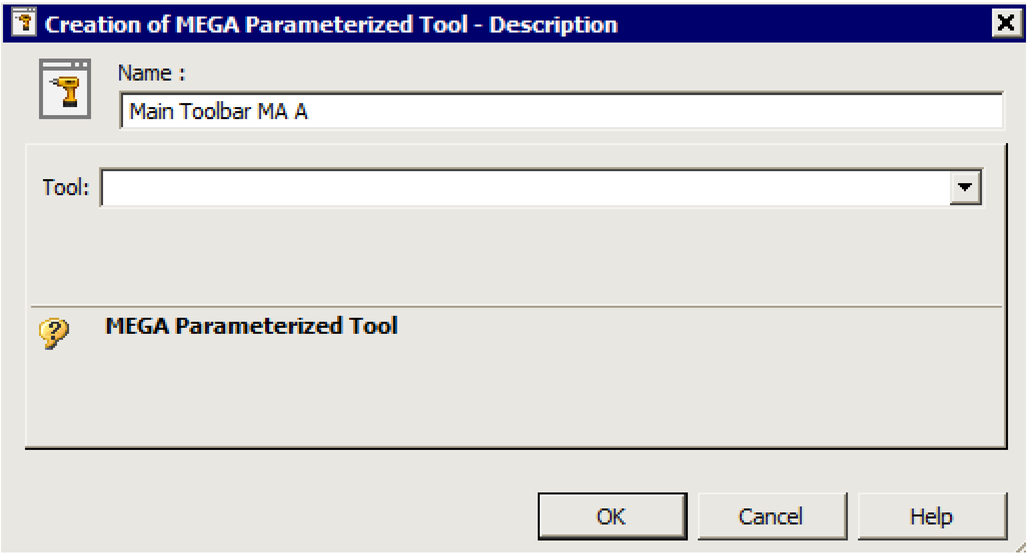
4. Click **Next**.

   ➢ The second step of the **Creation of Desktop Container** wizard appears: **Description**.

5. In the **Parameterized Tool** field, click the arrow and select **Create MEGA Parameterized Tool** (or **List MEGA Parameterized Tool** if the Parameterized Tool is already created).

   ➢ The first step of a new **Creation of MEGA Parameterized Tool** wizard appears: **Description**.

6. Enter the **Name** of the desktop/Parameterized Tool component (example: "Main Toolbar MA A").



Creation of MEGA Parameterized Tool - Description

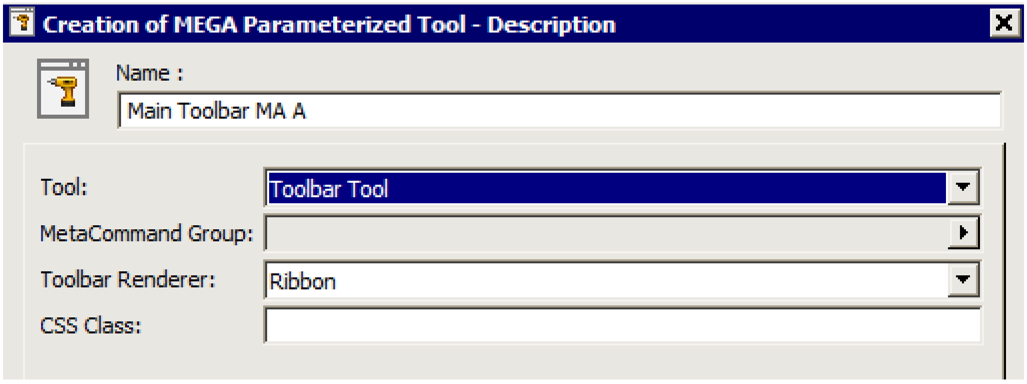
Name : Main Toolbar MA A

Tool:

MEGA Parameterized Tool

7. In the **Tool** field, click the drop-down menu arrow and select **Toolbar Tool**.

   ➢ **Toolbar Tool** is displayed in the **Tool** field. Fields **MetaCommand Group** and **Toolbar Renderer** appear.



Creation of MEGA Parameterized Tool - Description

Name : Main Toolbar MA A

Tool: Toolbar Tool

MetaCommand Group:

Toolbar Renderer: Ribbon

CSS Class:

8. In the **MetaCommand Group** field, click the arrow and select **Create MetaCommand Group** (or **List MetaCommand Group** if the required MetaCommand Group is already created).

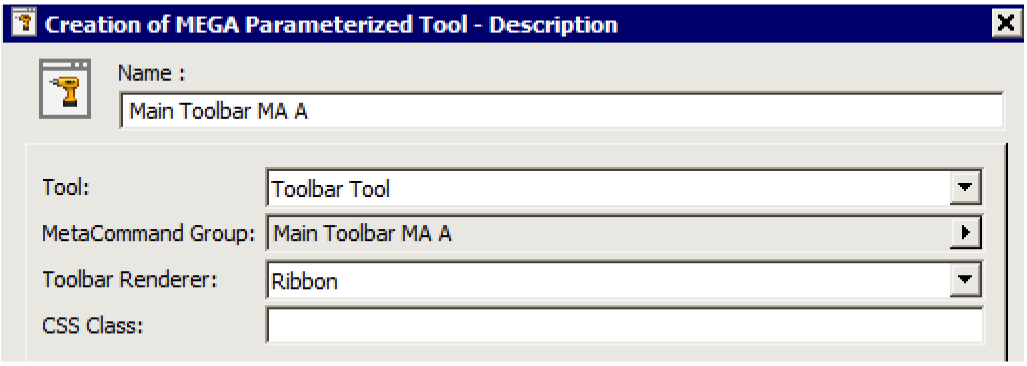
   ➢ The **MetaCommand Group** creation dialog box appears.

9. Enter the **Name** of the MetaCommand Group (example: "Main Toolbar MA A").

10. Click **OK**.

11. (Optional) In the **Toolbar Renderer** field, you can modify the toolbar type selected by default ("Ribbon").

    The toolbar can be presented in the form of a "Ribbon", a "Ribbon Without Frame" or a "Dropdown Menu.



Creation of MEGA Parameterized Tool - Description

Name : Main Toolbar MA A

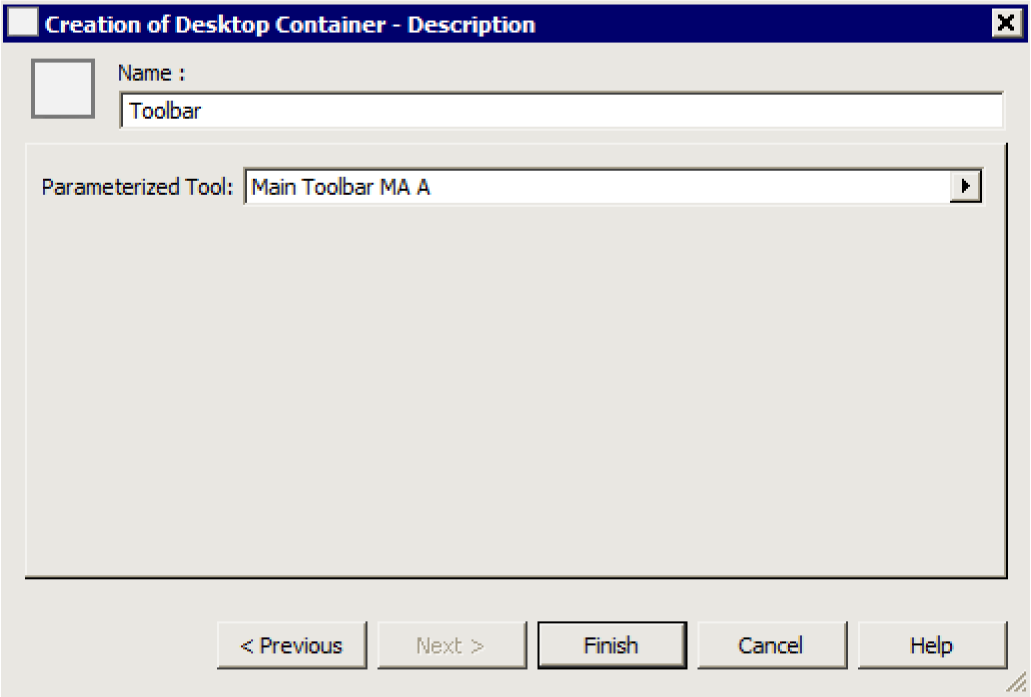
Tool: Toolbar Tool

MetaCommand Group: Main Toolbar MA A

Toolbar Renderer: Ribbon

CSS Class:

12.Click **OK**.



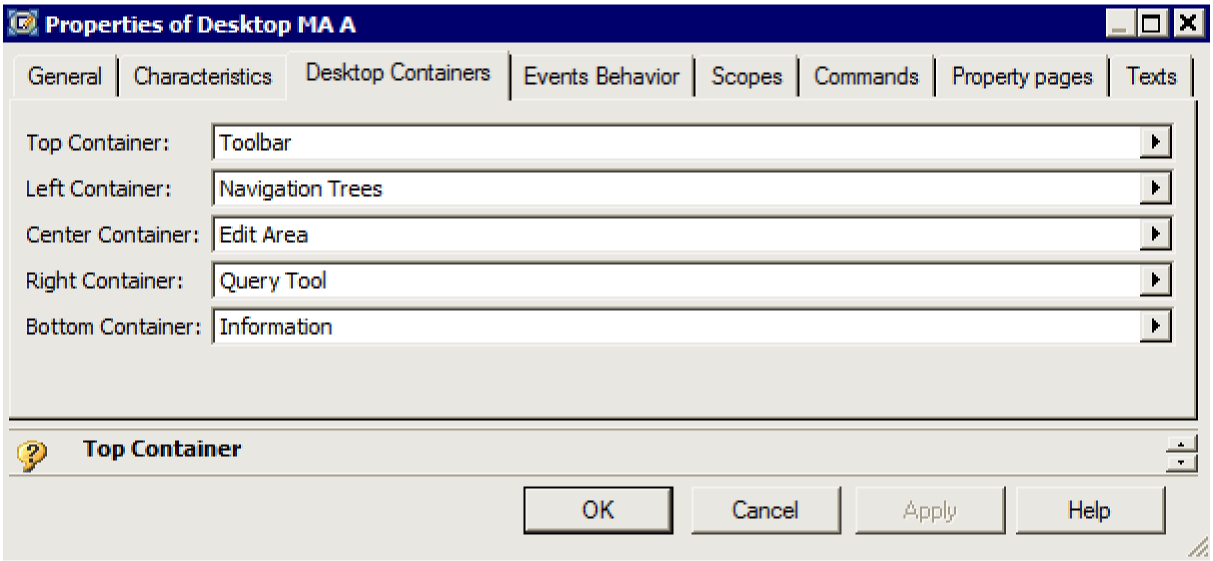
Creation of Desktop Container - Description

Name : Toolbar

Parameterized Tool: Main Toolbar MA A

13.Click **Finish**.

- ➢ The toolbar structure is created.



Properties of Desktop MA A

Top Container: Toolbar

Left Container: Navigation Trees

Center Container: Edit Area

Right Container: Query Tool

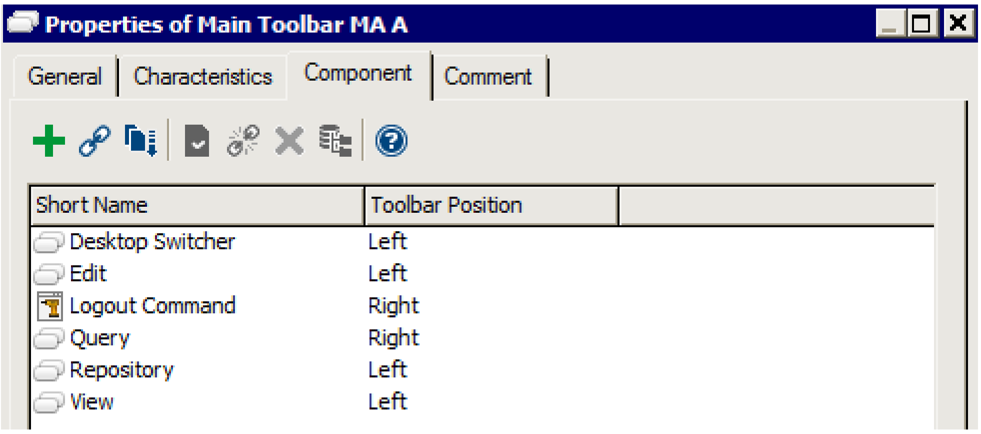
Bottom Container: Information

Top Container

8.6.2 Creating toolbar tool groups

To create tool groups (MetaCommand Groups) and other components (MetaCommand Item and/or MEGA Parameterized Tool) of your toolbar:

1. Open your toolbar properties (example: MetaCommand Group "MA A Main Toolbar").
2. In the Component tab, click **Create** (or **Connect**).
3. Select the type of **MetaClass** you want to create **MetaCommand Group**, **MetaCommand Item** or **MEGA Parameterized Tool**.
4. Click **OK**.
5. Enter the **Name** of the MetaCommand Group, MetaCommand Item or MEGA Parameterized Tool.
6. Repeat steps 2 to 5 and create (or connect) all the **MetaCommand Groups**, **MetaCommand Items** or **MEGA Parameterized Tools** required.
7. By default, the **toolbar components** are aligned left in the toolbar; to align a component on the right, click in the **Toolbar Position** field of the **MetaCommand Group**, **MetaCommand Item** or **Parameterized Tool**, and select **Right** (example: disconnection command "Logout Command").



Short Name	Toolbar Position
Desktop Switcher	Left
Edit	Left
Logout Command	Right
Query	Right
Repository	Left
View	Left

The order of presentation of the **MetaCommand Groups** in the list reflects the order of appearance of the tool groups and/or commands in the toolbar.

The value of the position of a **MetaCommand Group** (left/right) takes priority over its appearance order in the list.

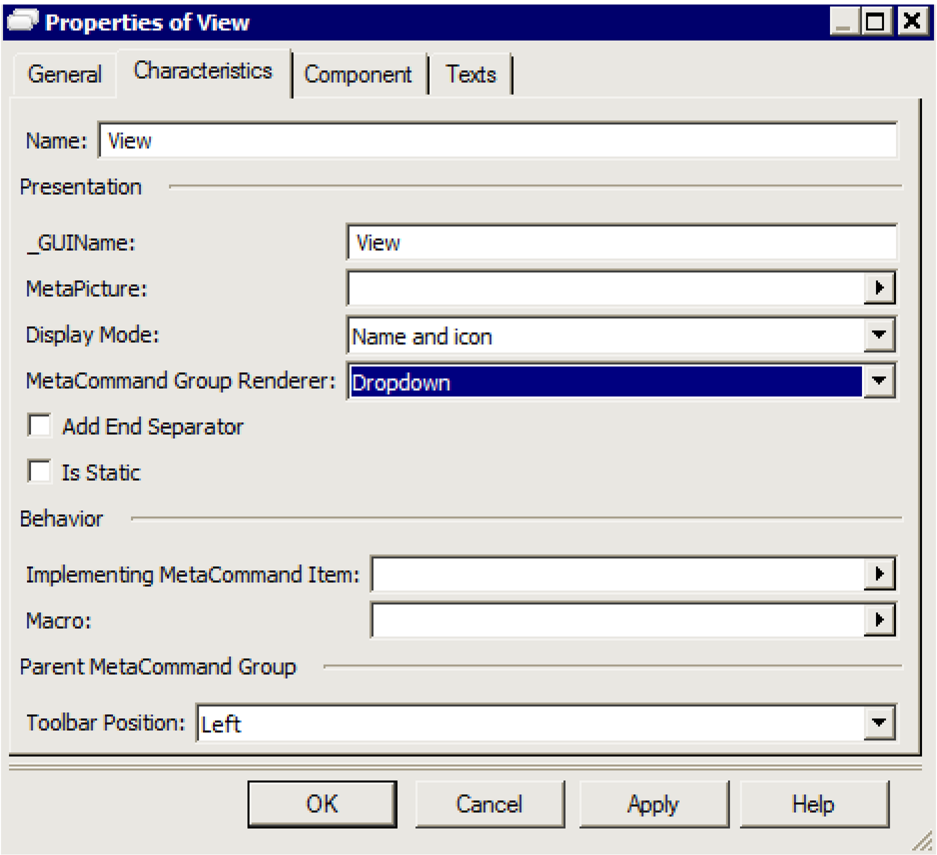
To modify organization of tool groups and/or commands, see Modifying the position of a tool group in the toolbar.

8.6.3 Configuring toolbar tool group display

To configure display of a toolbar tool group:

1. In **HOPEX**, display the **MetaStudio** navigation window.
2. Expand the **MEGA Application** folder, then the application and desktop (example: "Desktop MA A").
3. Right-click the desktop tool group (example: "Main Toolbar MA A") and select **Properties**.
4. Select the **Description** tab.
5. In the **MetaCommand Group** field (example: "Main Toolbar MA A"), click the arrow and select **Properties** of **MetaCommand Group** (example: "Main Toolbar MA A").
6. Select the **Component** tab.
7. In the list of components, right-click a tool group (example: "View") and select **Properties**.
8. Select the **Characteristics** tab.
9. In the **Presentation** frame:
   - in the **_GUIName** field, enter the name under which the **MetaCommand Group** will appear in the user interface.
   - if required, in the **MetaPicture** field, click the arrow and select the image of the MetaCommand Group that will appear in the user interface.
   - In the **MetaCommand Group Renderer** field, select tool presentation: Dropdown (drop-down) or Flat (horizontal).

     Note: this parameter is used essentially for drop-down menus.
   - Select **Add End Separator** if you want to add a separator bar at the end of the tool group.
   - Select **Is Static** to avoid recalculation of the Menu in Web.
10. Click **OK**.



Name: View

Presentation

_GUIName: View

MetaPicture:

Display Mode: Name and icon

MetaCommand Group Renderer: Dropdown

□ Add End Separator

□ Is Static

Behavior

Implementing MetaCommand Item:

Macro:

Parent MetaCommand Group

Toolbar Position: Left

8.6.4 Defining toolbar tool group commands

To define commands of a toolbar tool group:

1. In **HOPEX**, display the **MetaStudio** navigation window.
2. Expand the **MEGA Application** folder, then the application and desktop (example: "Desktop MA A").
3. Right-click the desktop tool group (example: "Main Toolbar MA A") and select **Properties**.
4. Select the **Description** tab.
5. In the **MetaCommand Group** field (example: "Main Toolbar MA A"), click the arrow and select **Properties** of **MetaCommand Group** (example: "Main Toolbar MA A").
6. Select the **Component** tab.
7. Right-click the tool group (example: **MetaCommand Group** "View") and select **Properties**.
8. Select the **Component** tab.

9. Click **Connect** (or **New** if the MetaCommand Item, MetaCommand Group or MEGA Parameterized Tool does not yet exist).

   ➢ The MetaClass selection dialog box opens.

10. Select the object type you want to connect. **MetaCommand Item**, **MetaCommand Group** or **MEGA Parameterized Tool**.

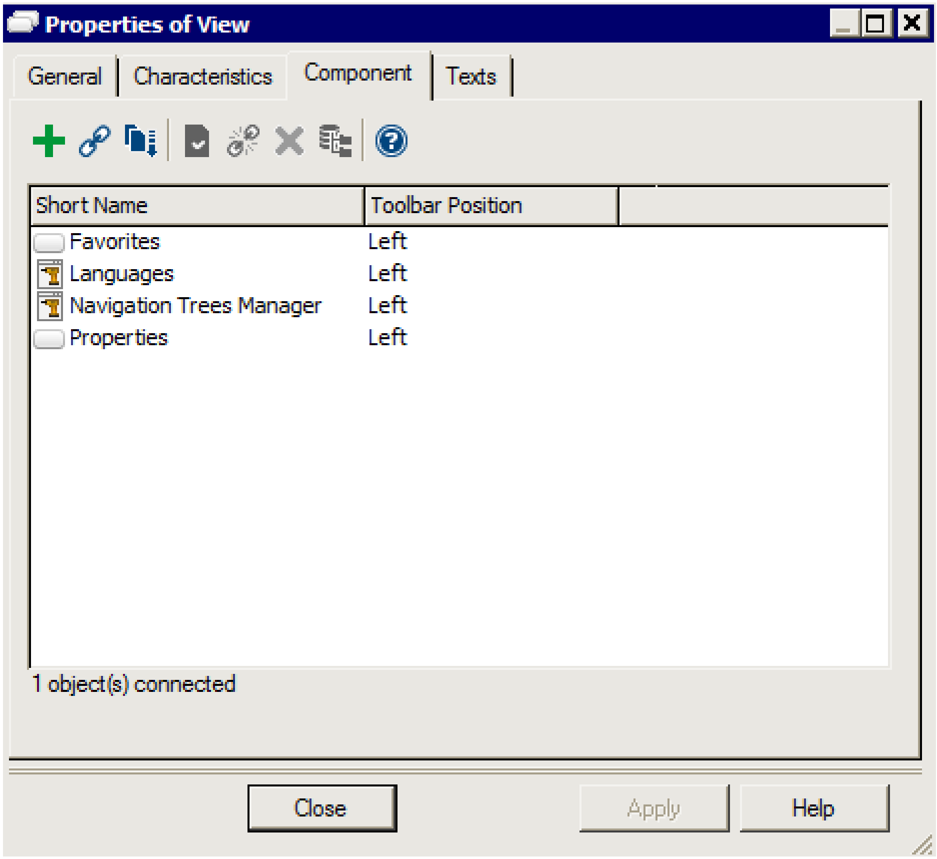
11. Click **Find**.

12. Select the object (MetaCommand Item, MetaCommand Group or MEGA Parameterized Tool) you want to connect.

13. Click **OK**.

14. Repeat steps 9 to 13 and create and/or connect all the components of the MetaCommand Group (example: "View").

    Example: Connect the **MetaCommand Items** "Favorites" and "Properties" and the **MEGA Parameterized Tools** "Navigation Trees Manager" and "Languages".



Short Name	Toolbar Position
Favorites	Left
Languages	Left
Navigation Trees Manager	Left
Properties	Left

1 object(s) connected

15. To change order of tool group components, see Modifying the position of a tool group in the toolbar.

16. Similarly specify each tool group (**MetaCommand Group**) of the toolbar.

8.6.5 Configuring toolbar commands display

To configure characteristics of commands (MetaCommand Groups):

1. In **HOPEX**, display the **MetaStudio** navigation window.
2. Expand the **MEGA Application** folder, then the application and desktop (example: "Desktop MA A").
3. Right-click the desktop tool group (example: "Main Toolbar MA A") and select **Properties**.
4. Select the **Description** tab.
5. In the **MetaCommand Group** field (example: "Main Toolbar MA A"), click the arrow and select **Properties** of the MetaCommand Group (example: "Main Toolbar MA A").
6. Select the **Component** tab.
7. In the list of toolbar components, right-click a component (example: "View") and select **Properties**.
8. Select the **Component** tab.
9. Right-click the **MetaCommand Item** for which you want to configure display and select **Properties**.
10. Select the **Characteristics** tab.
11. In the **Presentation** frame:
    - in the **_GUIName** field, enter the name under which the command will appear in the user interface.
    - in the **MetaPicture** field, click the arrow and select the image of the command that will appear in the user interface.
    - in the **Display Mode** field, select the display mode (image only, name only, or name and image).
    - Select **Is Separator** to add a separator between commands.
12. Click **OK**.

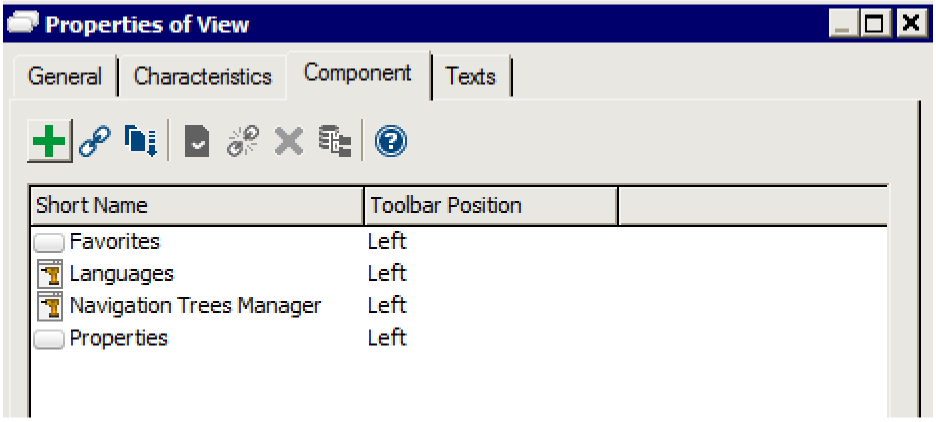
8.6.6 Command group configuration examples

Having defined commands of each group of commands of the desktop toolbar, you must configure each command (**MetaCommand Group**/**MetaCommand Item**).

Example 1: MetaCommand Group containing a **MEGA Parameterized Tool**

To configure a MetaCommand Group (example: "Navigation Tree") of a desktop toolbar:

1. In **HOPEX**, display the **MetaStudio** navigation window.
2. Expand the **MEGA Application** folder, then the application and desktop (example: "Desktop MA A").
3. Right-click the desktop tool group (example: "Main Toolbar MA A") and select **Properties**.
4. Select the **Description** tab.
5. In the **MetaCommand Group** field (example: "Main Toolbar MA A"), click the arrow and select **Properties** of the MetaCommand Group (example: "Main Toolbar MA A").
6. Select the **Component** tab.
7. Select the MetaCommand Group (example: "View") and open its Properties.
8. Select the **Component** tab.
9. Click **Connect** and select the MEGA Parameterized Tool (example: "**Navigation Trees Manager**") required.



Short Name	Toolbar Position
Favorites	Left
Languages	Left
Navigation Trees Manager	Left
Properties	Left

10. Click **OK**.

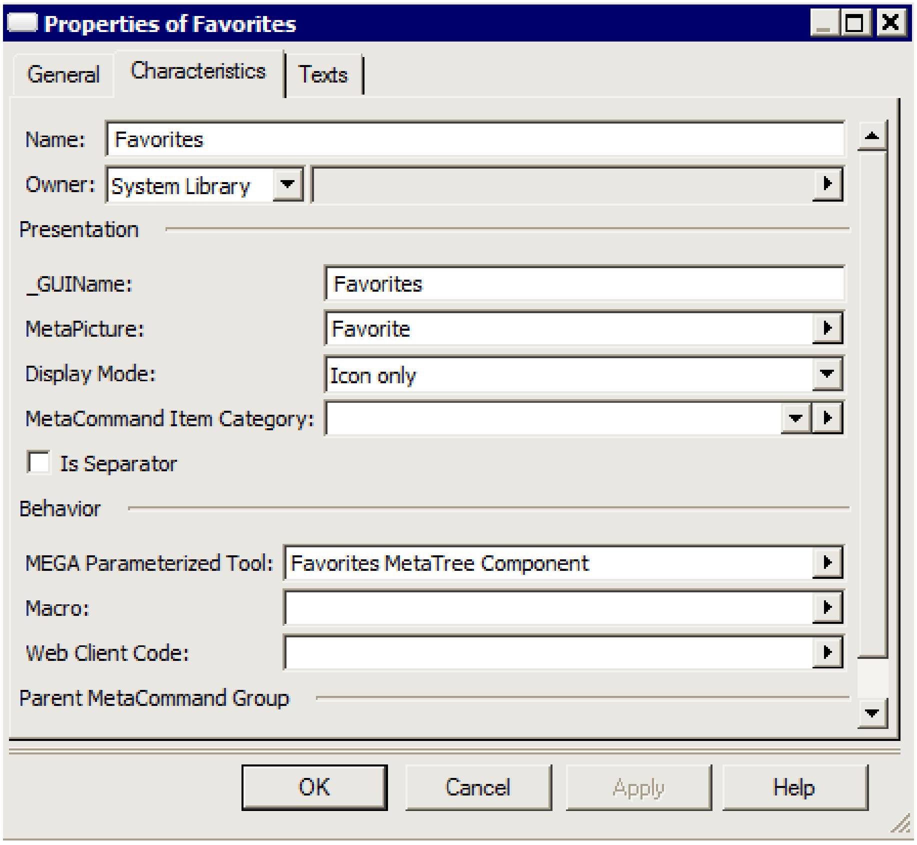
➔ In our case, the Left Container of the desktop is the Accordion type Container presenting navigation trees (see Creating a Desktop Container of Accordion type).

The **MEGA Parameterized Tool** "Navigation Trees Manager" is connected to Left Container. It shows the tools available in the selected Containers and hides the others. The Container is the tool parameter.

Example 2: **MetaCommand Group** containing a **MetaCommand Item** implemented by a **MEGA Parameterized Tool**

To configure a MetaCommand Item (example: "Favorites") of a desktop toolbar:

1. In **HOPEX**, display the **MetaStudio** navigation window.
2. Expand the **MEGA Application** folder, then the application and desktop (example: "Desktop MA A").
3. Right-click the desktop tool group (example: "Main Toolbar MA A") and select **Properties**.
4. Select the **Description** tab.
5. In the **MetaCommand Group** field (example: "Main Toolbar MA A"), click the arrow and select **Properties** of the MetaCommand Group (example: "Main Toolbar MA A").
6. Select the **Component** tab.
7. In the list of toolbar components, right-click a component (example: "View") and select **Properties**.
8. Select the **Component** tab.
9. Select the MetaCommand Item (example: "Favorites") and open its **Properties**.
10. Select the **Characteristics** tab.
11. In the **Behavior** frame, click the **MEGA Parameterized Tool** field arrow and select the tool (example: **Favorites MetaTree Component**) required.



Name: Favorites

Owner: System Library

Presentation

_GUIName: Favorites

MetaPicture: Favorite

Display Mode: Icon only

MetaCommand Item Category:

□ Is Separator

Behavior

MEGA Parameterized Tool: Favorites MetaTree Component

Macro:

Web Client Code:

Parent MetaCommand Group

12. Click **OK**.

Example 3: **MetaCommand Group** containing a **MetaCommand Item** implemented by a macro

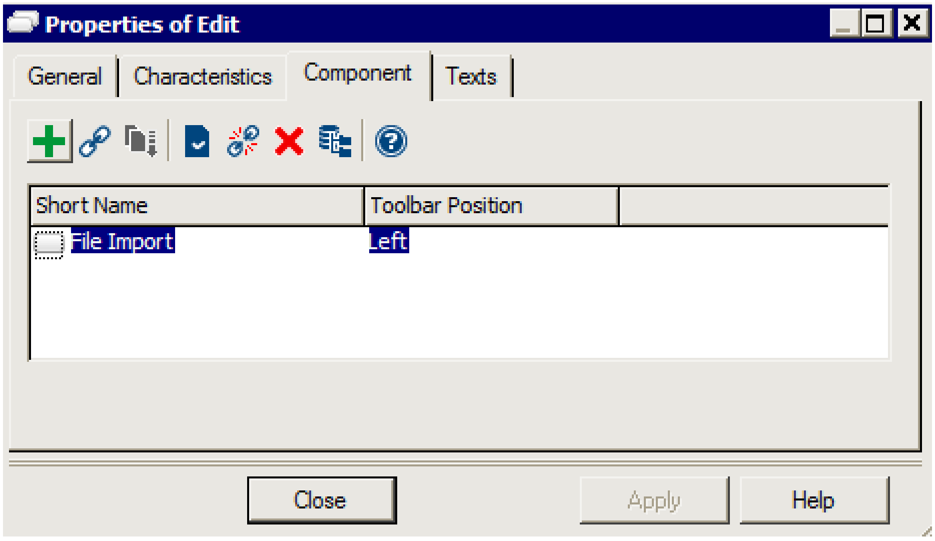
To configure a MetaCommand Item (example: "Documents") of a desktop toolbar:

1. In **HOPEX**, display the **MetaStudio** navigation window.
2. Expand the **MEGA Application** folder, then the application and desktop (example: "Desktop MA A").
3. Right-click the desktop tool group (example: "Main Toolbar MA A") and select **Properties**.
4. In the **MetaCommand Group** field (example: "Main Toolbar MA A"), click the arrow and select **Properties** of the MetaCommand Group (example: "Main Toolbar MA A").
5. Select the **Component** tab.
6. In the list of toolbar components, right-click a component (example: "Edit") and select **Properties**.
7. Select the **Component** tab.
8. Click **New** ✚.

9. In the **MetaClass** selection dialog box, select **MetaCommand item**.

10. Click **OK**.

11. In the **Name** field, enter the command name (example: "File Import").



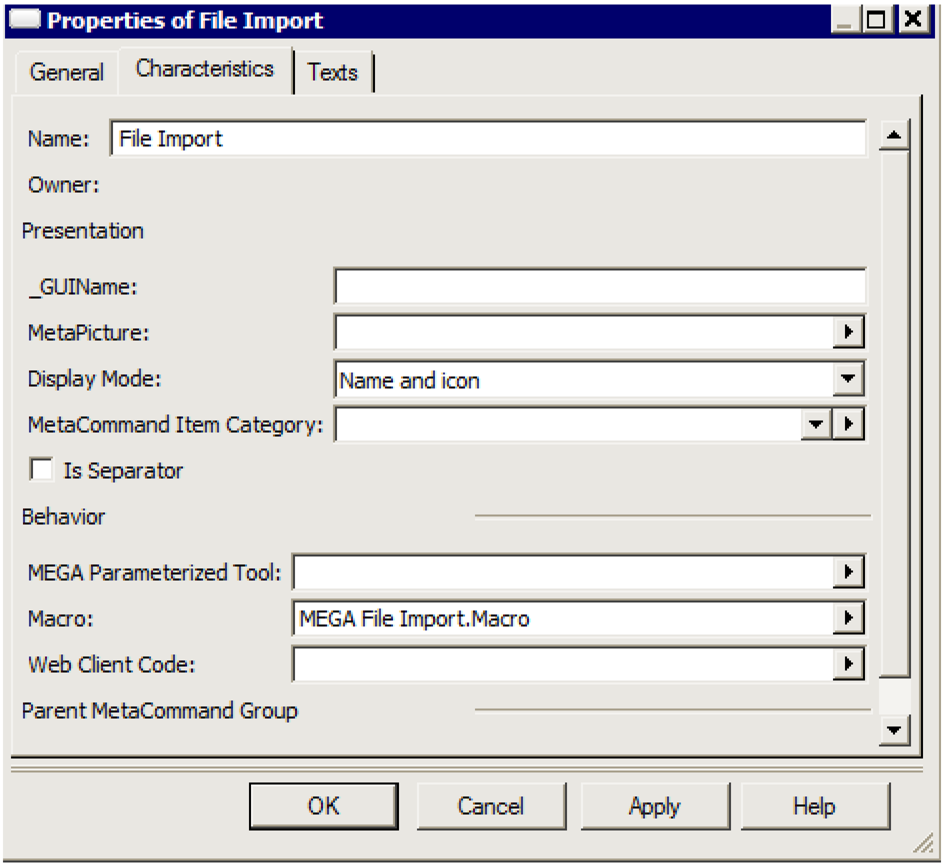
Short Name	Toolbar Position
File Import	Left

12. Open the Properties of the command you have just created (example: "File Import").

13. Select the **Characteristics** tab.

14. In the **Behavior** frame, click the **Macro** field arrow and select the required macro (example: "MEGA File Import.Macro").

15. Click **OK**.



Name: File Import

Owner:

Presentation

_GUIName:

MetaPicture:

Display Mode: Name and icon

MetaCommand Item Category:

□ Is Separator

Behavior

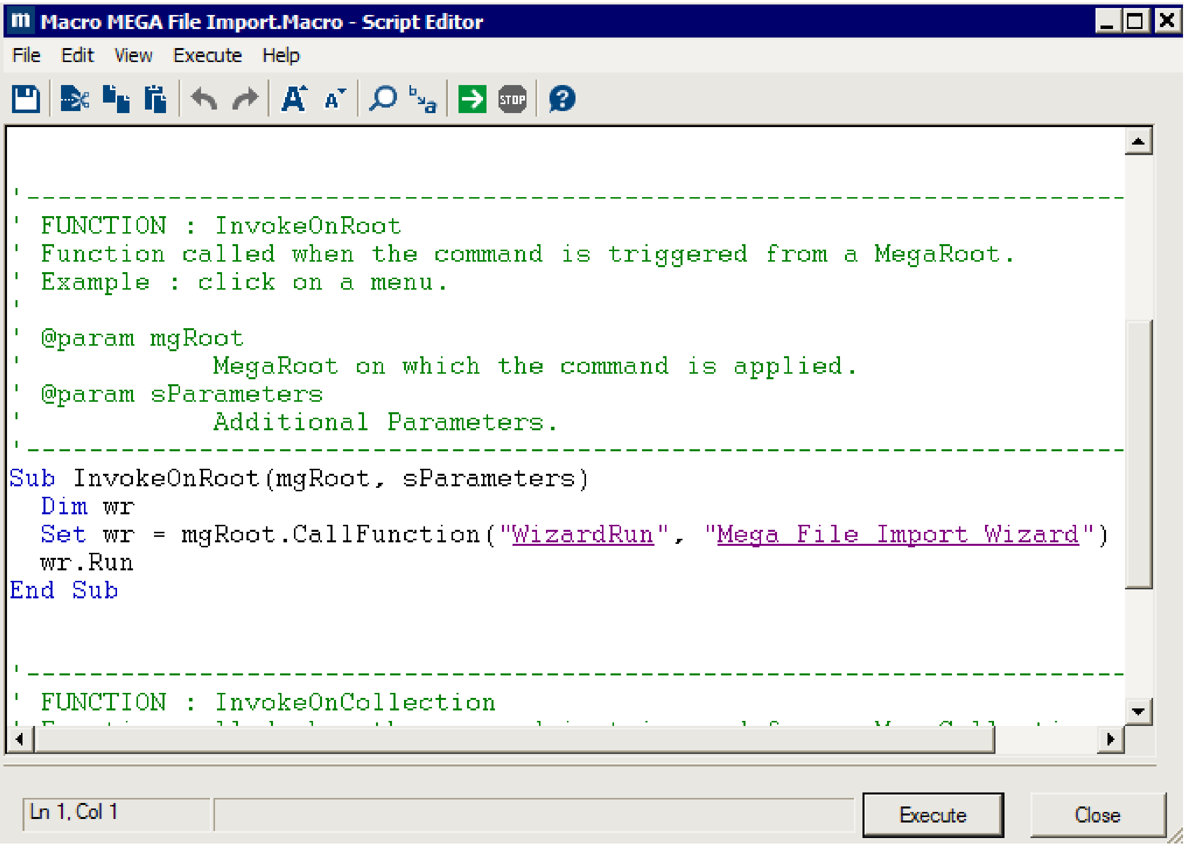
MEGA Parameterized Tool:

Macro: MEGA File Import.Macro

Web Client Code:

Parent MetaCommand Group

Example: macro "MEGA File Import.macro".



```
'-----------------------------------------------------------------------
' FUNCTION : InvokeOnRoot
' Function called when the command is triggered from a MegaRoot.
' Example : click on a menu.
'
' @param mgRoot
'            MegaRoot on which the command is applied.
' @param sParameters
'            Additional Parameters.
'-----------------------------------------------------------------------
Sub InvokeOnRoot(mgRoot, sParameters)
  Dim wr
  Set wr = mgRoot.CallFunction("WizardRun", "Mega File Import Wizard")
  wr.Run
End Sub


'-----------------------------------------------------------------------
' FUNCTION : InvokeOnCollection
```

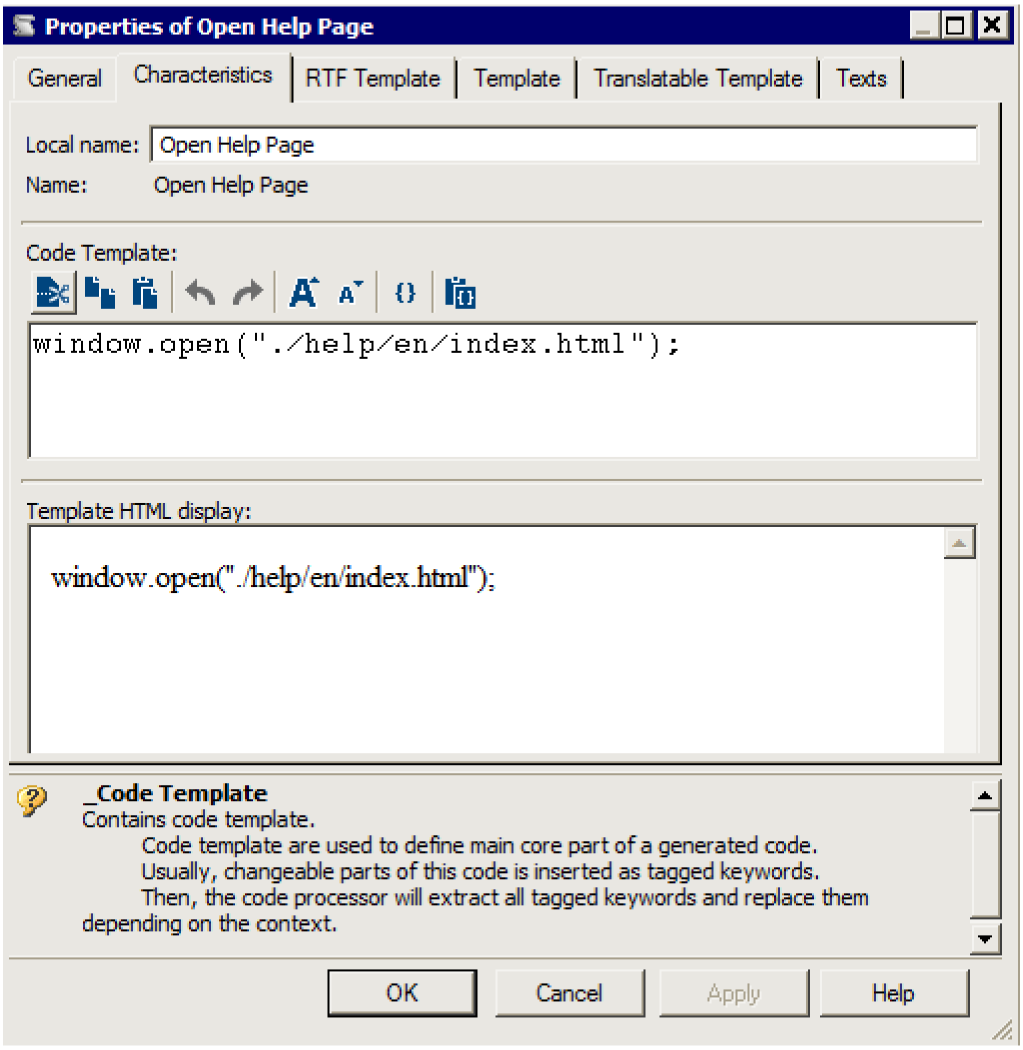
16.Click **OK**.

Example 4: MetaCommand Group containing a MetaCommand Item implemented by JavaScript code

To configure a MetaCommand Group (example: "Miscellaneous") in a desktop toolbar, containing a command implemented by JavaScript code:

1. In **HOPEX**, display the **MetaStudio** navigation window.
2. Expand the **MEGA Application** folder, then the application and desktop (example: "Desktop MA A").
3. Right-click the desktop tool group (example: "Main Toolbar MA A") and select **Properties**.
4. In the **MetaCommand Group** field (example: "Main Toolbar MA A"), click the arrow and select **Properties** of the MetaCommand Group (example: "Main Toolbar MA A").
5. Select the **Component** tab.
6. In the list of toolbar components, right-click a component (example: "Miscellaneous") and select **Properties**.
7. Select the **Component** tab.
8. Click **New** ➕.
9. In the **MetaClass** selection dialog box, select **MetaCommand item**.
10. Click **OK**.
11. in the **Name** field, enter the command name (example: "Help").
12. Click **OK**.
13. Open the Properties of the command you have just created (example: "Help").
14. Select the **Characteristics** tab.
15. In the **Behavior** frame, click the **Web Client Code** field arrow and select the _Code Template required (example: "Open Help Page").

Example: _Code Template "Open Help Page"



Local name: Open Help Page

Name: Open Help Page

Code Template:

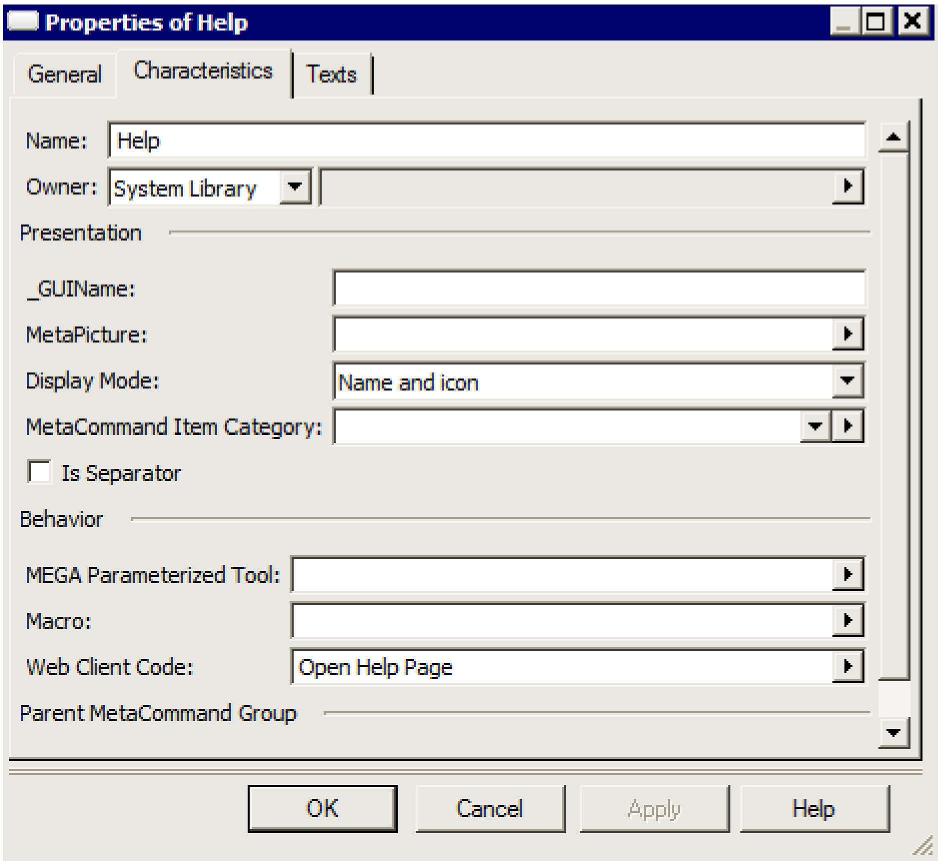
```
window.open("./help/en/index.html");
```

Template HTML display:

window.open("./help/en/index.html");

_Code Template
Contains code template.
Code template are used to define main core part of a generated code.
Usually, changeable parts of this code is inserted as tagged keywords.
Then, the code processor will extract all tagged keywords and replace them depending on the context.

16.Click **OK**.


Properties of Help

General | Characteristics | Texts

Name: Help

Owner: System Library

Presentation

_GUIName:

MetaPicture:

Display Mode: Name and icon

MetaCommand Item Category:

□ Is Separator

Behavior

MEGA Parameterized Tool:

Macro:

Web Client Code: Open Help Page

Parent MetaCommand Group

17.Click **OK**.

8.7 Customizing the Desktop style sheet

According to your needs, you may need to modify the Desktop style sheet. In that case, be careful that its color match with the default icon colors.

Else you can overload the default icons.

8.7.1 Modifying the Desktop style sheet

By default HOPEX (Web Front-End) desktops use the same style sheet. This style sheet is defined in **HOPEX Options > Installation > Web Application > Theme used in Web Application**.

Some of them already use their own style sheet.

> For example: HOPEX Explorer uses "HOPEX Explorer V2R1 Style" style sheet, Solutions like HOPEX GDPR based on Universal Desktop use the "HOPEX V2R1 Style" style sheet).

To modify the desktop style sheet:

1. In **HOPEX**, display the **MetaStudio** navigation window.
2. Expand the **Desktops** folder, and access the properties of the desktop concerned.

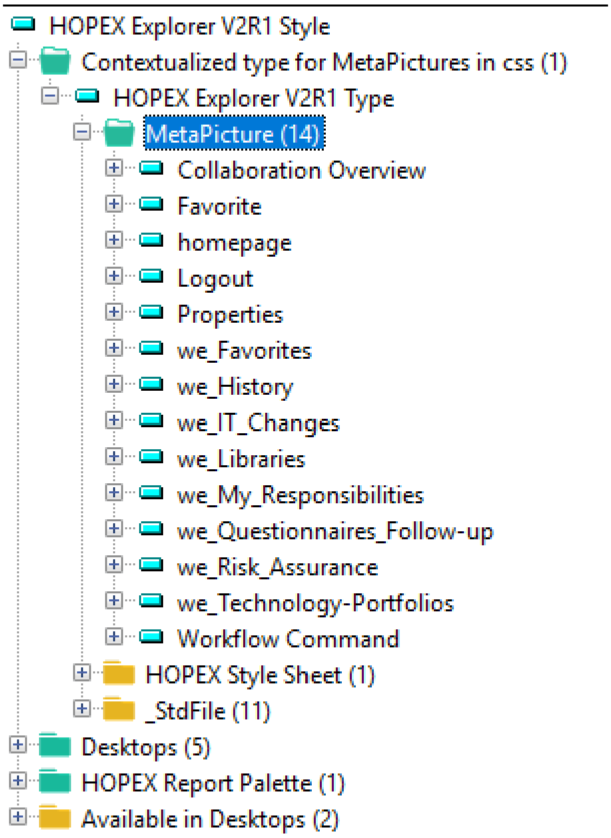
3. in the **Characteristics** tab, in **HOPEX Style Sheet** field, click the arrow and select **Connect HOPEX Style Sheet**.
4. Connect the HOPEX Style Sheet.

   When you change the style sheet, be careful that its color match with the default icons. Else you can customize the icons.

8.7.2 Customizing icons for a specific style sheet

You can customize icons according to a Solution. To customize icons for a specific style sheet, you need to define a type, and connect it to the style sheet.

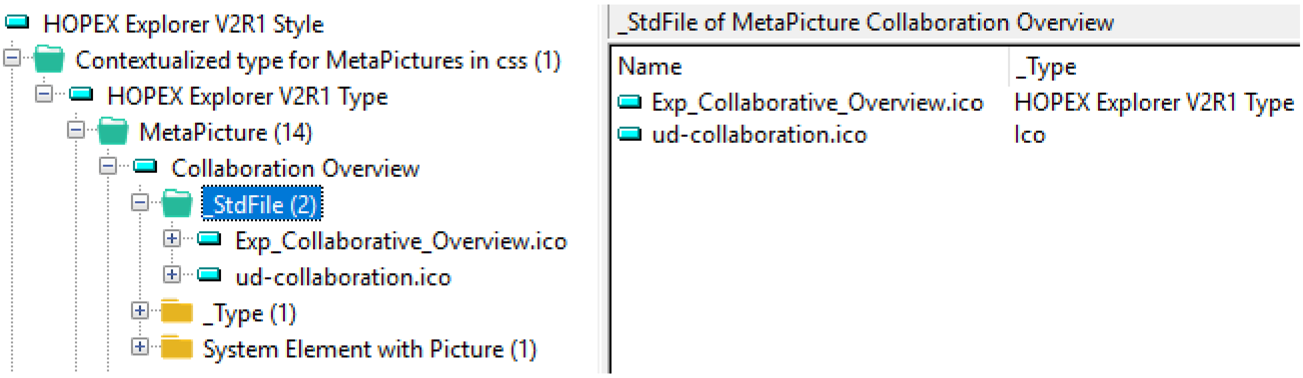
```
Example:

"HOPEX Explorer V2R1 Type" is connected to "HOPEX Explorer V2R1 Style" style sheet. It includes 14 pictures, which overload the corresponding 14 standard pictures in the context of "HOPEX Explorer V2R1 Style".
```



HOPEX Explorer V2R1 Style
- Contextualized type for MetaPictures in css (1)
  - HOPEX Explorer V2R1 Type
    - MetaPicture (14)
      - Collaboration Overview
      - Favorite
      - homepage
      - Logout
      - Properties
      - we_Favorites
      - we_History
      - we_IT_Changes
      - we_Libraries
      - we_My_Responsibilities
      - we_Questionnaires_Follow-up
      - we_Risk_Assurance
      - we_Technology-Portfolios
      - Workflow Command
    - HOPEX Style Sheet (1)
    - _StdFile (11)
- Desktops (5)
- HOPEX Report Palette (1)
- Available in Desktops (2)

An object MetaPicture (icon) is associated with an _StdFile. The icon displayed depends on the desktop style sheet.

```
Example:

For "Collaboration Overview", the MetaPicture used in the context of "HOPEX Explorer V2R1 Style", is Exp_Collaboration_Overview.ico instead of ud_collabortaion.ico.
```



HOPEX Explorer V2R1 Style
- Contextualized type for MetaPictures in css (1)
  - HOPEX Explorer V2R1 Type
    - MetaPicture (14)
      - Collaboration Overview
        - StdFile (2)
          - Exp_Collaborative_Overview.ico
          - ud-collaboration.ico
        - _Type (1)
        - System Element with Picture (1)

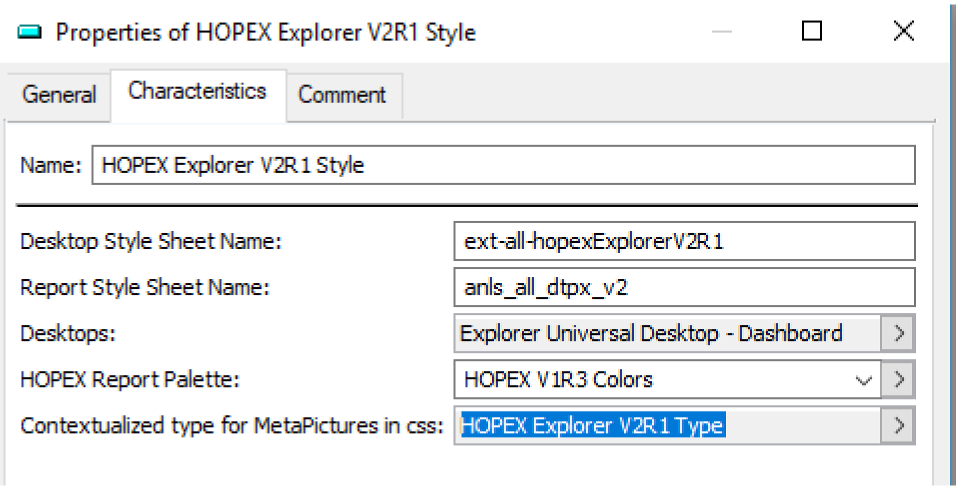
_StdFile of MetaPicture Collaboration Overview

Name	_Type
Exp_Collaborative_Overview.ico	HOPEX Explorer V2R1 Type
ud-collaboration.ico	Ico

To create a type:

1. Access the HOPEX Style Sheet properties associated with the Desktop concerned.
2. In its **Characteristics** tab, in the **Contextualized type for MetaPictures in css** field, click the arrow and select **Create_Type**.

   For example, in the "HOPEX Explorer V2R1 Style" style sheet properties, the "HOPEX Explorer V2R1 Type" is connected.



Properties of HOPEX Explorer V2R1 Style

Name: HOPEX Explorer V2R1 Style

Desktop Style Sheet Name: ext-all-hopexExplorerV2R1

Report Style Sheet Name: anls_all_dtpx_v2

Desktops: Explorer Universal Desktop - Dashboard

HOPEX Report Palette: HOPEX V1R3 Colors

Contextualized type for MetaPictures in css: HOPEX Explorer V2R1 Type

8.8 Defining a context-specific display for a Desktop Container

You can define the way containers are displayed (**Context Display Mode**: opened or closed) in a specific context at initialization.

The context applies on:

- tools
- macros: EmitCurrent in a context (e.g.: floating toolbar, Activity feed)

To do so, you need to create a Desktop Container Context.

To customize a Desktop Container default behavior:

1. In **HOPEX**, display the **MetaStudio** navigation window.

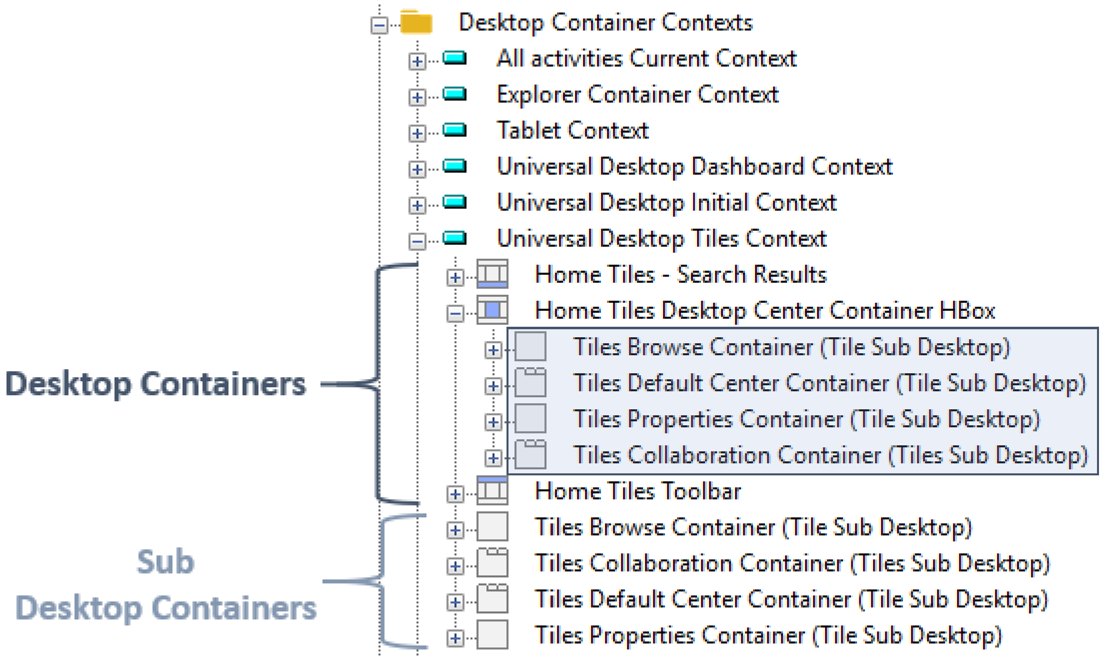
2. Right-click the **Desktop Container Contexts** folder and select **New > Desktop Container Context**.

3. In the **Name** field enter a name for the context.

4. Click **OK**.

5. Expand the **Desktops** folder and drag and drop all the Desktop Containers (including needed sub Desktop Containers) concerned in the Desktop Container Context you created.

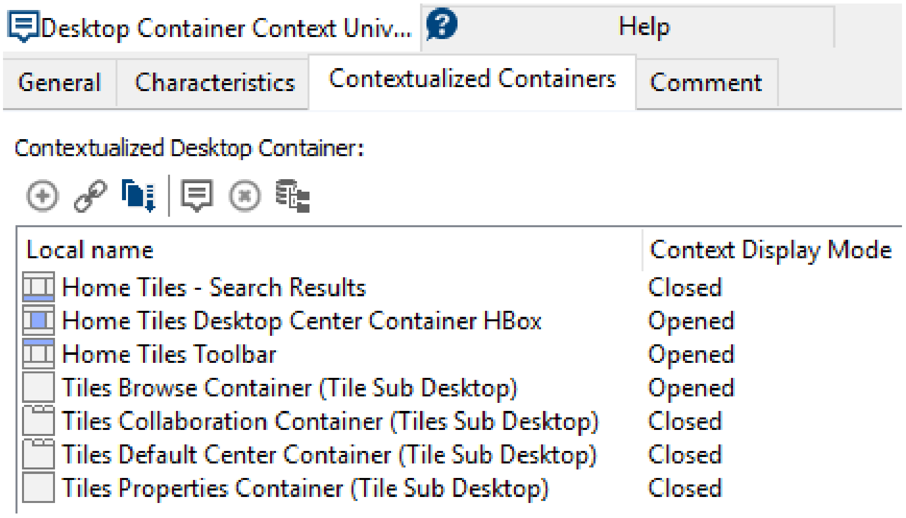
   For example: the four sub Desktop Containers of ''Home Tiles Desktop Center Container Hbox'' Desktop Container are also added in the Desktop Container Context.



- Desktop Container Contexts
  - All activities Current Context
  - Explorer Container Context
  - Tablet Context
  - Universal Desktop Dashboard Context
  - Universal Desktop Initial Context
  - Universal Desktop Tiles Context
    - Home Tiles - Search Results
    - Home Tiles Desktop Center Container HBox
      - Tiles Browse Container (Tile Sub Desktop)
      - Tiles Default Center Container (Tile Sub Desktop)
      - Tiles Properties Container (Tile Sub Desktop)
      - Tiles Collaboration Container (Tiles Sub Desktop)
    - Home Tiles Toolbar
    - Tiles Browse Container (Tile Sub Desktop)
    - Tiles Collaboration Container (Tiles Sub Desktop)
    - Tiles Default Center Container (Tile Sub Desktop)
    - Tiles Properties Container (Tile Sub Desktop)

6. In the Desktop Container Context properties, select the **Contextualized Containers** tab.

7. For each Desktop Container (and sub desktop Container), in its **Context Display Mode** field select its context display mode.



Contextualized Desktop Container:

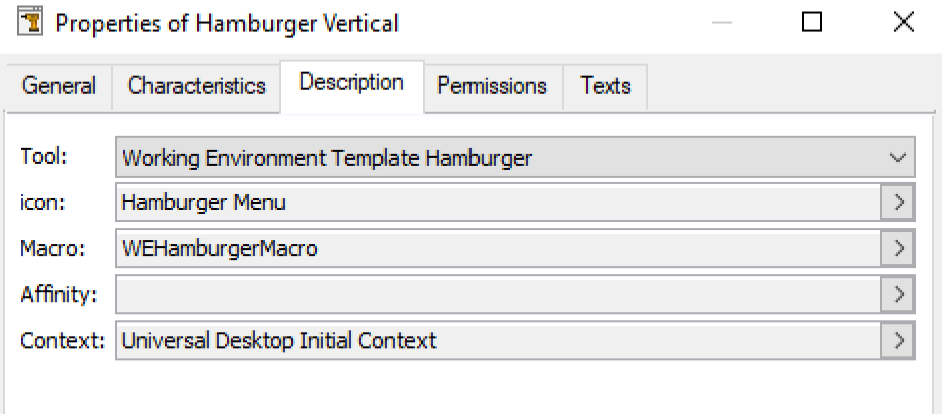
Local name	Context Display Mode
Home Tiles - Search Results	Closed
Home Tiles Desktop Center Container HBox	Opened
Home Tiles Toolbar	Opened
Tiles Browse Container (Tile Sub Desktop)	Opened
Tiles Collaboration Container (Tiles Sub Desktop)	Closed
Tiles Default Center Container (Tile Sub Desktop)	Closed
Tiles Properties Container (Tile Sub Desktop)	Closed

8. Access the properties of the MEGA Parameterized Tool concerned by the context.

9. In its **Description** tab, in the **Context** field, connect the Desktop Container Context you created.

   Example: in the ''Hamburger vertical'' **Mega Parameterized Tool**, the ''Universal Desktop Initial Context'' **Desktop Container Context** is connected to:

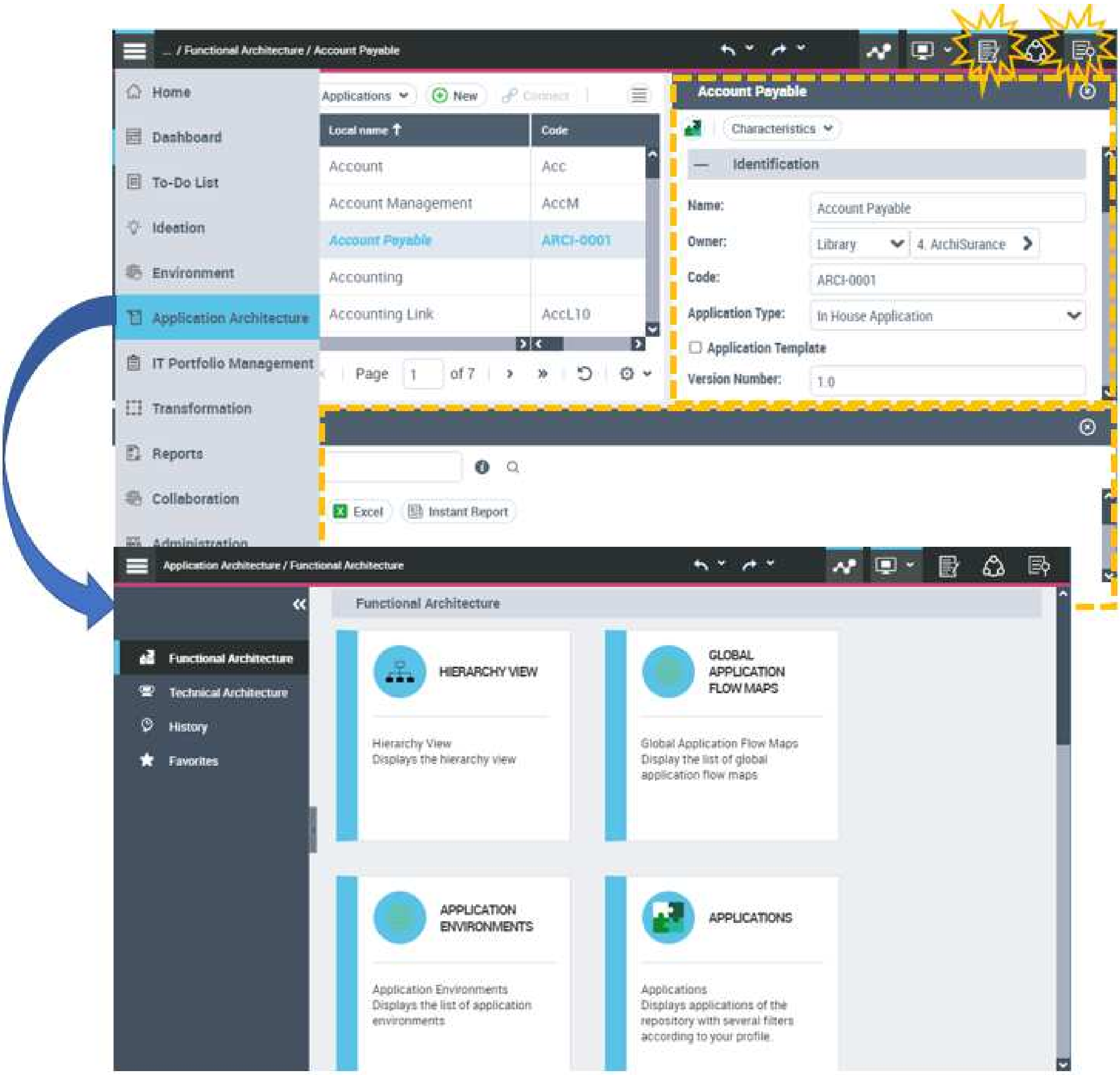
   - the ''Working Environment Template Hamburger'' **Tool** and
   - the ''WEHamburgerMacro'' **macro**



Properties of Hamburger Vertical

Tool: Working Environment Template Hamburger
icon: Hamburger Menu
Macro: WEHamburgerMacro
Affinity:
Context: Universal Desktop Initial Context

So that, in HOPEX, when the user clicks a Navigation menu, the Edit area is displayed (“Home Tiles Desktop Center Container Hbox and Home Tiles Toolbar”).



Local name ↑	Code
Account	Acc
Account Management	AccM
Account Payable	ARCI-0001
Accounting	
Accounting Link	AccL10

Page 1 of 7

Account Payable

— Identification

Name:	Account Payable
Owner:	Library 4. ArchiSurance
Code:	ARCI-0001
Application Type:	In House Application
☐ Application Template	
Version Number:	1.0

Functional Architecture

HIERARCHY VIEW

Hierarchy View
Displays the hierarchy view

GLOBAL APPLICATION FLOW MAPS

Global Application Flow Maps
Display the list of global application flow maps

APPLICATION ENVIRONMENTS

Application Environments
Displays the list of application environments

APPLICATIONS

Applications
Displays applications of the repository with several filters according to your profile.

9 MODIFYING A DESKTOP

When a desktop has been created, you can modify it.

You can:

- add a **Desktop Container** to the desktop,

    ➔ see Adding a Desktop Container to a desktop.

- delete a **Desktop Container** from the desktop,

    ➔ see Deleting a Desktop Container from a desktop.

- modify position of a tool group (**MetaCommand Group**) in the toolbar,

    ➔ see Modifying the position of a tool group in the toolbar.

- modify or customize a **Container**; to do this, you must specify its characteristics.

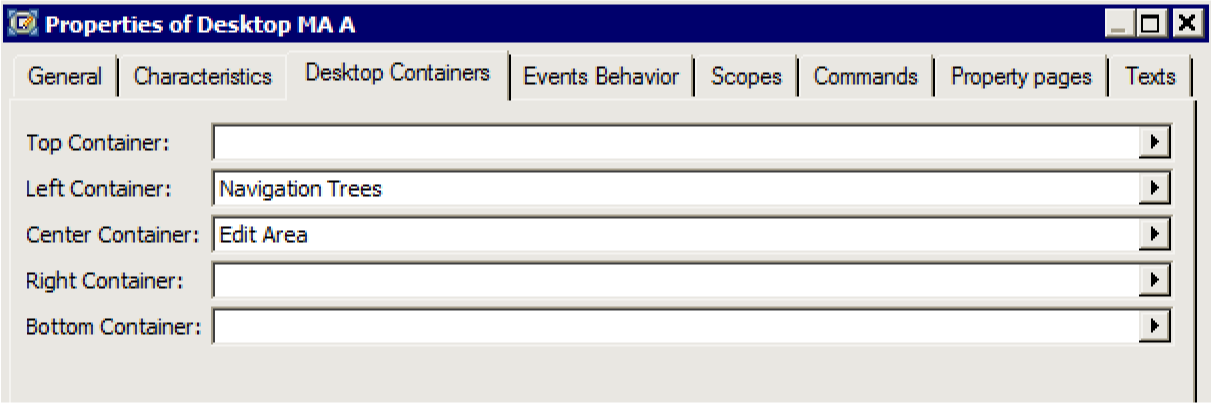
    ➔ see Defining Container characteristics.

9.1 Adding a Desktop Container to a desktop

To add a Desktop Container to an existing desktop:

1. In **HOPEX**, display the **MetaStudio** navigation window.
2. Expand the **MEGA Application** folder, then the application concerned.
3. Right-click the desktop (example: "Desktop MA A") and select **Properties**.

    ➢ The **Properties of Desktop "Desktop MA A"** appears.


Properties of Desktop MA A

Top Container:

Left Container: Navigation Trees

Center Container: Edit Area

Right Container:

Bottom Container:

4. To add a new **Desktop Container**, see Creating Desktop Containers and Step 1, start from the **Properties** of **Desktop** (instead of from the **Creation of Desktop – Desktop Containers** dialog box).

9.2 Deleting a Desktop Container from a desktop

To delete a Desktop Container from an existing desktop:

1. In **HOPEX**, display the **MetaStudio** navigation window.
2. Expand the **MEGA Application** folder, then the application concerned.
3. Expand the desktop concerned.
4. Right-click the **Desktop Container** you want to delete and select:
   - **Delete** to definitively delete the Desktop Container, or
   - **Disconnect** to retain the **Desktop Container** in the repository.

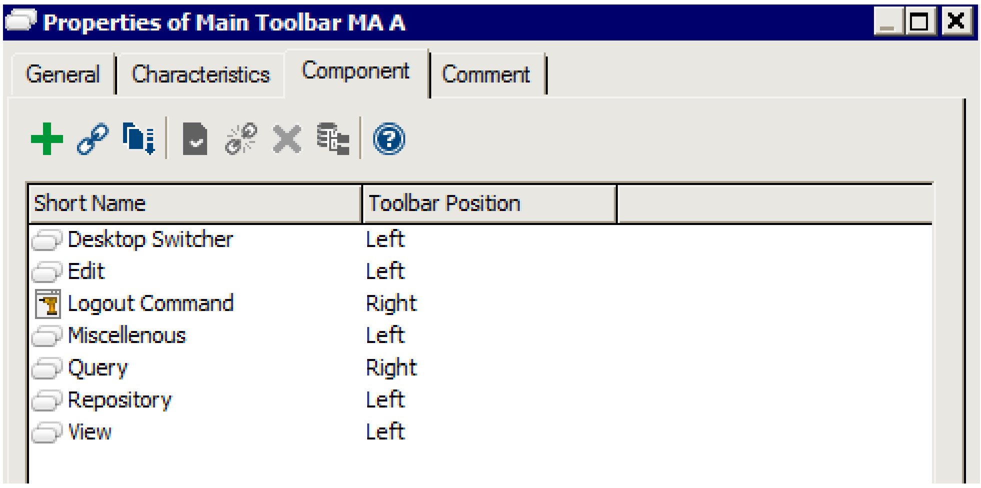
9.3 Modifying the position of a tool group in the toolbar

When the toolbar has been created, you can reorganize its tool groups.

The value of the position of an object (left/right) takes priority over its appearance order in the list.

To reorganize the toolbar:

1. In **HOPEX**, display the **MetaStudio** navigation window.
2. Expand the **MEGA Application** folder, then the application and desktop (example: "Desktop MA A").
3. Right-click the desktop tool group (example: "Main Toolbar MA A") and select **Properties**.
4. Select the **Description** tab.
5. In the **MetaCommand Group** field (example: "Main Toolbar MA A"), click the arrow and select **Properties** of the MetaCommand Group (example: "Main Toolbar MA A").
6. Select the **Component** tab.



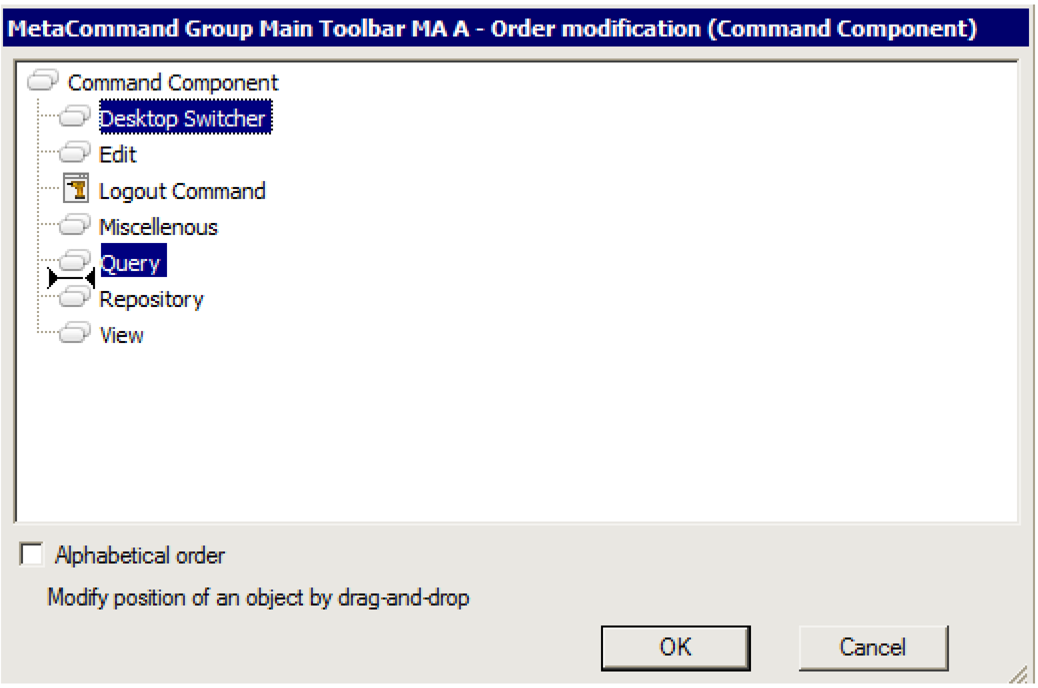
Short Name	Toolbar Position
Desktop Switcher	Left
Edit	Left
Logout Command	Right
Miscellenous	Left
Query	Right
Repository	Left
View	Left

7. Click **Reorder** .

   The **Order Modification (Command Component)** dialog box appears.

8. Clear **Alphabetic order**.

9. Select the **MetaCommand Group** you want to move and drag it to the required position.

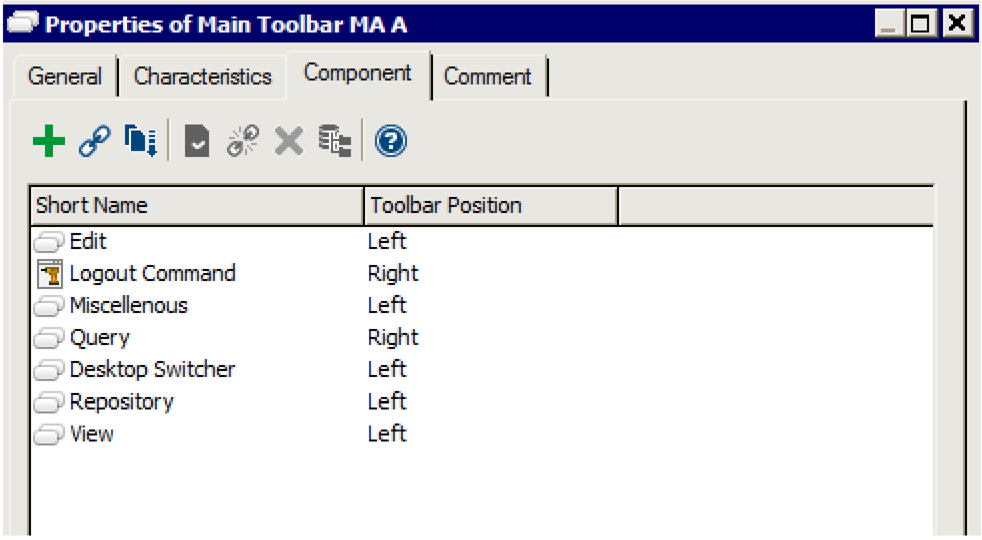

MetaCommand Group Main Toolbar MA A - Order modification (Command Component)

Command Component
- Desktop Switcher
- Edit
- Logout Command
- Miscellenous
- Query
- Repository
- View

☐ Alphabetical order

Modify position of an object by drag-and-drop

10. Click **OK**.



Short Name	Toolbar Position
Edit	Left
Logout Command	Right
Miscellenous	Left
Query	Right
Desktop Switcher	Left
Repository	Left
View	Left

10 IMPROVING A DESKTOP PERFORMANCE

10.1 Improving diagram opening time

Opening a diagram may take time. To improve the diagram opening time you can preload its DiagramType.

This improvement is customized at profile level and recommended for large diagrams.

To configure a profile to improve diagram time opening:

1. Access the profile **Properties**.
2. Display the **Tests > _Settings** tab.
3. Enter:

```
[DiagramTypePreLoad]
<DiagramType _HexaIdAbs>=1
```

At Desktop opening, the DiagramType is preloaded, in a dedicated thread with low priority so that the desktop opening is not impacted.

```
<DiagramType _HexaIdAbs>=0
```

At Desktop opening, the DiagramType is not preloaded.

Example:

Profile: Archimate Functional Administrator

"Archimate Diagram (V3)" **DiagramType _HexaIdAbs**: 347448DF59F31342

```
[DiagramTypePreLoad]
347448DF59F31342=1
```

"Archimate Diagram (V3)" DiagramType is preloaded at desktop opening, so that opening diagrams of this type is faster (can be 10 seconds faster).

11 ACTION FOLLOWING EVENT

The desktop can present tools distributed in a given space. Setup of this distribution is not however sufficient for desktop operation: if tools cannot communicate with each other, if there is no interaction between tools, the desktop is unusable.

➢ Tools must be able to communicate between themselves.

To do this, the desktop introduces two types of events in the application:

- **Current Change**: following an action (click by a user that will change the current) in **HOPEX**, the communication between two tools generates display update of the tool subscriber to the event.

  By default, the current object selected in **HOPEX** is notified to all desktop Containers and Tools. These Containers and Tools update themselves or not depending on this new current.

  > **Example:** When a dialog box is opened, clicking an object generates properties dialog box update as a function of the selected object (unless the action is overridden by the click manager).

- **Interaction**: following an action (click by a user) in **HOPEX**, communication between two tools generates an action of the tool subscriber to the event.

  > **Example:** In the Query tool, clicking ➔ generates an interaction that requests opening of a dialog box to display search results.

When a (**MEGA Parameterized Tool**) produces an item of information (example: change of state), it can dispatch this event in a Scope (this Scope is saved in the Desktop Container). Only the **Desktop Containers** or **MEGA Parameterized Tools** subscribers to this Scope will take account of this information to update their display as a function of the event (Scope), or execute another action.

11.1 Managing tool update as a function of current

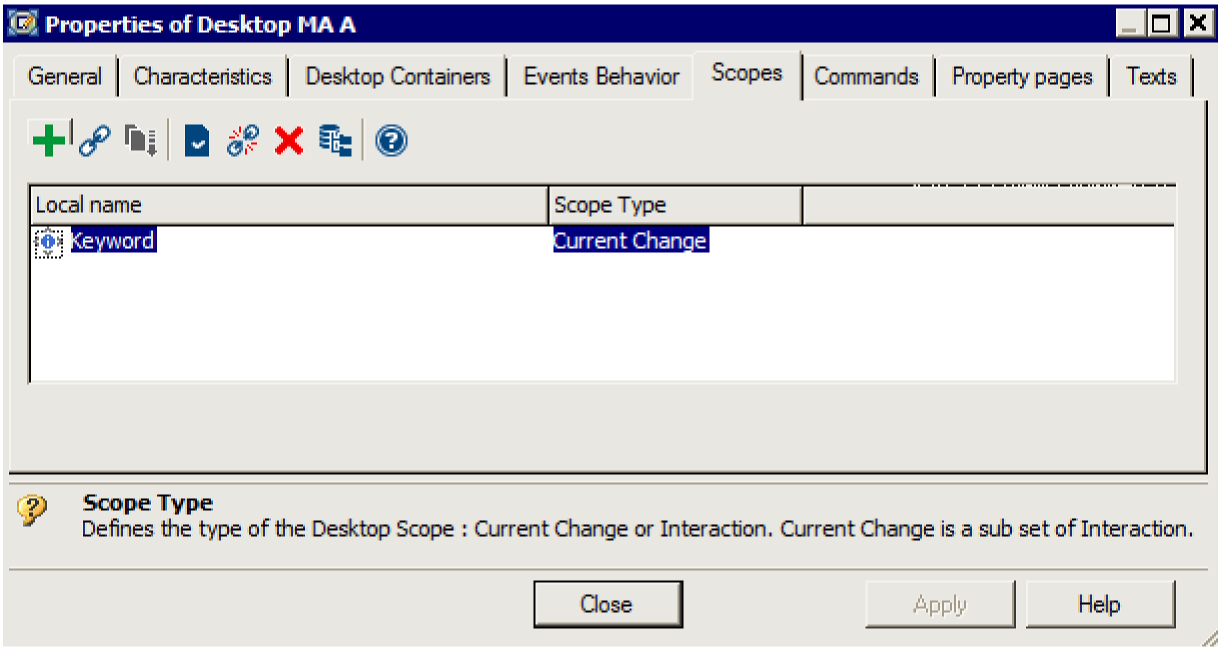
By default the desktop manages global current. This current is automatically sent to all tools, whether they process it or not.

To manage tool update as a function of another current, you must create an event (**Scope**) of **Current Change** type. This event (**Scope**) will be dispatched by a particular tool. Tools sensitive to this current will subscribe to this event (Scope). When the current changes in the source tool, the desktop immediately notifies the target tools, which react if they choose.

Defining an event (**Scope**) of current is to restrict current change to only two or three tools in the application.

To do this:

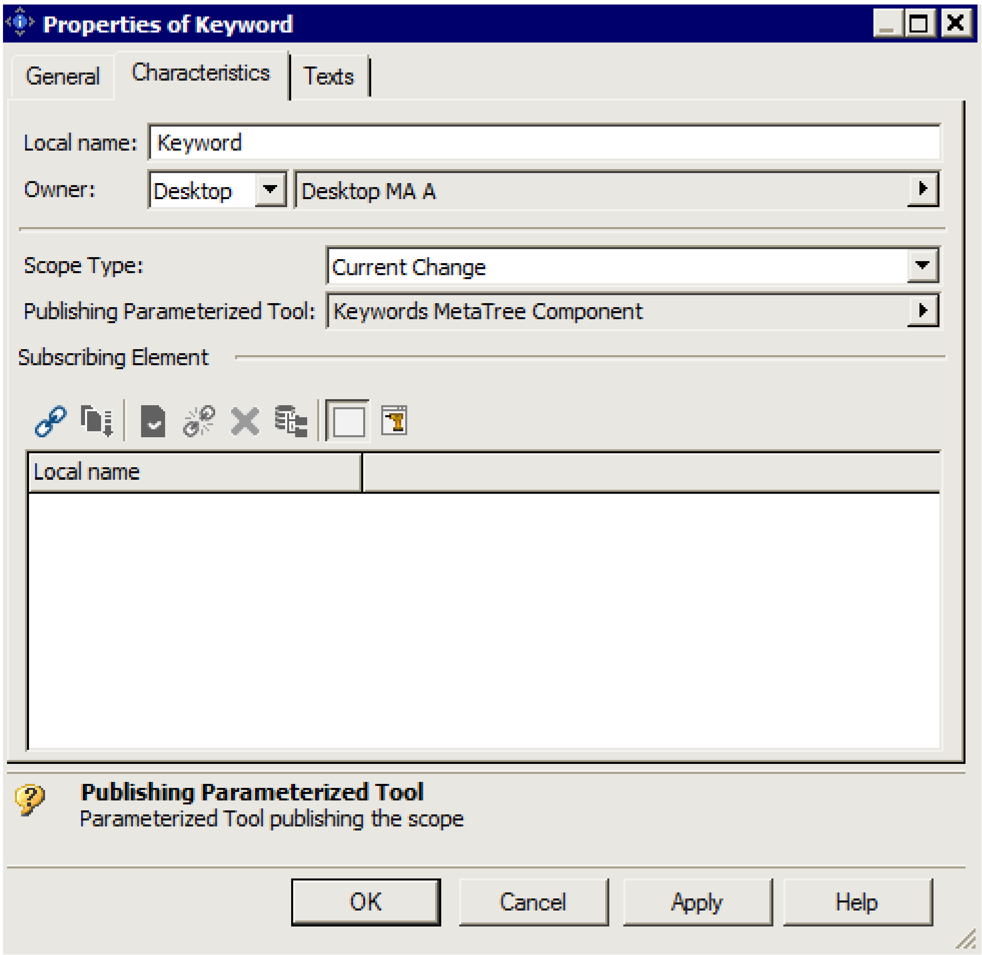
1. Open the Desktop Container **Properties** (e.g.: "Desktop MA A").
2. Select the **Scopes** tab.
3. Click **New** +.
4. In the **Local Name** field, enter the event name (Scope) (e.g.: "Keyword").
5. In the **Scope Type** field, select the **Current Change** event type.



Local name	Scope Type
Keyword	Current Change

Scope Type
Defines the type of the Desktop Scope : Current Change or Interaction. Current Change is a sub set of Interaction.

6. Open the new Scope **Properties** and select the **Characteristics** tab.
7. In the **Publishing Parameterized Tool** field, click the arrow and select the MEGA Parameterized Tool (example: "Keywords MetaTree Component") which will dispatch its modifications.
8. Click **OK**.



Properties of Keyword

Local name: Keyword

Owner: Desktop | Desktop MA A

Scope Type: Current Change

Publishing Parameterized Tool: Keywords MetaTree Component

Subscribing Element

Local name	

Publishing Parameterized Tool
Parameterized Tool publishing the scope

9. In the **Subscribing Element** frame, click **Connect** and select the **Containers** and/or **Parameterized Tools** that should be updated as a function of the event (example: modifications of MEGA Parameterized Tool "Keywords MetaTree Component").

10. Click **OK**.

11.2 Managing an action following an interaction

Generating an action following an interaction consists of managing any other event type different from the current. For example so that the action of clicking on a tool generates an action, such as opening a dialog box, you must create an event.

> **Example:** Clicking **Query** generates opening of the query results dialog box and displays the result.

To do this:

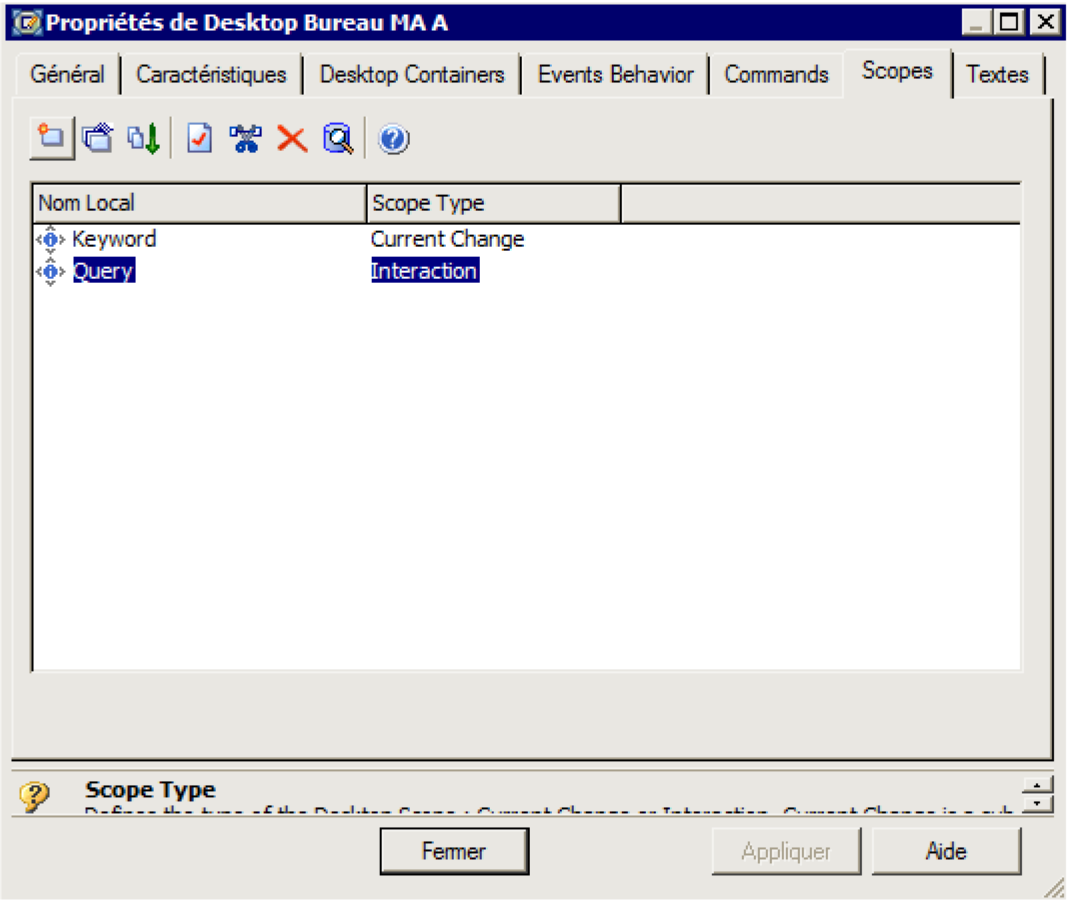
1. Open the Desktop Container **Properties** (example: "Desktop MA A").

2. Select the **Scopes** tab.

3. Click **New** .

4. In the **Local Name** field, enter the event name (Scope) (e.g.: "Query").

5. In the **Scope Type** field, select the **Interaction** event type.



Propriétés de Desktop Bureau MA A

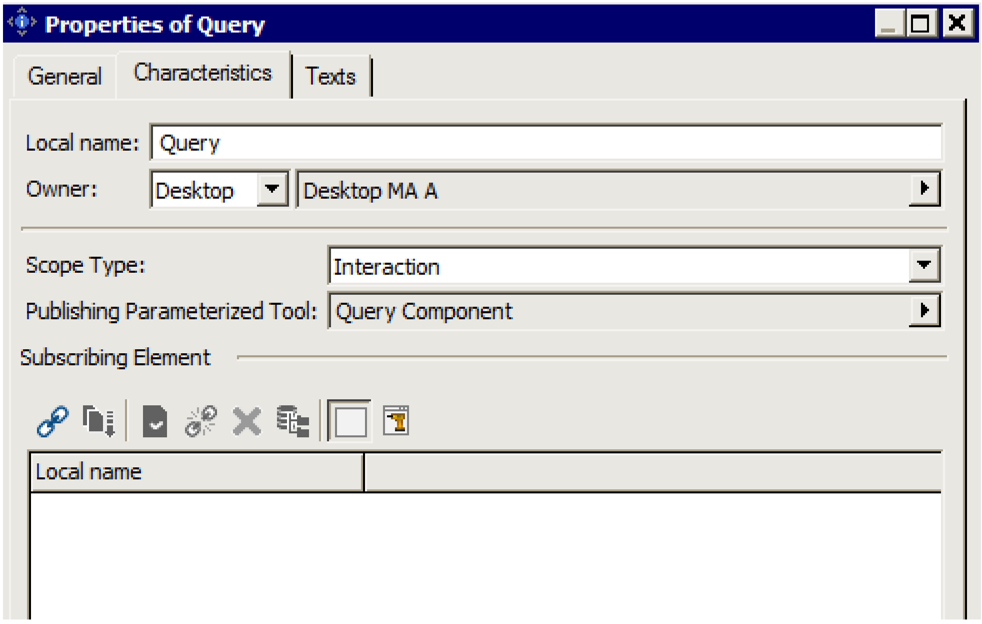
Nom Local	Scope Type
Keyword	Current Change
Query	Interaction

Scope Type

Defines the type of the Desktop Scope : Current Change or Interaction. Current Change is a sub

6. Open the new Scope **Properties** and select the **Characteristics** tab.

7. In the **Publishing Parameterized Tool** field, click the arrow and select the MEGA Parameterized Tool source of the action (example: query tool "Query Component").


Properties of Query

Local name: Query

Owner: Desktop | Desktop MA A

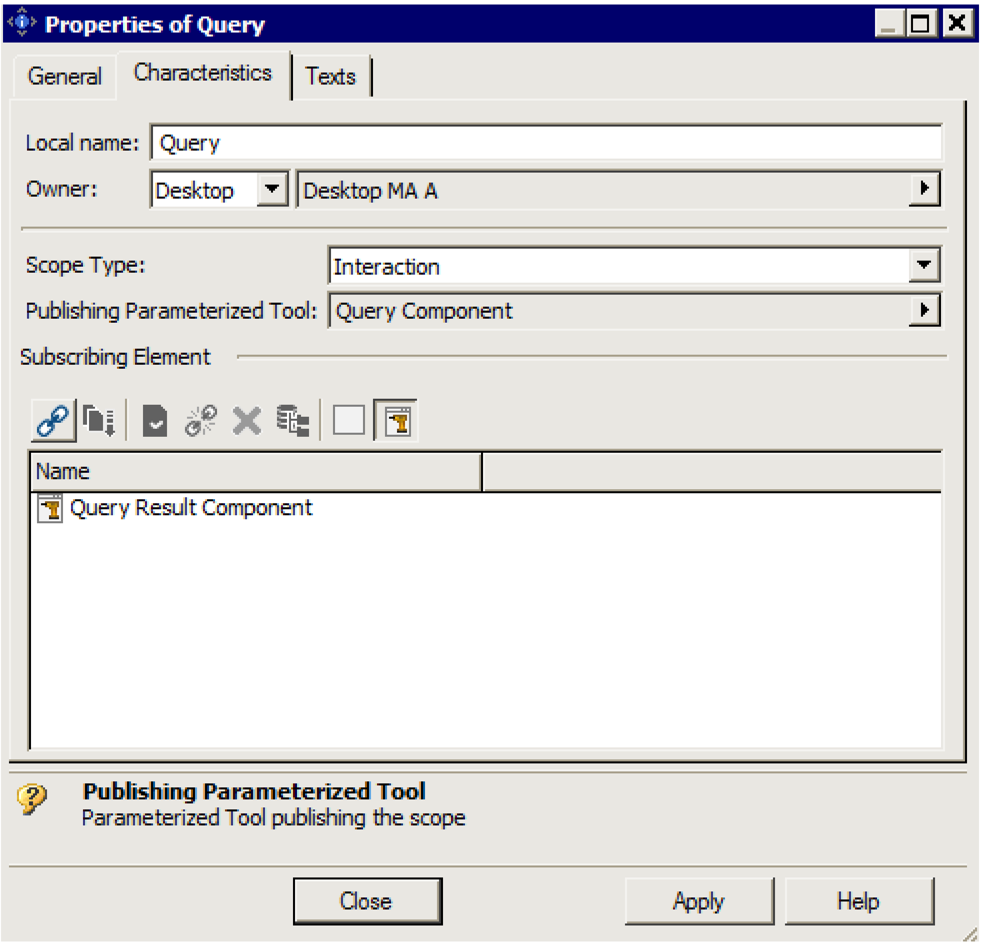
Scope Type: Interaction

Publishing Parameterized Tool: Query Component

Subscribing Element

Local name	

8. In the **Subscribing Element** frame, click Subscribing **MEGA Parameterized Tool** then **Connect** and select the tool that will present result of the action (example: query result tool "Query Result Component").


Properties of Query

Local name: Query

Owner: Desktop | Desktop MA A

Scope Type: Interaction

Publishing Parameterized Tool: Query Component

Subscribing Element

Name	
Query Result Component	

Publishing Parameterized Tool
Parameterized Tool publishing the scope

9. Click **Apply**.

12 USING A WORKING ENVIRONMENT TEMPLATE (WET)

12.1 Working Environment Template Overview

12.1.1 WET-based connection diagram

A **Person** is assigned as many **Profiles** as required.

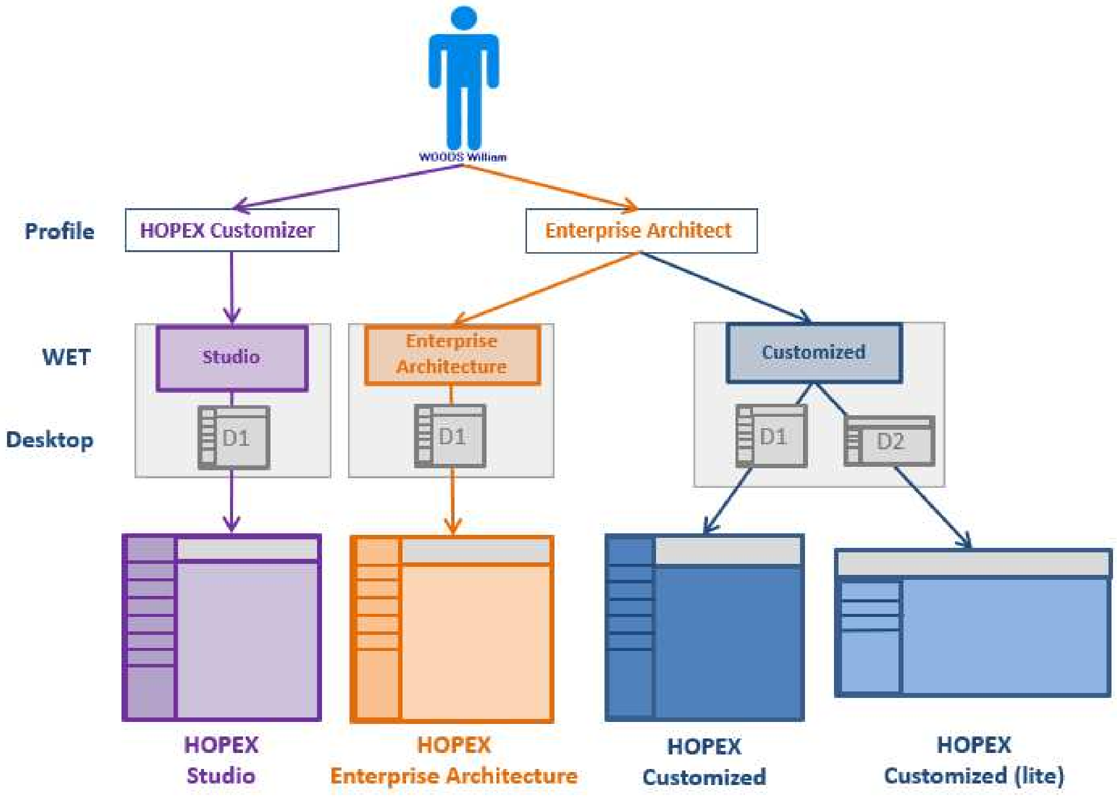
A **Profile** can be connected to one or several **Working Environment Templates** (WETs), each of them giving access to a specific desktop based on the same desktop layout (Universal Desktop).

For example:

HOPEX Customizer profile is connected to the **Studio** WET, which gives access to **HOPEX Studio** desktop.

Enterprise Architect profile is connected to both **Enterprise Architecture** WET and **Customized** WET, which give access to **HOPEX Enterprise Architecture** desktop and **HOPEX Customized** desktop respectively.

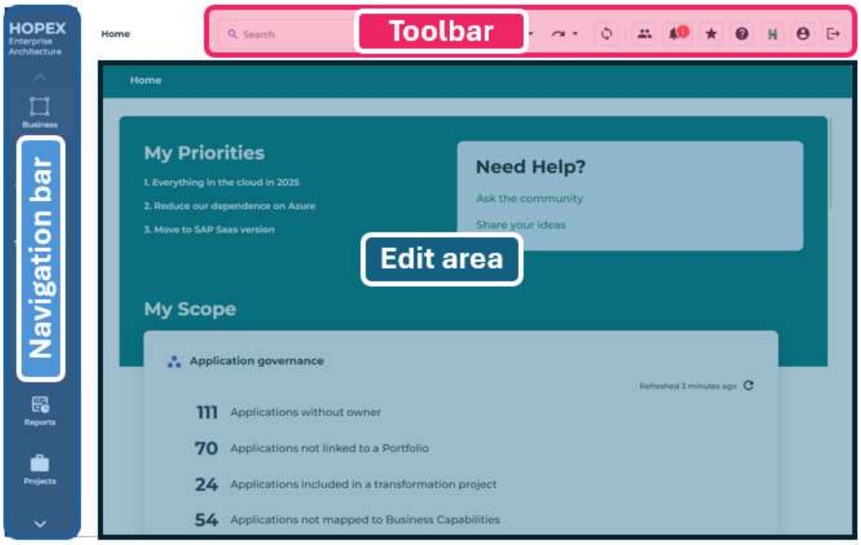
HOPEX Report Studio, **HOPEX Enterprise Architecture** and **HOPEX Customized** desktops are all based on the "Universal Desktop V6" desktop. **HOPEX Customized Lite** desktops are based on the Universal Lite desktop.



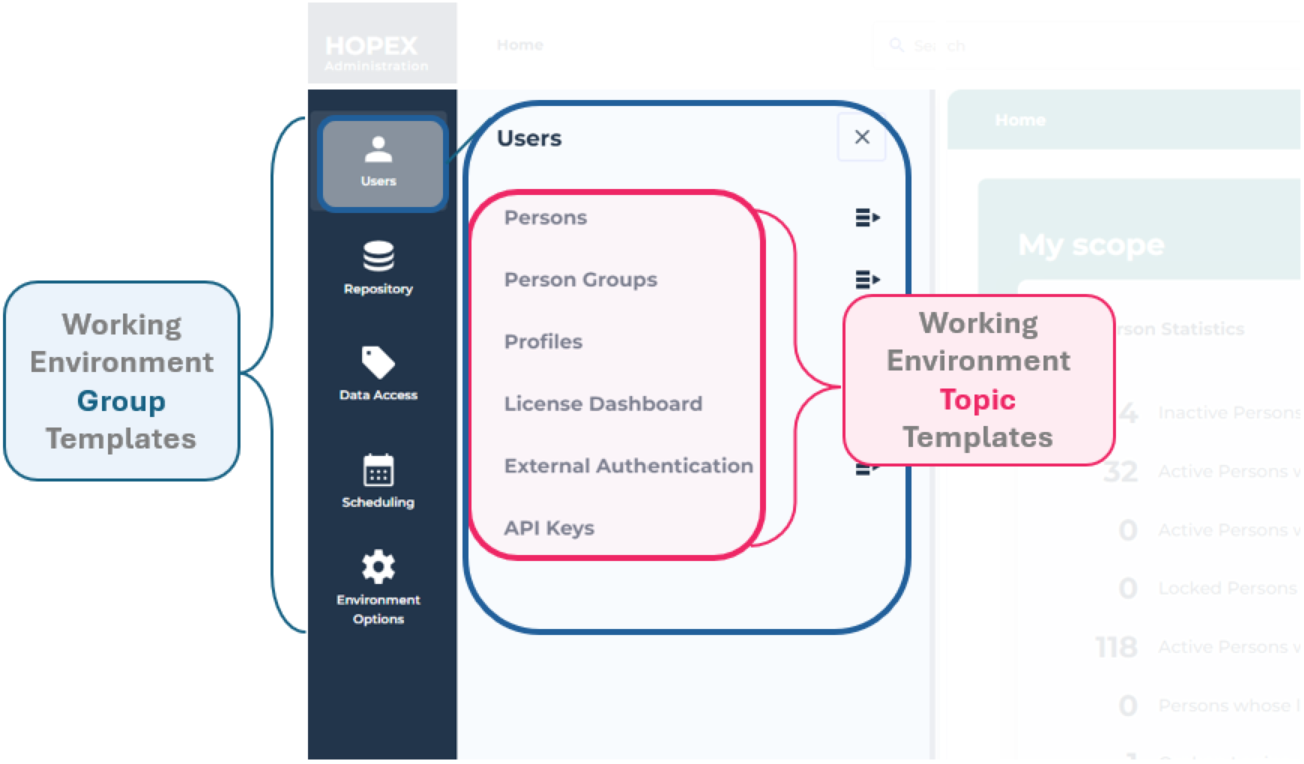
12.1.2 WET-based desktop principle

With the Working Environment Templates (WET), HOPEX desktops display the same desktop layout, with:

- at the top, a **static toolbar** common to all HOPEX desktops
- on the left, a customizable **navigation bar**
- a main container: the **Edit area**


HOPEX
Enterprise Architecture
Home
Search
Toolbar
Business
Navigation bar
Reports
Projects
Home
My Priorities
1. Everything in the cloud in 2025
2. Reduce our dependence on Azure
3. Move to SAP Saas version
Need Help?
Ask the community
Share your ideas
Edit area
My Scope
Application governance
Refreshed 3 minutes ago
111 Applications without owner
70 Applications not linked to a Portfolio
24 Applications included in a transformation project
54 Applications not mapped to Business Capabilities

The navigation bar includes navigation menus (**Working Environment Group Templates**) giving access to sub-menus (**Working Environment Topic Templates**).



12.1.3 WET-based desktop example

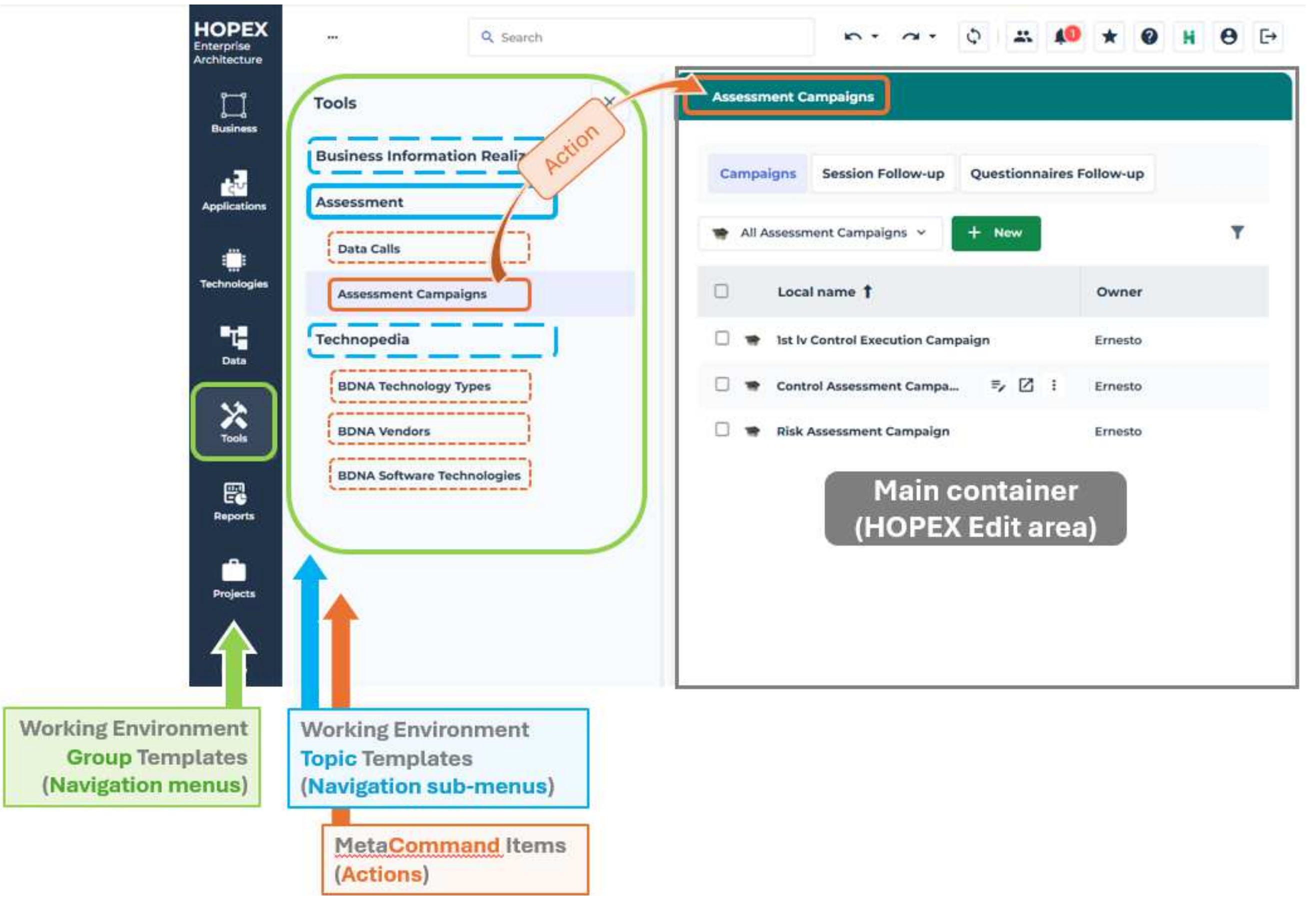
In the HOPEX Enterprise Architecture desktop:

the **Tools** navigation menu (Working Environment **Group** Template)

gives access, in particular, to the **Assessment** sub-menu (Working Environment **Topic** Template),

which gives access, in particular, to the **Assessment Campaigns** action,

which displays all the Assessment Campaigns in the **Edit area** (main container).



□	Local name ↑	Owner
□	1st lv Control Execution Campaign	Ernesto
□	Control Assessment Campa...	Ernesto
□	Risk Assessment Campaign	Ernesto

12.1.4 WET-based desktop creation and customization main steps

To create and customize a WET-based desktop, you should follow these steps:

Step	Action	See section
1.	Create the WET	Creating a WET
2.	Define the WET characteristics	Defining the WET characteristics
3.	Define the WET Homepage	Customizing the Homepage for a WET
4.	Add group(s) to the WET	Adding Navigation menus to a WET
5.	Add topic(s) to each Navigation menu (else connect a desktop to the topic)	Adding topics to a Working Environment Group Template (Defining a Desktop to a Working Environment Group Template)
6.	Add action(s) to each topic	Adding actions to a Working Environment Topic Template
7.	Customize a desktop according to a profile	Profile and Working Environment Template

12.1.5 Advanced configuration

To customize a WET-based desktop, you might need to perform these actions:

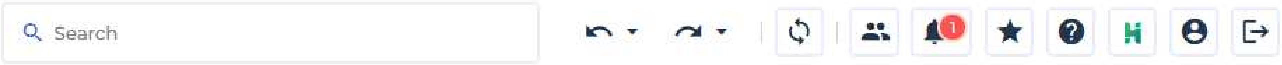
Action	See section
Add a filter at property page level according to a profile	WET advanced customization at property page level
Customize a desktop for certain users	Customizing a desktop for certain users

12.1.6 Elements of a WET-based Desktop

To customize the work environment of users, the main following elements are available:

- **a Toolbar**

  This toolbar includes components common to all WET-based desktops and is not customizable.



- Working Environment Group Template (**Navigation menu**):

  A Working Environment Group Template includes at least one Working Environment Topic Template, which defines at least one action.

  ➢ See Adding Navigation menus to a WET.

- Working Environment Topic Template (topic):

  Each Working Environment Topic Template must include at least one action (**MetaCommand Item**).

  Each action is defined by a configuration, i.e. a macro, or a **MEGA Parameterized Tool** including a tool (**MEGA Tool**).

  ```
  MEGA Tool examples: MetaTree Tool, Docked MetaPropertyPage
  Tool, Docked MetaWizard Tool, Desktop viewer.
  ```

  You can define:

  - a standard action on the WET, see WET properties.
  - a default action on the topic, see Adding actions to a Working Environment Topic Template.

The following element is also available for advanced customization:

- **PropertyPage ViewPort**

12.1.7 Accessing the WET metamodel element properties

To access the metamodel elements related to a WET:

1. Connect to HOPEX with the HOPEX Customizer profile.

2. Display the MetaStudio tab (**View > Navigation Windows > MetaStudio**).

3. To access:

   - the **WET** properties:

     a. Expand the **Working Environment Template** folder.

     b. Right-click the WET and select **Properties**.

   - the **Working Environment Group Template** properties:

   a. Expand the **GroupsTemplate** folder.

   b. Right-click the Working Environment Group Template and select **Properties**.

- the **Homepage** properties:

   a. Expand the **Home Pages** folder.

   b. Right-click the Homepage and select **Properties**.

- the **Desktop** properties:

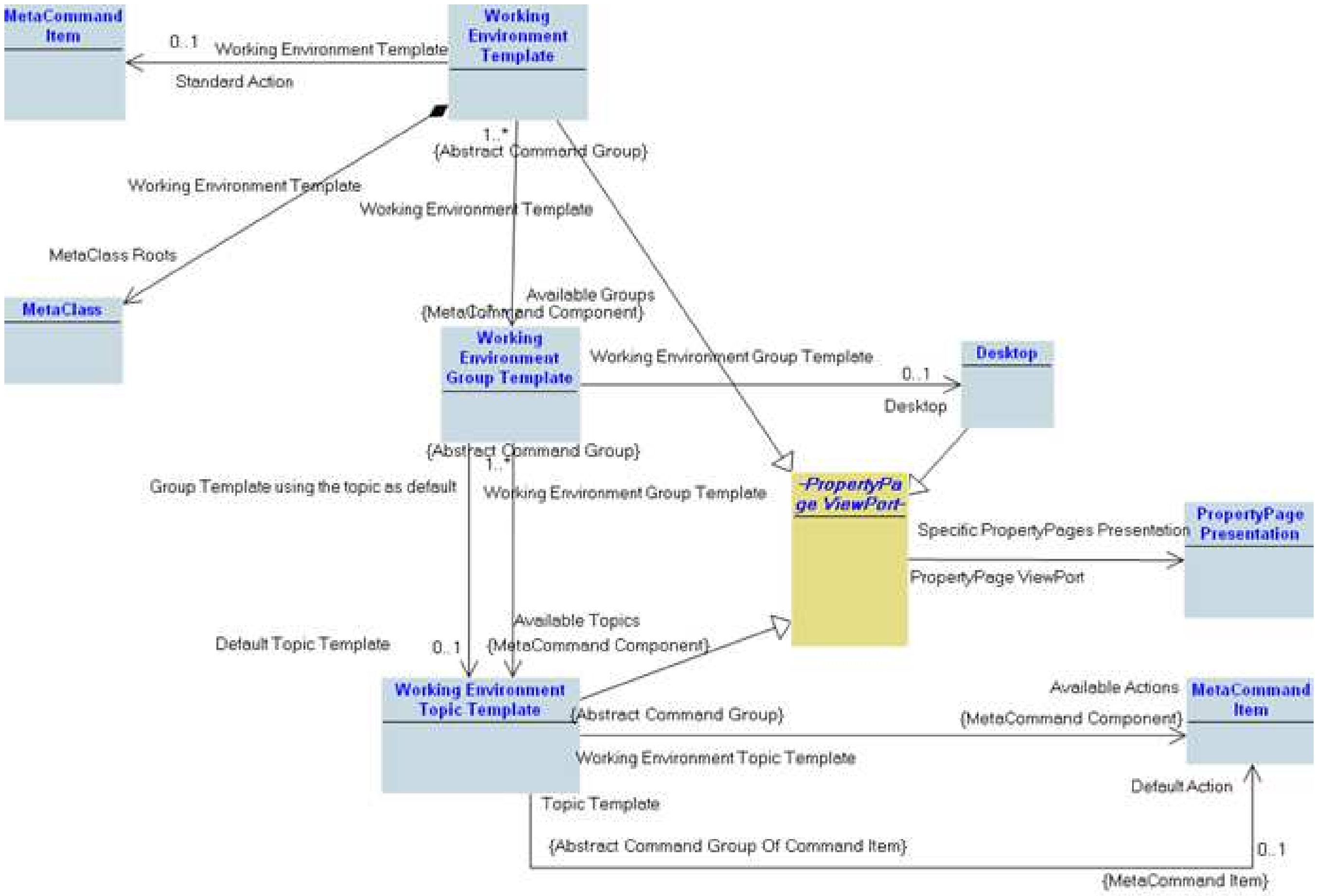
   a. Expand the **Desktops > Other Desktops > Universal Desktop V6** folders.

   b. Right-click the desktop concerned and select **Properties**.

12.1.8 WET metamodel elements

The following metamodel schema shows the architecture and links between the following WET related metamodel elements:

- **Working Environment Template**
- **Working Environment Group Template**
- **Working Environment Topic Template**
- **Desktop**
- **Desktop components**
- **PropertyPage ViewPort**


MetaCommand Item
0..1
Working Environment Template
Standard Action
Working Environment Template
1..*
{Abstract Command Group}
Working Environment Template
Working Environment Template
MetaClass Roots
MetaClass
Available Groups
{MetaCommand Component}
Working Environment Group Template
Working Environment Group Template
0..1
Desktop
Desktop
{Abstract Command Group}
1..*
Group Template using the topic as default
Working Environment Group Template
-PropertyPage ViewPort-
Specific PropertyPages Presentation
PropertyPage Presentation
PropertyPage ViewPort
Available Topics
Default Topic Template
0..1
{MetaCommand Component}
Working Environment Topic Template
{Abstract Command Group}
Available Actions
{MetaCommand Component}
MetaCommand Item
Working Environment Topic Template
Topic Template
Default Action
{Abstract Command Group Of Command Item}
0..1
{MetaCommand Item}

12.2 Creating a Working Environment Template

12.2.1 WET properties

A WET has the following characteristics:

See Accessing the WET metamodel element properties.

Sessions Access Mode:

The **Sessions Access Mode** defines the way the application is opened:

- **Public Workspace**: the application opens at the current time and can be updated. All the updates are available for the end users connected in the same way.
- **Private Workspace** (default value): the application opens at the current time and can be updated. All the end user updates are kept private until the end user dispatches his work.

MetaPicture:

The **MetaPicture** defines the WET icon displayed in HOPEX.

Standard Action:

(Optional parameter) The **Standard Action** defines an action that is frequently used in the Working Environment Topic Template definitions. When you define a Working Environment Topic Template, the Standard Action is automatically proposed in the action Connecting window (see Creating a WET and Defining a Navigation Menu (Working Environment Group Template)).

See Adding actions to a Working Environment Topic Template.

Session Connection Mode:

The **Session Connection Mode** defines whether the end-users share the same process ("Multi Session") or not ("Single Session", default value). If they share the same process, they must have the same view on the repository.

WET Specialization Assignment:

The **WET Specialization Assignment** defines the desktop type regarding its display. (E.g.: "Universal Desktop V6 WET Assignment" for Aquila type desktop).

_GUIName:

The **_GUIName** defines the WET display name in HOPEX.

No Assignment Required for Working Environment Instance:

When **No Assignment Required for Working Environment Instance** is selected (by default) any Working Environment created from this WET automatically takes advantage of the entire WET description (Groups and Topics).

(Optional) MetaClass:

You can connect a **MetaClass** to the WET. This is done through the creation of a Working Environment, which is a WET instance. The MetaClass is the entry point, which enables to instantiate the Working Environment.

See Customizing a desktop for certain users.

12.2.2 Creating a WET

To create a WET:

1. Connect to **HOPEX** with the HOPEX Customizer profile.
2. Display the **MetaStudio** tab (**View > Navigation Windows > MetaStudio**).
3. Right-click the **Working Environment Template** folder and select **New > Working Environment Template**.

4. Enter a **Name** to your WET.

5. (Optional) In the **Standard Action** field, connect a MetaCommandItem representing an action you want to add in several Working Environment Topic Templates or select a MetaCommandItem already defined as Standard Action.

   ➢ See Accessing the WET metamodel element properties.

6. Click **OK**.

7. You must define:

   - the WET characteristics, see Defining the WET characteristics.
   - the WET Homepage, see Customizing the Homepage for a WET
   - the WET Navigation menus, see Adding a Navigation menu to a WET
   - the actions available from a Navigation menu, see Defining a Navigation Menu (Working Environment Group Template)

12.2.3 Defining the WET characteristics

To define the WET characteristics:

1. Access the WET properties.

   ➢ See Accessing the WET metamodel element properties.

2. In the **_GUIName** field, enter the WET name displayed in HOPEX.

3. (Optional) In the **MetaPicture** field, click the arrow and connect an image.

   Else the default WET image is displayed in HOPEX.

4. (if needed) Modify the **Session Connection Mode** and **Session Access Mode** default values.

   ➢ See WET properties.

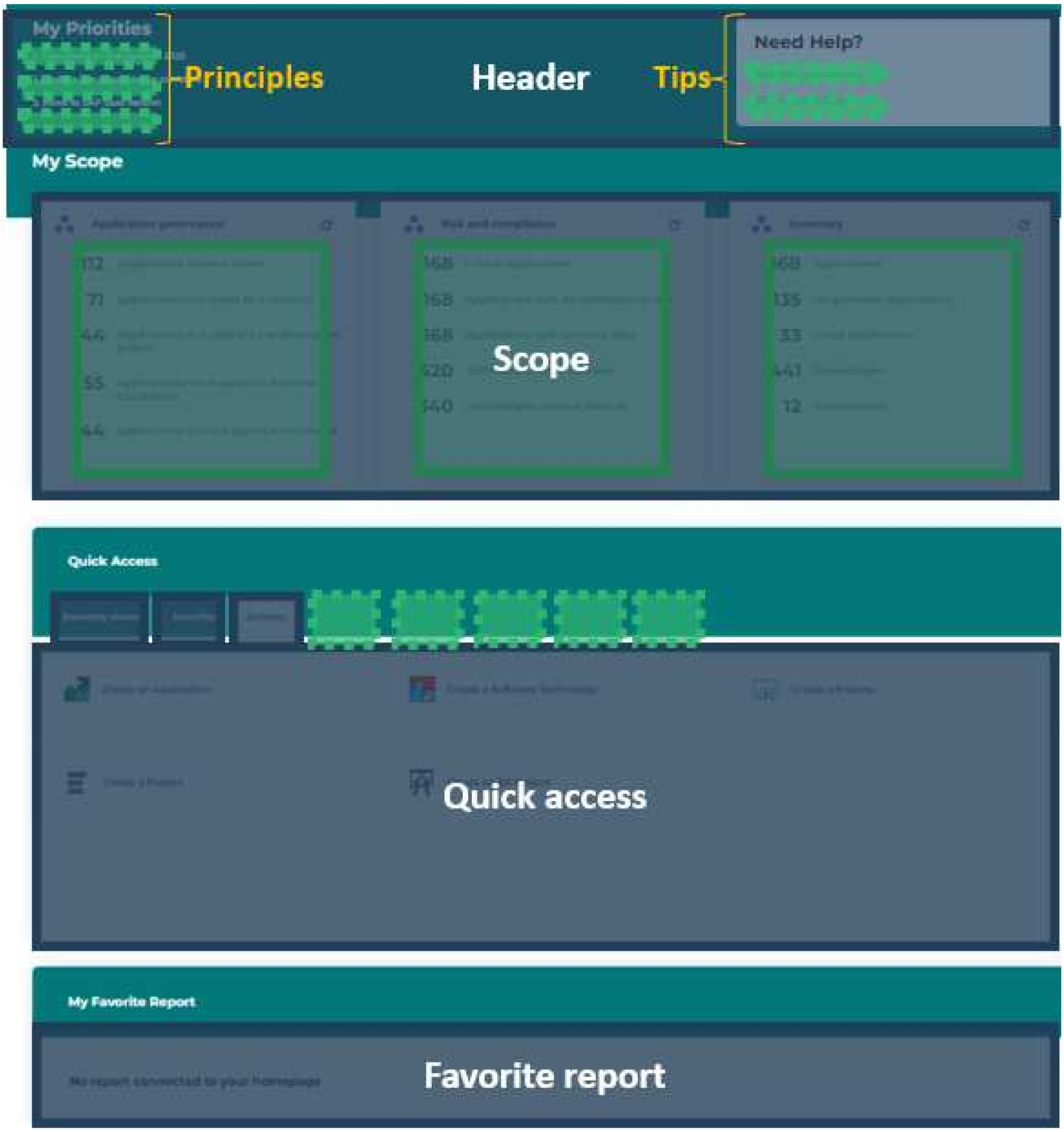
5. In the **WET Specialization Assignment** field, click the arrow and select the desktop type:

   - “Universal Desktop V6 WET Assignment” for Aquila type desktop

12.3 Customizing the Homepage for a WET

Each Homepage shows four customizable blocks:

- **Header,** with **Principles** and **Tips**
- **Scope**
- **Quick access**
- **Favorite report**



12.3.1 Accessing the Homepage Properties

The first three blocks are defined in the homepage properties associated with the profile.

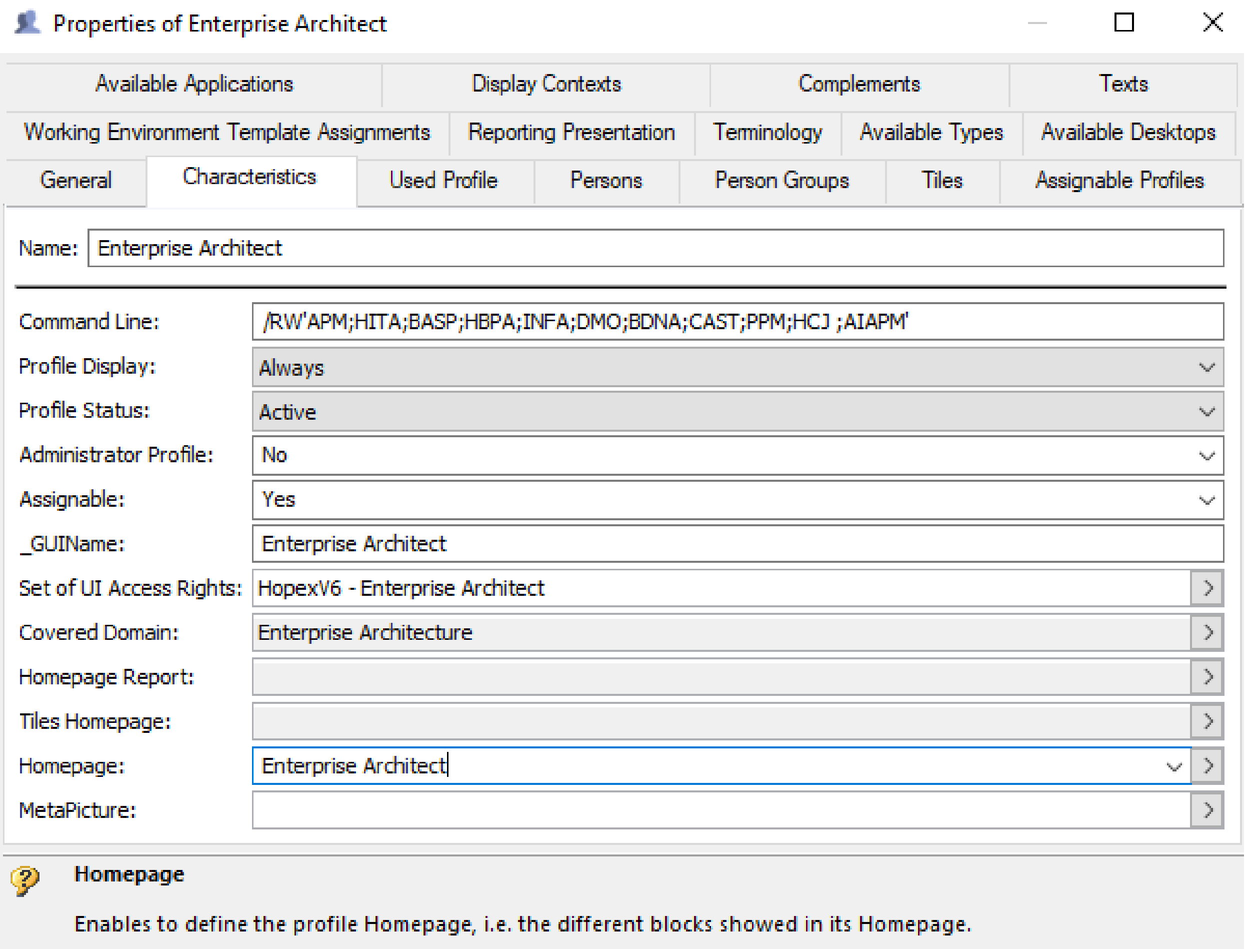
Note: Part of the Header can also be customized by each Functional Administrator of a Solution (Web Frent-End).

To access the Homepage properties associated with a profile:

1. Connect to HOPEX with the **HOPEX Customizer** profile.
2. Display the **MetaStudio** tab (**View > Navigation Windows > MetaStudio**).
3. Access the Homepage **Properties** associated with the profile:

   a. In the **Profiles > Used profiles – Using profiles** folder, right-click the profile concerned and select **Properties**.

   `E.g.: Enterprise Architect profile.`



Name: Enterprise Architect

Command Line: /RW'APM;HITA;BASP;HBPA;INFA;DMO;BDNA;CAST;PPM;HCJ ;AIAPM'

Profile Display: Always

Profile Status: Active

Administrator Profile: No

Assignable: Yes

_GUIName: Enterprise Architect

Set of UI Access Rights: HopexV6 - Enterprise Architect

Covered Domain: Enterprise Architecture

Homepage Report:

Tiles Homepage:

Homepage: Enterprise Architect

MetaPicture:

Homepage

Enables to define the profile Homepage, i.e. the different blocks showed in its Homepage.

   b. From the **Characteristics** tab, access its **Homepage** properties.

   `E.g.: Enterprise Architect homepage.`

   ➢ Alternative: from **Home Pages** folder, right-click the homepage, associated with the profile concerned, and select **Properties**.

12.3.2 Customizing the left Header: Principles

The left header, or **Principles**, includes:

- a **title**

  E.g.: My priorities

  The title is defined on the profile associated homepage.

- three **links** (the **Strategic Themes**) retrieved by a query

  The query retrieves the first three **Strategic Themes** defined for the **Covered Domain** (Methodological Domain) of the current profile:

  ```
  Select [Strategic Theme] Where [Methodological Domain].[Applicable Profile] &"CurrentProfile" Order By [Name]
  ```

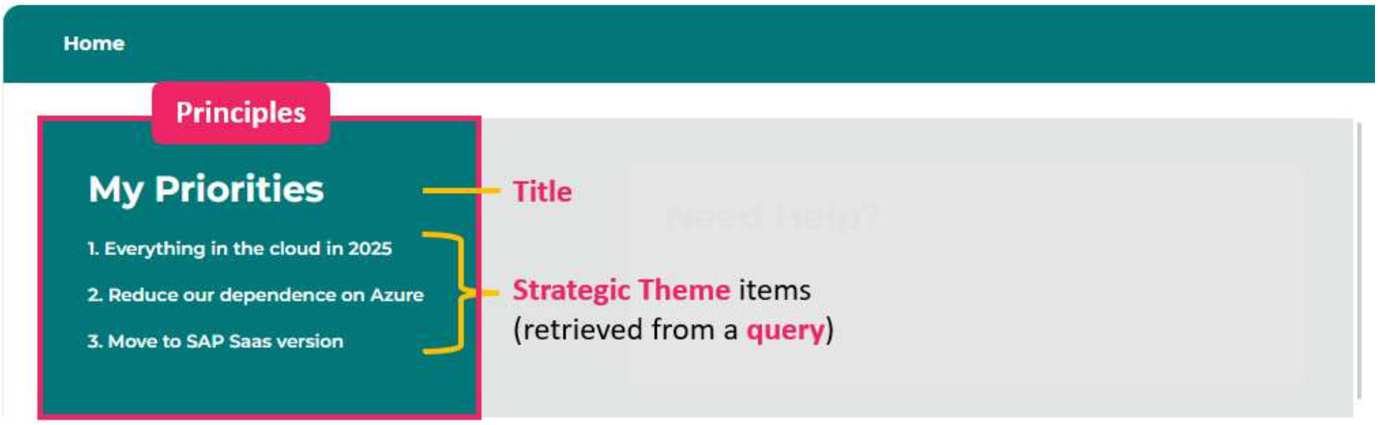
  The strategic themes are accessible from the profile.

You can:

- keep the query and modify the **title** and/or the **Strategic Themes** displayed.

  The Strategic Themes can also be modified in HOPEX (Web Front-End) with the Administrator of the Solution.

- modify the **query**.

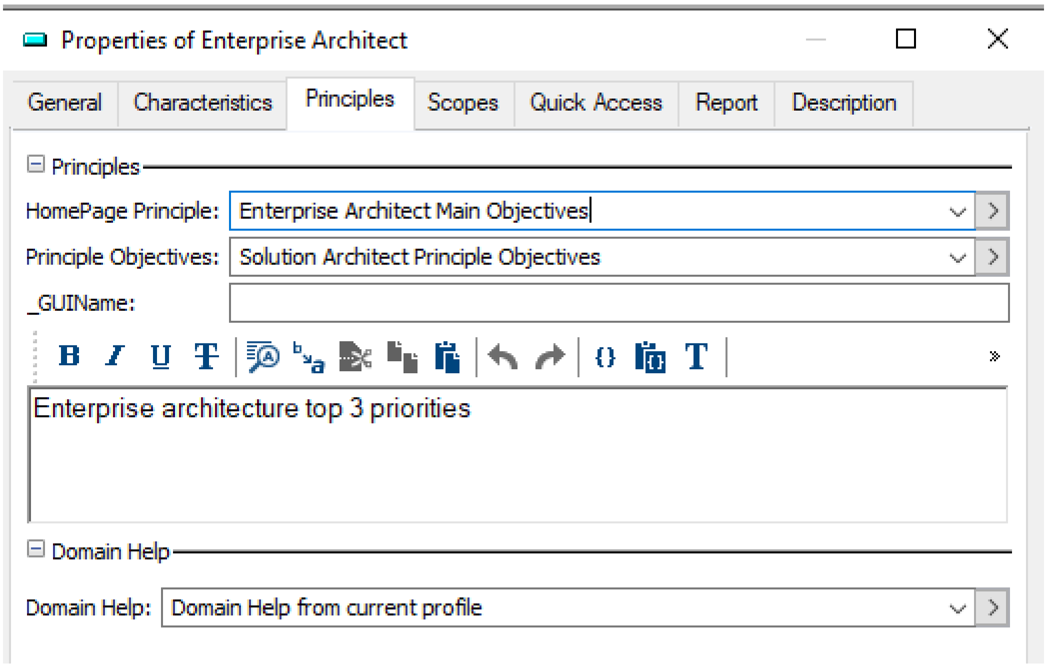

Home
Principles
My Priorities
1. Everything in the cloud in 2025
2. Reduce our dependence on Azure
3. Move to SAP Saas version
Title
Strategic Theme items
(retrieved from a query)

To customize the Principles (header left part):

1. Access the Homepage properties associated with the Profile.

   E.g.: Enterprise Architect homepage.

2. Display its **Principles** tab.



HomePage Principle: Enterprise Architect Main Objectives

Principle Objectives: Solution Architect Principle Objectives

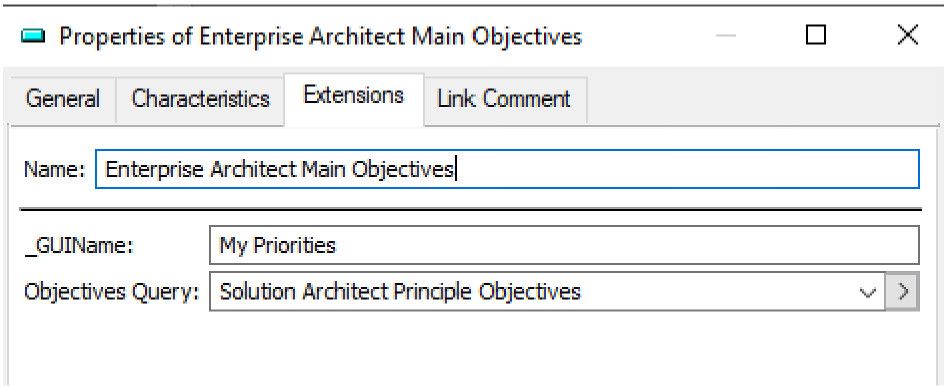
_GUIName:

Enterprise architecture top 3 priorities

Domain Help: Domain Help from current profile

The **Principles** section defines the Principles (header left part)

3. Access the properties of the **Homepage Principle**:
   - the **_GUIName** field defines the title.
   - the **Objectives Query** field defines the query that retrieves the links.



Name: Enterprise Architect Main Objectives

_GUIName: My Priorities

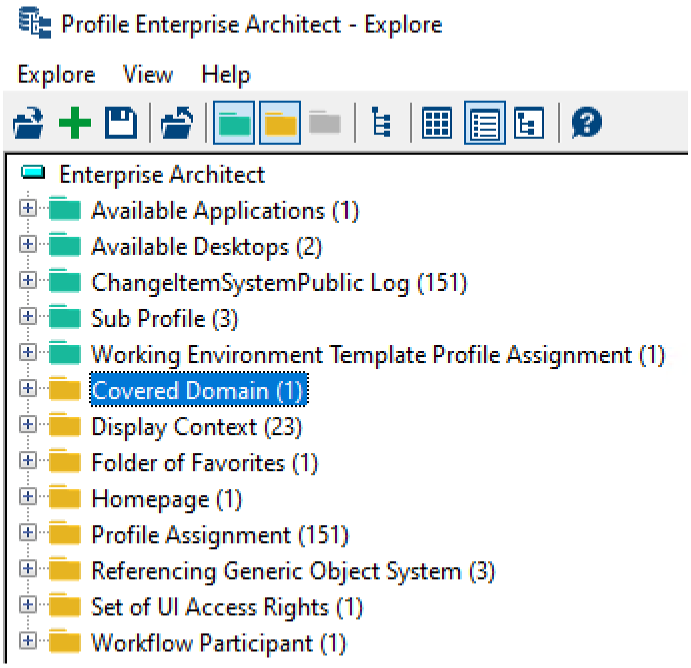
Objectives Query: Solution Architect Principle Objectives

4. To modify the query, in the **Objectives Query** field, connect another query.

5. To customize the title, in the **_GUIName** field, enter another title.

6. To customize the links, modify the **Strategic Theme** items associated with the profile:

   a. Explore the profile.



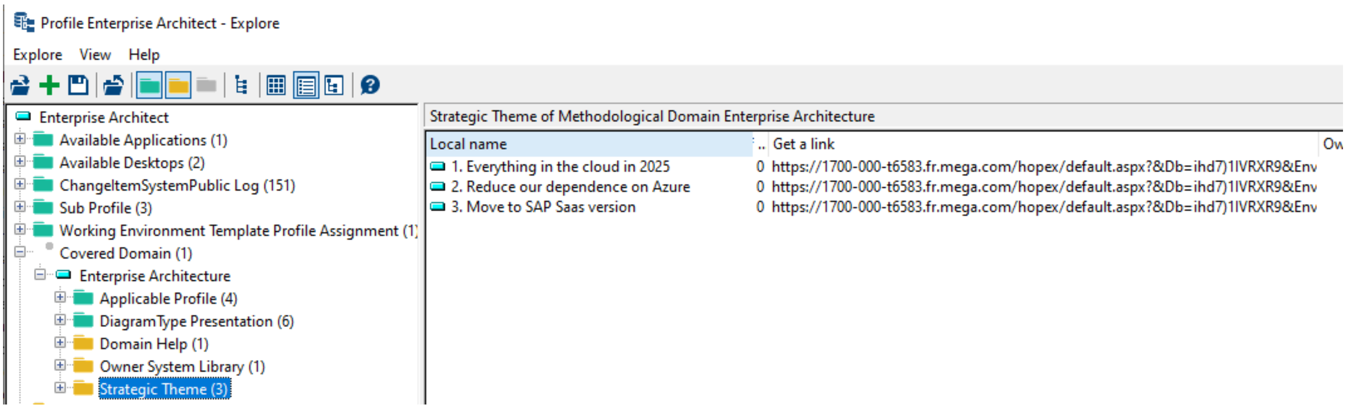
Profile Enterprise Architect - Explore

Explore View Help

- Enterprise Architect
  - Available Applications (1)
  - Available Desktops (2)
  - ChangeItemSystemPublic Log (151)
  - Sub Profile (3)
  - Working Environment Template Profile Assignment (1)
  - Covered Domain (1)
  - Display Context (23)
  - Folder of Favorites (1)
  - Homepage (1)
  - Profile Assignment (151)
  - Referencing Generic Object System (3)
  - Set of UI Access Rights (1)
  - Workflow Participant (1)

b. Expand **Covered Domain > <Methodological Domain name>** folder.

c. Select **Strategic Theme** folder.



Profile Enterprise Architect - Explore

Explore View Help

- Enterprise Architect
  - Available Applications (1)
  - Available Desktops (2)
  - ChangeItemSystemPublic Log (151)
  - Sub Profile (3)
  - Working Environment Template Profile Assignment (1)
  - Covered Domain (1)
    - Enterprise Architecture
      - Applicable Profile (4)
      - DiagramType Presentation (6)
      - Domain Help (1)
      - Owner System Library (1)
      - Strategic Theme (3)

Strategic Theme of Methodological Domain Enterprise Architecture

Local name	..	Get a link	Ow
1. Everything in the cloud in 2025	0	https://1700-000-t6583.fr.mega.com/hopex/default.aspx?&Db=ihd7)1IVRXR9&Env	
2. Reduce our dependence on Azure	0	https://1700-000-t6583.fr.mega.com/hopex/default.aspx?&Db=ihd7)1IVRXR9&Env	
3. Move to SAP Saas version	0	https://1700-000-t6583.fr.mega.com/hopex/default.aspx?&Db=ihd7)1IVRXR9&Env	

d. In the right pane, for each item, you can modify:

- the **Local Name**
- the **link**

e. You can also create a Strategic theme: right-click **Strategic Theme** folder, click New and enter a **Name**.

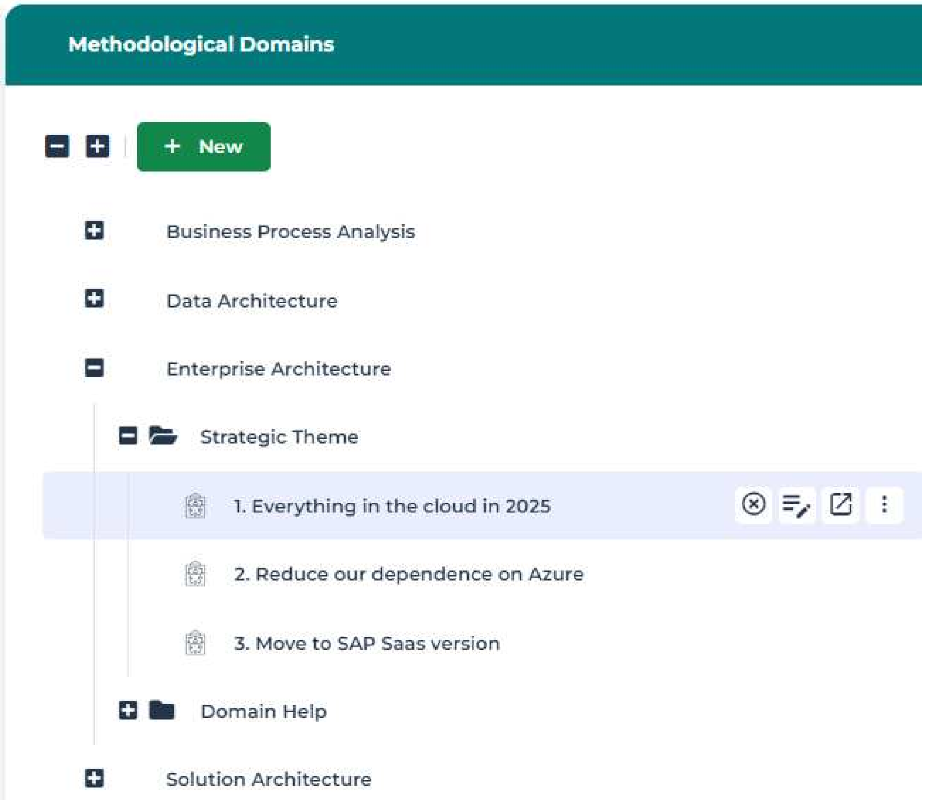
The Strategic Theme items are sorted by alphabetical order, and the query retrieves the first three of them.

To modify the Strategic Theme items in HOPEX (Web Front-End):

The **Administrator of a Solution** can customize the **Strategic Themes** of each profile linked to the Solution (in HOPEX Web Front-End).

1. Connect to HOPEX with the Functional Administrator Profile of the Solution.

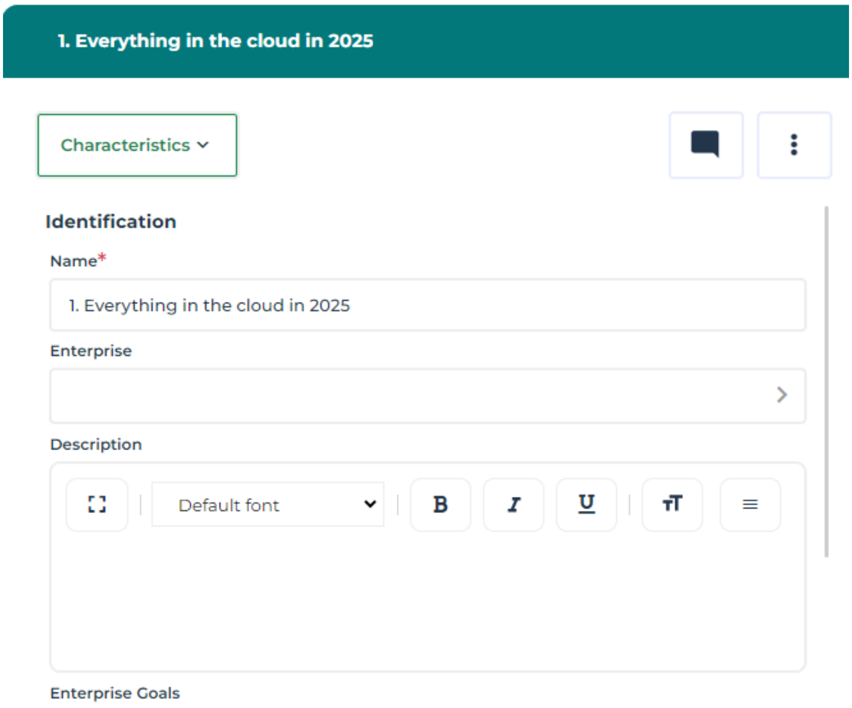
2. Access **Administration > Methodological Domains >** <name of the methodological domain associated with the profile> **> Strategic Theme** folder.


Methodological Domains
+ New
Business Process Analysis
Data Architecture
Enterprise Architecture
Strategic Theme
1. Everything in the cloud in 2025
2. Reduce our dependence on Azure
3. Move to SAP Saas version
Domain Help
Solution Architecture

3. To modify a **Strategic theme**:

   a. Access its **Properties**, and display its **Characteristics** page.

   b. In the **Name** field, modify the name of the link.


1. Everything in the cloud in 2025

Characteristics

Identification

Name*: 1. Everything in the cloud in 2025

Enterprise

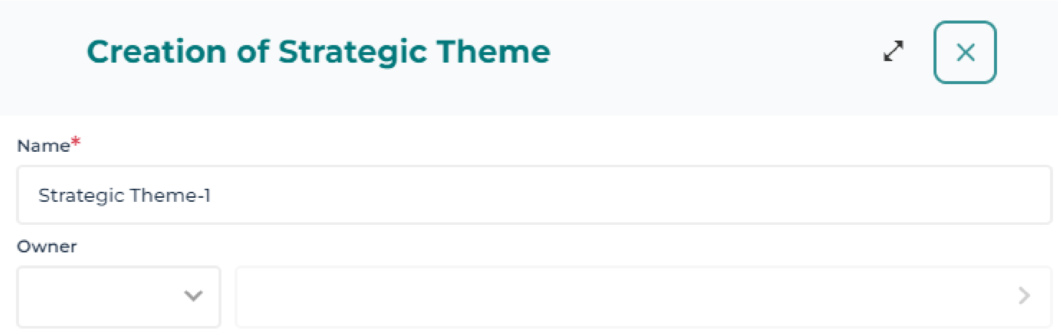
Description

Enterprise Goals

4. To create a **Strategic Theme**:

   a. Hover the cursor over **Strategic Theme** and click +.

   b. In the **Name** field, enter the name of the link.



Creation of Strategic Theme

Name*

Strategic Theme-1

Owner

The Strategic Theme items are sorted by alphabetical order, and the query retrieves the first three of them.

12.3.3 Customizing the right Header: Tips

The right header, or **Tips**, includes:

- a **title**

  ```
  E.g.: Need Help?
  ```

  The title is defined on the profile associated homepage.

- a content displaying two **links**

  This content displays the **Domain Help** description linked to the **Covered Domain (Methodological Domain)** defined for the current Profile.

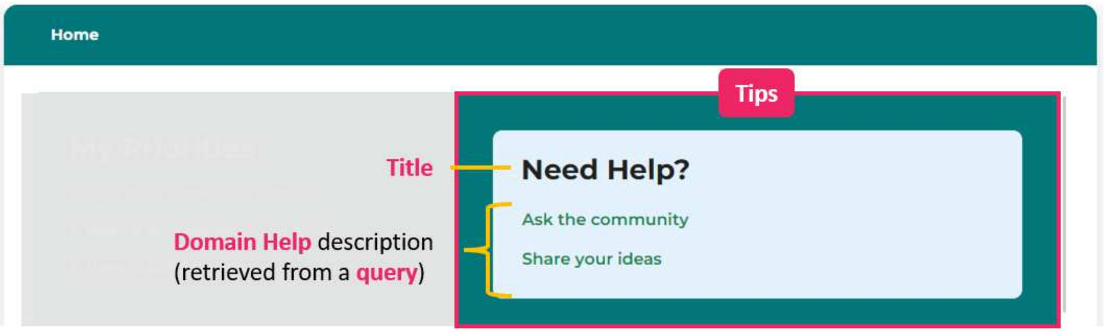
  It is retrieved by the following query:

  ```
  Select [Domain Help] Where [Methodological Domain].[Applicable
  Profile] &"Current Profile"
  ```

  The Domain Help description can also be modified in HOPEX (Web Front-End) with the Administrator of the Solution.

You can:

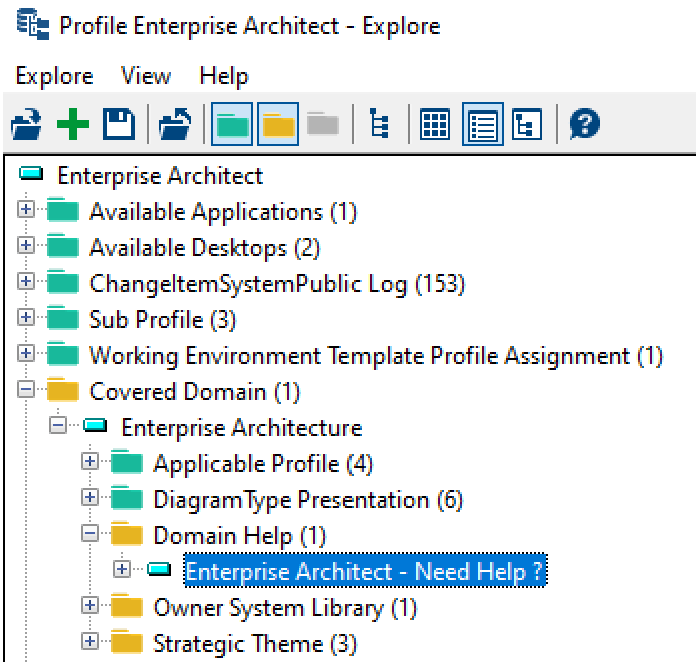
- keep the query and modify the **title** and/or the **Domain Help** description displayed
- modify the **query**



To customize the Tips (header right part):

To keep the query and customize the links and/or title, you need to modify the **Domain Help** associated with the profile.

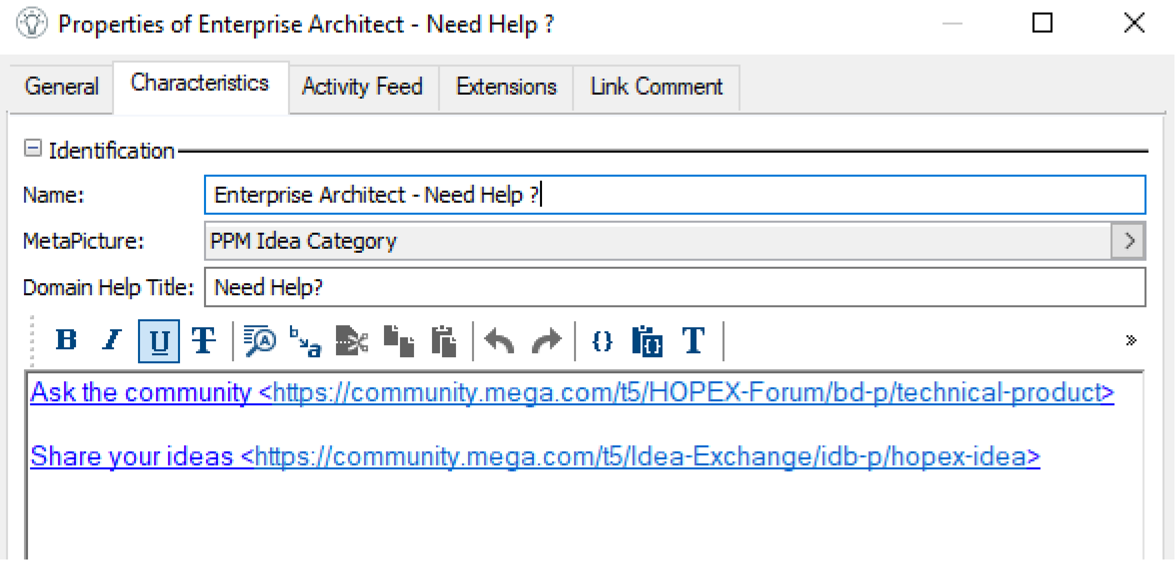
1. Explore the profile.
2. Expand **Covered Domain >** <Methodological Domain name> **> Domain Help** folder.



Profile Enterprise Architect - Explore

- Enterprise Architect
  - Available Applications (1)
  - Available Desktops (2)
  - ChangeItemSystemPublic Log (153)
  - Sub Profile (3)
  - Working Environment Template Profile Assignment (1)
  - Covered Domain (1)
    - Enterprise Architecture
      - Applicable Profile (4)
      - DiagramType Presentation (6)
      - Domain Help (1)
        - Enterprise Architect - Need Help ?
      - Owner System Library (1)
      - Strategic Theme (3)

3. Right-click the domain help and select **Properties**.



Name: Enterprise Architect - Need Help ?

MetaPicture: PPM Idea Category

Domain Help Title: Need Help?

Ask the community <https://community.mega.com/t5/HOPEX-Forum/bd-p/technical-product>

Share your ideas <https://community.mega.com/t5/Idea-Exchange/idb-p/hopex-idea>

4. In the **Characteristics** tab, customize the Tips block:
   - In the **MetaPicture** field: connect another picture
   - In the **Domain Help Title** field: enter another title
   - In the pane: modify the text and links

To modify the query (header right part):

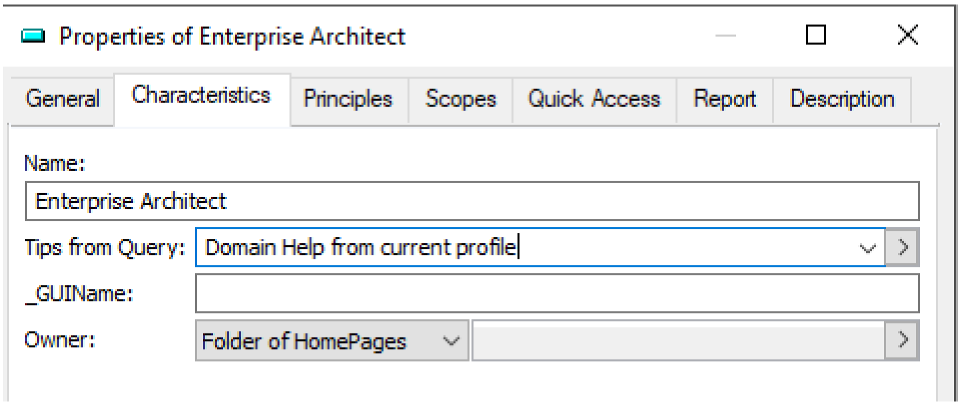
To modify the query, you need to modify the query defined in the Homepage associated with the profile.

1. Access the Homepage properties associated with the Profile.

   `E.g.: Enterprise Architect homepage.`

2. Display its **Characteristics** tab.

   The **Tips from Query** field defines the query that retrieves the items.



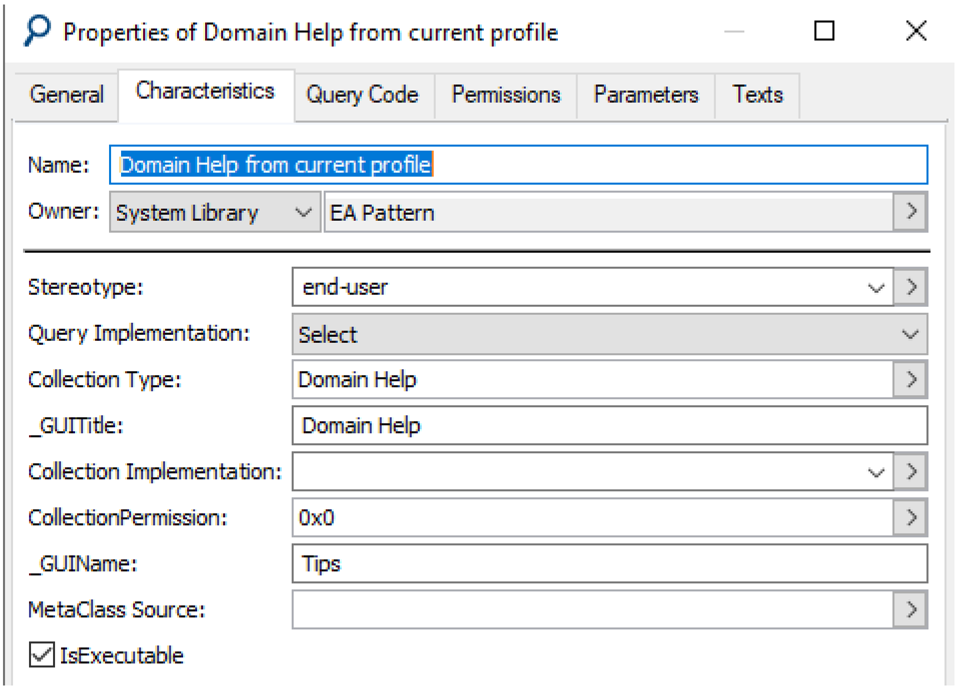
Name: Enterprise Architect

Tips from Query: Domain Help from current profile

_GUIName:

Owner: Folder of HomePages

3. From the **Tips from Query** field, access the query properties:
   - the **_GUIName** field defines the title.
   - the **Query Code** tab defines the query code that retrieves the Domain Help Description.



Name: Domain Help from current profile

Owner: System Library | EA Pattern

Stereotype: end-user

Query Implementation: Select

Collection Type: Domain Help

_GUITitle: Domain Help

Collection Implementation:

CollectionPermission: 0x0

_GUIName: Tips

MetaClass Source:

☑ IsExecutable

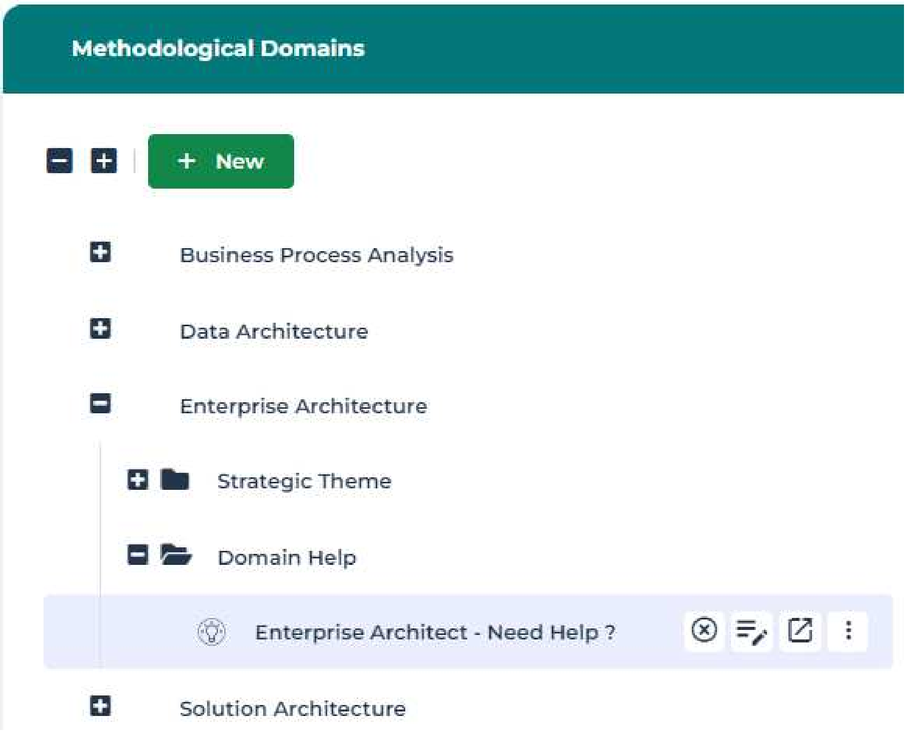
4. From the **Tips from Query** field, connect another query.

To customize the Tips in HOPEX (Web Front-End):

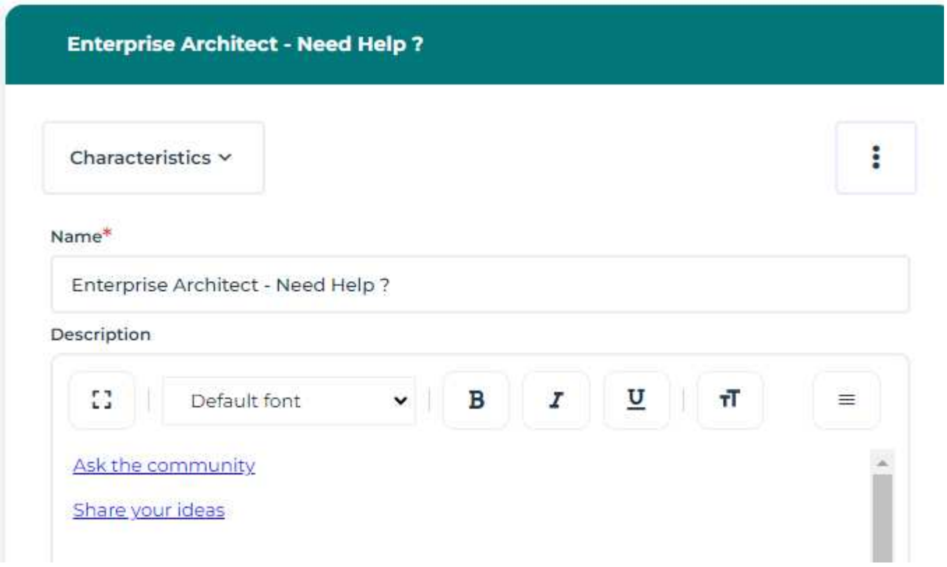
The **Administrator of a Solution** can customize the links (**Domain Help**) of each profile linked to the Solution (in HOPEX Web Front-End).

1. Connect to HOPEX with the Administrator Profile of the Solution.

2. Access **Administration > Methodological Domains >** <name of the methodological domain associated with the profile> **> Domain Help** folder.



3. In the **Properties** of the domain help, display its **Characteristics** page.



Enterprise Architect - Need Help ?

Characteristics

Name*

Enterprise Architect - Need Help ?

Description

Ask the community

Share your ideas

4. Modify the **Description**.

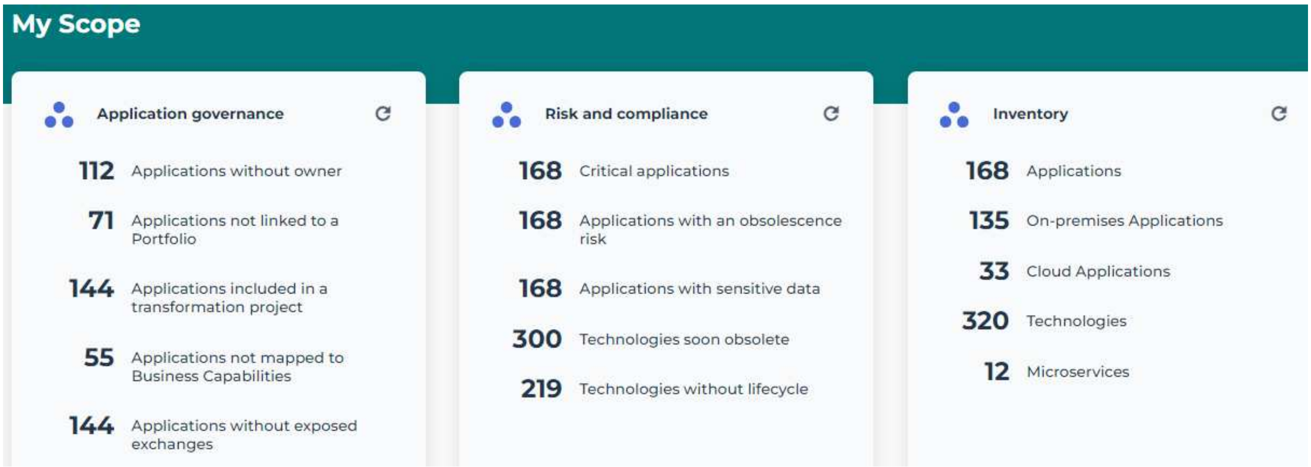
12.3.4 Customizing the Scope block

The Scope block displays a maximum of three scope items. For a better readability, each of them should include a maximum of five indicators.

You can:

- create a scope item
- modify an existing scope item:
  - its title and indicators
  - the indicator order
  - each indicator display type: count (by default) or percentage.

  With the percentage type, you can define a total count different than the maximum one.
- modify the scope item display order



My Scope

Application governance

112	Applications without owner
71	Applications not linked to a Portfolio
144	Applications included in a transformation project
55	Applications not mapped to Business Capabilities
144	Applications without exposed exchanges

Risk and compliance

168	Critical applications
168	Applications with an obsolescence risk
168	Applications with sensitive data
300	Technologies soon obsolete
219	Technologies without lifecycle

Inventory

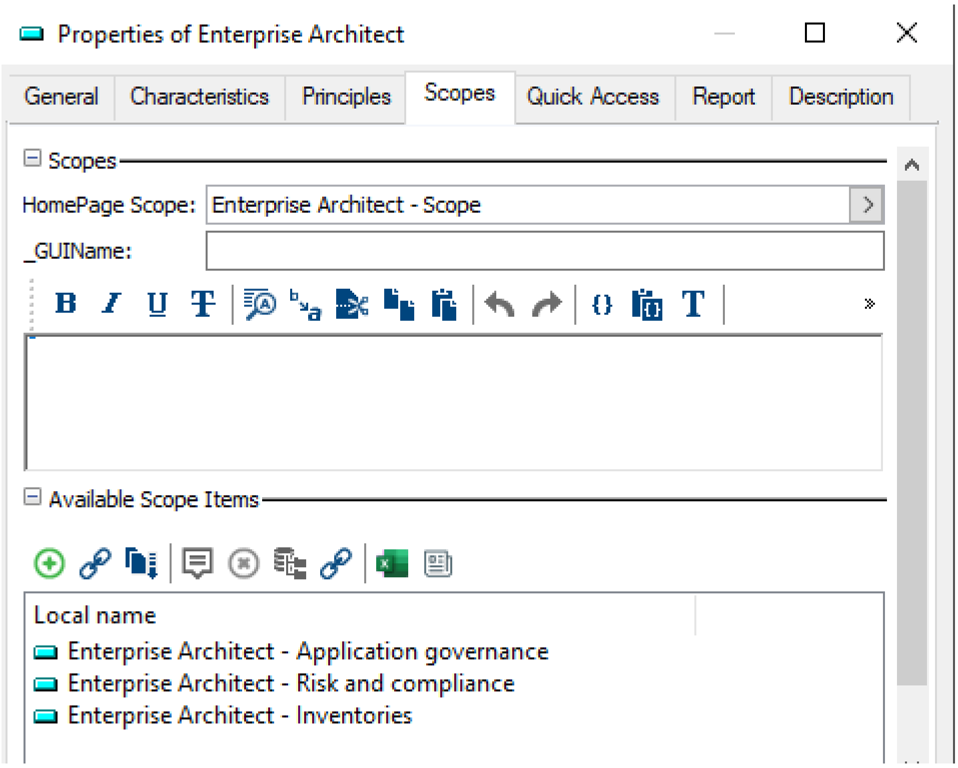
168	Applications
135	On-premises Applications
33	Cloud Applications
320	Technologies
12	Microservices

To customize the Scope block:

1. Access the **Homepage** properties associated with the Profile.

   `E.g.: Enterprise Architect homepage.`

2. Display its **Scopes** tab.



Properties of Enterprise Architect

General | Characteristics | Principles | Scopes | Quick Access | Report | Description

Scopes

HomePage Scope: Enterprise Architect - Scope

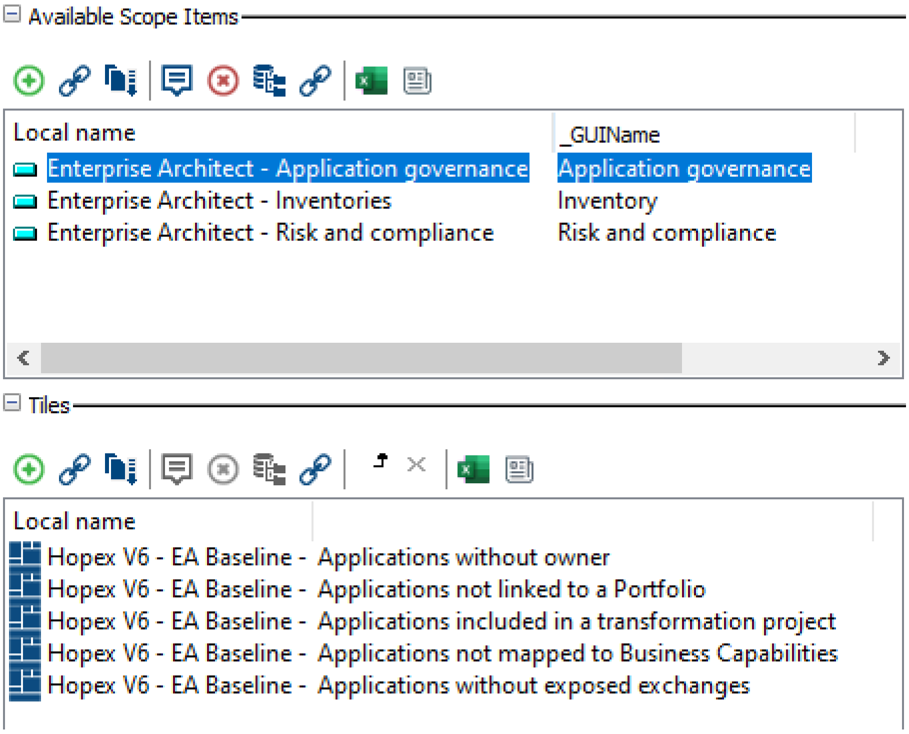
_GUIName:

Available Scope Items

Local name
Enterprise Architect - Application governance
Enterprise Architect - Risk and compliance
Enterprise Architect - Inventories

3. To modify an existing scope item:

   a. In the **Available Scope Items** section, select the scope item.



Available Scope Items

Local name	_GUIName
Enterprise Architect - Application governance	Application governance
Enterprise Architect - Inventories	Inventory
Enterprise Architect - Risk and compliance	Risk and compliance

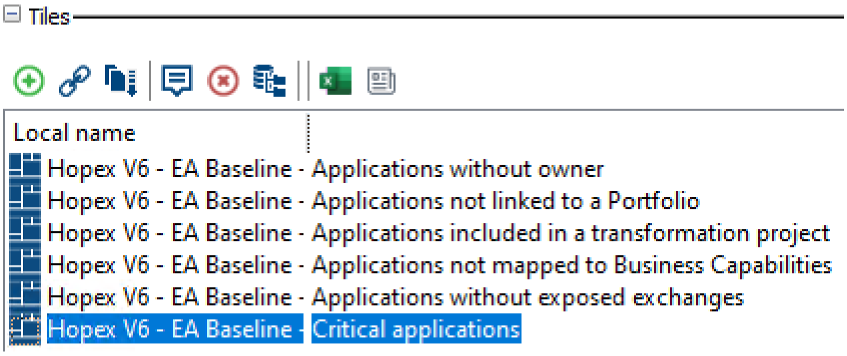
Tiles

Local name
Hopex V6 - EA Baseline - Applications without owner
Hopex V6 - EA Baseline - Applications not linked to a Portfolio
Hopex V6 - EA Baseline - Applications included in a transformation project
Hopex V6 - EA Baseline - Applications not mapped to Business Capabilities
Hopex V6 - EA Baseline - Applications without exposed exchanges

   b. To modify the title, in its **_GUIName** field, enter another title.

   c. To modify the indicators, in its **Tiles** section, connect the tiles corresponding to the indicators you want to display.

```
E.g.: "Hopex V6 – EA Baseline – Tile – critical applications".
```

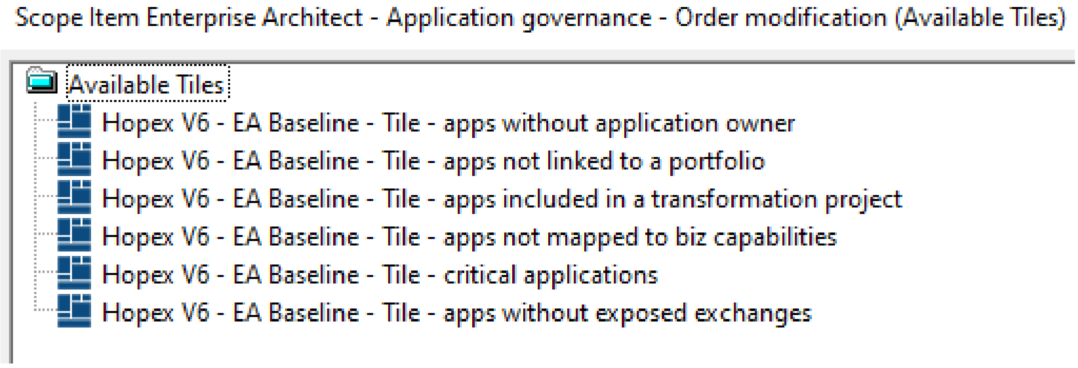


Local name
Hopex V6 - EA Baseline - Applications without owner
Hopex V6 - EA Baseline - Applications not linked to a Portfolio
Hopex V6 - EA Baseline - Applications included in a transformation project
Hopex V6 - EA Baseline - Applications not mapped to Business Capabilities
Hopex V6 - EA Baseline - Applications without exposed exchanges
Hopex V6 - EA Baseline - Critical applications

Reorganize the tiles so as to display the indicators as required, else remove the indicators you are not interested in.

d. To modify the indicator display order, in the **Tiles** section, click **Reorganize** and modify (drag and drop) their order.

```
E.g.: move "Critical applications" to the fifth position.
```

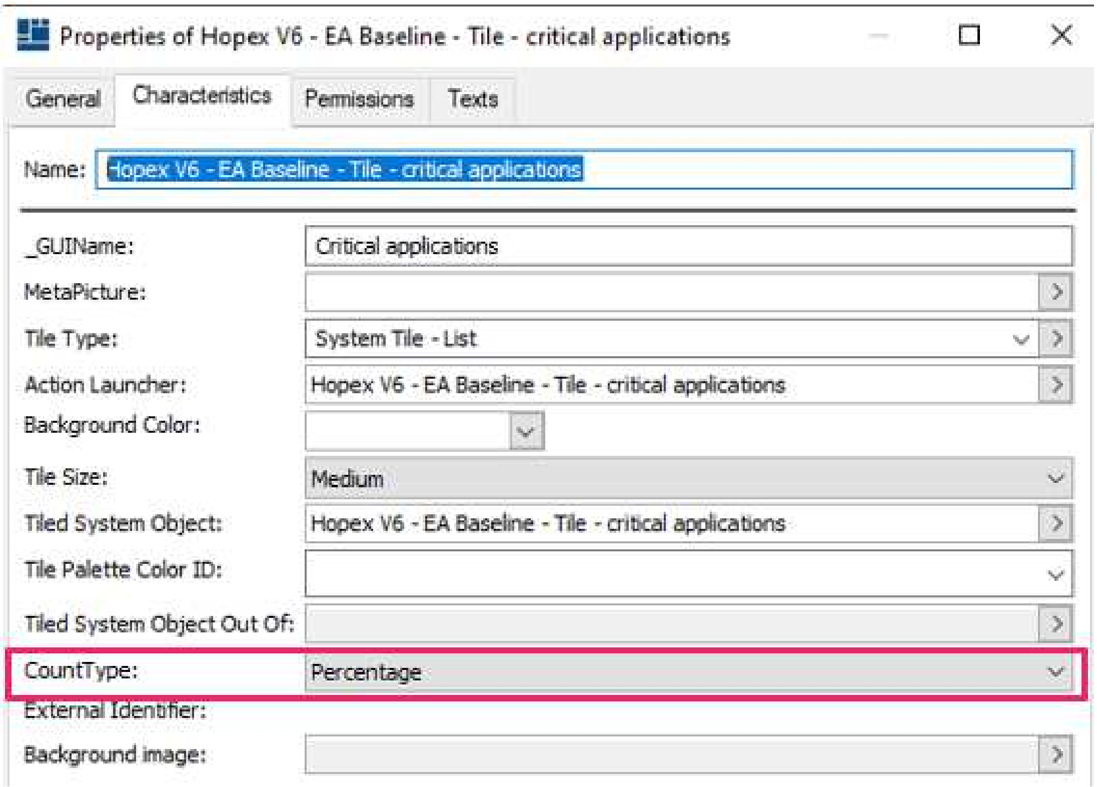

Scope Item Enterprise Architect - Application governance - Order modification (Available Tiles)

Available Tiles
Hopex V6 - EA Baseline - Tile - apps without application owner
Hopex V6 - EA Baseline - Tile - apps not linked to a portfolio
Hopex V6 - EA Baseline - Tile - apps included in a transformation project
Hopex V6 - EA Baseline - Tile - apps not mapped to biz capabilities
Hopex V6 - EA Baseline - Tile - critical applications
Hopex V6 - EA Baseline - Tile - apps without exposed exchanges

e. To modify an indicator display, in the corresponding Tile **Properties** > **Characteristics**, modify its **CountType** field value:

- Cardinal (by default)
- Percentage

```
E.g.: display the "Critical applications" indicator in "Percentage".
```



Properties of Hopex V6 - EA Baseline - Tile - critical applications

Name: Hopex V6 - EA Baseline - Tile - critical applications

_GUIName: Critical applications

MetaPicture:

Tile Type: System Tile - List

Action Launcher: Hopex V6 - EA Baseline - Tile - critical applications

Background Color:

Tile Size: Medium

Tiled System Object: Hopex V6 - EA Baseline - Tile - critical applications

Tile Palette Color ID:

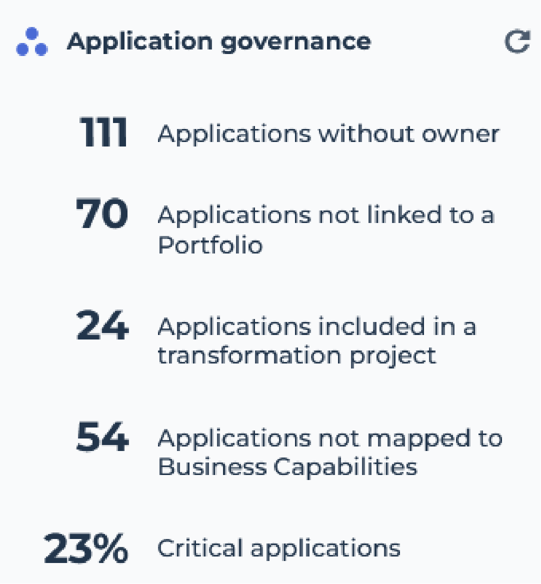
Tiled System Object Out Of:

CountType: Percentage

External Identifier:

Background image:

The "Critical applications" indicator shows the percentage of "Critical applications" out of all the applications:

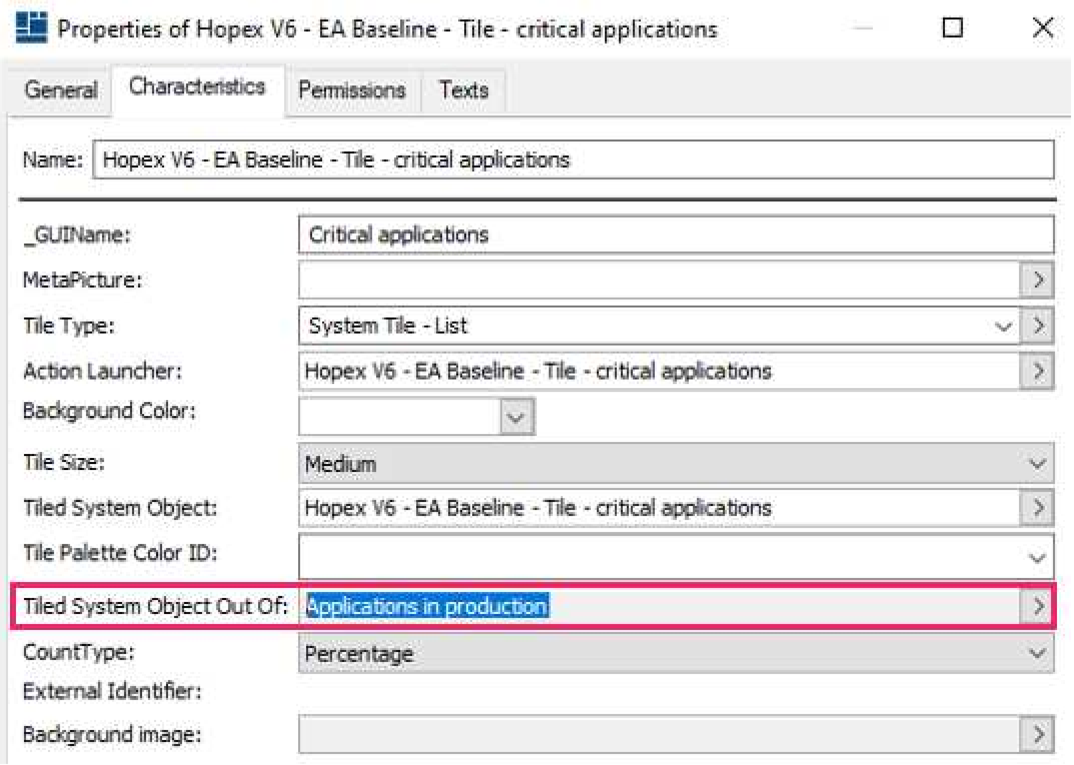


Application governance

111	Applications without owner
70	Applications not linked to a Portfolio
24	Applications included in a transformation project
54	Applications not mapped to Business Capabilities
23%	Critical applications

f. (With an indicator in percentage) To modify the total count: in the corresponding Tile **Properties** > **Characteristics**, in the **Tiled System Object Out Of** field connect the query that calculates the total count.

E.g.: the count of "Applications in production".



Properties of Hopex V6 - EA Baseline - Tile - critical applications

Name: Hopex V6 - EA Baseline - Tile - critical applications

_GUIName: Critical applications

MetaPicture:

Tile Type: System Tile - List

Action Launcher: Hopex V6 - EA Baseline - Tile - critical applications

Background Color:

Tile Size: Medium

Tiled System Object: Hopex V6 - EA Baseline - Tile - critical applications

Tile Palette Color ID:

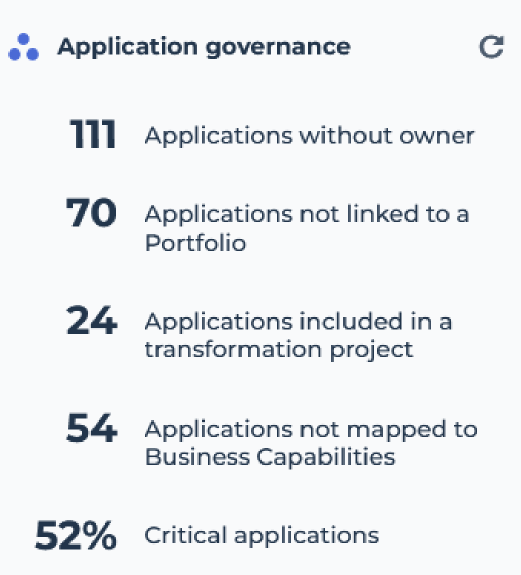
Tiled System Object Out Of: Applications in production

CountType: Percentage

External Identifier:

Background image:

The "Critical applications" indicator shows the percentage of critical applications that are in production.



Application governance

111	Applications without owner
70	Applications not linked to a Portfolio
24	Applications included in a transformation project
54	Applications not mapped to Business Capabilities
52%	Critical applications

4. To create a Scope item:

   a. In the **Available Scope Items** section, click **New**.

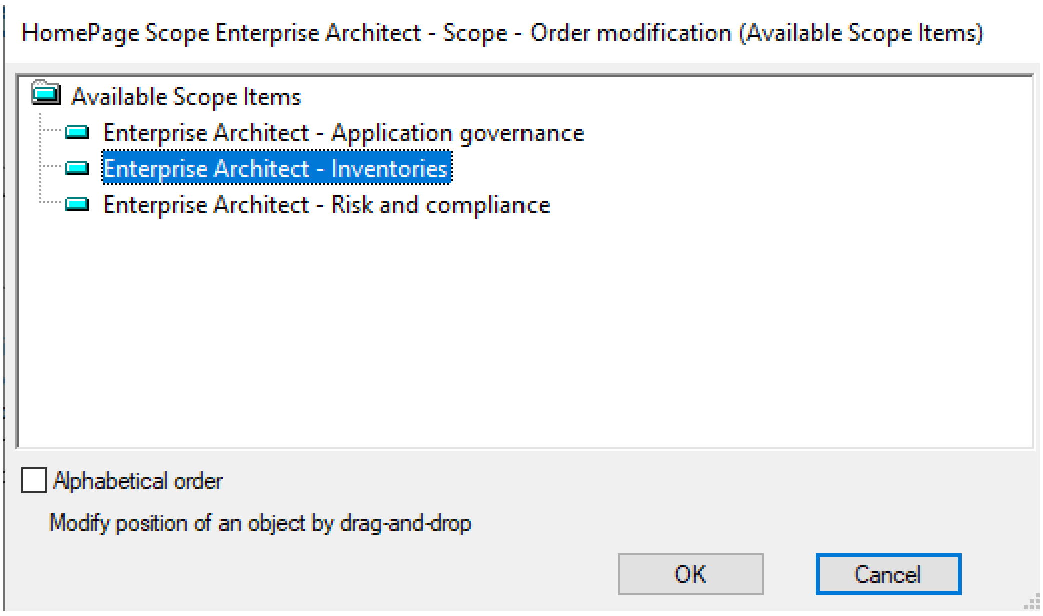
   b. Access the scope item properties.

   c. In the **Scope** section, enter its **Name** and **_GUIName** (title).

   d. In the **Tiles** section, connect the tiles corresponding to the indicators you want to display.

e. To modify the Scope item order, in the **Available Scope Items** section, click **Reorganize** and modify their order as required.

E.g.: Move the "Inventory" Scope Item to the middle.



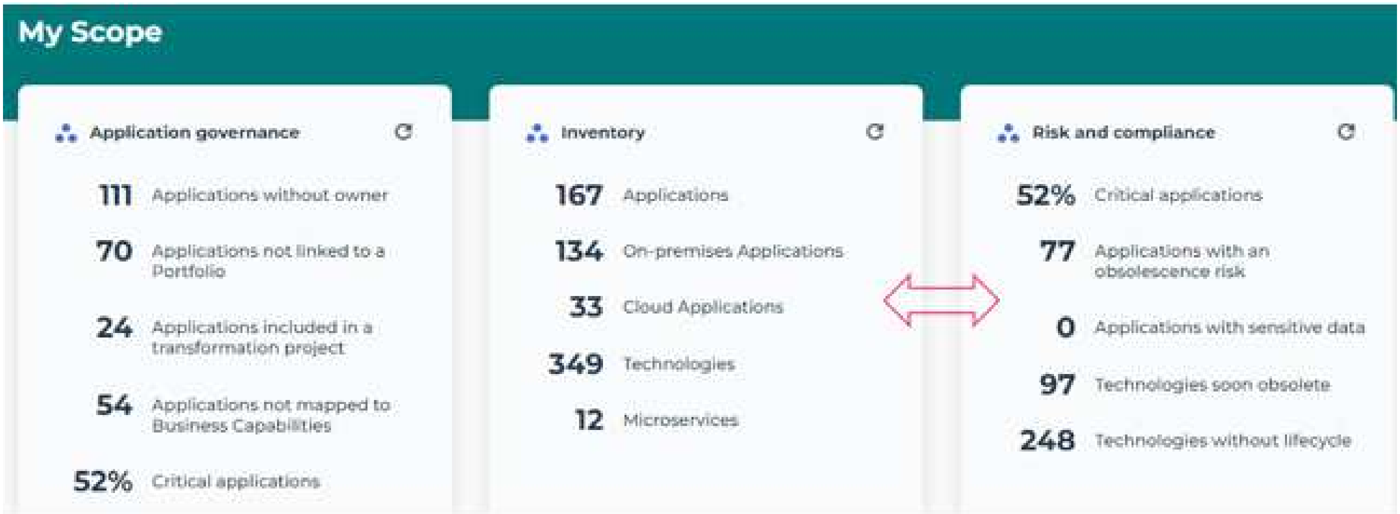
HomePage Scope Enterprise Architect - Scope - Order modification (Available Scope Items)

- Available Scope Items
  - Enterprise Architect - Application governance
  - Enterprise Architect - Inventories
  - Enterprise Architect - Risk and compliance

☐ Alphabetical order

Modify position of an object by drag-and-drop

The "Inventory" and "Risk and compliance" Scope Items are swapped.



My Scope

Application governance

111 Applications without owner

70 Applications not linked to a Portfolio

24 Applications included in a transformation project

54 Applications not mapped to Business Capabilities

52% Critical applications

Inventory

167 Applications

134 On-premises Applications

33 Cloud Applications

349 Technologies

12 Microservices

Risk and compliance

52% Critical applications

77 Applications with an obsolescence risk

0 Applications with sensitive data

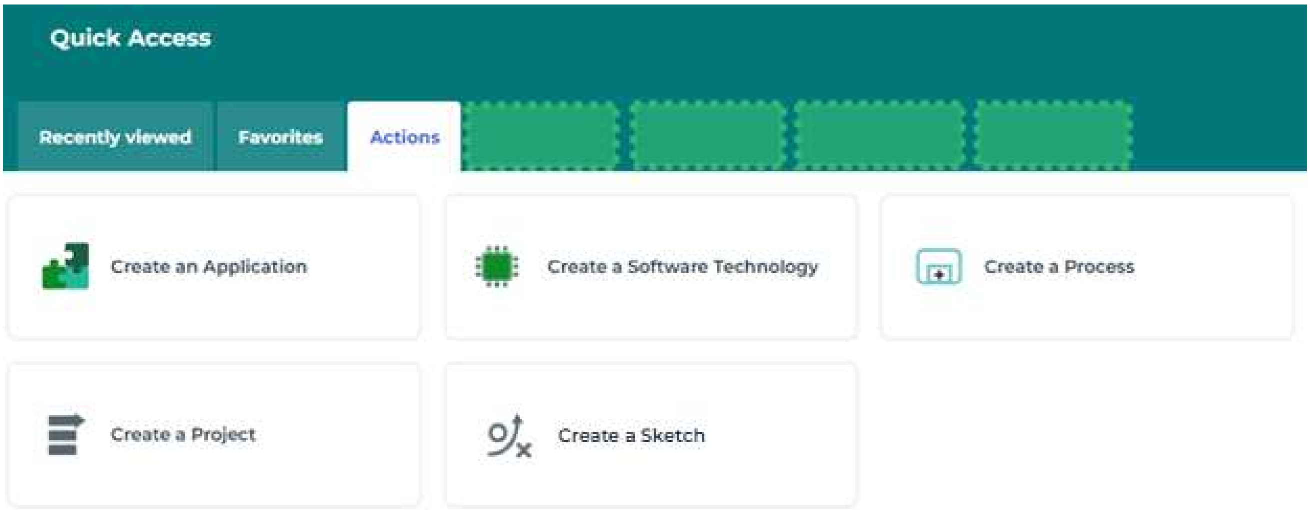
97 Technologies soon obsolete

248 Technologies without lifecycle

12.3.5 Customizing the Quick Access block

By default, the Quick access block includes the following tabs:

- **Recently viewed** (mandatory)
- **Favorites** (mandatory)
- **Actions**



You can:

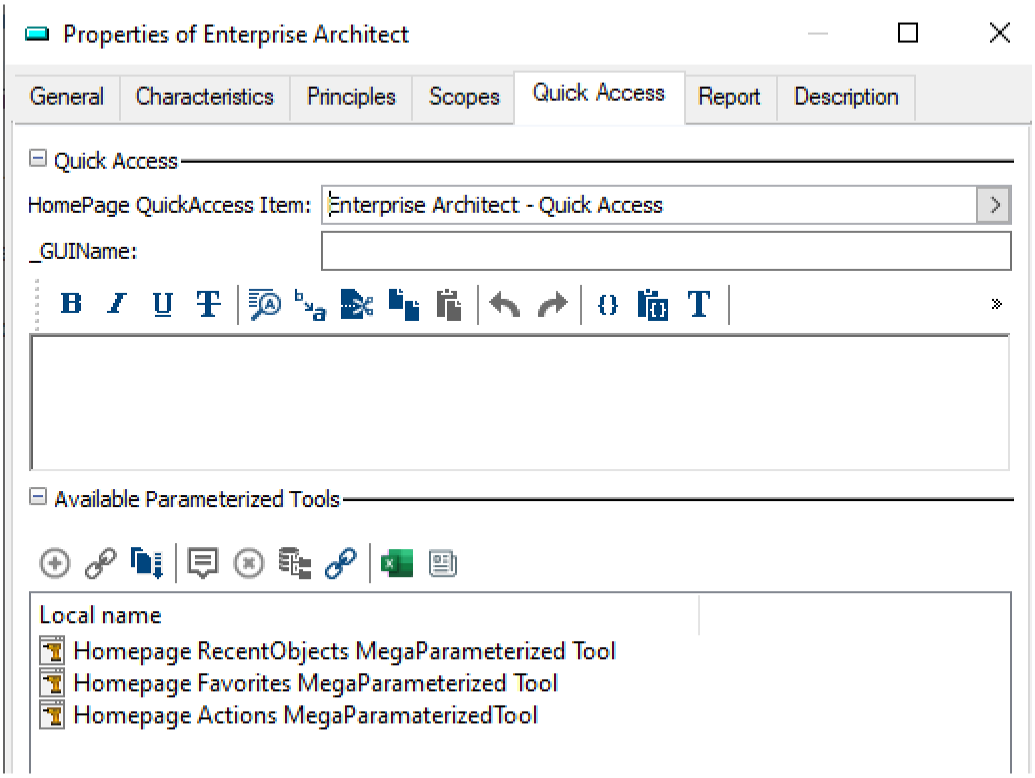
- modify the Quick Access **title**
- modify or remove the **Actions** tab
- create, add, and remove tabs
- reorganize the tab order

To customize the Quick Access block:

1. Access the Homepage properties associated with the Profile.

   `E.g.: Enterprise Architect homepage.`

2. Display its **Quick Access** tab.


Properties of Enterprise Architect

Quick Access

HomePage QuickAccess Item: Enterprise Architect - Quick Access

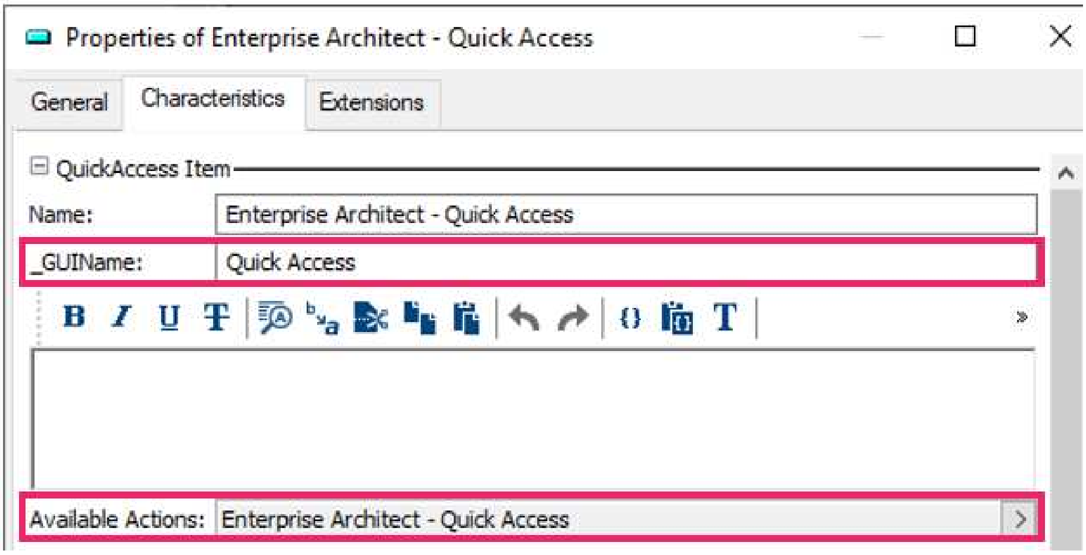
_GUIName:

Available Parameterized Tools

Local name

- Homepage RecentObjects MegaParameterized Tool
- Homepage Favorites MegaParameterized Tool
- Homepage Actions MegaParamaterizedTool

3. In the **Quick Access** section, access the **HomePage QuickAccess Item** properties.


Properties of Enterprise Architect - Quick Access

QuickAccess Item

Name: Enterprise Architect - Quick Access

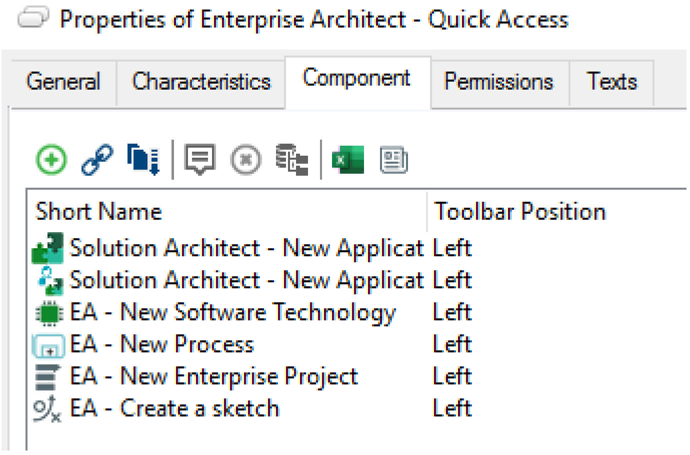
_GUIName: Quick Access

Available Actions: Enterprise Architect - Quick Access

4. To modify the Quick Access title: in the **QuickAccess Item** section, modify the **_GuiName**.

5. To modify the actions: in the **Available Actions** field, access the “Enterprise Architect – Quick Access” MetaCommand Group properties.

   Its **Component** tab displays the available actions (MetaCommand items).

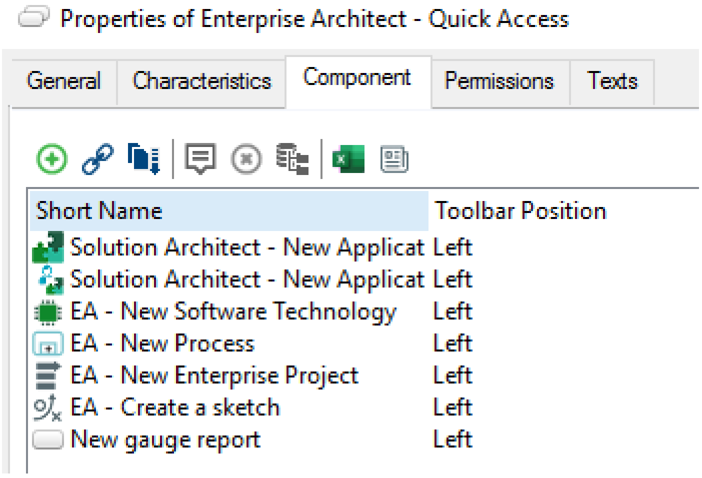

Properties of Enterprise Architect - Quick Access

Short Name	Toolbar Position
Solution Architect - New Applicat	Left
Solution Architect - New Applicat	Left
EA - New Software Technology	Left
EA - New Process	Left
EA - New Enterprise Project	Left
EA - Create a sketch	Left

6. You can modify the **Actions** tab:

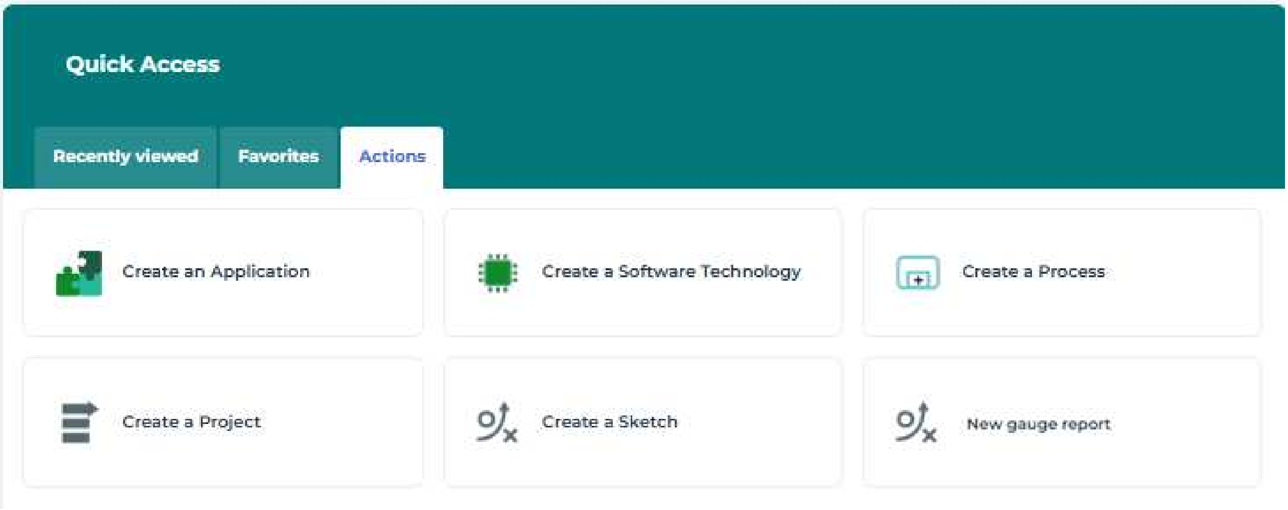
- Remove an action from the **Actions** Tab: select the MetaCommand item and click **Remove**.
- Add an action to the **Actions** Tab: click **Connect** and select the MetaCommand item.

E.g.: New Gauge report

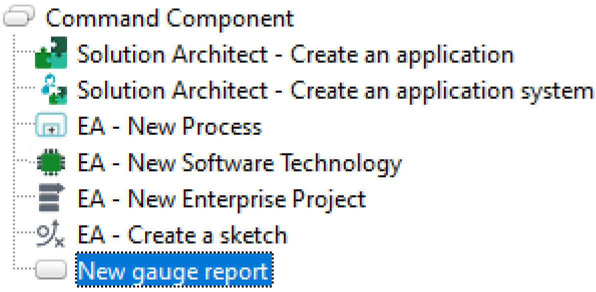

Properties of Enterprise Architect - Quick Access

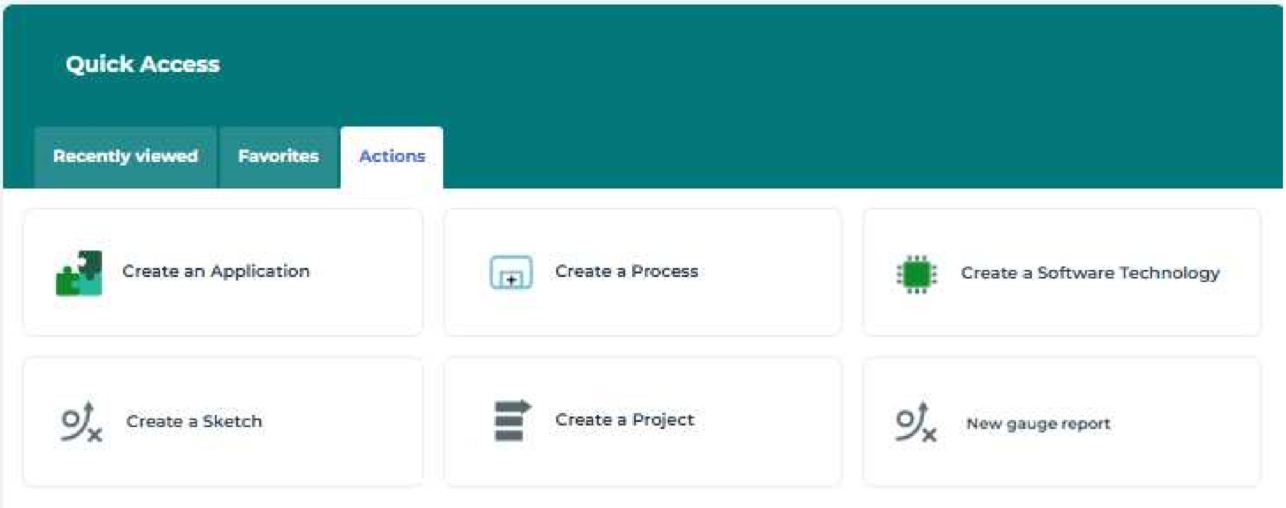
Short Name	Toolbar Position
Solution Architect - New Applicat	Left
Solution Architect - New Applicat	Left
EA - New Software Technology	Left
EA - New Process	Left
EA - New Enterprise Project	Left
EA - Create a sketch	Left
New gauge report	Left

The Action **New gauge report** is added to the Quick access block.

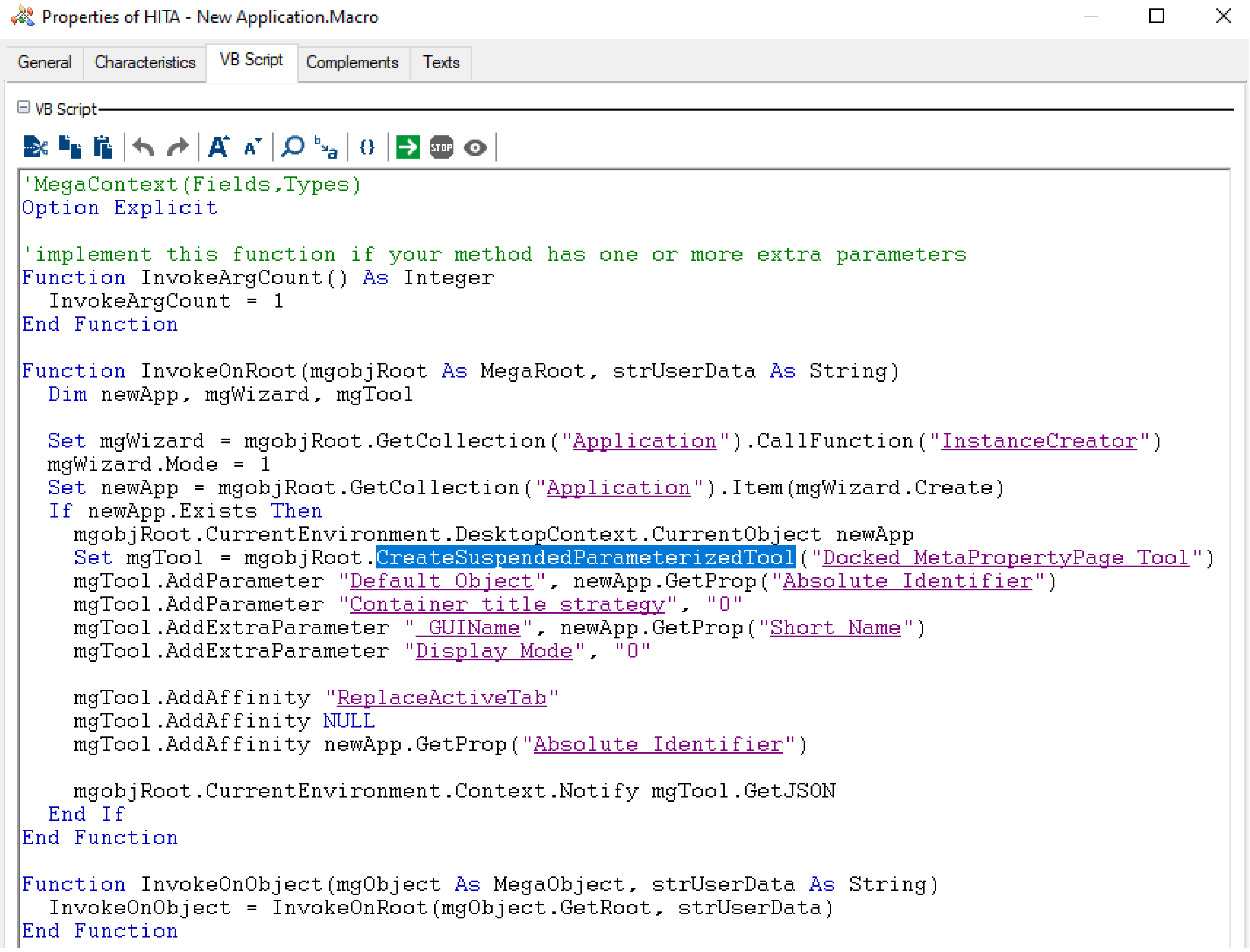


- Modify the action display position: click **Reorganize** and move (drag and drop) the items according to the display you want. Items are displayed from left to right, and top to bottom.


Command Component
Solution Architect - Create an application
Solution Architect - Create an application system
EA - New Process
EA - New Software Technology
EA - New Enterprise Project
EA - Create a sketch
New gauge report



Note: if the MetaCommand item is defined by a macro, it must include a SuspendedParameterizedTool.

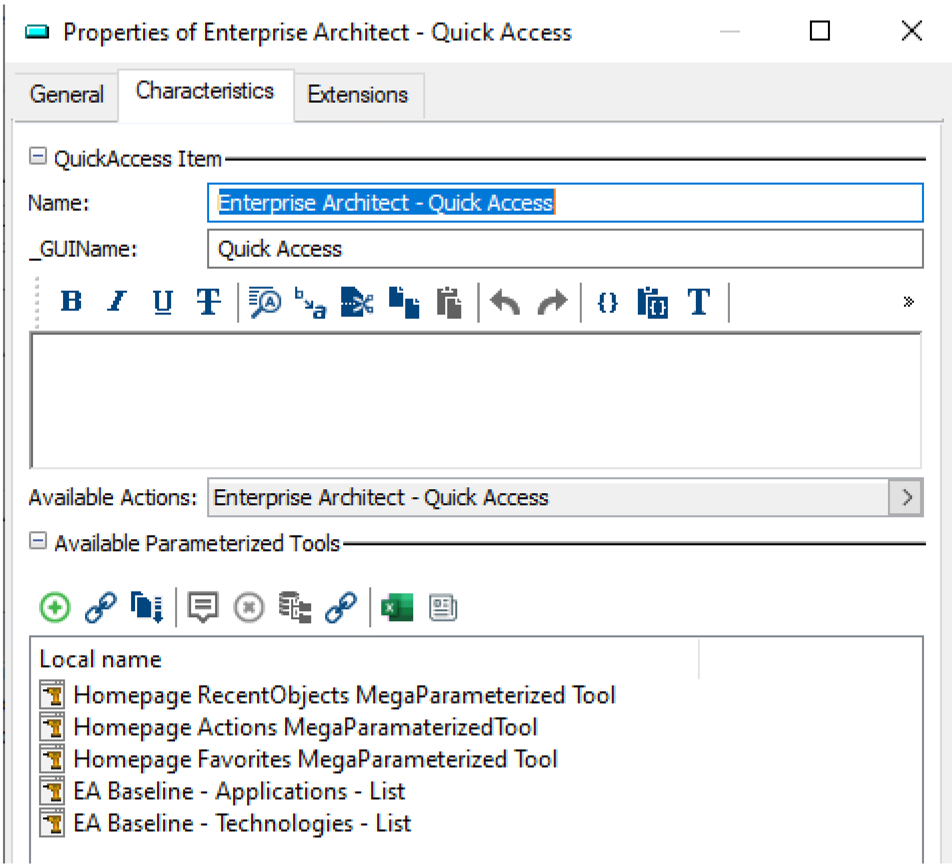


```
'MegaContext(Fields,Types)
Option Explicit

'implement this function if your method has one or more extra parameters
Function InvokeArgCount() As Integer
  InvokeArgCount = 1
End Function

Function InvokeOnRoot(mgobjRoot As MegaRoot, strUserData As String)
  Dim newApp, mgWizard, mgTool

  Set mgWizard = mgobjRoot.GetCollection("Application").CallFunction("InstanceCreator")
  mgWizard.Mode = 1
  Set newApp = mgobjRoot.GetCollection("Application").Item(mgWizard.Create)
  If newApp.Exists Then
    mgobjRoot.CurrentEnvironment.DesktopContext.CurrentObject newApp
    Set mgTool = mgobjRoot.CreateSuspendedParameterizedTool("Docked MetaPropertyPage Tool")
    mgTool.AddParameter "Default Object", newApp.GetProp("Absolute Identifier")
    mgTool.AddParameter "Container title strategy", "0"
    mgTool.AddExtraParameter "_GUIName", newApp.GetProp("Short Name")
    mgTool.AddExtraParameter "Display Mode", "0"

    mgTool.AddAffinity "ReplaceActiveTab"
    mgTool.AddAffinity NULL
    mgTool.AddAffinity newApp.GetProp("Absolute Identifier")

    mgobjRoot.CurrentEnvironment.Context.Notify mgTool.GetJSON
  End If
End Function

Function InvokeOnObject(mgObject As MegaObject, strUserData As String)
  InvokeOnObject = InvokeOnRoot(mgObject.GetRoot, strUserData)
End Function
```

7. To add a tab:

   In the **Quick Access** tab, **Available Parameterized Tool** section, connect the MEGA Parameterized Tool.

   E.g.: "EA – Baseline – Application – List" and "EA – Baseline – Technologies – List".



Properties of Enterprise Architect - Quick Access

QuickAccess Item

Name: Enterprise Architect - Quick Access

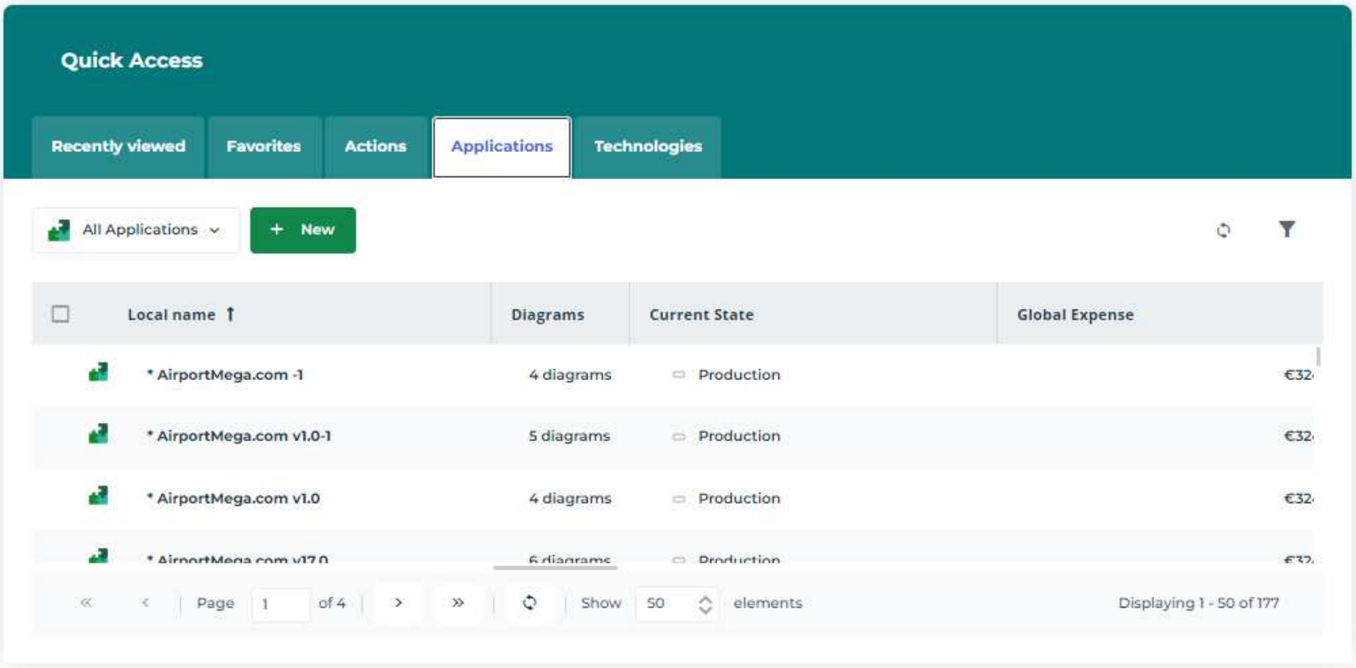
_GUIName: Quick Access

Available Actions: Enterprise Architect - Quick Access

Available Parameterized Tools

Local name
Homepage RecentObjects MegaParameterized Tool
Homepage Actions MegaParamaterizedTool
Homepage Favorites MegaParameterized Tool
EA Baseline - Applications - List
EA Baseline - Technologies - List

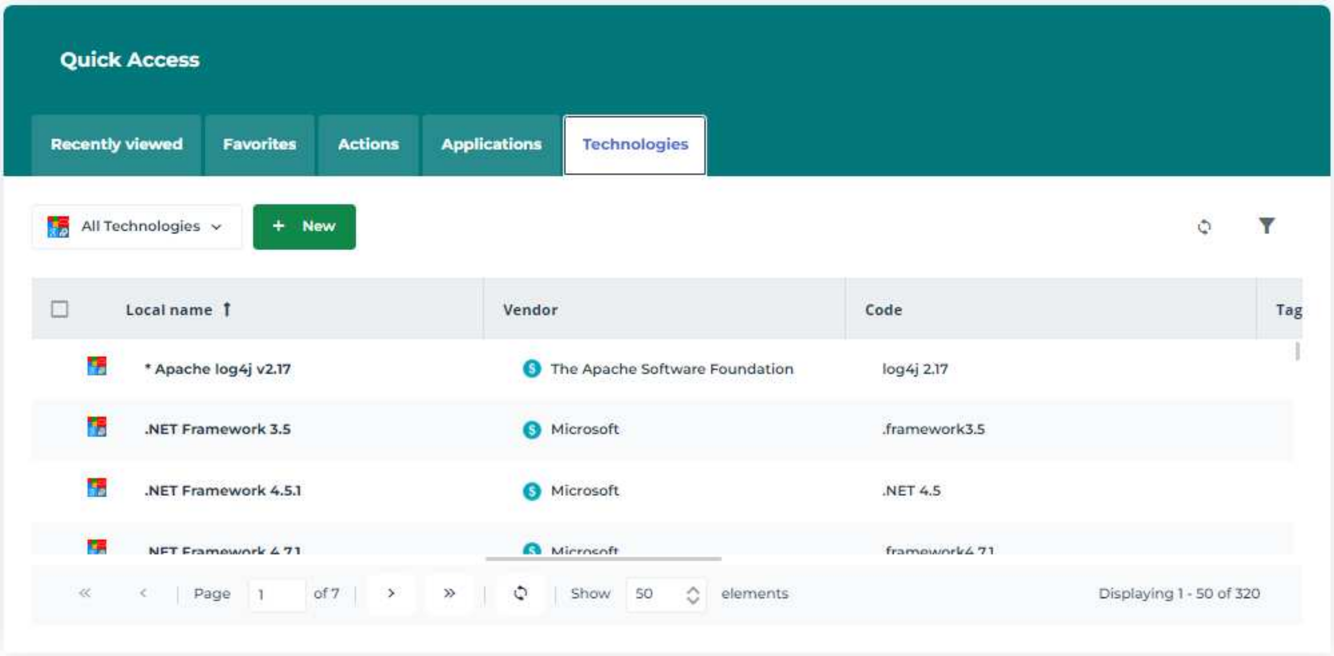
Both new tabs (**Applications** and **Technologies**) are added to the Quick Access:



Quick Access

	Local name ↑	Diagrams	Current State	Global Expense
	* AirportMega.com -1	4 diagrams	Production	€32
	* AirportMega.com v1.0-1	5 diagrams	Production	€32
	* AirportMega.com v1.0	4 diagrams	Production	€32
	* AirportMega.com v17.0	6 diagrams	Production	€32

Page 1 of 4 Show 50 elements Displaying 1 - 50 of 177

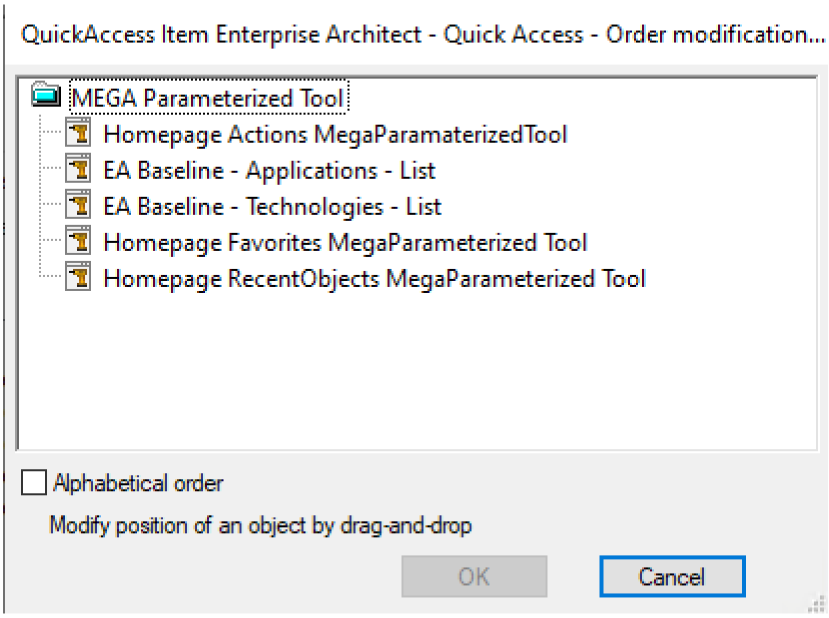


Quick Access

	Local name	Vendor	Code	Tag
	* Apache log4j v2.17	The Apache Software Foundation	log4j 2.17	
	.NET Framework 3.5	Microsoft	.framework3.5	
	.NET Framework 4.5.1	Microsoft	.NET 4.5	
	.NET Framework 4.71	Microsoft	.framework4.71	

Displaying 1 - 50 of 320

8. To modify the tab order, in the **Quick Access** tab, **Available Parameterized Tools** section, click **Reorganize** , drag and drop the item to the required position and click **OK**.

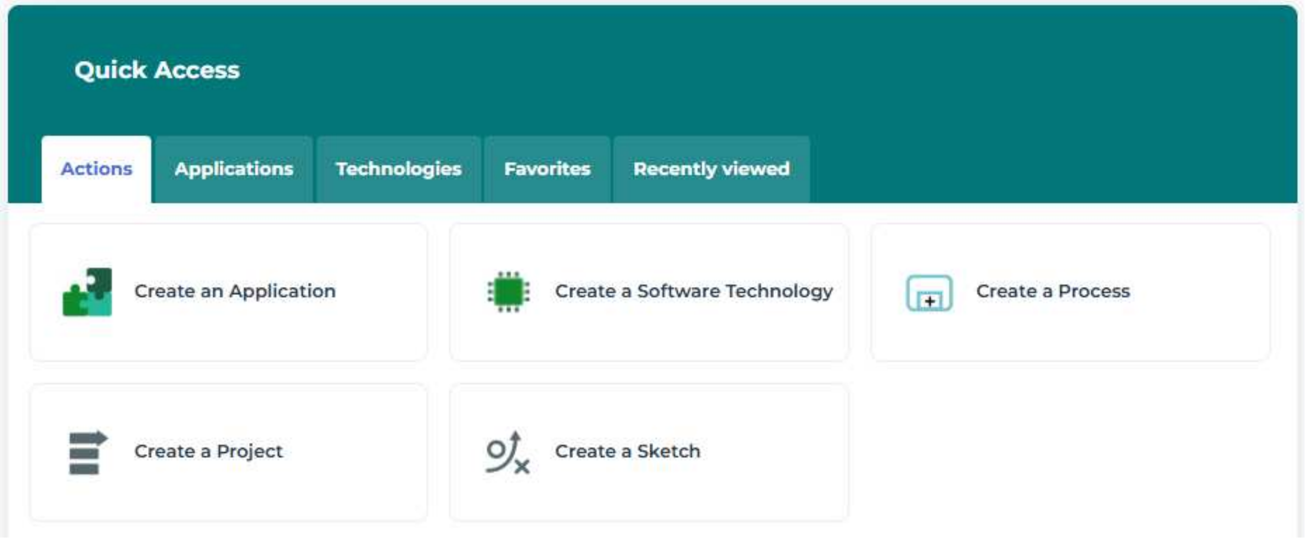


QuickAccess Item Enterprise Architect - Quick Access - Order modification...

- MEGA Parameterized Tool
  - Homepage Actions MegaParamaterizedTool
  - EA Baseline - Applications - List
  - EA Baseline - Technologies - List
  - Homepage Favorites MegaParameterized Tool
  - Homepage RecentObjects MegaParameterized Tool

☐ Alphabetical order

Modify position of an object by drag-and-drop



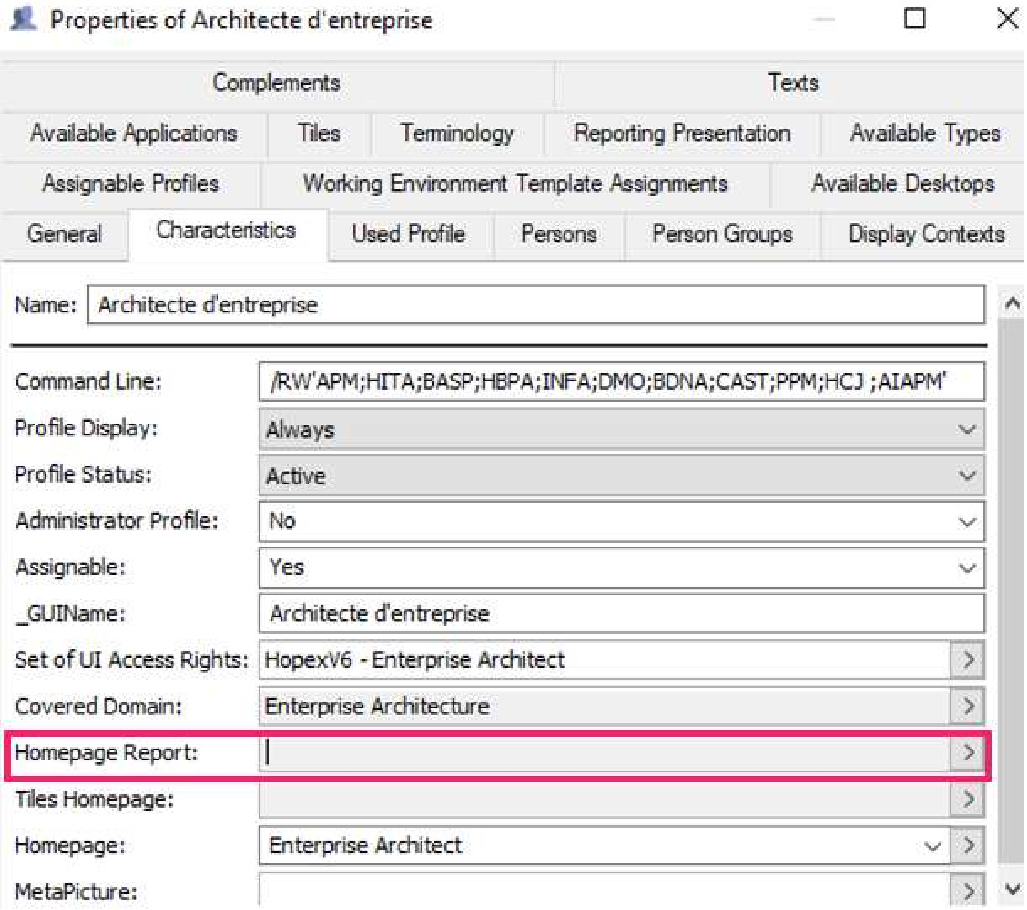
Quick Access

12.3.6 Defining the default report

The default report is the report associated with the profile. It can be defined by HOPEX Customizer profile or by HOPEX Administrator.

To modify the default report:

1. Access the profile properties.
2. In its **Characteristics** tab, in the **Homepage Report** field, click the arrow and connect a report.



Name: Architecte d'entreprise

Command Line: /RW'APM;HITA;BASP;HBPA;INFA;DMO;BDNA;CAST;PPM;HCJ ;AIAPM'

Profile Display: Always

Profile Status: Active

Administrator Profile: No

Assignable: Yes

_GUIName: Architecte d'entreprise

Set of UI Access Rights: HopexV6 - Enterprise Architect

Covered Domain: Enterprise Architecture

Homepage Report:

Tiles Homepage:

Homepage: Enterprise Architect

MetaPicture:

The report is defined by default for all the users connected with this profile.

Users can change the default report by another report.

12.4 Adding Navigation menus to a WET

A Navigation menu is defined by a Working Environment Group Template.

12.4.1 Working Environment Group Template characteristics

A Working Environment Group Template has the following characteristics:

Always Available:

Always Available defines whether the Navigation menu is always available to the end user. This is particularly useful when the WET is instantiated. the Navigation menu is always available except when filtered by the CRUD or a dedicated product.

If not defined the group is filtered according to the CRUD defined or dedicated product.

Display Mode:

The Display Mode defines the way the command is displayed:

- name and icon: both name and command icon are displayed
- name only: the command name is displayed only
- icon only: the command icon is displayed only

Add End Separator:

Add End Separator enables to automatically add a separator at the end of the command group.

_GUIName:

_GUIName defines the Working Environment Group Template display name in HOPEX.

Desktop:

Desktop enables to define/customize a desktop different from the Universal Desktop.

> For example, **Home** and **Dashboards** Working Environment Group Templates are based on specific desktops (Universal Desktop – Home Tiles Desktop and Universal Desktop – Dashboard respectively).

MetaPicture:

MetaPicture defines the Working Environment Group Template icon displayed in HOPEX.

Default Topic Template:

Default Topic Template defines a default Working Environment Topic for the Working Environment Group Template.

12.4.2 Creating a Working Environment Group Template

WET-based desktops include a Navigation menu, which gives access to Navigation menus (defined through Working Environment Group Templates).

A Working Environment Group Template is not specific to a single WET.

> For example, The Administration Navigation menu is included in all the HOPEX Functional Administration desktops.

To create a Working Environment Group Template:

1. Connect to HOPEX with the HOPEX Customizer profile.
2. Display the **MetaStudio** tab (**View > Navigation Windows > MetaStudio**).
3. Right-click the **GroupsTemplate** folder and select **New > Working Environment group template**.
4. In the **Name** field, enter a name to your Working Environment Group Template.
5. In the **_GUIName** field, enter the Navigation menu name displayed in HOPEX.
6. In the **MetaPicture** field, click the arrow and connect an image.

   Else the default Working Environment Group Template image ▬ is displayed in HOPEX.
7. (If needed) Define a **Display Mode** to modify the default behavior.
   - ➢ See Working Environment Group Template characteristics.
8. (If needed) Define the **Always Available** value to modify the default behavior.
   - ➢ See Working Environment Group Template characteristics.
9. (If needed) Select **Add End Separator** to add a separator before the next group.
10. In most cases you must add topics (Working Environment Topic Templates) to the group.
    - ➢ See Adding topics to a Working Environment Group Template.

    In case you do not want to add a topic, you need to define a desktop, see Defining a Desktop to a Working Environment Group Template.
11. Click **OK**.
12. Define the Navigation menu.

- ➢ See Adding a Navigation menu to a WET.

12.4.3 Adding a Navigation menu to a WET

You can add as many Navigation menus (Working Environment Group Templates) as needed to a WET.

To add a Navigation menu to a WET:

1. Access the WET properties.
   - ➢ See Accessing the WET metamodel element properties.
2. Select the **Characteristics** tab.
3. In the **Available Groups** section, click **Connect** 🔗.
4. Select the Working Environment Group Template (Navigation menu) you want to add to the WET.

   You can select as many Working Environment Group Templates as needed.
5. Click **Connect**.

   All the Working Environment Group Templates (Navigation menus) selected are added to the WET.
6. Click **Close**.

12.5 Defining a Navigation Menu (Working Environment Group Template)

To define a Navigation menu:

- **common use**
  Create a topic (Working Environment Topic Template), and define:
  - one or several actions (**MetaCommandItems**), and/or
  - a default action, which is automatically launched when the end-user clicks the Navigation menu.
- **rare and advanced use**
  Create and customize a desktop.

12.5.1 Working Environment Topic Template properties

A Working Environment Topic Template has the following characteristics:

MetaPicture:

The **MetaPicture** defines the Working Environment Topic Template icon displayed in HOPEX.

_GUIName:

The **_GUIName** defines the Working Environment Topic Template display name in HOPEX.

Default Action:

(Optional parameter) The **Default Action** defines the action performed when the end-user clicks the topic.

Actions:

(Mandatory if no **Default Action** is defined) The **Actions** defines the actions that can be performed from the topic.

12.5.2 Creating a Working Environment Topic Template

To create a Working Environment Topic Template:

1. Access the Working Environment Group Template properties.

   ➢ See Accessing the WET metamodel element properties**Error! Reference source not found.**.

2. In the **Name** field, enter your Working Environment Topic Template.
3. In the **MetaPicture** field, click the arrow and connect an image.
4. (Optional) In the **Default Action** field, select an action.

   The action is performed when the end-user clicks the Navigation menu.

5. Click **Next**.
6. In the **_GUIName** field, enter the topic name displayed in HOPEX.
7. In the **Available Actions** pane:

   a. Click **Connect** 🔗.

   b. (if needed) If a WET **Standard Action** is defined, this action is automatically proposed, click **Connect** to add it.

c. Click **Connect** 🔗.

d. To add other action(s), in the first field select **MetaCommand Item** and click 🔍.

e. Select all the actions you want to add and click **Connect**.

12.5.3 Adding topics to a Working Environment Group Template

To add topics to a Working Environment Group Template:

1. Access the Working Environment Group Template properties.

   ➢ See Accessing the WET metamodel element properties.

2. In the **Characteristics** tab, **Available Topics** section, click **Connect** 🔗.

3. In the **Working Environment Topic Template** list, select all the topics you want to add and click **Connect**.

   The selected topics are added to the Working Environment Group Template.

4. Define the topic, see Defining a Working Environment Topic Template (topic).

12.5.4 Defining a Desktop to a Working Environment Group Template

Prerequisite: the desktop you want to connect to the Working Environment Group Template must be created, see Creating a desktop.

To define a Desktop to a Working Environment Group Template:

1. Access the Working Environment Group Template properties.

   ➢ See Accessing the WET metamodel element properties.

2. In the **Characteristics** tab, in the **Desktop** field, click the arrow and select **Connect Desktop**.

3. Select the desktop concerned and click **OK**.

12.5.5 Defining a Working Environment Topic Template (topic)

To define a Working Environment Topic template (topic) you must define its:

- displayed characteristics
- actions

If needed, you can also define a default action to the topic. This action is performed when the end-user clicks the topic.

To define a Working Environment Topic Template:

1. Access the Working Environment Topic Template properties.

   ➢ See Accessing the WET metamodel element properties.

2. In the Characteristics tab:

   - In the **MetaPicture** field, click the arrow and select the image you want to be displayed for the topic.
   - In the **_GUIName** field, enter the name you want to be displayed for the topic.

3. In the **Characteristics** tab, **Available Actions** section, you need to define at least one action and/or default action:

   - (if needed) In the **Default Action** field, click the right-oriented arrow and select a MetaCommanditem to define the action you want to be performed by default when the end-user clicks the topic.
   - If you did not define a default action, or if you need to add actions to the Working Environment Topic Template.

     ➢ See Adding actions to a Working Environment Topic Template.

12.5.6 Adding actions to a Working Environment Topic Template

You can add as many actions as needed to the Working Environment Topic Template.

When the WET has a **Standard Action** defined, this action is automatically proposed when connecting actions to the Working Environment Topic Template.

To configure the Actions, see Defining the Actions with Views.

To add actions to a Working Environment Topic Template:

1. Access the Working Environment Topic Template properties.

   ➢ See Accessing the WET metamodel element properties.

2. Select **Characteristics** tab.

3. In the **Available Actions** section, click **Connect** 🔗:

   - If the WET has a **Standard Action** defined, the action is automatically added to the list of actions.
   - To add other actions, in the first field select **MetaCommandItem** field, and select an action in the result list.

4. Click **Connect** to add the action.

   You can add as many actions as needed.

12.6 Defining the Actions with Views

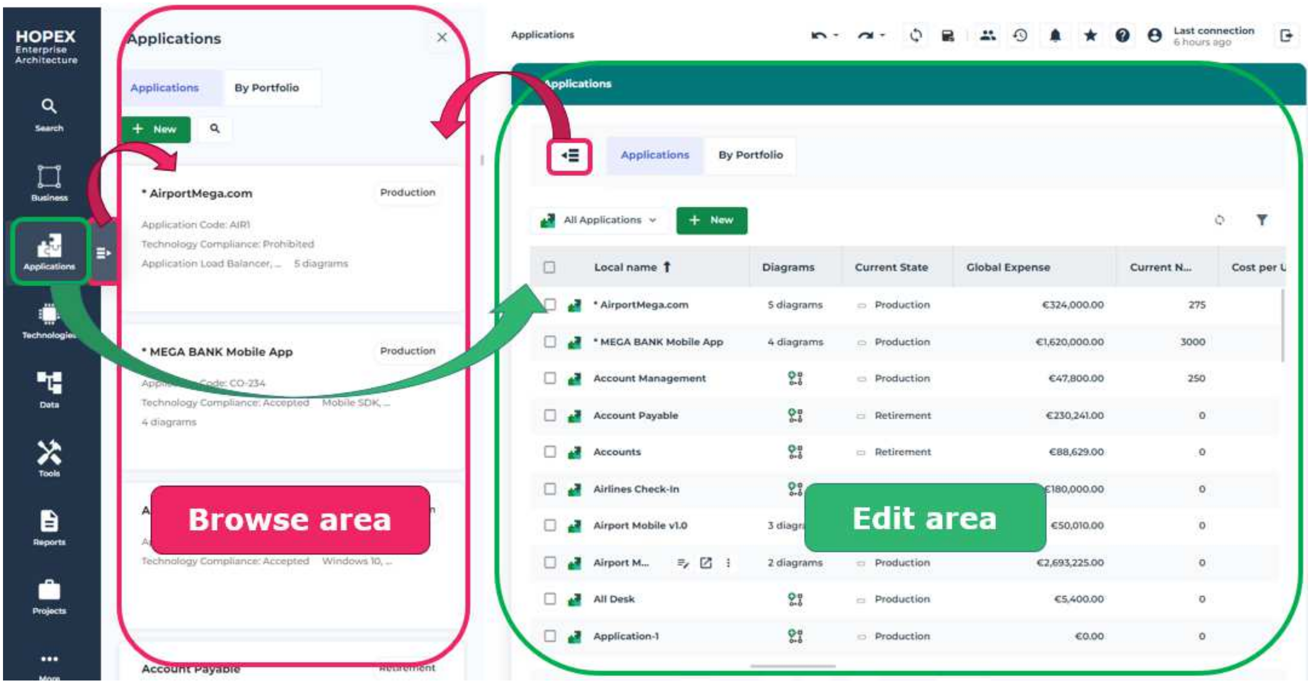
HOPEX Aquila type desktop shows a layout with two communicating areas:

- the Edit area (large View)
- the Browse area (small View)

The tools displayed in both areas are similar and there is a mapping between them. It is required to have a bidirectional list of parameterized tools.

When displaying information in:

- the Browse area, the existing navigation remains in the breadcrumb
- the Edit area, a new navigation starts (breadcrumb reset).



* AirportMega.com — Production
Application Code: AIR1
Technology Compliance: Prohibited
Application Load Balancer, ... 5 diagrams

* MEGA BANK Mobile App — Production
Application Code: CO-234
Technology Compliance: Accepted Mobile SDK, ...
4 diagrams

Technology Compliance: Accepted Windows 10, ...

Account Payable — Retirement

□	Local name	Diagrams	Current State	Global Expense	Current N...	Cost per U
□	* AirportMega.com	5 diagrams	Production	€324,000.00	275	
□	* MEGA BANK Mobile App	4 diagrams	Production	€1,620,000.00	3000	
□	Account Management		Production	€47,800.00	250	
□	Account Payable		Retirement	€230,241.00	0	
□	Accounts		Retirement	€88,629.00	0	
□	Airlines Check-In			€180,000.00	0	
□	Airport Mobile v1.0	3 diagr		€50,010.00	0	
□	Airport M...	2 diagrams	Production	€2,693,225.00	0	
□	All Desk		Production	€5,400.00	0	
□	Application-1		Production	€0.00	0	

HOPEX Aquila type desktop includes the following concepts:

- **View Tool**

  The **View Tool** defines the list of **Mega Parameterized Tools** available for a specific **MetaClass** according to the **View** type (large/small).

  The **View Tool** is defined on the **Display Context** associated with the MetaClass.

- **Display Context**

  The **Display Context** enables to customize a single MetaClass for a profile. A profile has a Display Context for each MetaClass.

  The same Display Context can be used for several profiles.

These MEGA Parameterized Tools are defined in the **WET actions** as a dedicated tool: **MetaClass View Tool** (in the MEGA Parameterized Tool properties **Description > Tool**).

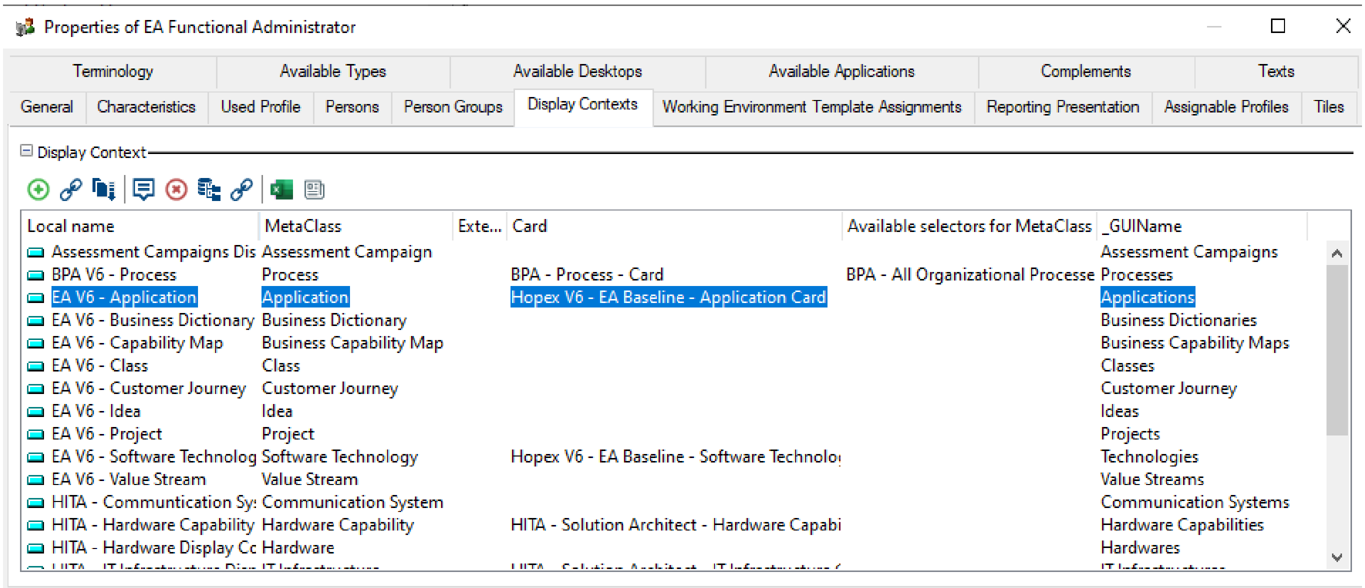
- ➢ See Adding actions to a Working Environment Topic Template.

When the View tool is launched, the current profile Display Context indicates the MEGA Parameterized Tools launched according to the display area.

To define the actions available in the Views:

1. In the profile properties, display its **Display Contexts** tab.

   E.g.: the **Enterprise Architect** profile.



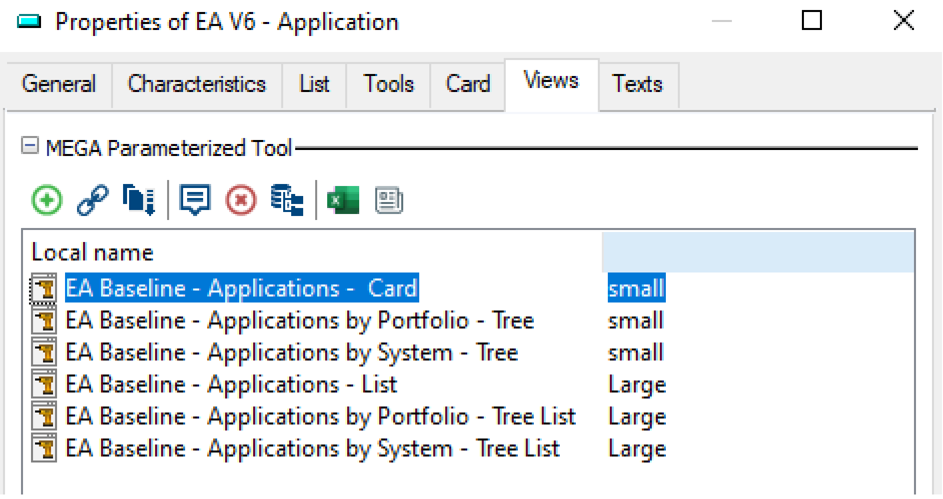
Local name	MetaClass	Exte...	Card	Available selectors for MetaClass	_GUIName
Assessment Campaigns Dis	Assessment Campaign				Assessment Campaigns
BPA V6 - Process	Process		BPA - Process - Card	BPA - All Organizational Processe	Processes
EA V6 - Application	Application		Hopex V6 - EA Baseline - Application Card		Applications
EA V6 - Business Dictionary	Business Dictionary				Business Dictionaries
EA V6 - Capability Map	Business Capability Map				Business Capability Maps
EA V6 - Class	Class				Classes
EA V6 - Customer Journey	Customer Journey				Customer Journey
EA V6 - Idea	Idea				Ideas
EA V6 - Project	Project				Projects
EA V6 - Software Technolog	Software Technology		Hopex V6 - EA Baseline - Software Technolo		Technologies
EA V6 - Value Stream	Value Stream				Value Streams
HITA - Communtication Sy:	Communication System				Communication Systems
HITA - Hardware Capability	Hardware Capability		HITA - Solution Architect - Hardware Capabi		Hardware Capabilities
HITA - Hardware Display Cc	Hardware				Hardwares

2. Access the Display Context properties of the MetaClass concerned.

   E.g.: "EA V6 – Application" Display Context of the **Application** MetaClass.

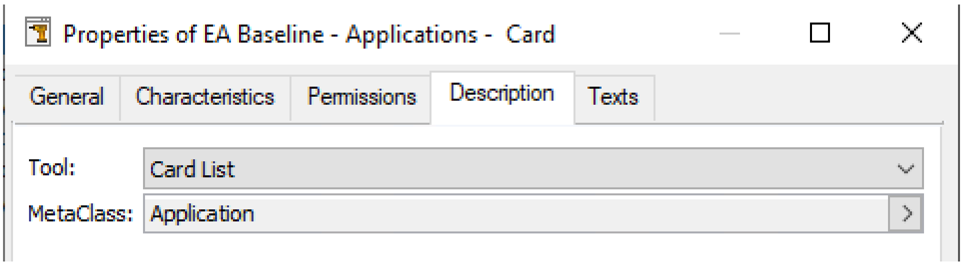
3. Display its **Views** property.

   It lists the **MEGA Parameterized Tools** used for each View Context (small, large)



Local name	
EA Baseline - Applications - Card	small
EA Baseline - Applications by Portfolio - Tree	small
EA Baseline - Applications by System - Tree	small
EA Baseline - Applications - List	Large
EA Baseline - Applications by Portfolio - Tree List	Large
EA Baseline - Applications by System - Tree List	Large

EA Baseline – Applications – Card **MEGA Parameterized Tool** is launched as the Card List **Tool** in the Browse area (small View) for the Application MetaClass.

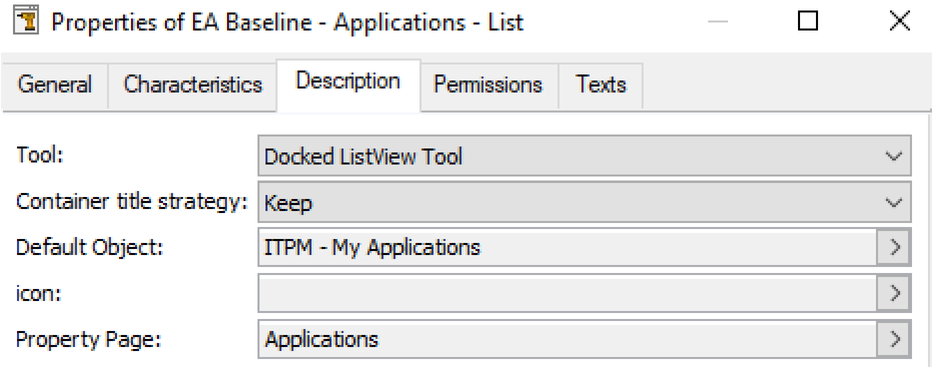


Properties of EA Baseline - Applications - Card

Tool: Card List

MetaClass: Application

EA Baseline – Applications – List **MEGA Parameterized Tool** is launched as the Docked ListView **Tool** in the Edit area (large View) for the Application MetaClass.



Properties of EA Baseline - Applications - List

Tool: Docked ListView Tool

Container title strategy: Keep

Default Object: ITPM - My Applications

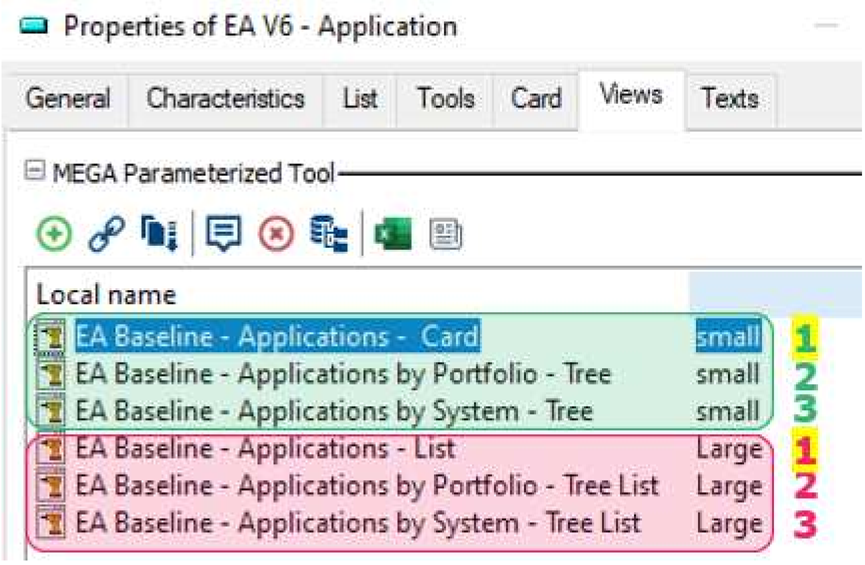
icon:

Property Page: Applications

4. Order the tools so as to get the mapping between the Edit and Browse areas: click **Reorganize** .

   The mapping between the tools in Edit (large) area and Browse (small) area is performed with the tool display order in **Views**. You must have as many tools defined in both Browse and Edit areas.

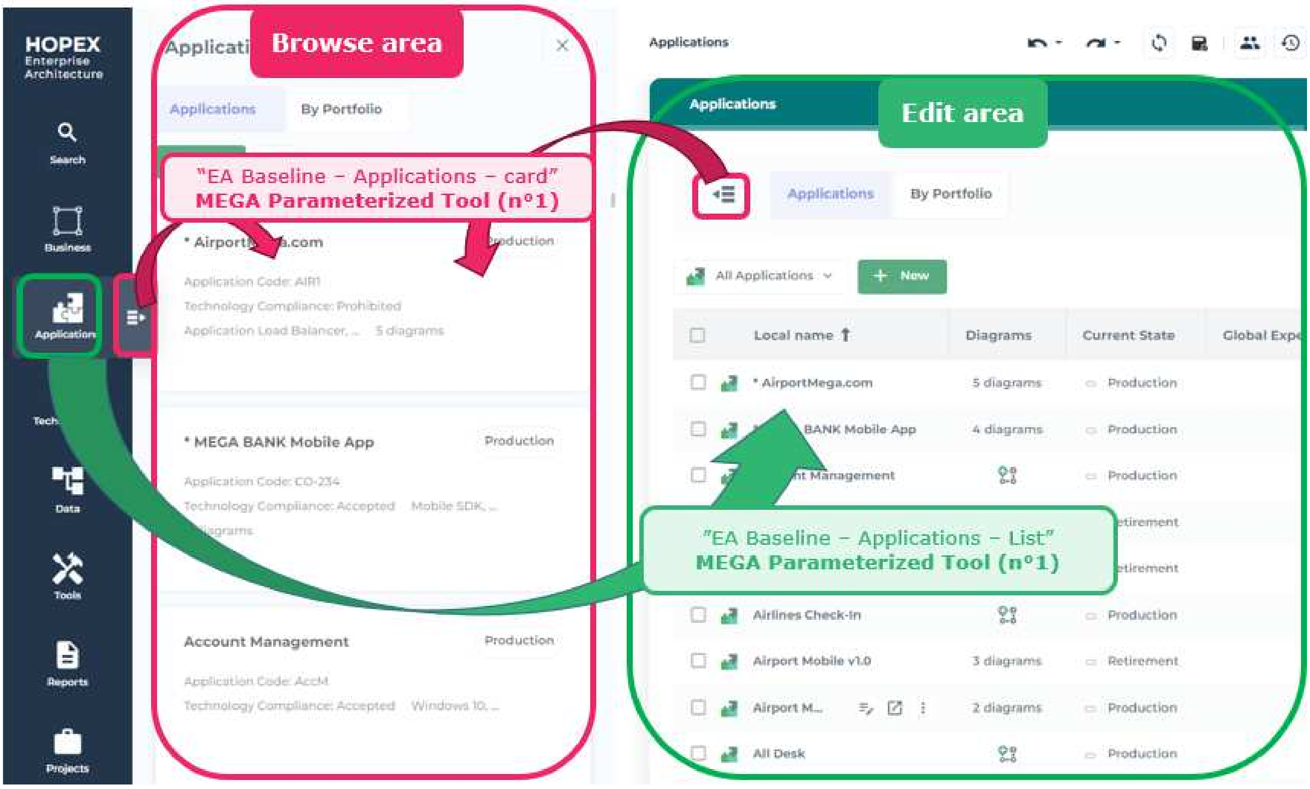
   For example, for **Application** MetaClass "EA Baseline – Application – Card" tool n°1 in Browse area is mapped with "EA Baseline – Application – List" n°1 in Edit area.



Properties of EA V6 - Application

MEGA Parameterized Tool

Local name		
EA Baseline - Applications - Card	small	1
EA Baseline - Applications by Portfolio - Tree	small	2
EA Baseline - Applications by System - Tree	small	3
EA Baseline - Applications - List	Large	1
EA Baseline - Applications by Portfolio - Tree List	Large	2
EA Baseline - Applications by System - Tree List	Large	3



"EA Baseline – Applications – card"
MEGA Parameterized Tool (n°1)

* AirportM...a.com Production
Application Code: AIR1
Technology Compliance: Prohibited
Application Load Balancer, ... 5 diagrams

* MEGA BANK Mobile App Production
Application Code: CO-234
Technology Compliance: Accepted Mobile SDK, ...
...iagrams

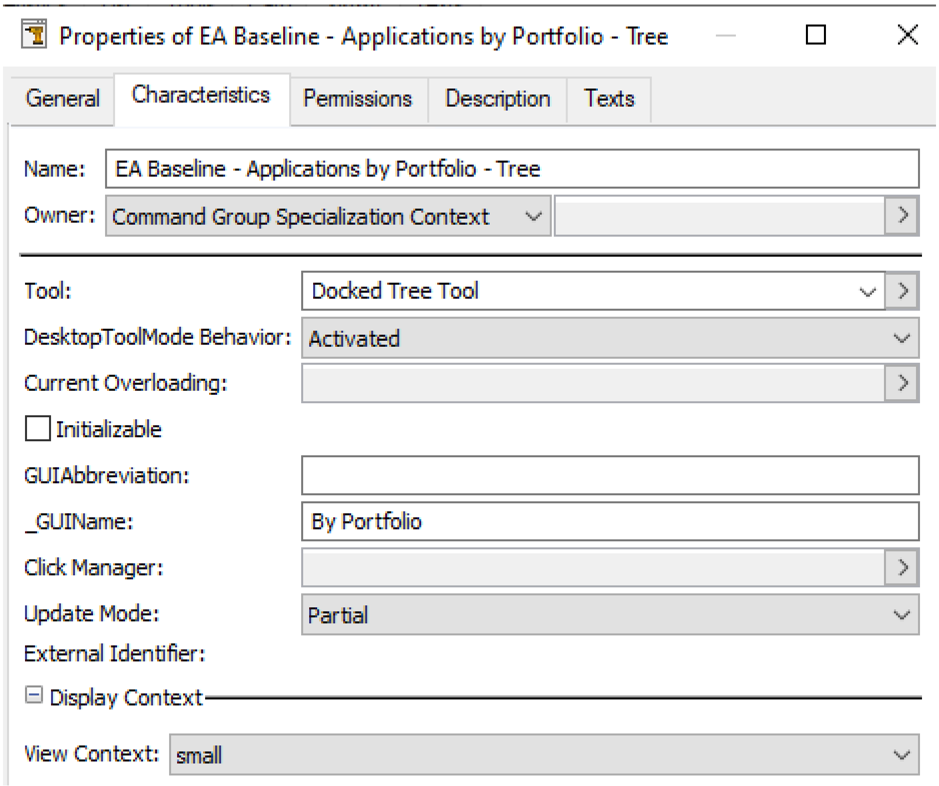
Account Management Production
Application Code: AccM
Technology Compliance: Accepted Windows 10, ...

"EA Baseline – Applications – List"
MEGA Parameterized Tool (n°1)

Local name	Diagrams	Current State	Global Expe
* AirportMega.com	5 diagrams	Production	
... BANK Mobile App	4 diagrams	Production	
...nt Management		Production	
		...etirement	
		...etirement	
Airlines Check-In		Production	
Airport Mobile v1.0	3 diagrams	Retirement	
Airport M...	2 diagrams	Production	
All Desk		Production	

5. (Browse area) Define the button name: in the Mega Parameterized Tool Characteristics (with View Context: “small”), in **_GUIName** field: enter the button name.

   EA Baseline – Applications by Portfolio – Tree **MEGA Parameterized Tool** is launched as the Docked Tree Tool in the Browse area (small View) for the Application MetaClass, with the button name **By Portfolio** (_GUIName).



Properties of EA Baseline - Applications by Portfolio - Tree

Name: EA Baseline - Applications by Portfolio - Tree

Owner: Command Group Specialization Context

Tool: Docked Tree Tool

DesktopToolMode Behavior: Activated

Current Overloading:

□ Initializable

GUIAbbreviation:

_GUIName: By Portfolio

Click Manager:

Update Mode: Partial

External Identifier:

⊟ Display Context

View Context: small

12.7 Defining the Cards

HOPEX desktops include the following concepts:

- **Card**

  A card shows the main information that identifies an object.

  Cards can be displayed in:

  - the Edit area, as the **Overview** page of object properties
  - the Browse area, as a **card list** in the navigation submenus

  You can configure the card content. If no specific card is defined for the MetaClass, the default card is applied (**Name** and **Description** only).

  A card is defined on the **Display Context** associated with the MetaClass.

- **Display Context**

  The **Display Context** enables to customize a single MetaClass. A profile has a Display Context for each MetaClass. The same Display Context can be used for several profiles.

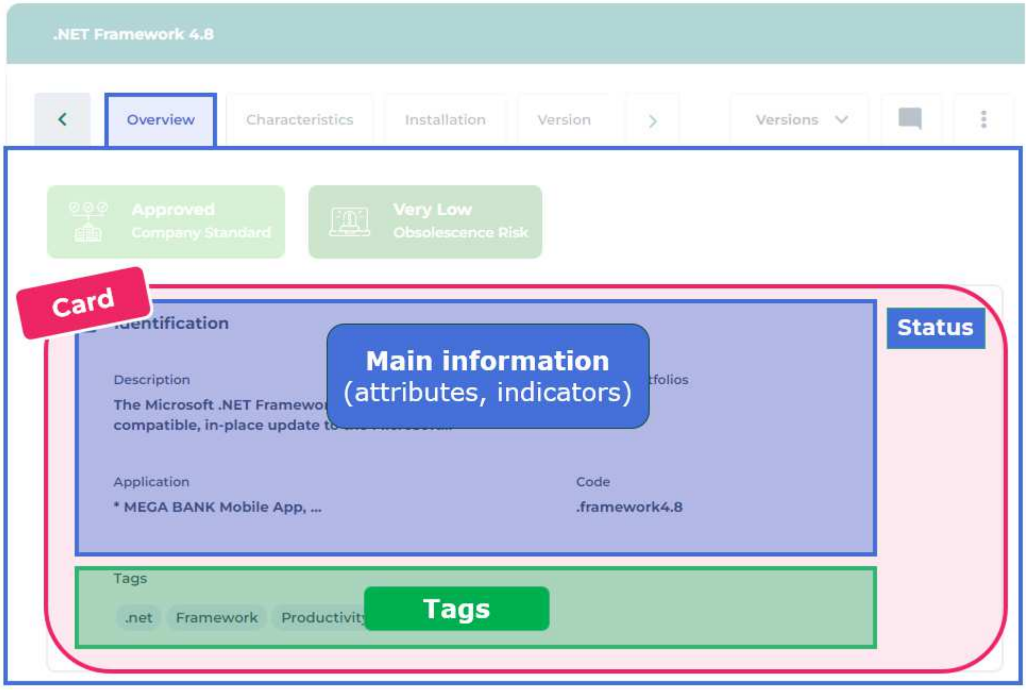
The card is defined on the Display Context of the profile. You can define one card by profile.

12.7.1 Modifying a card display

You can define:

- where the card is displayed: in both the Edit and Browse areas or in only one of them
- the card items: a MetaAttribute, a MetaAssociationEnd, or a query
- each information display location: Attributes (Main/General information), Indicators (Main/General information), Name, Status, or Tags

Edit area: **Overview** property page



Identification

Description
The Microsoft .NET Framewo
compatible, in-place update to

…tfolios

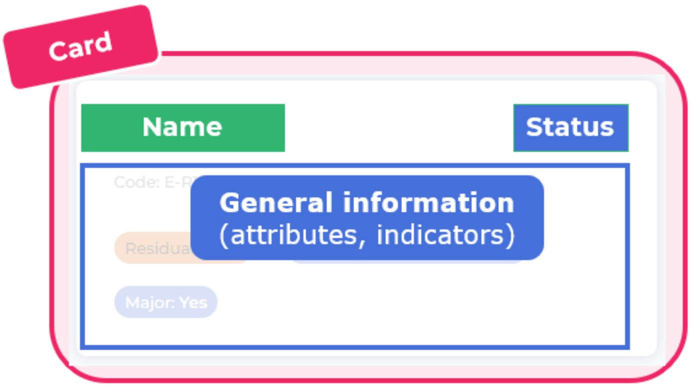
Application
* MEGA BANK Mobile App, ...

Code
.framework4.8

Tags
.net Framework Productivity

Card Item **Name** is not available in the **Overview** page. The page name is the object one.

Browse area: **card list**



Code: E-R

Residua

Major: Yes

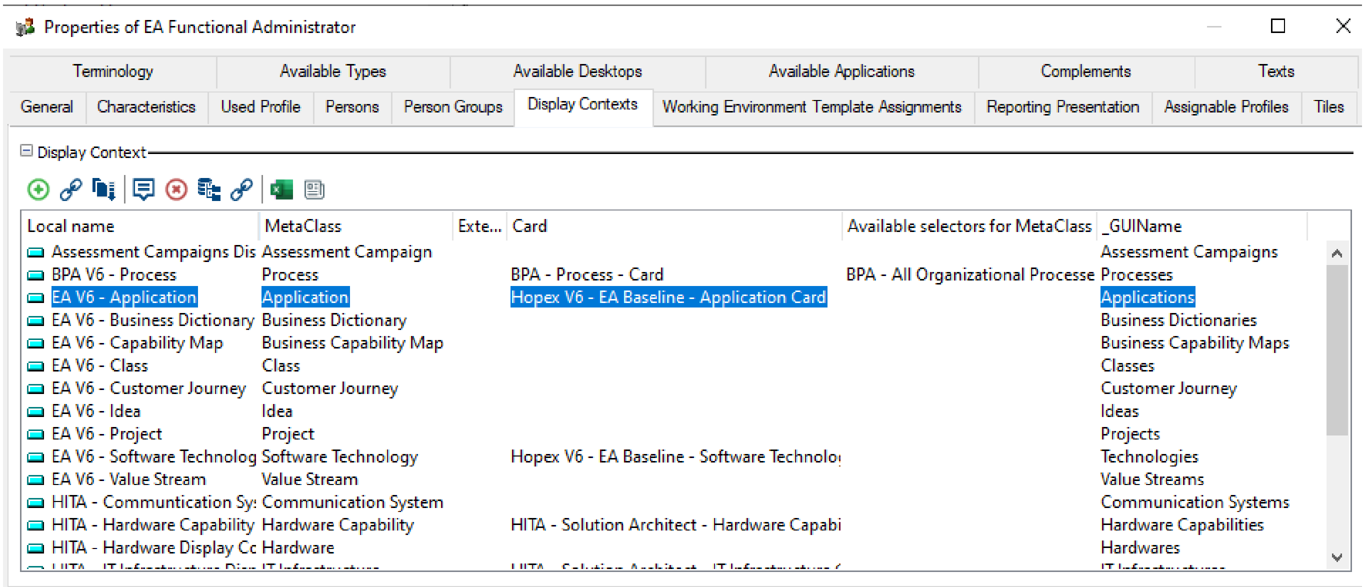
As smaller in the Browse area, the card shows restricted amount of information.

Tags are not available in the card list.

To modify a card display:

1. In the profile properties, display its **Display Contexts** tab.

   E.g.: the **Enterprise Architect** profile.



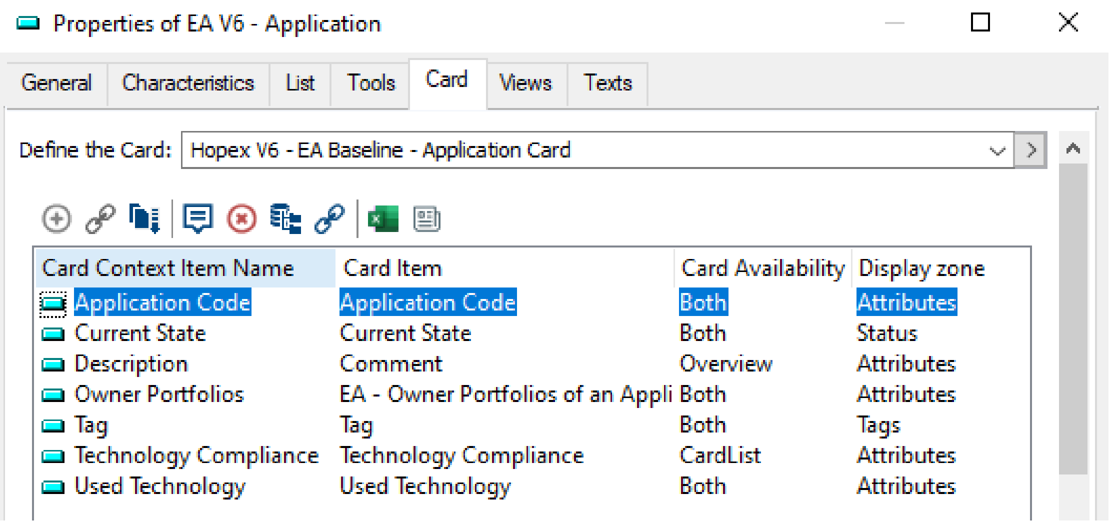
Local name	MetaClass	Exte...	Card	Available selectors for MetaClass	_GUIName
Assessment Campaigns Dis	Assessment Campaign				Assessment Campaigns
BPA V6 - Process	Process		BPA - Process - Card	BPA - All Organizational Processe	Processes
EA V6 - Application	Application		Hopex V6 - EA Baseline - Application Card		Applications
EA V6 - Business Dictionary	Business Dictionary				Business Dictionaries
EA V6 - Capability Map	Business Capability Map				Business Capability Maps
EA V6 - Class	Class				Classes
EA V6 - Customer Journey	Customer Journey				Customer Journey
EA V6 - Idea	Idea				Ideas
EA V6 - Project	Project				Projects
EA V6 - Software Technolog	Software Technology		Hopex V6 - EA Baseline - Software Technolo		Technologies
EA V6 - Value Stream	Value Stream				Value Streams
HITA - Communtication Sy	Communication System				Communication Systems
HITA - Hardware Capability	Hardware Capability		HITA - Solution Architect - Hardware Capabi		Hardware Capabilities
HITA - Hardware Display Cc	Hardware				Hardwares

2. Access the Display Context properties of the MetaClass concerned.

    E.g.: "EA V6 – Application" Display Context of the **Application** MetaClass.

3. Display its **Card** property.

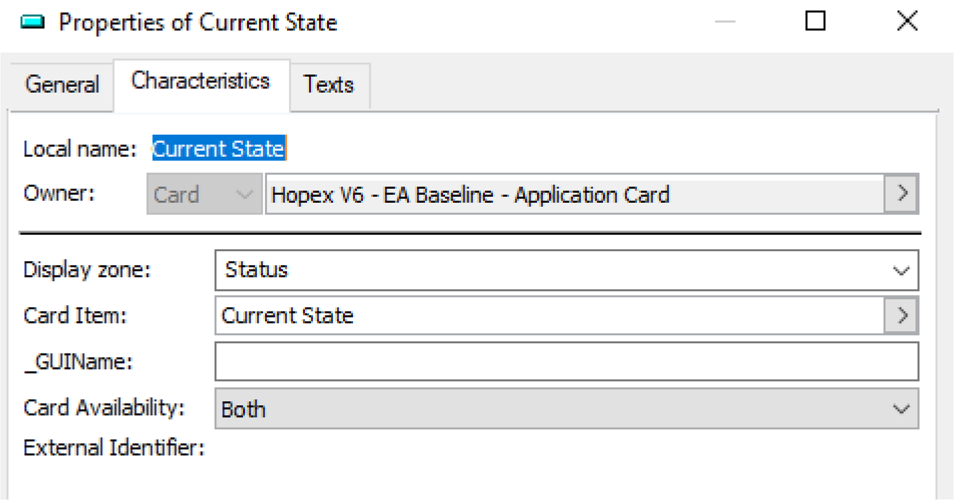
    It lists the object information (**Card Item**) available in the card. For each information, it shows its availability (**Card Availability**) and **Display zone**.



Define the Card: Hopex V6 - EA Baseline - Application Card

Card Context Item Name	Card Item	Card Availability	Display zone
Application Code	Application Code	Both	Attributes
Current State	Current State	Both	Status
Description	Comment	Overview	Attributes
Owner Portfolios	EA - Owner Portfolios of an Appli	Both	Attributes
Tag	Tag	Both	Tags
Technology Compliance	Technology Compliance	CardList	Attributes
Used Technology	Used Technology	Both	Attributes

4. Access the properties of the information concerned.



Properties of Current State

Local name: Current State

Owner: Card | Hopex V6 - EA Baseline - Application Card

Display zone: Status

Card Item: Current State

_GUIName:

Card Availability: Both

External Identifier:

5. Modify its values as required:

   - **Display zone** field (Attributes, Indicators, Name, Status, Tags)
   - **Card Item** field (card name in the Browse area)
   - **_GUIName** field to overload the name
   - **Card Availability** field (CardList, Overview, Both)

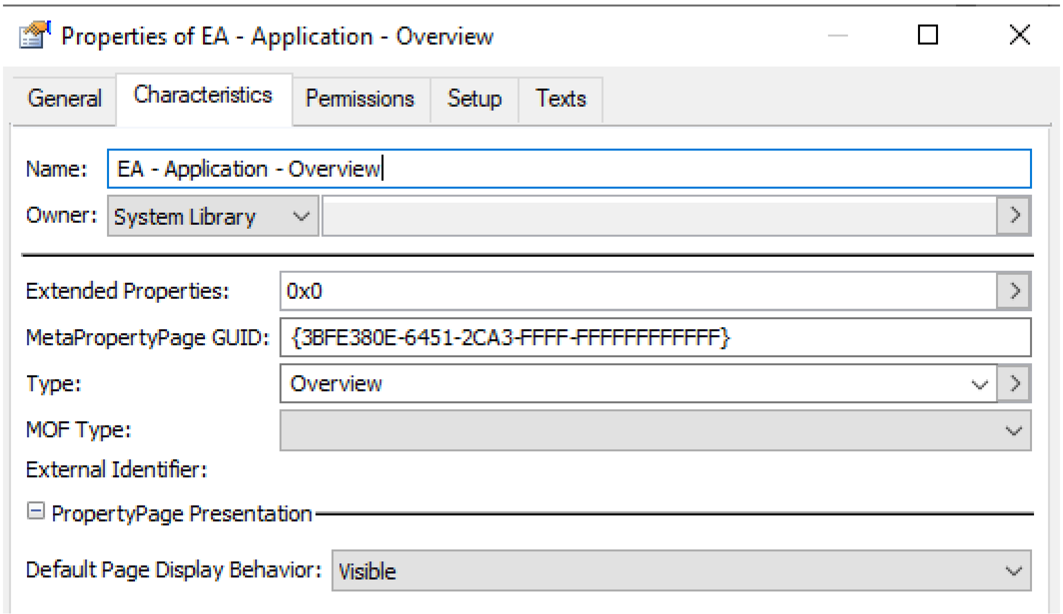
12.7.2 Using the control card

To overload the **Overview** (or **Characteristics**) page, the MetaPropertyPage must be configured ad follows:

- in **Characteristics**:

  Type: "Overview" (or "Characteristics")

```
For example, the EA – Application – Overview
```



Properties of EA - Application - Overview

Name: EA - Application - Overview

Owner: System Library

Extended Properties: 0x0

MetaPropertyPage GUID: {3BFE380E-6451-2CA3-FFFF-FFFFFFFFFFFF}

Type: Overview

MOF Type:

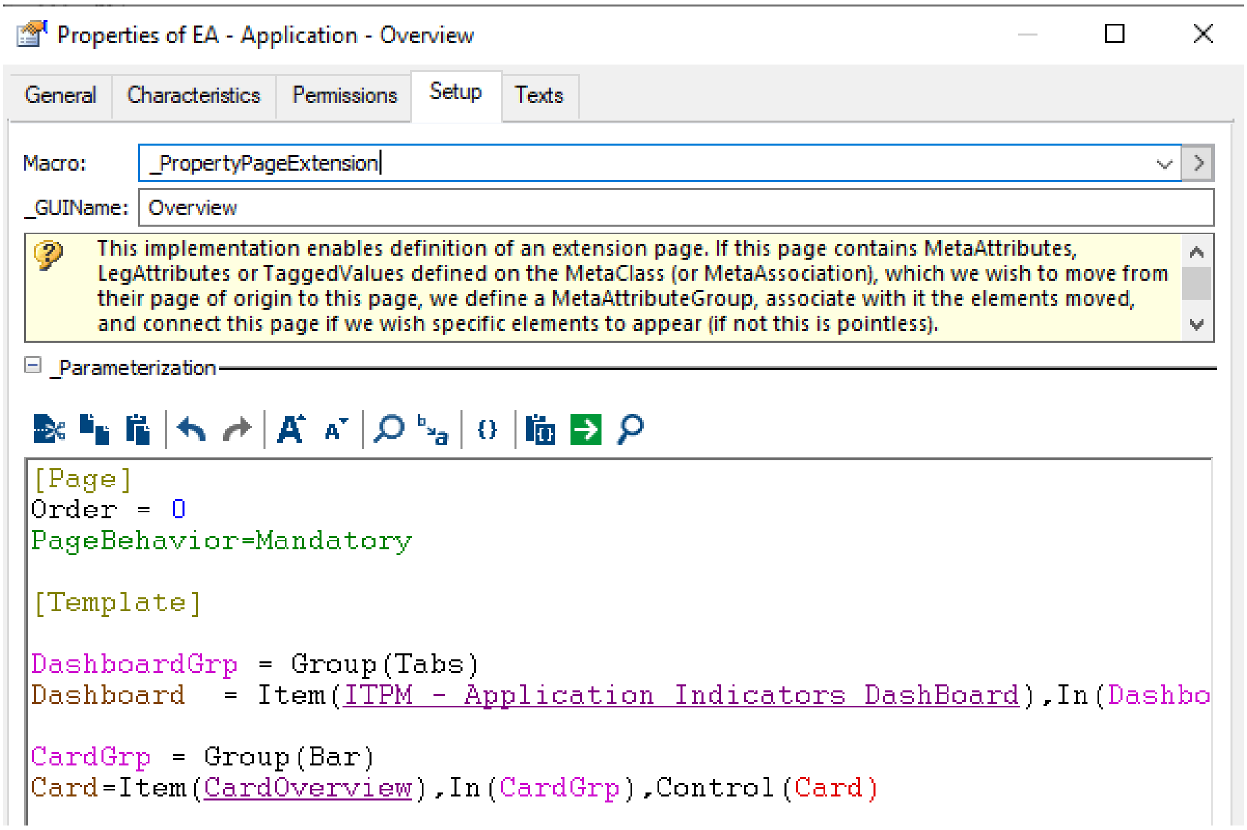
External Identifier:

⊟ PropertyPage Presentation

Default Page Display Behavior: Visible

- in **Setup**

  _Parameterization: Control(Card)



Properties of EA - Application - Overview

Macro: _PropertyPageExtension

_GUIName: Overview

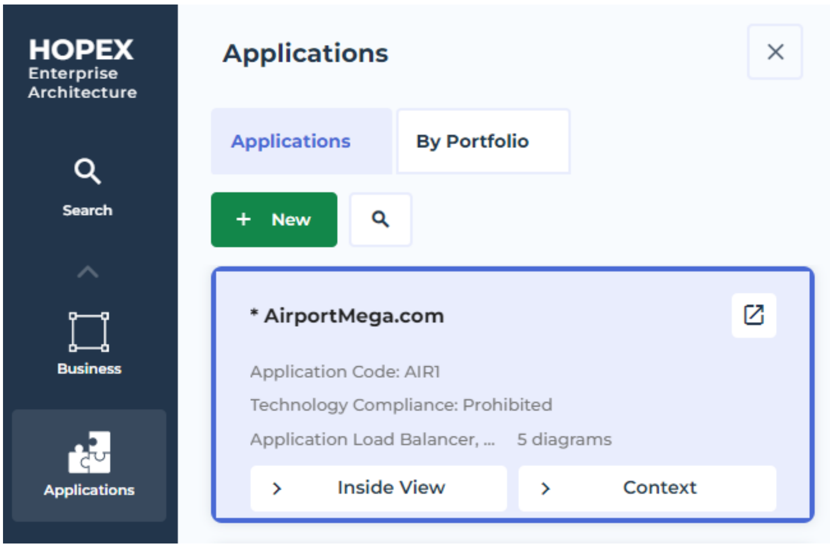
This implementation enables definition of an extension page. If this page contains MetaAttributes, LegAttributes or TaggedValues defined on the MetaClass (or MetaAssociation), which we wish to move from their page of origin to this page, we define a MetaAttributeGroup, associate with it the elements moved, and connect this page if we wish specific elements to appear (if not this is pointless).

_Parameterization

```
[Page]
Order = 0
PageBehavior=Mandatory

[Template]

DashboardGrp = Group(Tabs)
Dashboard   = Item(ITPM - Application Indicators DashBoard),In(Dashbo

CardGrp = Group(Bar)
Card=Item(CardOverview),In(CardGrp),Control(Card)
```

12.7.3 Adding lists/trees to cards (Browse area)

In the Browse area, for specific objects, you can configure a maximum of two buttons displayed when hovering the cursor over the card. These buttons give access to objects that can be drag & drop into the Edit area (e.g.: a diagram, or another object property pages).

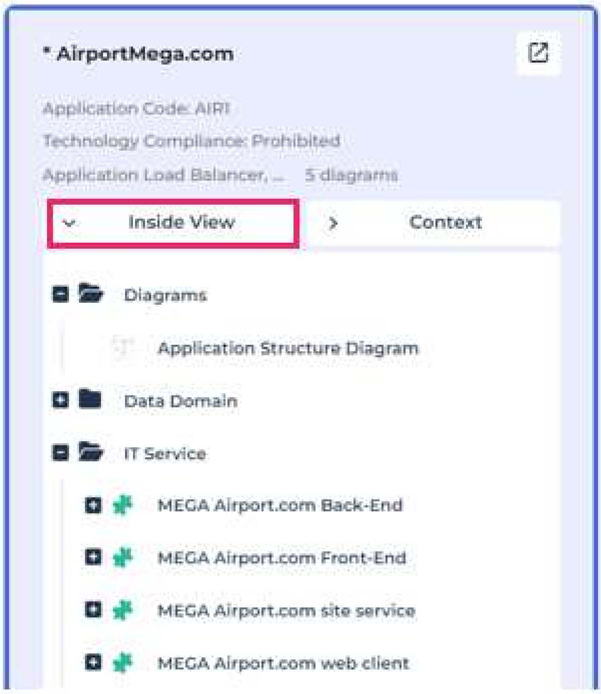
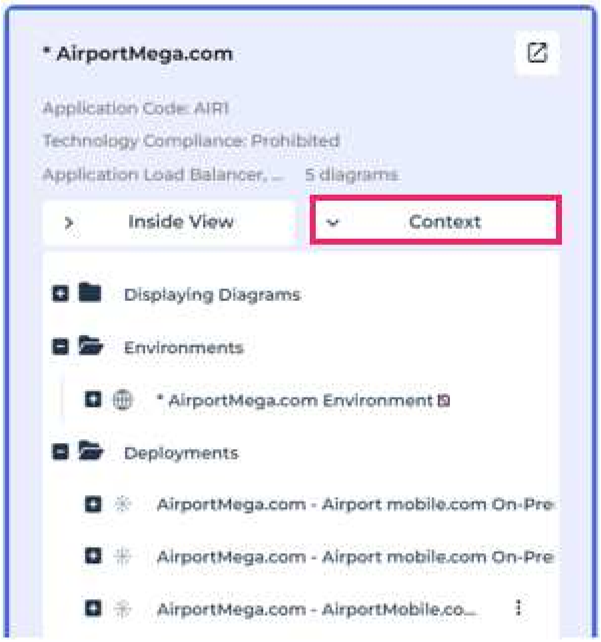


* AirportMega.com

Application Code: AIR1

Technology Compliance: Prohibited

Application Load Balancer, ...   5 diagrams

Inside view to access a functional synthesis of the object	**Context** to access the object context
 * AirportMega.com Application Code: AIRI Technology Compliance: Prohibited Application Load Balancer, ... 5 diagrams Inside View Context Diagrams Application Structure Diagram Data Domain IT Service MEGA Airport.com Back-End MEGA Airport.com Front-End MEGA Airport.com site service MEGA Airport.com web client	 * AirportMega.com Application Code: AIRI Technology Compliance: Prohibited Application Load Balancer, ... 5 diagrams Inside View Context Displaying Diagrams Environments * AirportMega.com Environment Deployments AirportMega.com - Airport mobile.com On-Pre AirportMega.com - Airport mobile.com On-Pre AirportMega.com - AirportMobile.co...

> Example: for the *AirportMega.com **Application**, the end-user can display its **Application Structure Diagram** in the Edit area, and then drag & drop **IT services** from the tree into the diagram.

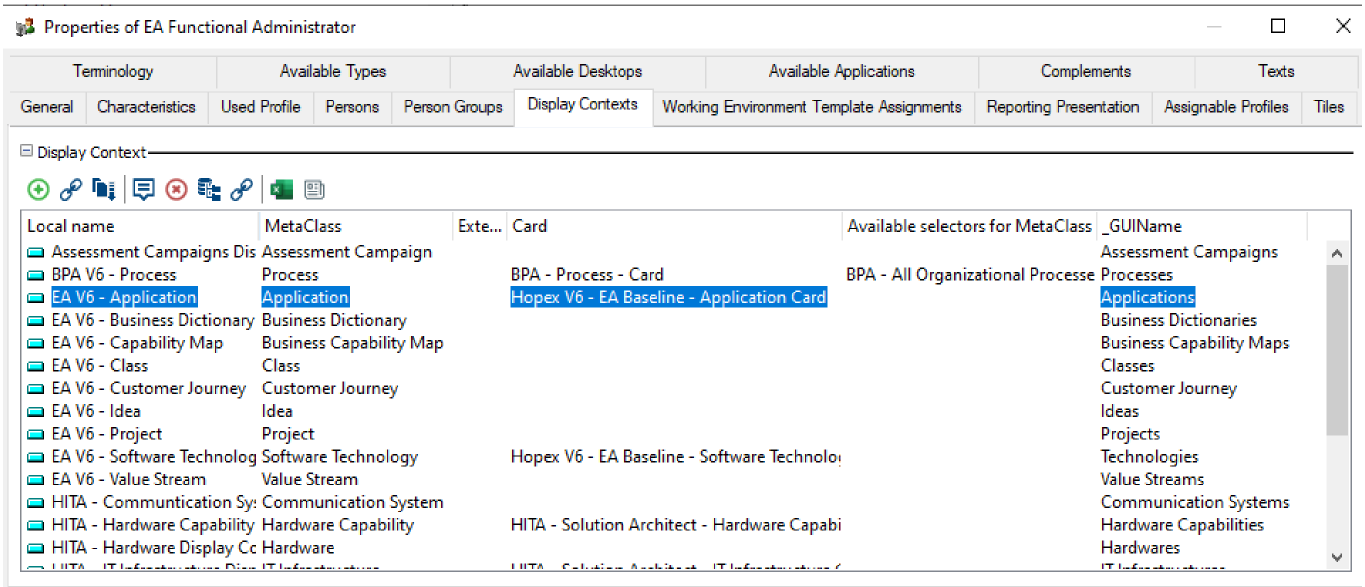
You can define:

- each button name (e.g.: **Inside View** and **Context**)
- the tree/list displayed:
  - use a Metatree to display objects in a tree format
  - use a query to display an object list (the query must be performant)

To configure the buttons on card (Browse area):

1. In the profile properties, display its **Display Contexts** tab.

   E.g.: the **Enterprise Architect** profile.



Properties of EA Functional Administrator

Display Context

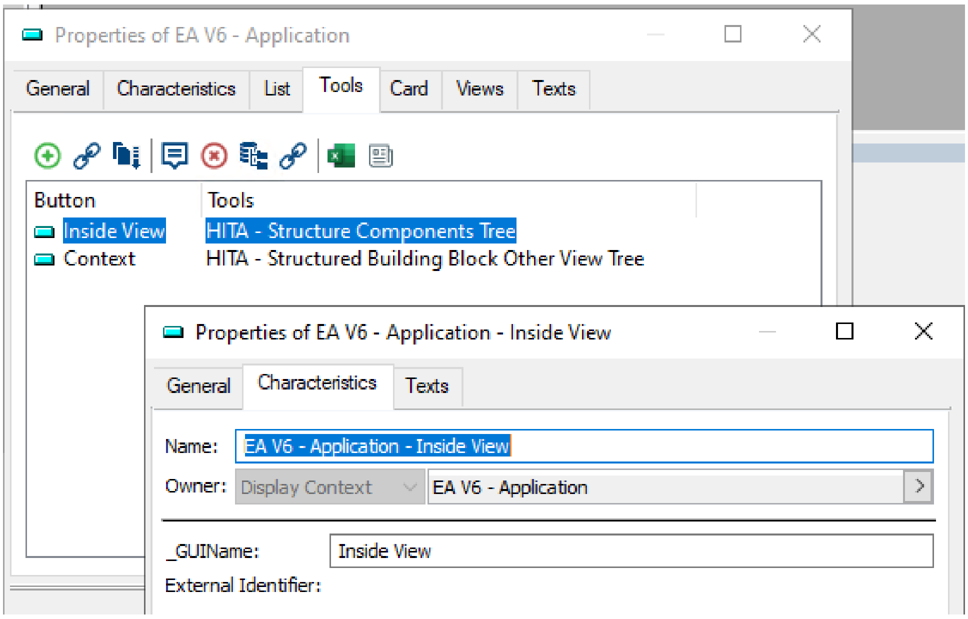
Local name	MetaClass	Exte...	Card	Available selectors for MetaClass	_GUIName
Assessment Campaigns Dis	Assessment Campaign				Assessment Campaigns
BPA V6 - Process	Process		BPA - Process - Card	BPA - All Organizational Processe	Processes
EA V6 - Application	Application		Hopex V6 - EA Baseline - Application Card		Applications
EA V6 - Business Dictionary	Business Dictionary				Business Dictionaries
EA V6 - Capability Map	Business Capability Map				Business Capability Maps
EA V6 - Class	Class				Classes
EA V6 - Customer Journey	Customer Journey				Customer Journey
EA V6 - Idea	Idea				Ideas
EA V6 - Project	Project				Projects
EA V6 - Software Technolog	Software Technology		Hopex V6 - EA Baseline - Software Technolo		Technologies
EA V6 - Value Stream	Value Stream				Value Streams
HITA - Communtication Sy:	Communication System				Communication Systems
HITA - Hardware Capability	Hardware Capability		HITA - Solution Architect - Hardware Capabi		Hardware Capabilities
HITA - Hardware Display Cc	Hardware				Hardwares

2. Access the Display Context properties of the MetaClass concerned.

   E.g.: "EA V6 – Application" Display Context of the **Application** MetaClass.

3. Display its **Tools** property.

4. Add/modify the buttons: **_GUIname** and MetaTree/query.



Properties of EA V6 - Application

Button	Tools
Inside View	HITA - Structure Components Tree
Context	HITA - Structured Building Block Other View Tree

Properties of EA V6 - Application - Inside View

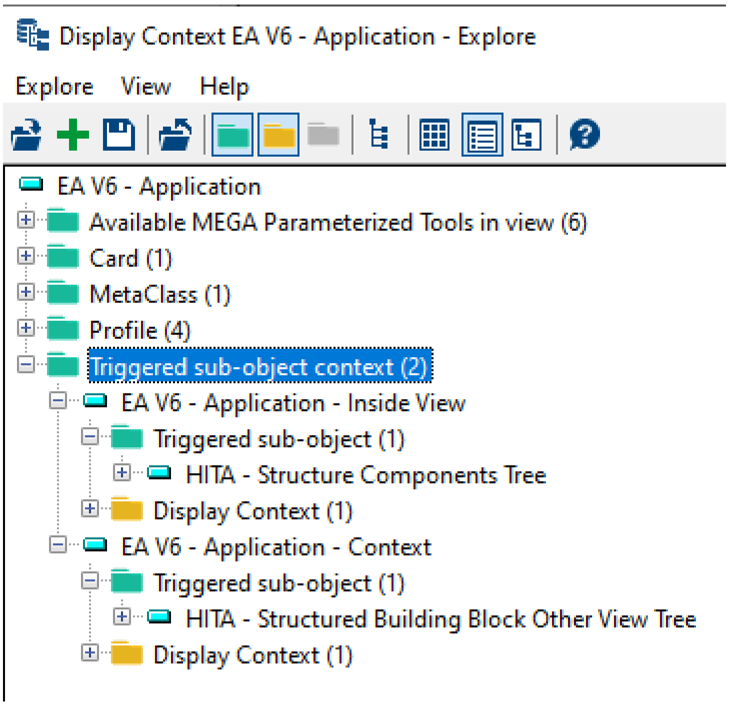
Name: EA V6 - Application - Inside View

Owner: Display Context EA V6 - Application

_GUIName: Inside View

External Identifier:

To modify or add a tree (using a MetaTree) or a list (using a query), you might need to perform the action by **Exploring** the corresponding Display Context.


Display Context EA V6 - Application - Explore

Explore View Help

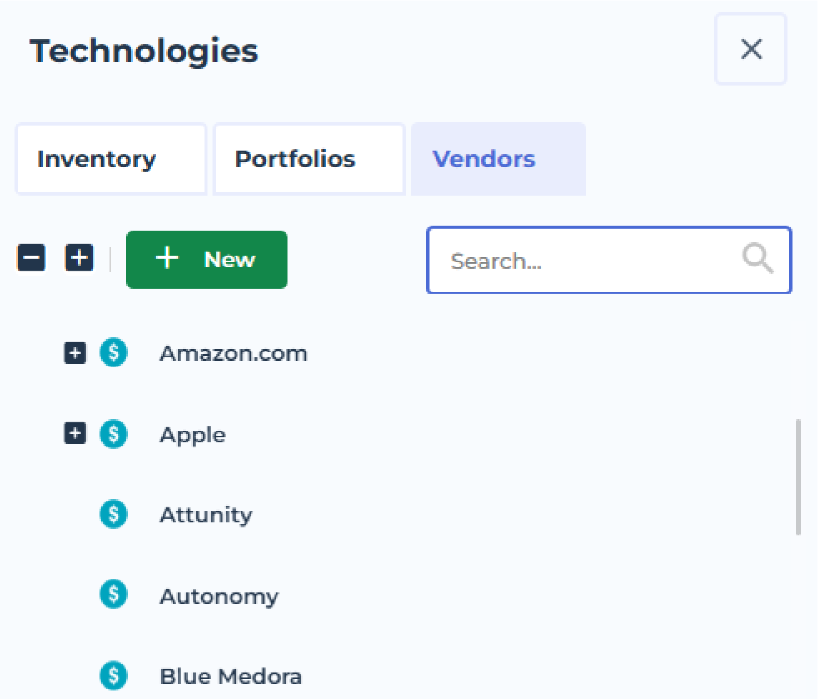
- EA V6 - Application
  - Available MEGA Parameterized Tools in view (6)
  - Card (1)
  - MetaClass (1)
  - Profile (4)
  - Triggered sub-object context (2)
    - EA V6 - Application - Inside View
      - Triggered sub-object (1)
        - HITA - Structure Components Tree
      - Display Context (1)
    - EA V6 - Application - Context
      - Triggered sub-object (1)
        - HITA - Structured Building Block Other View Tree
      - Display Context (1)

If you add more than two tools, only the first two ones are taken into account.

12.7.4 Adding the search tool to a tree (Browse area)

You can add the search tool in a tree of a small View (Browse area). This is performed on the corresponding **MetaTree**.

E.g.: you can add the search tool in the **Technologies** by **Vendors** tree.



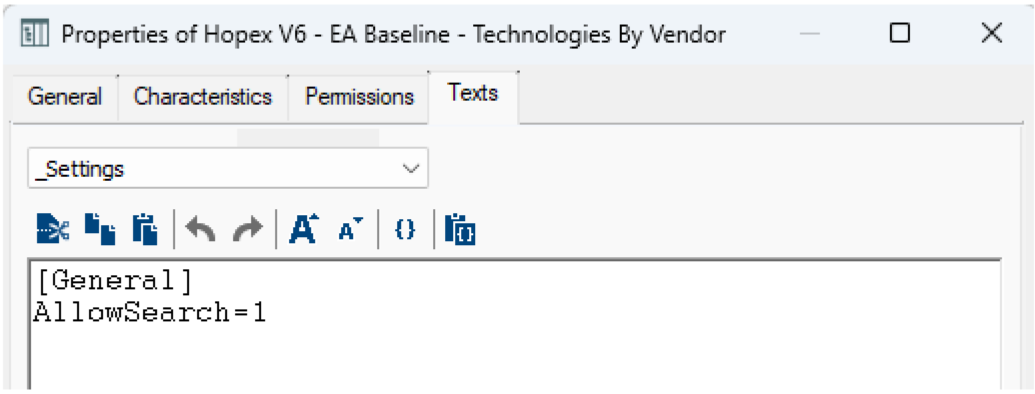
To add the search tool to a tree (Browse area):

1. Access the **MetaTree** properties corresponding to the tree.

   `E.g.: Hopex V6 – EA Baseline – Technologies By Vendor.`

2. In its **Texts > _Settings** page add:

```
[General]

AllowSearch=1
```

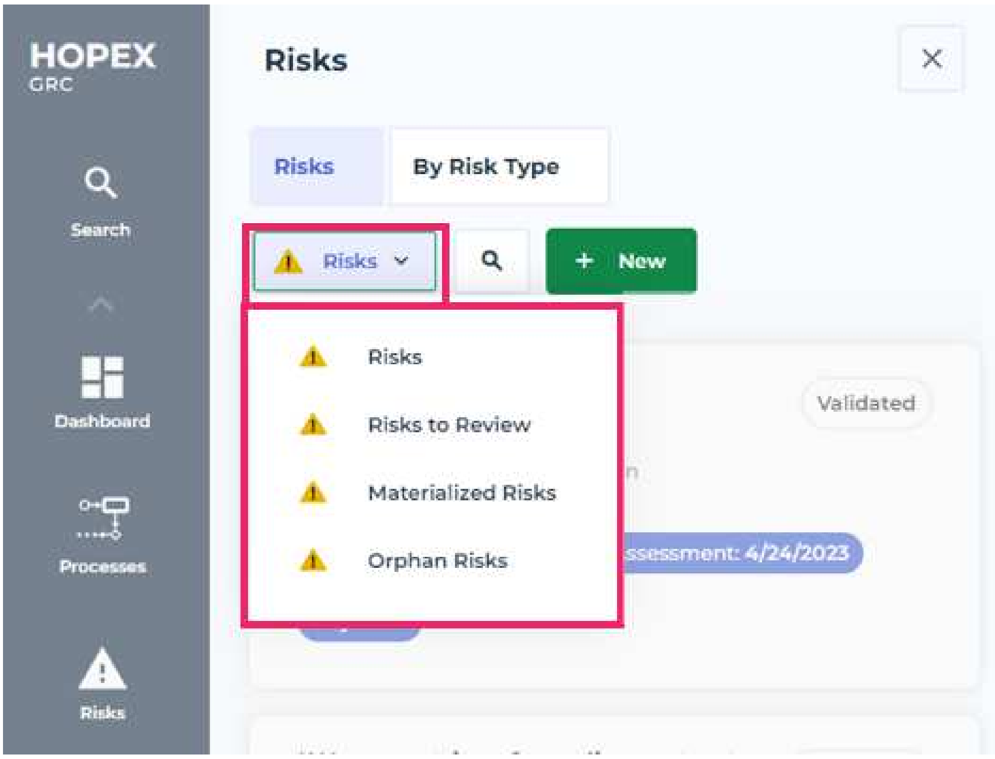

Properties of Hopex V6 - EA Baseline - Technologies By Vendor
General
Characteristics
Permissions
Texts
_Settings

```
[General]
AllowSearch=1
```

12.7.5 Adding list filters in the card list (Browse area)

In the card list (Browse area), you can add a button to filter the list. This is performed with a **Parameterized Alternate List**.

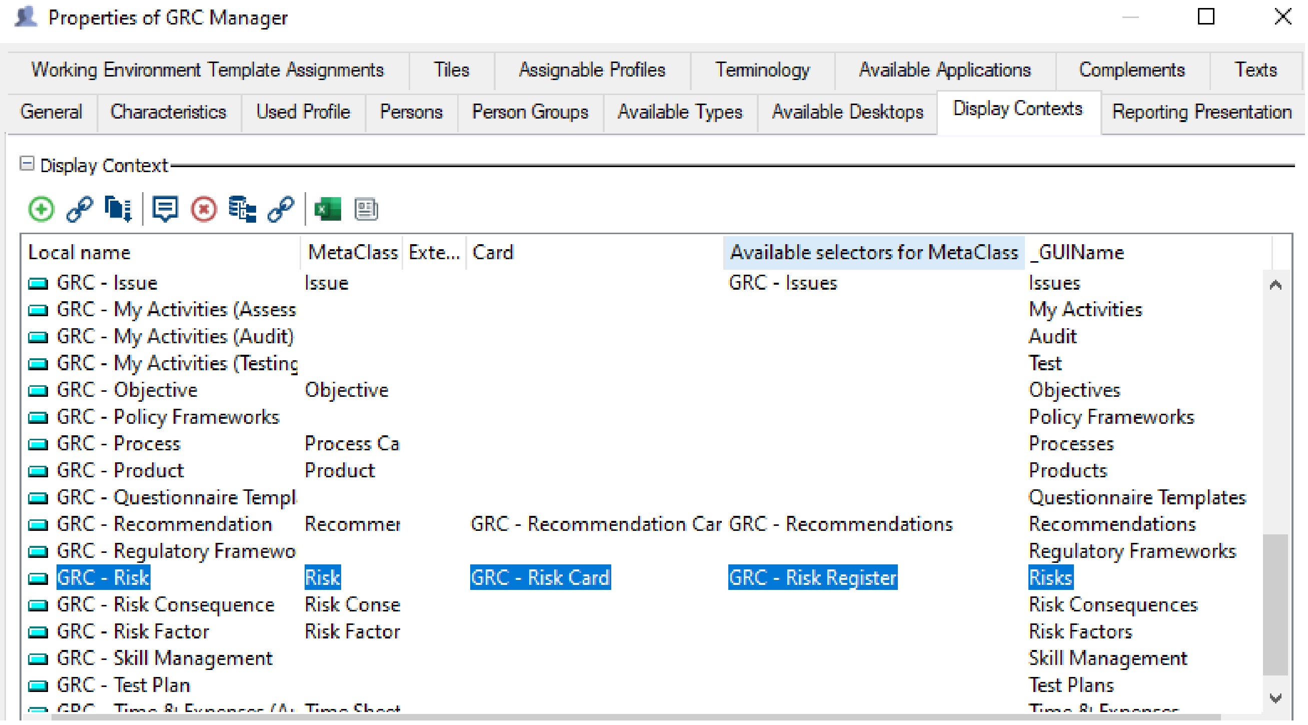
> E.g.: in GRC Solution, cards on **Risk** MetaClass can be filtered according to specific criteria: risks to review, materialized risks, orphan risks, or all of the risks.



To configure the buttons on card (Browse area):

1. In the profile properties, display its **Display Contexts** tab.

   `E.g.: the` **`GRC Manager`** `profile.`

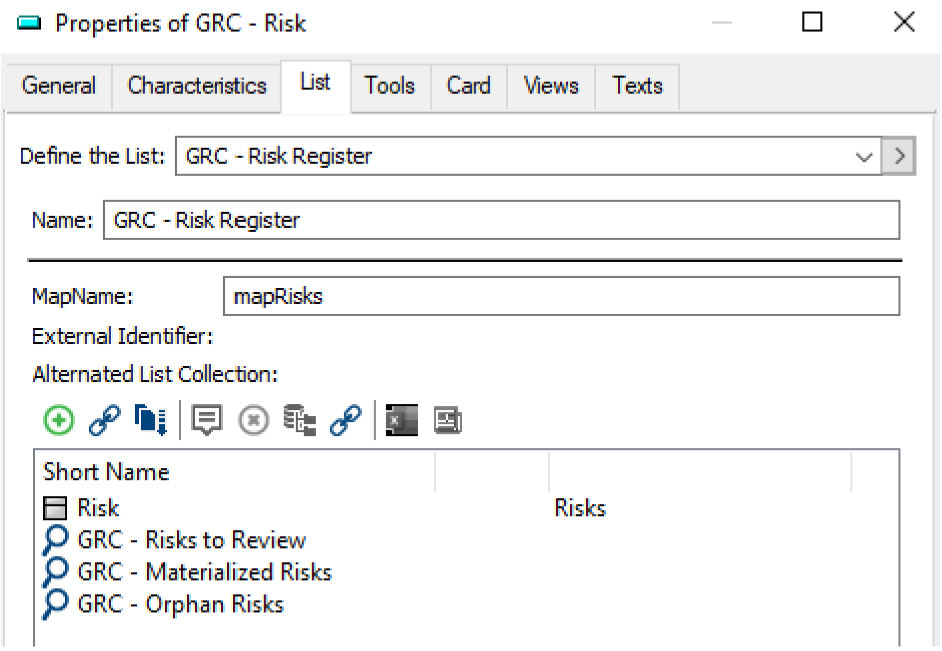


Local name	MetaClass	Exte...	Card	Available selectors for MetaClass	_GUIName
GRC - Issue	Issue			GRC - Issues	Issues
GRC - My Activities (Assess					My Activities
GRC - My Activities (Audit)					Audit
GRC - My Activities (Testing					Test
GRC - Objective	Objective				Objectives
GRC - Policy Frameworks					Policy Frameworks
GRC - Process	Process Ca				Processes
GRC - Product	Product				Products
GRC - Questionnaire Templ					Questionnaire Templates
GRC - Recommendation	Recommer		GRC - Recommendation Car	GRC - Recommendations	Recommendations
GRC - Regulatory Framewo					Regulatory Frameworks
GRC - Risk	Risk		GRC - Risk Card	GRC - Risk Register	Risks
GRC - Risk Consequence	Risk Conse				Risk Consequences
GRC - Risk Factor	Risk Factor				Risk Factors
GRC - Skill Management					Skill Management
GRC - Test Plan					Test Plans
GRC - Time & Expenses (A:	Time Sheet				Time & Expenses

2. Access the Display Context properties of the MetaClass concerned.

   `E.g.: "GRC – Risk" Display Context of the` **`Risk`** `MetaClass.`

3. Display its **List** property.



Properties of GRC - Risk

Define the List: GRC - Risk Register

Name: GRC - Risk Register

MapName: mapRisks

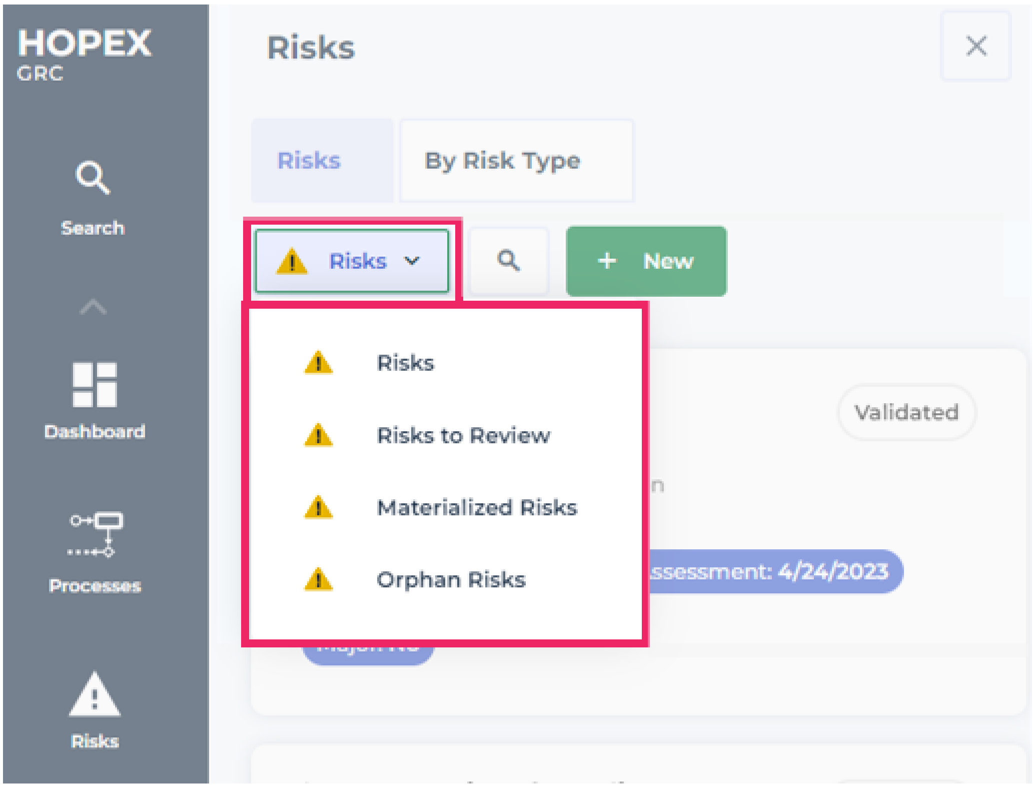
External Identifier:

Alternated List Collection:

Short Name		Risks
Risk		Risks
GRC - Risks to Review		
GRC - Materialized Risks		
GRC - Orphan Risks		

4. Connect your object collection.

```
E.g.: here the Risk MetaClass collection and three queries
retrieving the objects (Risks to Review, Materialized Risks, and
Orphan Risks).
```



12.8 Customizing the Property Pages for a WET

12.8.1 Modifying the default visibility of a property page

By default, a property page can be either:

- **Mandatory**: always displayed, the end-user cannot hide it.
- **Visible/Hidden**: by default, the page is visible/hidden but the end-user can hide/display it.

Note that both the **Overview** and **Diagrams** pages are specifically designed to be mandatory.

To modify the default visibility of a property page:

1. Access the WET properties.

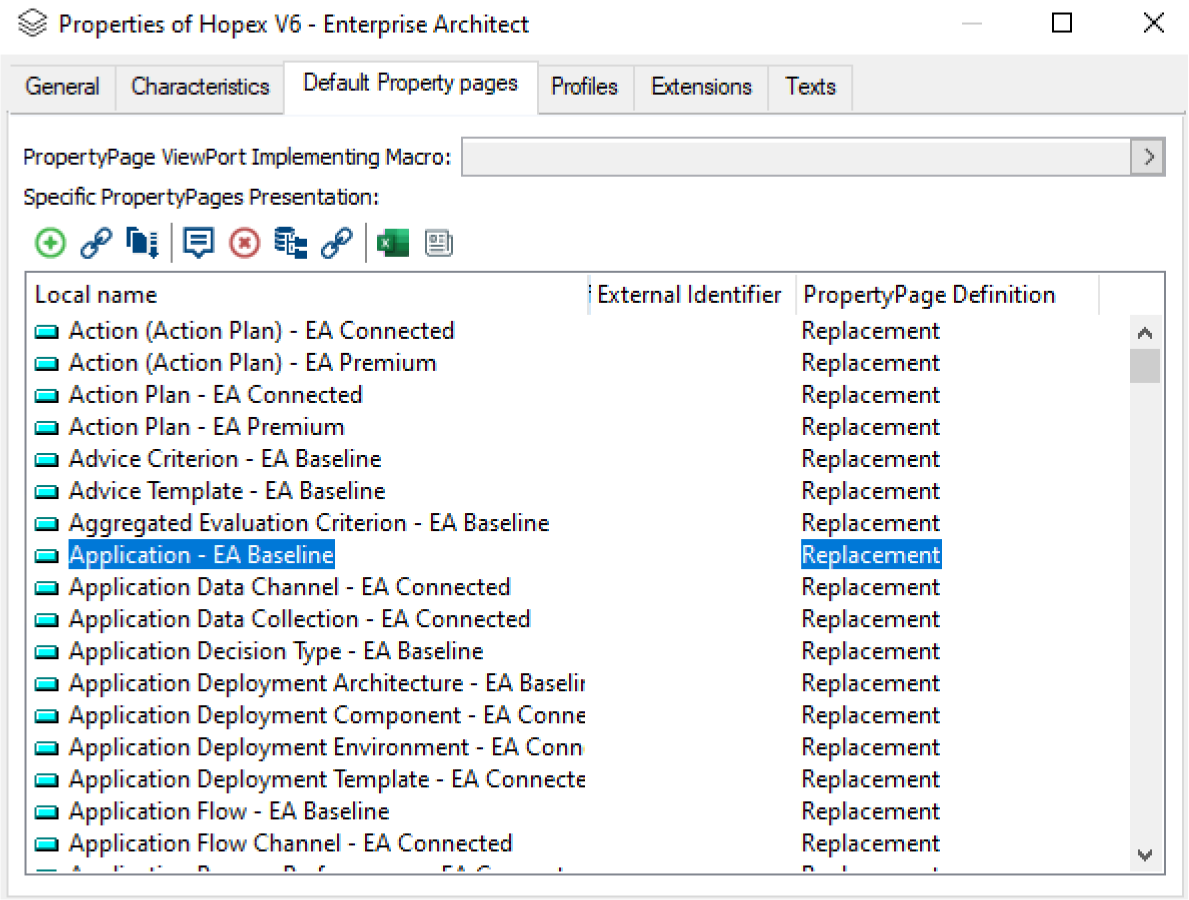
   E.g.: "HOPEX V6 – Enterprise Architect".

   ➢ See Accessing the WET metamodel element properties.

2. Display its **Default Property pages** tab.

   Each **PropertyPage Presentation** listed details the pages that replace (**PropertyPage Definition** = "Replacement") the standard set of pages linked to the corresponding MetaClass.

   E.g.: "Application – EA Baseline" for the **Application** MetaClass.



PropertyPage ViewPort Implementing Macro:

Specific PropertyPages Presentation:

Local name	External Identifier	PropertyPage Definition
Action (Action Plan) - EA Connected		Replacement
Action (Action Plan) - EA Premium		Replacement
Action Plan - EA Connected		Replacement
Action Plan - EA Premium		Replacement
Advice Criterion - EA Baseline		Replacement
Advice Template - EA Baseline		Replacement
Aggregated Evaluation Criterion - EA Baseline		Replacement
Application - EA Baseline		Replacement
Application Data Channel - EA Connected		Replacement
Application Data Collection - EA Connected		Replacement
Application Decision Type - EA Baseline		Replacement
Application Deployment Architecture - EA Baselir		Replacement
Application Deployment Component - EA Conne		Replacement
Application Deployment Environment - EA Conn		Replacement
Application Deployment Template - EA Connecte		Replacement
Application Flow - EA Baseline		Replacement
Application Flow Channel - EA Connected		Replacement

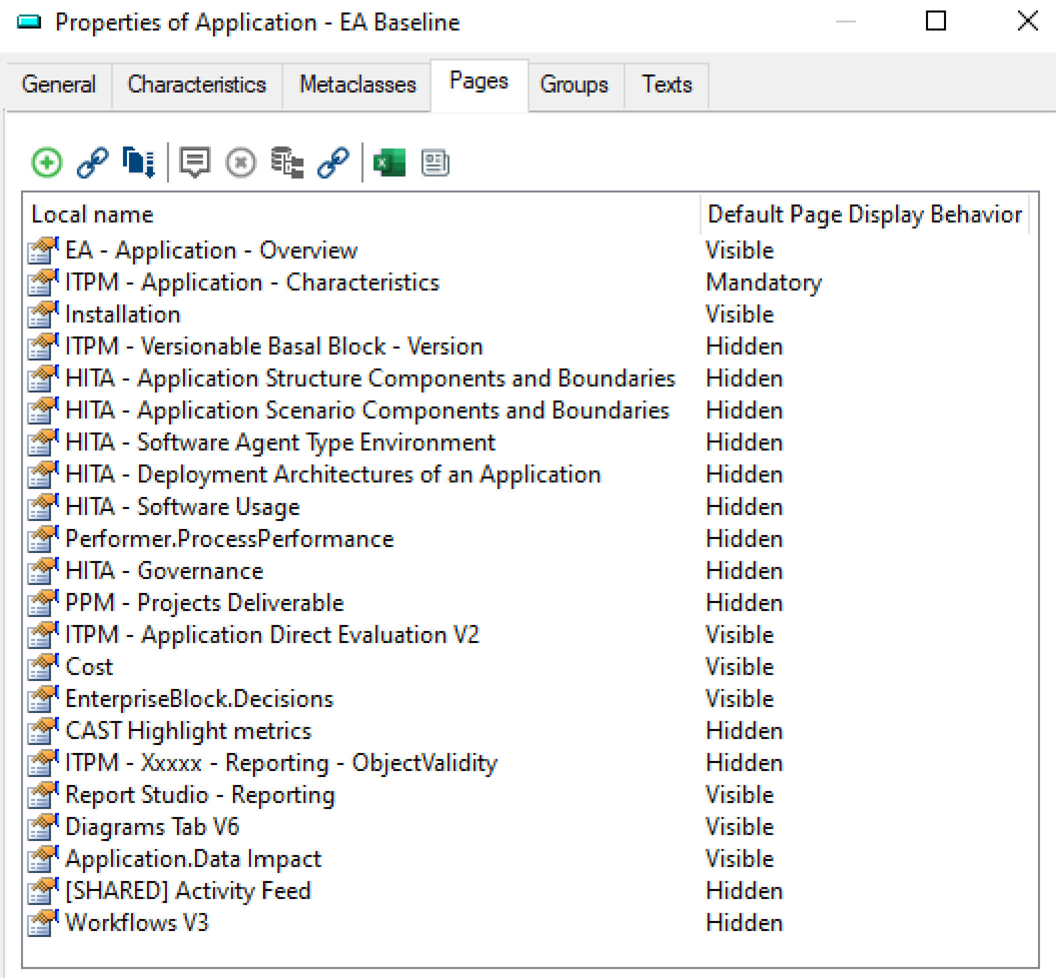
3. From the **Specific PropertyPages Presentation** list, select a **PropertyPage Presentation** and display its properties.

   E.g.: "Application – EA Baseline" for the **Application** MetaClass.

4. Its **Pages** tab details for each **MetaPropertyPage** its default display behavior.

   Default value: Visible.

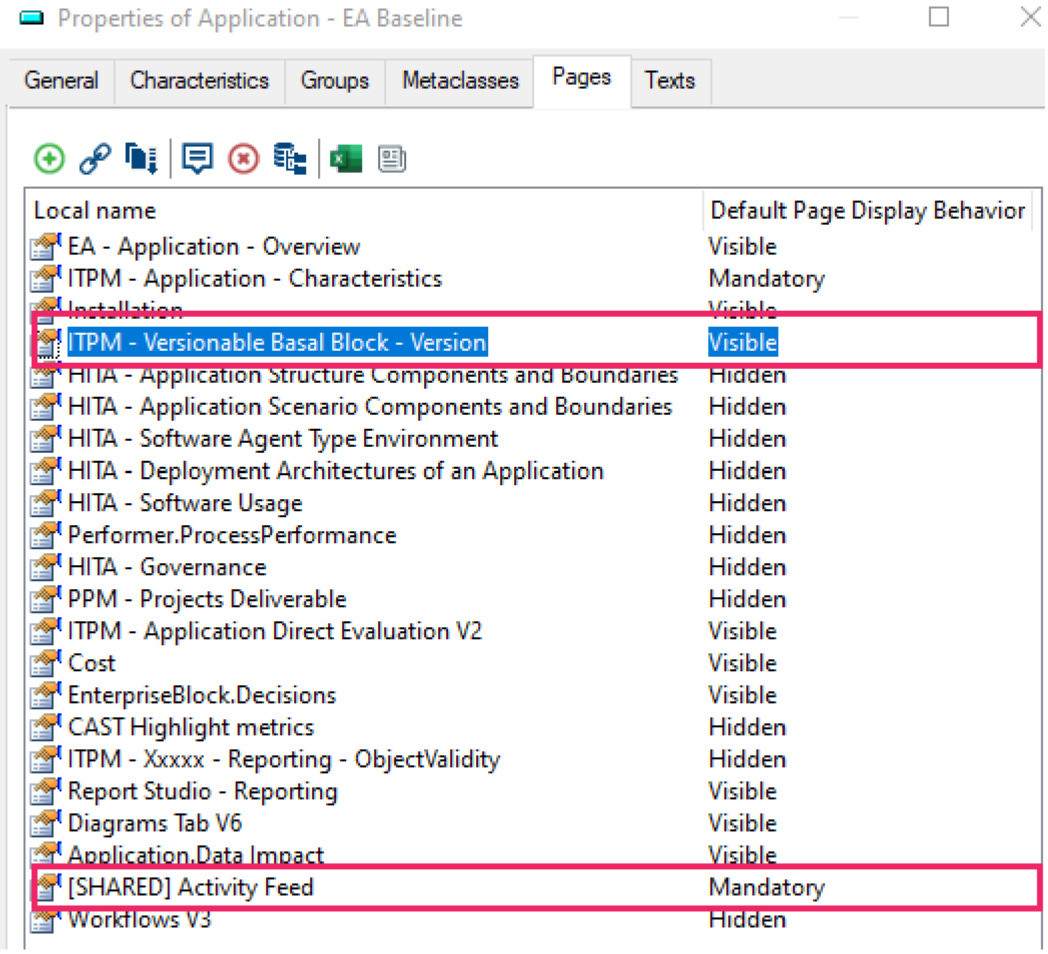
   E.g.: for the **Application** MetaClass, the **Characteristics** page is mandatory, **Installation** is displayed (but can be hidden), **Activity Feed** is hidden (but can be displayed by default).



Local name	Default Page Display Behavior
EA - Application - Overview	Visible
ITPM - Application - Characteristics	Mandatory
Installation	Visible
ITPM - Versionable Basal Block - Version	Hidden
HITA - Application Structure Components and Boundaries	Hidden
HITA - Application Scenario Components and Boundaries	Hidden
HITA - Software Agent Type Environment	Hidden
HITA - Deployment Architectures of an Application	Hidden
HITA - Software Usage	Hidden
Performer.ProcessPerformance	Hidden
HITA - Governance	Hidden
PPM - Projects Deliverable	Hidden
ITPM - Application Direct Evaluation V2	Visible
Cost	Visible
EnterpriseBlock.Decisions	Visible
CAST Highlight metrics	Hidden
ITPM - Xxxxx - Reporting - ObjectValidity	Hidden
Report Studio - Reporting	Visible
Diagrams Tab V6	Visible
Application.Data Impact	Visible
[SHARED] Activity Feed	Hidden
Workflows V3	Hidden

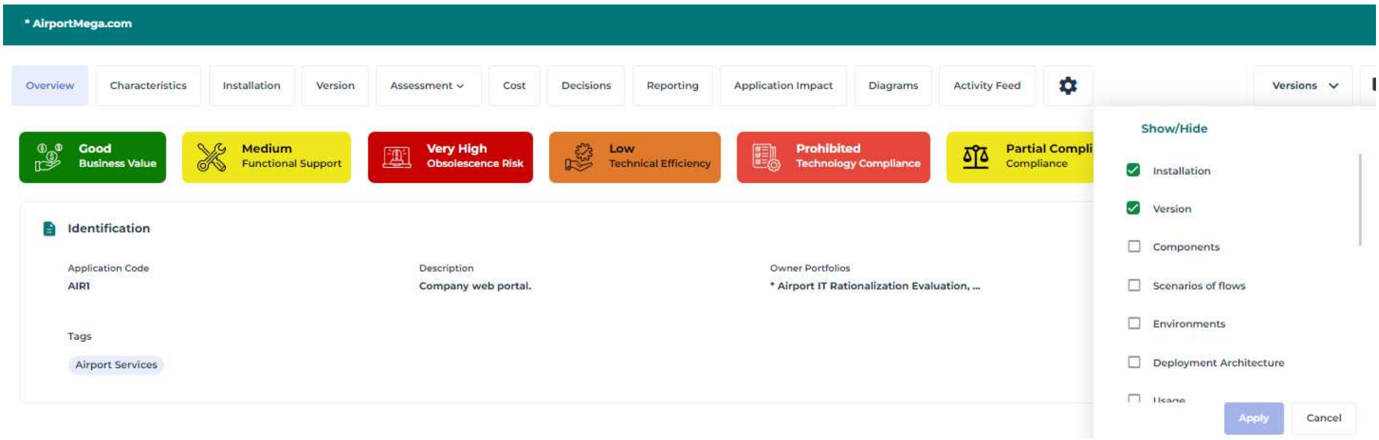
5. Click in each corresponding **Default Page Display Behavior** field and select the required default behavior.

   E.g.: Make the **Version** page visible by default, and the **Activity feed** mandatory.


Properties of Application - EA Baseline

Local name	Default Page Display Behavior
EA - Application - Overview	Visible
ITPM - Application - Characteristics	Mandatory
Installation	Visible
ITPM - Versionable Basal Block - Version	Visible
HITA - Application Structure Components and Boundaries	Hidden
HITA - Application Scenario Components and Boundaries	Hidden
HITA - Software Agent Type Environment	Hidden
HITA - Deployment Architectures of an Application	Hidden
HITA - Software Usage	Hidden
Performer.ProcessPerformance	Hidden
HITA - Governance	Hidden
PPM - Projects Deliverable	Hidden
ITPM - Application Direct Evaluation V2	Visible
Cost	Visible
EnterpriseBlock.Decisions	Visible
CAST Highlight metrics	Hidden
ITPM - Xxxxx - Reporting - ObjectValidity	Hidden
Report Studio - Reporting	Visible
Diagrams Tab V6	Visible
Application.Data Impact	Visible
[SHARED] Activity Feed	Mandatory
Workflows V3	Hidden

E.g.: **Installation** page is displayed by default and **Activity Feed** cannot be removed.



Identification

Application Code
AIR1

Description
Company web portal.

Owner Portfolios
* Airport IT Rationalization Evaluation, ...

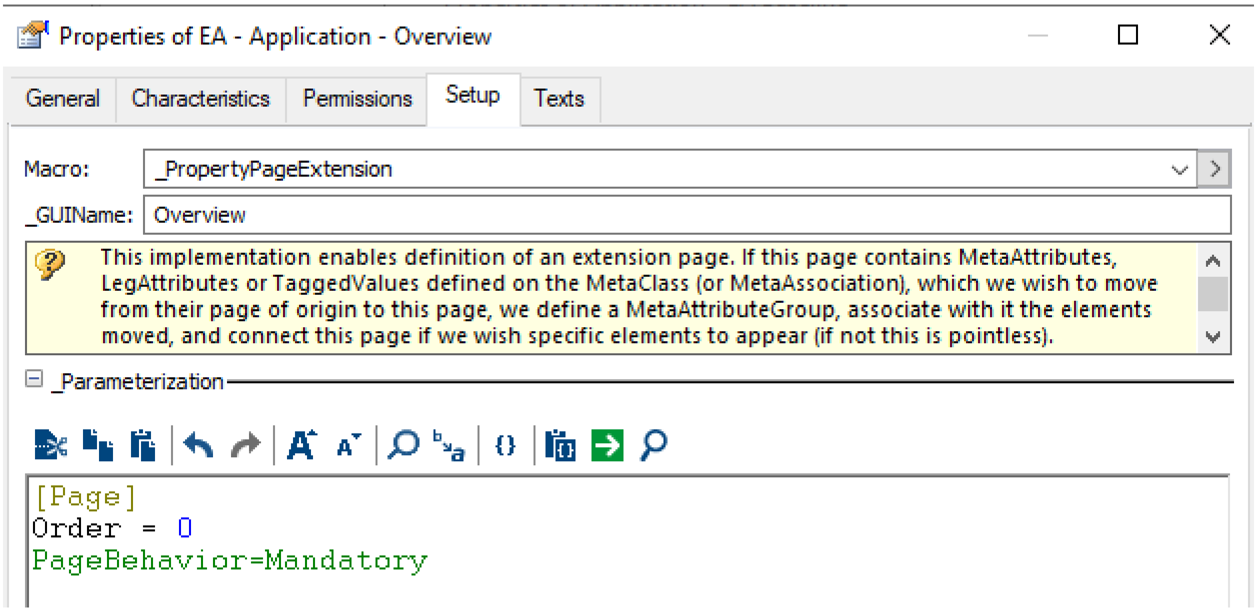
Tags
Airport Services

☑ Installation
☑ Version
☐ Components
☐ Scenarios of flows
☐ Environments
☐ Deployment Architecture
☐ Usage

Note

The **Overview** page as well as the **Diagrams** page are specifically designed to be mandatory. The **Overview** page is in first position, its **Setup** page shows Order = 0:

```
[Page]
Order = 0
PageBehavior=Mandatory
```



Macro: _PropertyPageExtension

_GUIName: Overview

This implementation enables definition of an extension page. If this page contains MetaAttributes, LegAttributes or TaggedValues defined on the MetaClass (or MetaAssociation), which we wish to move from their page of origin to this page, we define a MetaAttributeGroup, associate with it the elements moved, and connect this page if we wish specific elements to appear (if not this is pointless).

_Parameterization

```
[Page]
Order = 0
PageBehavior=Mandatory
```

12.8.2 Modifying the default display of a section

With a property page including sections, you can define whether the section is:

- expanded or collapsed at first opening.

  The end-user can expand/collapse each section, this action is kept at next opening.

  The expanded/collapsed behavior is managed through:

  ```
  Initially(IsClosed)
  ```

  E.g.: for an Application, in its **Characteristics** page, **Identification** section is expanded, **Service Level Agreement**, **Functional Scope**, **Responsibility**, and **Technology** sections are collapsed.

- hidden or displayed at first opening.

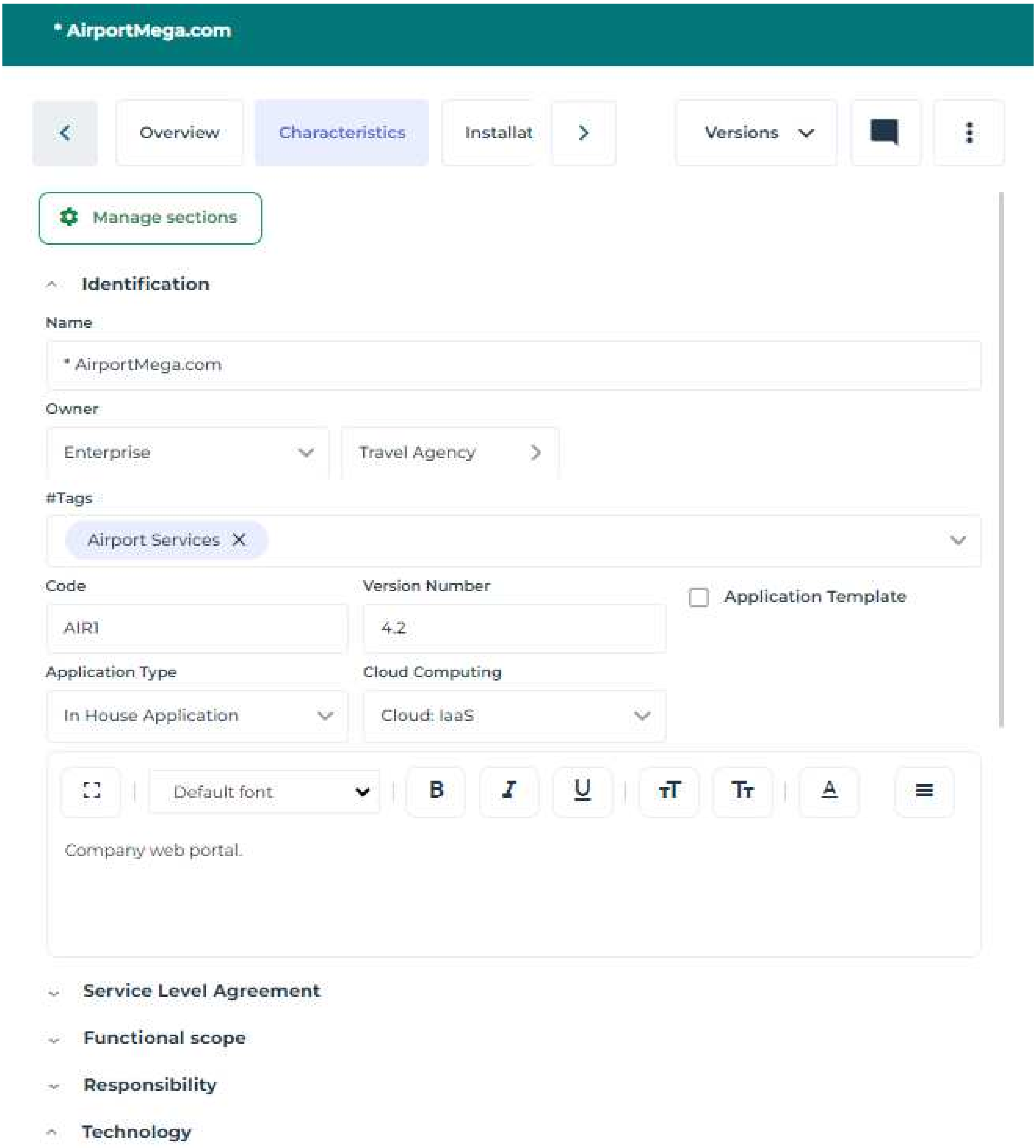
  The end-user can display/hide each section, this action is kept at next opening.

  The hidden/displayed behavior is managed through:

  ```
  Initially(Hidden)
  ```

If nothing is specified, the section is displayed and expanded.

For detailed information on section configuration, see the documentation regarding the **Groups** in **Forms**.



Identification

Name: * AirportMega.com

Owner: Enterprise | Travel Agency

#Tags: Airport Services

Code: AIR1

Version Number: 4.2

□ Application Template

Application Type: In House Application

Cloud Computing: Cloud: IaaS

Company web portal.

Service Level Agreement

Functional scope

Responsibility

Technology

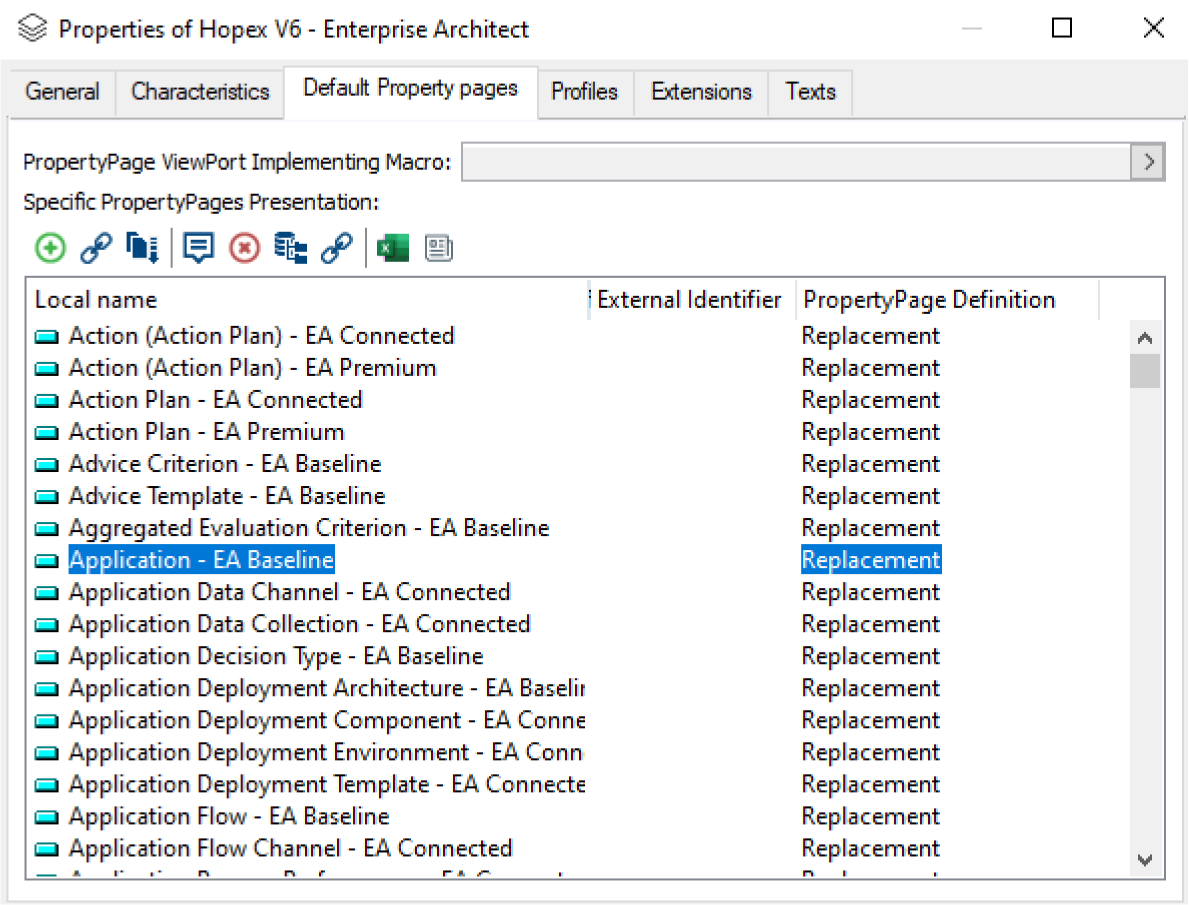
To modify the default display of a property page section:

1. Access the WET properties.

   `E.g.: "HOPEX V6 – Enterprise Architect".`

   ➢ See Accessing the WET metamodel element properties.

2. Display its **Default Property pages** tab.



PropertyPage ViewPort Implementing Macro:

Specific PropertyPages Presentation:

Local name	External Identifier	PropertyPage Definition
Action (Action Plan) - EA Connected		Replacement
Action (Action Plan) - EA Premium		Replacement
Action Plan - EA Connected		Replacement
Action Plan - EA Premium		Replacement
Advice Criterion - EA Baseline		Replacement
Advice Template - EA Baseline		Replacement
Aggregated Evaluation Criterion - EA Baseline		Replacement
Application - EA Baseline		Replacement
Application Data Channel - EA Connected		Replacement
Application Data Collection - EA Connected		Replacement
Application Decision Type - EA Baseline		Replacement
Application Deployment Architecture - EA Baselir		Replacement
Application Deployment Component - EA Conne		Replacement
Application Deployment Environment - EA Conn		Replacement
Application Deployment Template - EA Connecte		Replacement
Application Flow - EA Baseline		Replacement
Application Flow Channel - EA Connected		Replacement

3. From the **Specific PropertyPages Presentation** list, select a **PropertyPage Presentation** and display its properties.

   E.g.: "Application - EA Baseline" for the **Application** MetaClass.

4. In its **Pages** tab select the **MetaPropertyPage**.

   E.g.: "ITPM - Application - Characteristics" for the **Application Characteristics** page.

5. Access the **MetaPropertyPage** Properties **> Setup** tab.

6. By default, the section is displayed and expanded.

   If you want the section to be:

   - collapsed at page opening, add Initially(IsClosed) to the description of the section
   - hidden at page opening, add Initially(Hidden) to the description of the section

   For example, for an Application, in its **Characteristics** page, **to** hide **Service Level Agreement** section at opening.

12.8.3 WET advanced customization at property page level

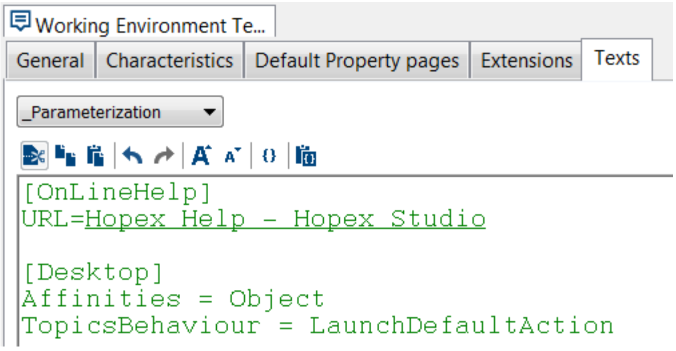
You can add customization on Property Pages at Topic level or at WET level, in the **Texts > _Parameterization**.

> For example, Application Property Pages display a tab that in a specific context you do not want to be displayed.

The _Parameterization is first read on a Topic, and if not defined, the _Parameterization is read on the WET, and if not found, there is no specific behavior.

To customize a WET:

1. Access the WET properties.
   - ➢ See Accessing the WET metamodel element properties.
2. Select the **Texts** tab.
3. In the **_Parameterization** section, enter your code.



```
[OnLineHelp]
URL=Hopex Help – Hopex Studio

[Desktop]
Affinities = Object
TopicsBehaviour = LaunchDefaultAction
```

12.8.4 Hiding the Overview page for a specific WET MetaClass

Standard pages are defined by a Type. For example:

- **Overview** page is of type **Overview**
- **Characteristics** page is of type **Characteristics**
- **Reporting** page is of type **Reporting**

Each object inherits automatically from these standard pages.

To overload such a page, you need to create a MetaPropertyPage and define it of the same type. For example:

- **EA – Application - Overview** MetaPropertyPage of type **Overview** overloads the standard **Overview** page
- **ITPM – Application - Characteristics** MetaPropertyPage of type **Characteristics** overloads the standard **Characteristics** page

For a specific object type, the **Overview** page may not be of interest. In its **PropertyPage Presentation**, you can define the **Overview** page as hidden using its link to **_Type**.

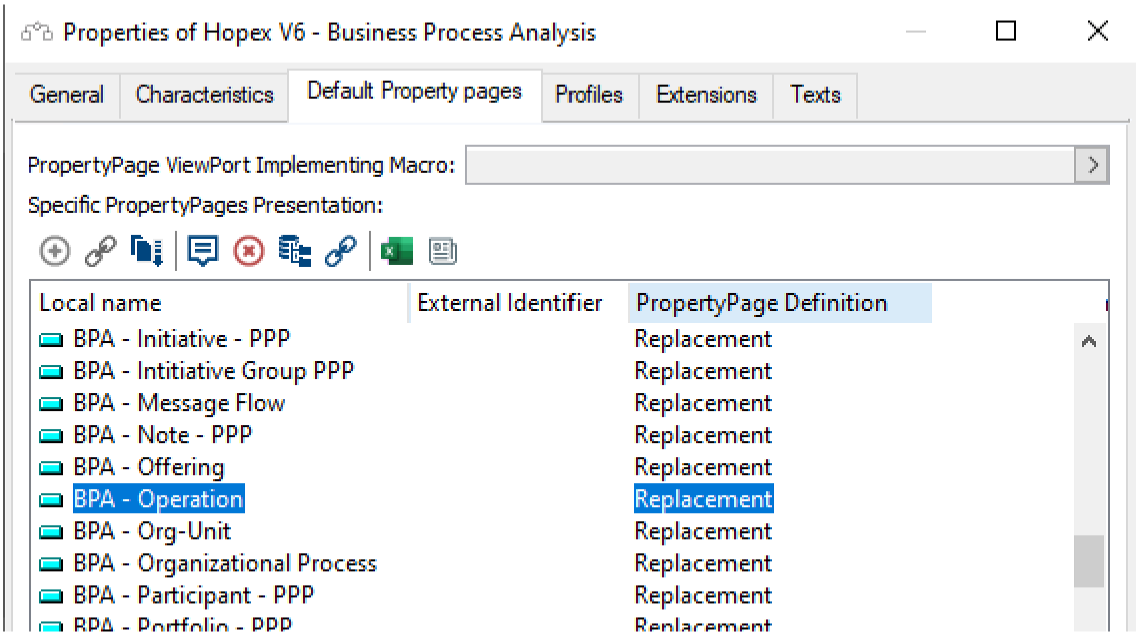
To hide the overview page for a MetaClass of a specific WET:

1. Access the WET properties.

   ➢ See Accessing the WET metamodel element properties.

   E.g.: HOPEX V6 – Business Process Analysis to hide the **Overview** page on **Operation** MetaClass.

2. Display its **Default Property Pages** tab.



Properties of Hopex V6 - Business Process Analysis

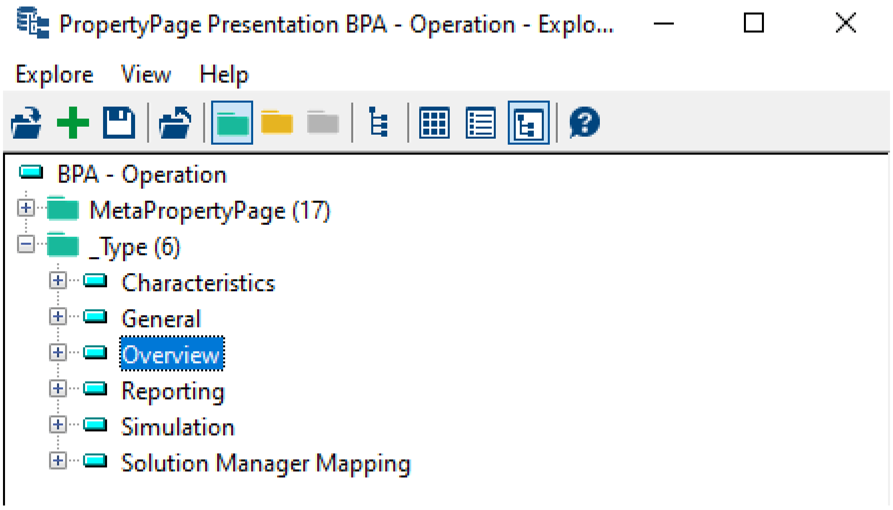
PropertyPage ViewPort Implementing Macro:

Specific PropertyPages Presentation:

Local name	External Identifier	PropertyPage Definition
BPA - Initiative - PPP		Replacement
BPA - Intitiative Group PPP		Replacement
BPA - Message Flow		Replacement
BPA - Note - PPP		Replacement
BPA - Offering		Replacement
BPA - Operation		Replacement
BPA - Org-Unit		Replacement
BPA - Organizational Process		Replacement
BPA - Participant - PPP		Replacement
BPA - Portfolio - PPP		Replacement

3. In the **Specific PropertyPages Presentation** list, explore the **PropertyPage Presentation** concerned.

   E.g.: "BPA – Operation" PropertyPage Presentation.

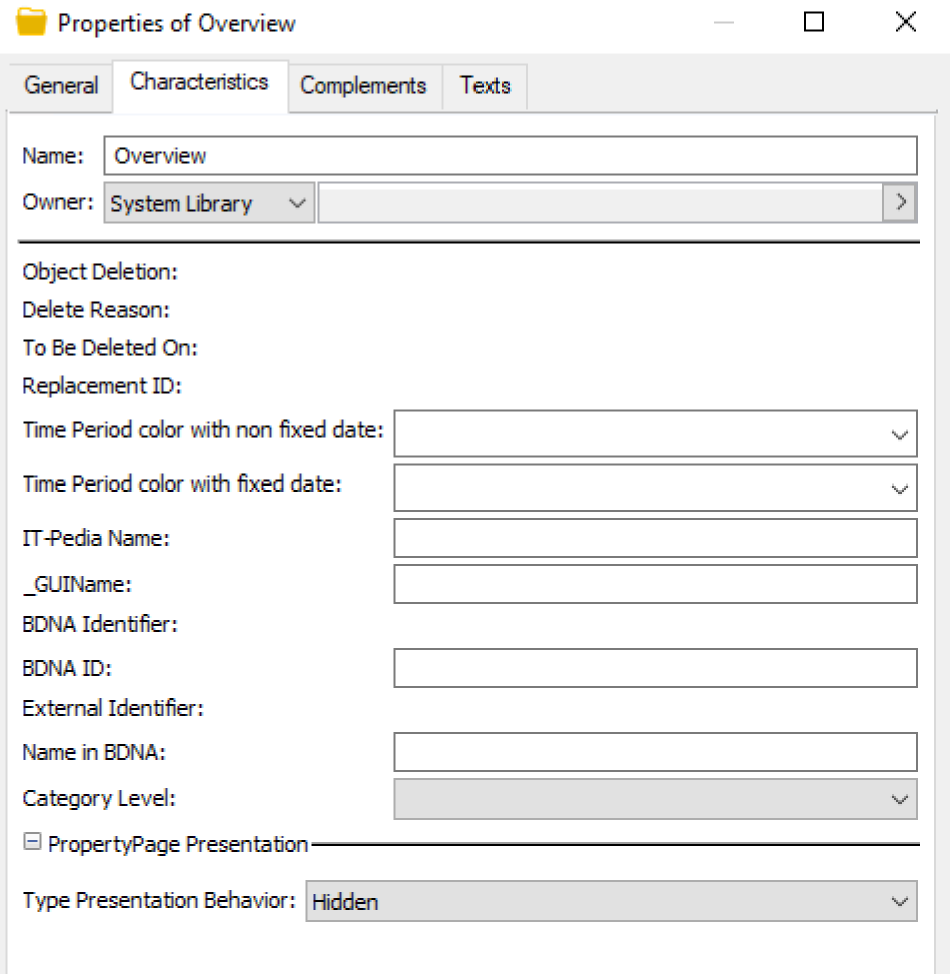


PropertyPage Presentation BPA - Operation - Explo...

- BPA - Operation
  - MetaPropertyPage (17)
  - _Type (6)
    - Characteristics
    - General
    - Overview
    - Reporting
    - Simulation
    - Solution Manager Mapping

4. Access the **Characteristics** properties of the _Type concerned.

   `E.g.: "Overview" _Type.`



Name: Overview

Owner: System Library

Object Deletion:

Delete Reason:

To Be Deleted On:

Replacement ID:

Time Period color with non fixed date:

Time Period color with fixed date:

IT-Pedia Name:

_GUIName:

BDNA Identifier:

BDNA ID:

External Identifier:

Name in BDNA:

Category Level:

⊟ PropertyPage Presentation

Type Presentation Behavior: Hidden

5. In the **PropertyPage Presentation** section, modify the **Type Presentation Behavior** to "Hidden".

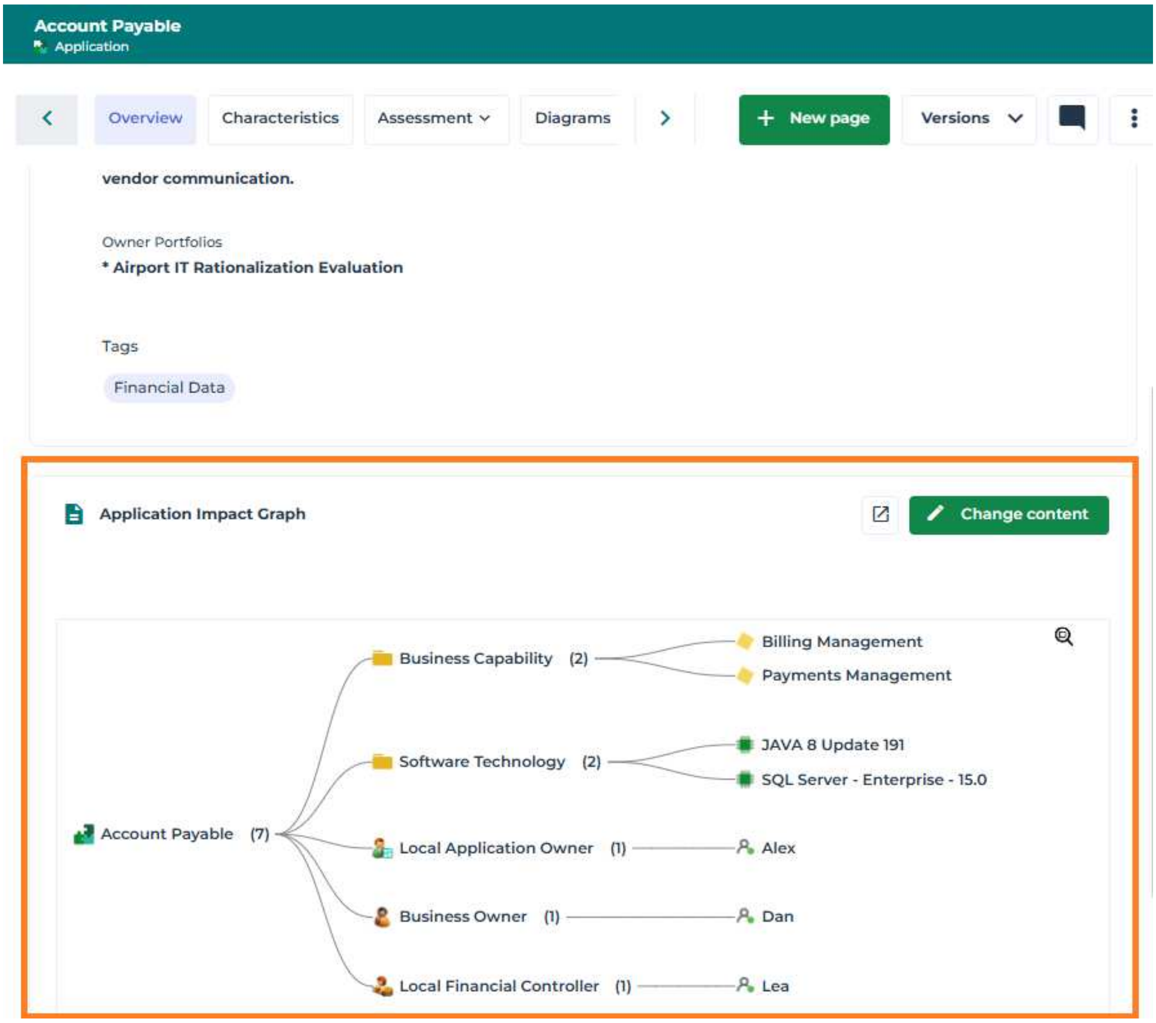
   Default value: "Standard" (i.e. the page is available)

12.8.5 Defining a default report/diagram for an object Overview page

An object **Overview** page includes a dedicated pane to display a report or a diagram. You can define a default Report Template or diagram type for a specific MetaClass for a Profile, through its display Context.

You can also remove the default selection.

The Functional Administrator of the Solution can change or reset the default selection.


Account Payable
Application

vendor communication.

Owner Portfolios
* Airport IT Rationalization Evaluation

Tags
Financial Data

Application Impact Graph

To define the default report or diagram:

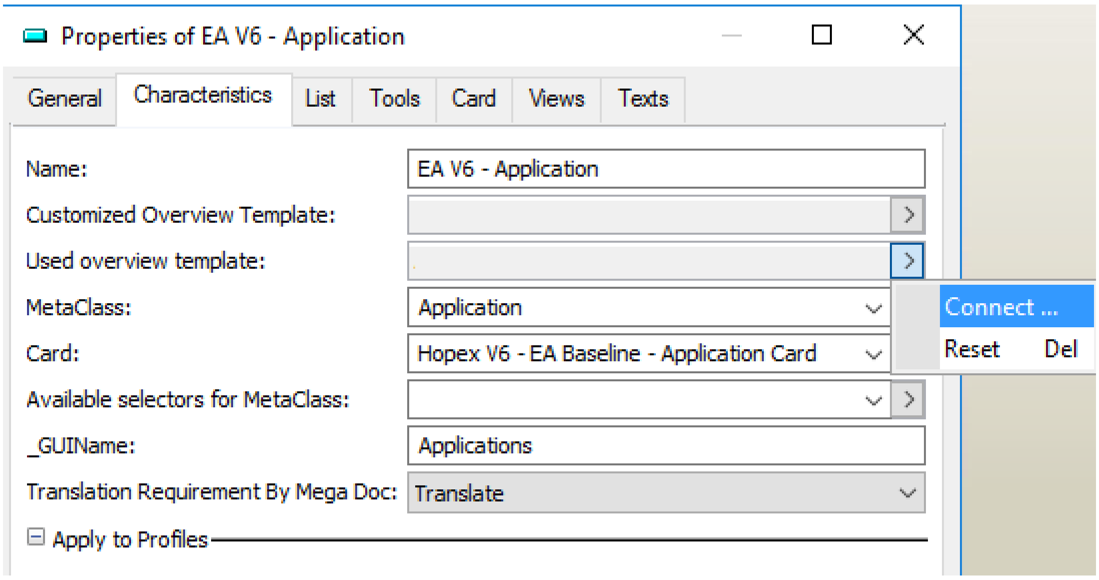
1. In the profile properties, display its **Display Context** tab.

   E.g.: Enterprise Architect profile

2. Access the Display Context properties of the MetaClass concerned.

   E.g.: "EA V6 – Application" Display Context of the **Application** MetaClass.

3. In **Characteristics**, **Used Overview Template** field, click the arrow and select **Connect**.


Properties of EA V6 - Application

Name:	EA V6 - Application
Customized Overview Template:	
Used overview template:	
MetaClass:	Application
Card:	Hopex V6 - EA Baseline - Application Card
Available selectors for MetaClass:	
_GUIName:	Applications
Translation Requirement By Mega Doc:	Translate

4. In the first search field, select diagram type or report Template.

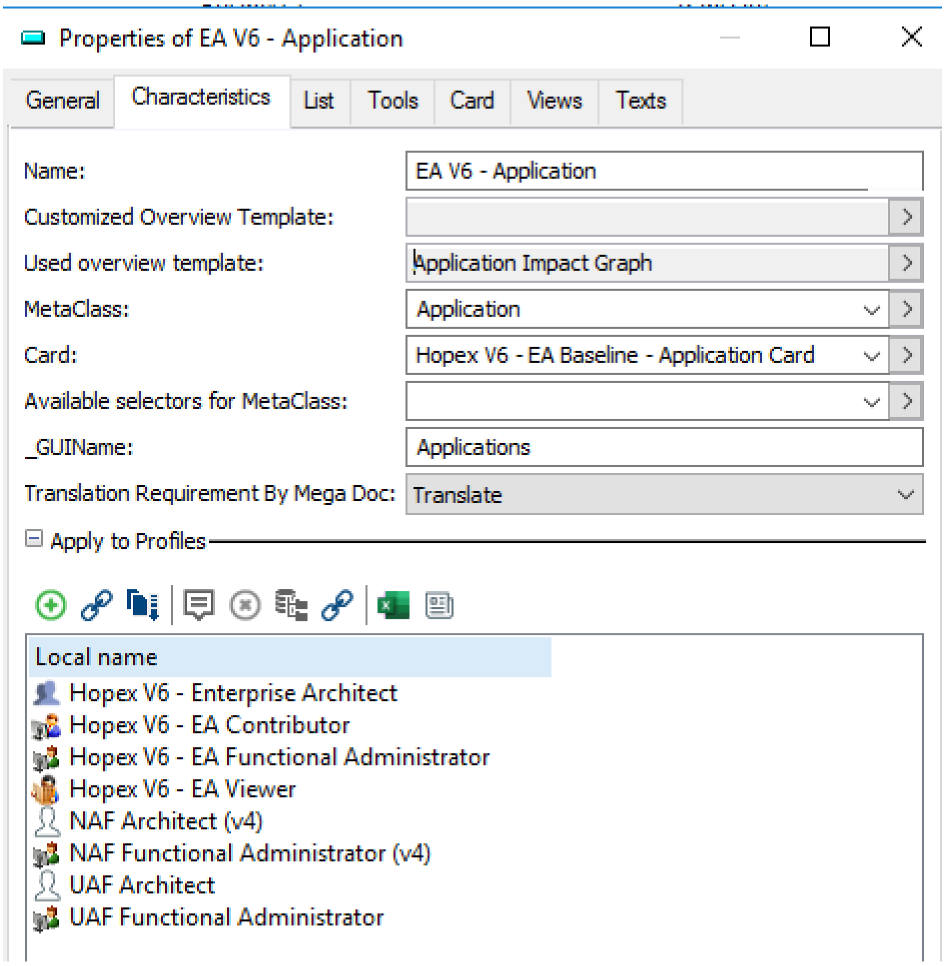
   `E.g.: ''Report templates available for current display context''`

5. Select the diagram type or Report Template and click **Connect**.

   `E.g.: « Application Impact Graph » report Template.`

   - Note: to remove the default selection, click **Reset**.

For each object its corresponding diagram or report is displayed.



Properties of EA V6 - Application

Name: EA V6 - Application

Customized Overview Template:

Used overview template: Application Impact Graph

MetaClass: Application

Card: Hopex V6 - EA Baseline - Application Card

Available selectors for MetaClass:

_GUIName: Applications

Translation Requirement By Mega Doc: Translate

⊟ Apply to Profiles

Local name
Hopex V6 - Enterprise Architect
Hopex V6 - EA Contributor
Hopex V6 - EA Functional Administrator
Hopex V6 - EA Viewer
NAF Architect (v4)
NAF Functional Administrator (v4)
UAF Architect
UAF Functional Administrator

12.8.6 Customizing the Overview page for a specific WET MetaClass

Standard pages are defined by a Type. For example:

- **Overview** page is of type **Overview**
- **Characteristics** page is of type **Characteristics**
- **Reporting** page is of type **Reporting**

Each object inherits automatically from these standard pages.

To overload such a page, you need to create a MetaPropertyPage and define it of the same type. For example:

- **EA – Application - Overview** MetaPropertyPage of type **Overview** overloads the standard **Overview** page
- **ITPM – Application - Characteristics** MetaPropertyPage of type **Characteristics** overloads the standard **Characteristics** page

For a specific object type, you might want to add information to the standard **Overview** page. This is performed in the MetaPropertyPage (built with the **_PropertyPageExtension** macro) in its **Setup** tab.

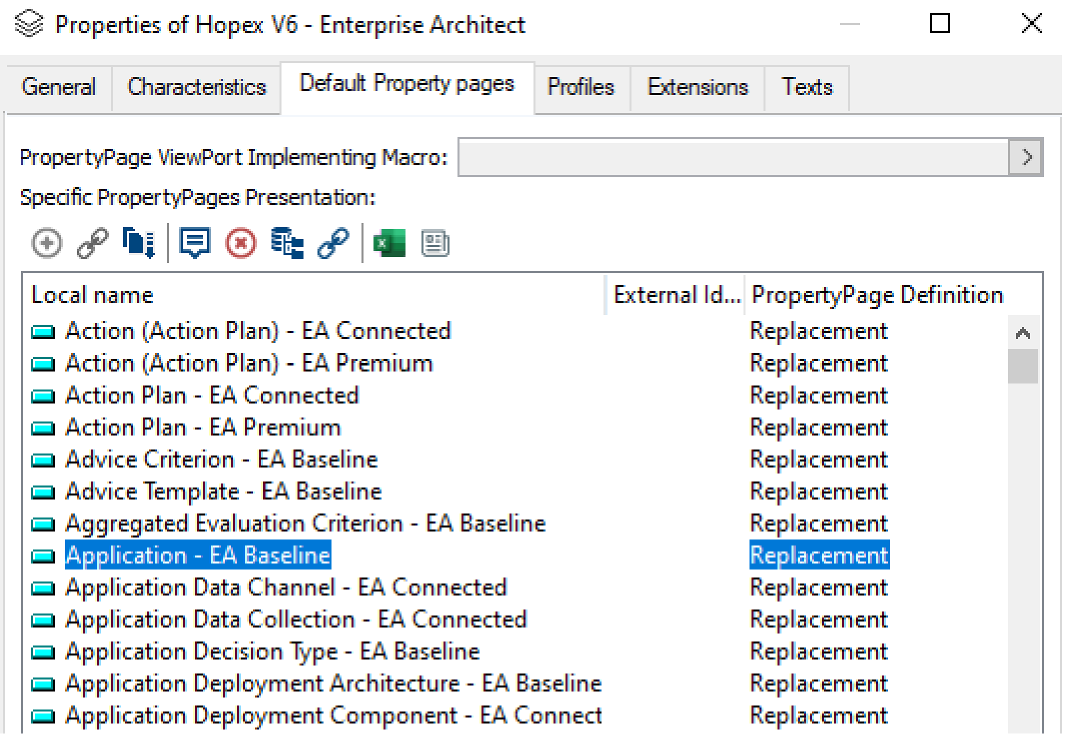
To customize the Overview page for a specific MetaClass:

1. Access the WET properties.

   ➢ See Accessing the WET metamodel element properties.

   `E.g.: HOPEX V6 – Enterprise Architext to customize the **Overview** page on **Application** MetaClass.`

2. Display its **Default Property pages** tab.



PropertyPage ViewPort Implementing Macro:

Specific PropertyPages Presentation:

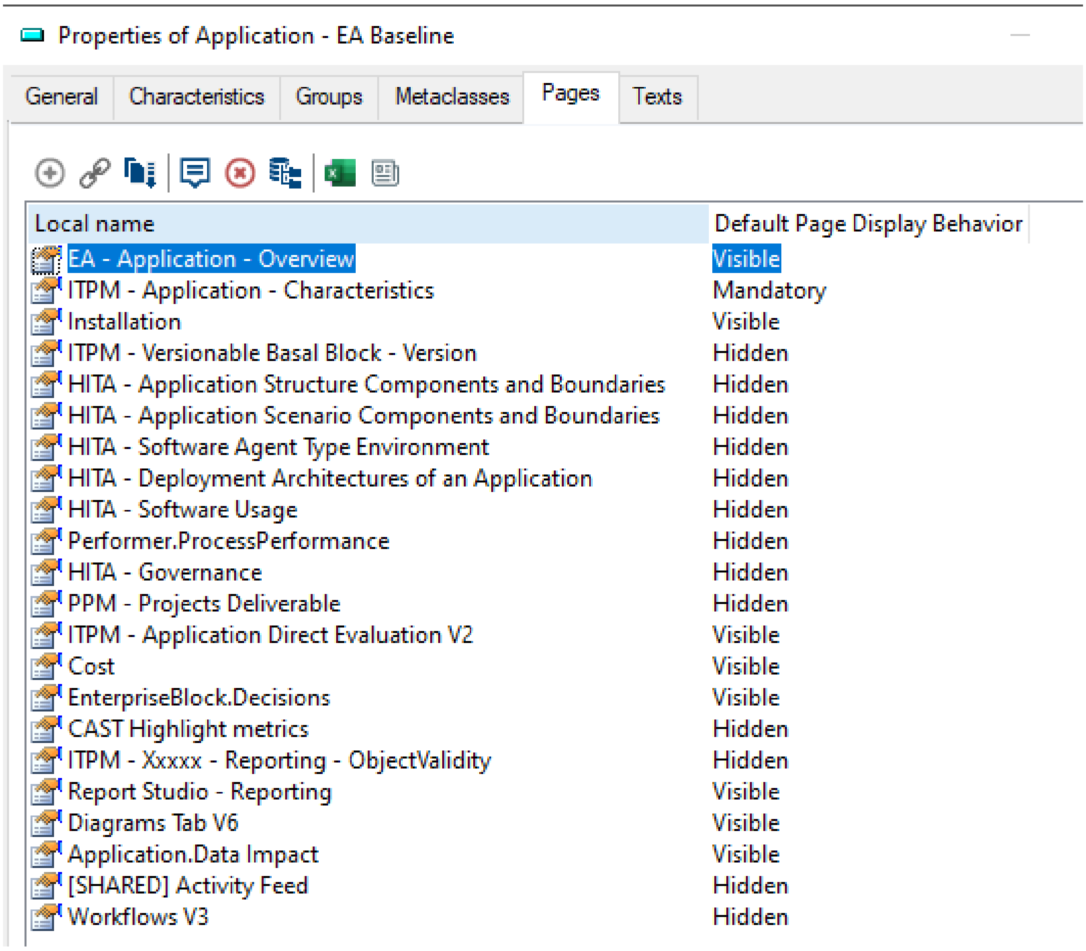
Local name	External Id...	PropertyPage Definition
Action (Action Plan) - EA Connected		Replacement
Action (Action Plan) - EA Premium		Replacement
Action Plan - EA Connected		Replacement
Action Plan - EA Premium		Replacement
Advice Criterion - EA Baseline		Replacement
Advice Template - EA Baseline		Replacement
Aggregated Evaluation Criterion - EA Baseline		Replacement
Application - EA Baseline		Replacement
Application Data Channel - EA Connected		Replacement
Application Data Collection - EA Connected		Replacement
Application Decision Type - EA Baseline		Replacement
Application Deployment Architecture - EA Baseline		Replacement
Application Deployment Component - EA Connect		Replacement

3. From the **Specific PropertyPages Presentation** list, select a **PropertyPage Presentation** and display its properties.

   `E.g.: "Application – EA Baseline" for the **Application** MetaClass.`

4. Display its **Pages** tab.

   All these pages are built with the **_PropertyPageExtension** macro.


Properties of Application - EA Baseline

Local name	Default Page Display Behavior
EA - Application - Overview	Visible
ITPM - Application - Characteristics	Mandatory
Installation	Visible
ITPM - Versionable Basal Block - Version	Hidden
HITA - Application Structure Components and Boundaries	Hidden
HITA - Application Scenario Components and Boundaries	Hidden
HITA - Software Agent Type Environment	Hidden
HITA - Deployment Architectures of an Application	Hidden
HITA - Software Usage	Hidden
Performer.ProcessPerformance	Hidden
HITA - Governance	Hidden
PPM - Projects Deliverable	Hidden
ITPM - Application Direct Evaluation V2	Visible
Cost	Visible
EnterpriseBlock.Decisions	Visible
CAST Highlight metrics	Hidden
ITPM - Xxxxx - Reporting - ObjectValidity	Hidden
Report Studio - Reporting	Visible
Diagrams Tab V6	Visible
Application.Data Impact	Visible
[SHARED] Activity Feed	Hidden
Workflows V3	Hidden

5. Display the properties of the **MetaPropertyPage**.

   E.g.: "ITPM – Application – Overview" for the **Application Overview** page.

Its **Setup** tab shows the **_Parameterization** of the page.

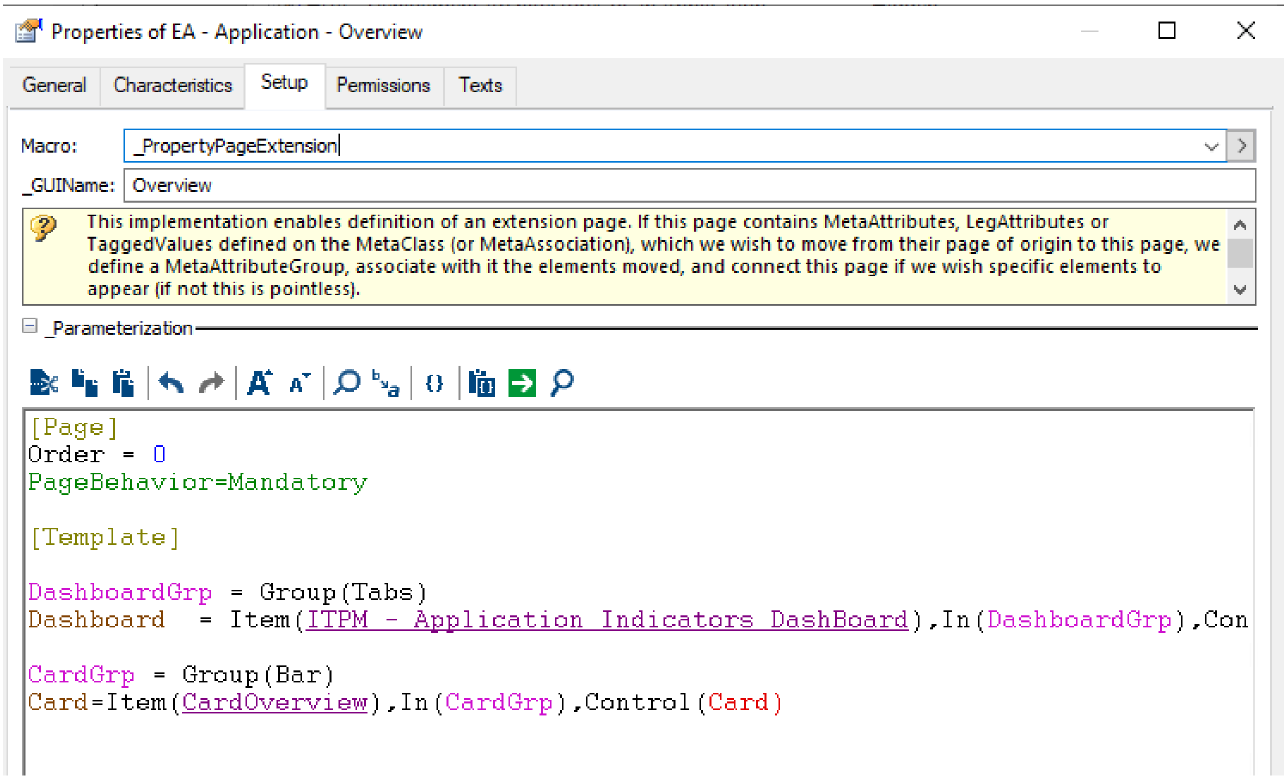
The information in **[Page]** (PageBehavior=Mandatory) has priority over the **Default Page Display Behavior**.(Value= Visible)

The code order in **[Template]** is taken into account in the display.

E.g.: the **Overview** page of **Application** MetaClass

– is in first position and mandatory

– shows indicators above its card



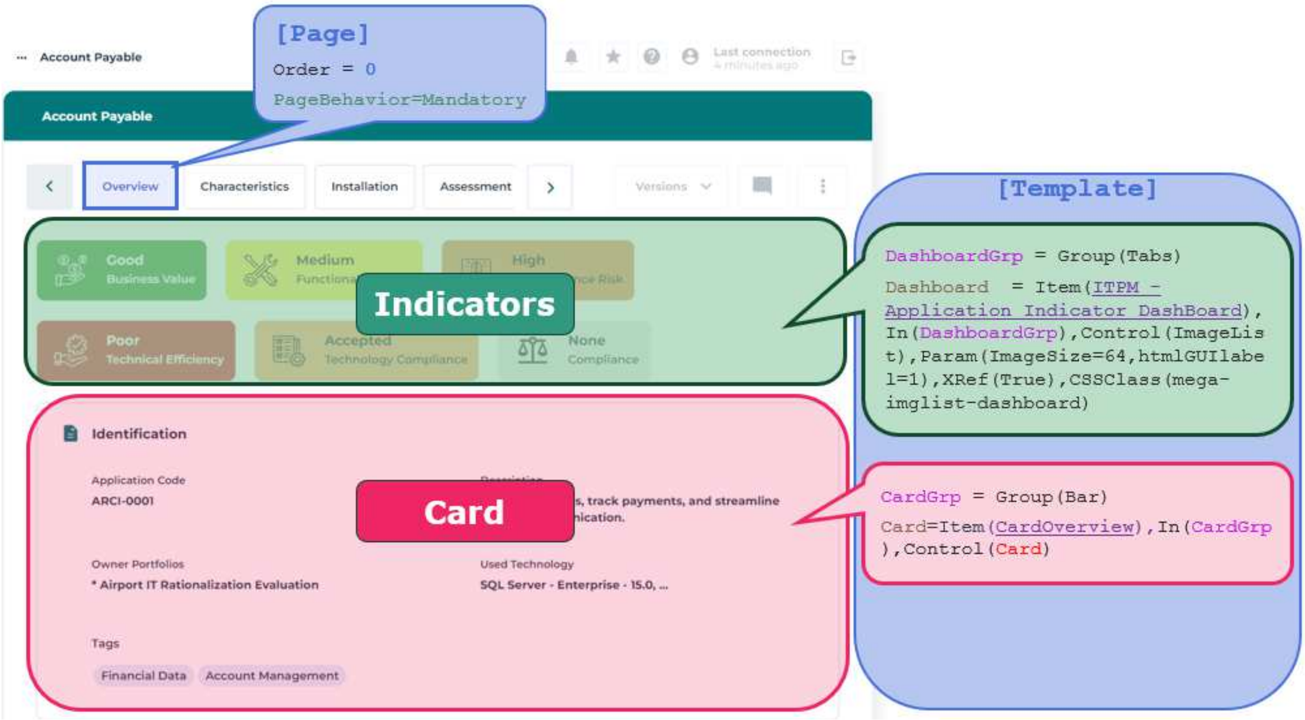
Macro: _PropertyPageExtension

_GUIName: Overview

This implementation enables definition of an extension page. If this page contains MetaAttributes, LegAttributes or TaggedValues defined on the MetaClass (or MetaAssociation), which we wish to move from their page of origin to this page, we define a MetaAttributeGroup, associate with it the elements moved, and connect this page if we wish specific elements to appear (if not this is pointless).

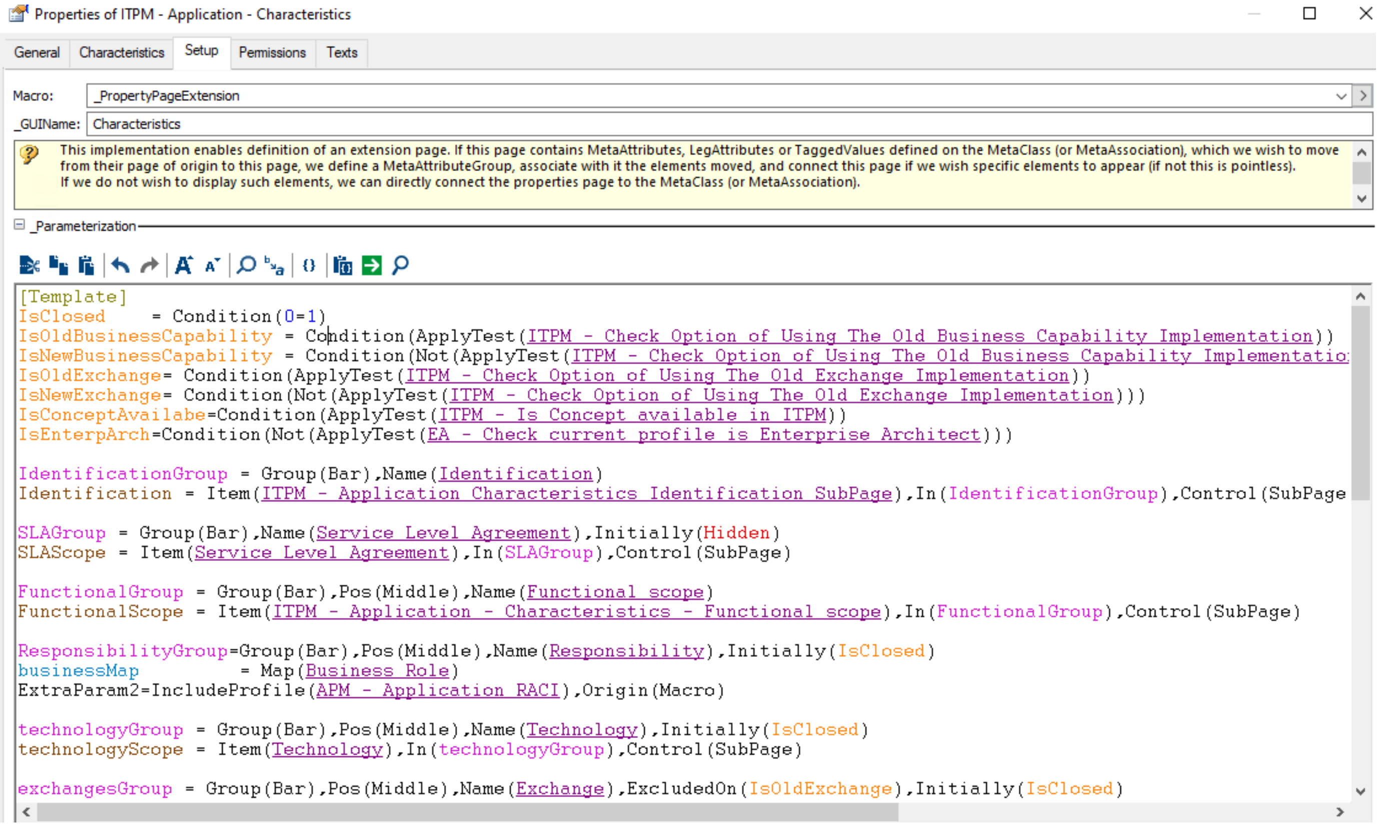
_Parameterization

```
[Page]
Order = 0
PageBehavior=Mandatory

[Template]

DashboardGrp = Group(Tabs)
Dashboard   = Item(ITPM - Application Indicators DashBoard),In(DashboardGrp),Con

CardGrp = Group(Bar)
Card=Item(CardOverview),In(CardGrp),Control(Card)
```



```
[Page]
Order = 0
PageBehavior=Mandatory
```

```
[Template]

DashboardGrp = Group(Tabs)
Dashboard  = Item(ITPM -
Application Indicator DashBoard),
In(DashboardGrp),Control(ImageLis
t),Param(ImageSize=64,htmlGUIlabe
l=1),XRef(True),CSSClass(mega-
imglist-dashboard)

CardGrp = Group(Bar)
Card=Item(CardOverview),In(CardGrp
),Control(Card)
```



Macro: _PropertyPageExtension

_GUIName: Characteristics

This implementation enables definition of an extension page. If this page contains MetaAttributes, LegAttributes or TaggedValues defined on the MetaClass (or MetaAssociation), which we wish to move from their page of origin to this page, we define a MetaAttributeGroup, associate with it the elements moved, and connect this page if we wish specific elements to appear (if not this is pointless).
If we do not wish to display such elements, we can directly connect the properties page to the MetaClass (or MetaAssociation).

```
[Template]
IsClosed     = Condition(0=1)
IsOldBusinessCapability = Condition(ApplyTest(ITPM - Check Option of Using The Old Business Capability Implementation))
IsNewBusinessCapability = Condition(Not(ApplyTest(ITPM - Check Option of Using The Old Business Capability Implementatio
IsOldExchange= Condition(ApplyTest(ITPM - Check Option of Using The Old Exchange Implementation))
IsNewExchange= Condition(Not(ApplyTest(ITPM - Check Option of Using The Old Exchange Implementation)))
IsConceptAvailabe=Condition(ApplyTest(ITPM - Is Concept available in ITPM))
IsEnterpArch=Condition(Not(ApplyTest(EA - Check current profile is Enterprise Architect)))

IdentificationGroup = Group(Bar),Name(Identification)
Identification = Item(ITPM - Application Characteristics Identification SubPage),In(IdentificationGroup),Control(SubPage

SLAGroup = Group(Bar),Name(Service Level Agreement),Initially(Hidden)
SLAScope = Item(Service Level Agreement),In(SLAGroup),Control(SubPage)

FunctionalGroup = Group(Bar),Pos(Middle),Name(Functional scope)
FunctionalScope = Item(ITPM - Application - Characteristics - Functional scope),In(FunctionalGroup),Control(SubPage)

ResponsibilityGroup=Group(Bar),Pos(Middle),Name(Responsibility),Initially(IsClosed)
businessMap          = Map(Business Role)
ExtraParam2=IncludeProfile(APM - Application RACI),Origin(Macro)

technologyGroup = Group(Bar),Pos(Middle),Name(Technology),Initially(IsClosed)
technologyScope = Item(Technology),In(technologyGroup),Control(SubPage)

exchangesGroup = Group(Bar),Pos(Middle),Name(Exchange),ExcludedOn(IsOldExchange),Initially(IsClosed)
```

12.8.7 Adding a Property page to a MetaClass

When you create a MetaPropertyPage for a specific MetaClass (for example, see How to Add a Report DataSet in an object Property Pages) you must define for which profile it is displayed, i.e. you must define the **property page presentation** corresponding to its WET.

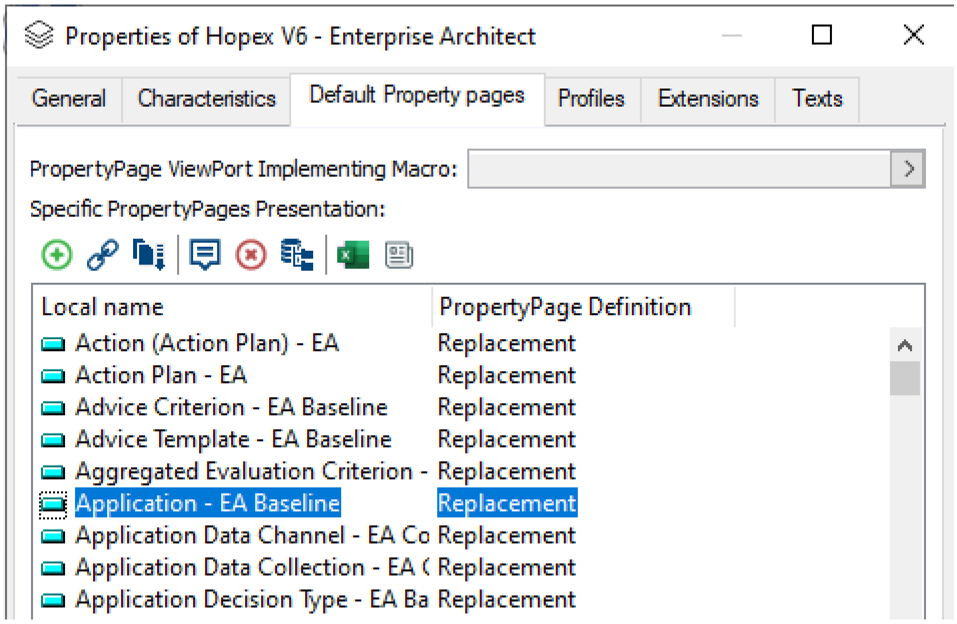
To add a property page to a MetaClass:

1. Access the WET properties.

   ➢ See Accessing the WET metamodel element properties.

   E.g.: "HOPEX V6 – Enterprise Architect" WET to add a Property page to the **Application** MetaClass.

2. Display its **Default Property pages** tab.



Properties of Hopex V6 - Enterprise Architect

PropertyPage ViewPort Implementing Macro:

Specific PropertyPages Presentation:

Local name	PropertyPage Definition
Action (Action Plan) - EA	Replacement
Action Plan - EA	Replacement
Advice Criterion - EA Baseline	Replacement
Advice Template - EA Baseline	Replacement
Aggregated Evaluation Criterion -	Replacement
Application - EA Baseline	Replacement
Application Data Channel - EA Co	Replacement
Application Data Collection - EA (	Replacement
Application Decision Type - EA Ba	Replacement

3. From the **Specific PropertyPages Presentation** list, access the required **PropertyPage Presentation** properties.

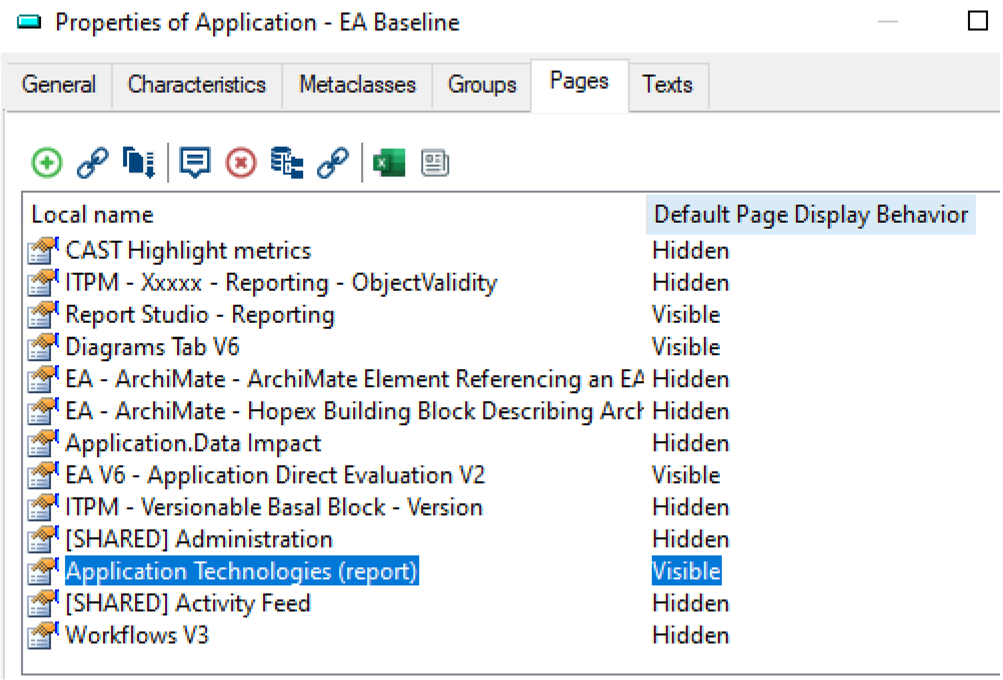
   E.g.: "Application – EA baseline" which defines Application MetaClass property page replacement.

4. In its **Pages** tab, click **Connect** 🔗 and add the MetaPropertyPage.

   E.g.: "Application Technologies (report)" MetaPropertyPage.

5. Define its visibility as required, see **Error! Reference source not found.**.

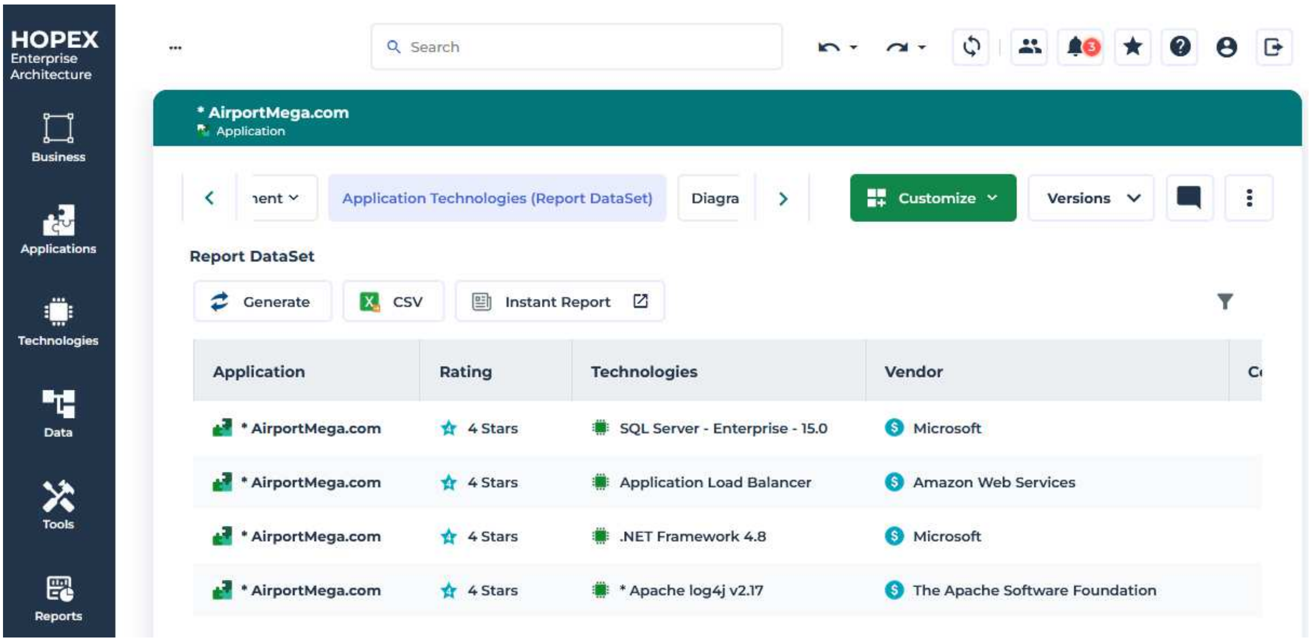
   E.g.: "Visible" to display it by default.



Local name	Default Page Display Behavior
CAST Highlight metrics	Hidden
ITPM - Xxxxx - Reporting - ObjectValidity	Hidden
Report Studio - Reporting	Visible
Diagrams Tab V6	Visible
EA - ArchiMate - ArchiMate Element Referencing an EA	Hidden
EA - ArchiMate - Hopex Building Block Describing Arch	Hidden
Application.Data Impact	Hidden
EA V6 - Application Direct Evaluation V2	Visible
ITPM - Versionable Basal Block - Version	Hidden
[SHARED] Administration	Hidden
Application Technologies (report)	Visible
[SHARED] Activity Feed	Hidden
Workflows V3	Hidden

6. Check that the page is added to the objects.

   E.g.: In HOPEX Enterprise Architecture, the "AirportMega.com" application shows the Application Technologies (Report DataSet).



Application	Rating	Technologies	Vendor
* AirportMega.com	4 Stars	SQL Server - Enterprise - 15.0	Microsoft
* AirportMega.com	4 Stars	Application Load Balancer	Amazon Web Services
* AirportMega.com	4 Stars	.NET Framework 4.8	Microsoft
* AirportMega.com	4 Stars	* Apache log4j v2.17	The Apache Software Foundation

12.9 Customizing diagrams for a WET

12.9.1 Hiding the diagram column in lists

The diagram column in lists enables to view the number of diagrams already created for the object, and access them directly. It also enables to create a diagram if none is available for the object. You can hide the diagram column in all the lists of your Solution at the following levels:

- WET
- Desktop
- Desktop container

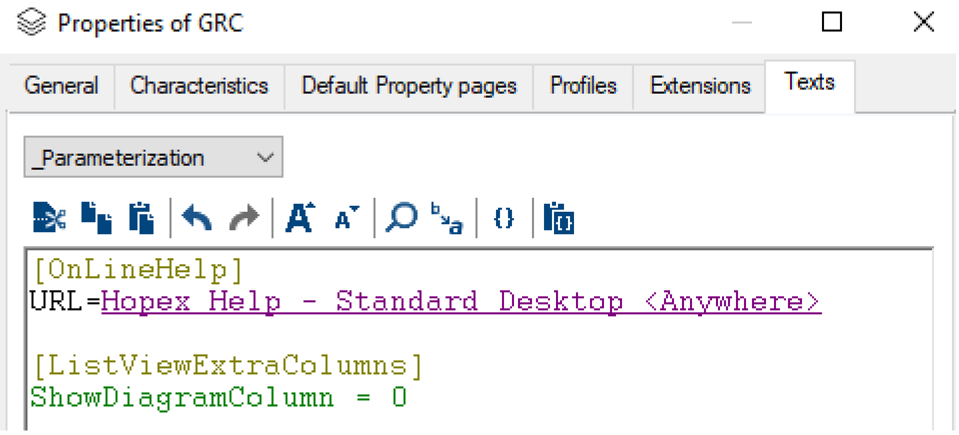
To hide the diagram column in lists:

1. Access the properties of the element concerned.
2. In its **Texts** page, enter the following code:

```
[ListViewExtraColumns]
ShowDiagramColumn = 0
```

With **ShowDiagramColumn** values: "0" the column is hidden, "1" the column is displayed.

For example: in the GRC Solution where diagrams are not used, in the GRC WET properties, **Texts** shows:



```
[OnLineHelp]
URL=Hopex Help - Standard Desktop <Anywhere>

[ListViewExtraColumns]
ShowDiagramColumn = 0
```

12.9.2 Hiding the diagram column in a specific list

Instead of hiding the diagram column for all the lists, you can hide it for a specific list only.

To hide the diagram column in a specific list:

1. Enter the following code:

```
ShowDiagramColumn=0
    Example:
    ApplicationList =
    Item(~MrUiM9B5iyM0[Application]),Contains(Applications),Control(Li
    stView),Title(No),Param(NoDefaultColumn,ReOrder,ShowAlternate=Drop
    Down,ToolBar[-
    R],MultiSelection,ExportCommands=XG,ShowDiagramColumn=0)
```

12.10 Profile and Working Environment Template

At HOPEX connection, a user must select a profile.

Most of HOPEX desktops are WET-based. When using the WETs, to complete the profile configuration, you must assign a WET to the profile, and define the desktop(s) associated with this WET assignment.

Thanks to the **Desktop Manager**, you can define a device-specific desktop, so that when a user connects to the application from a tablet or from a computer, the desktop layout is adapted to the device.

For specific purposes, you may also need to assign several WETs to a profile.

See:

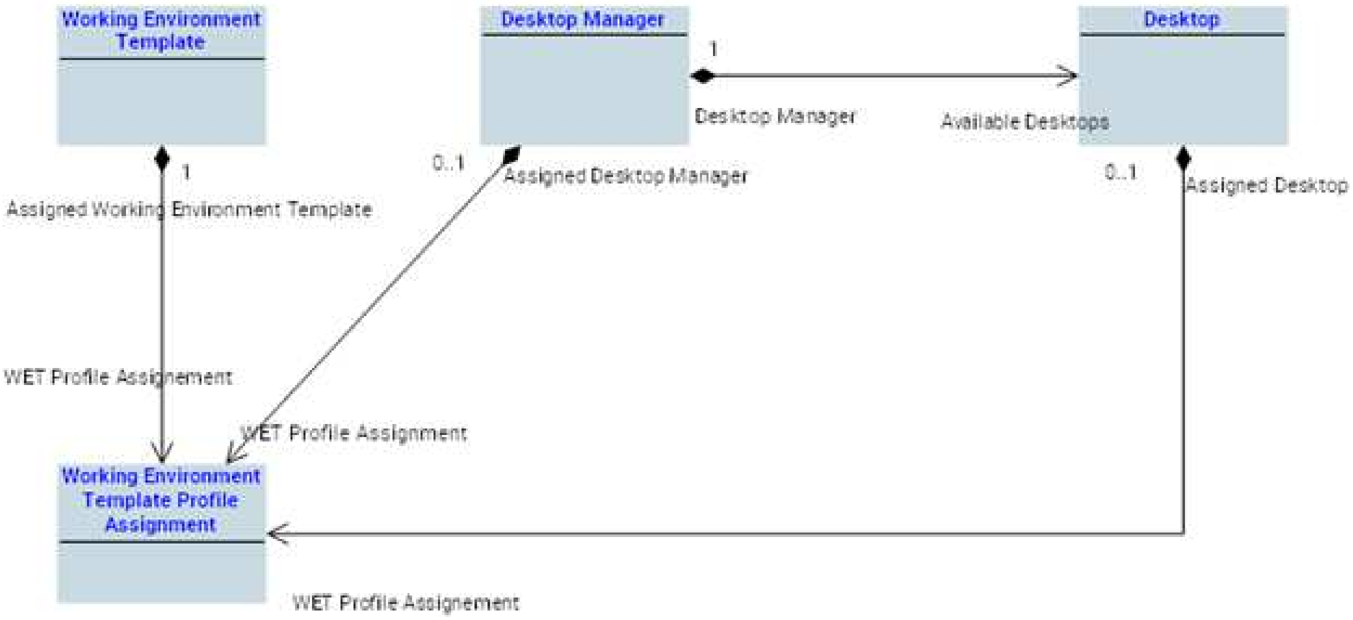
- WET assignment metamodel
- Assigning a WET to a Profile

12.10.1 WET assignment metamodel

The following Metamodel schema shows the architecture and links between the following MetaModel elements:

- **Working Environment Template Profile Assignment**
- **Working Environment Template**
- **Desktop Manager**
- **Desktop**

Note that a **WET-based desktop** configuration does not include **MEGA Application**.



12.10.2 Assigning a WET to a Profile

With a WET-based desktop configuration, you must assign a WET to the profile (with one or several associated desktops).

The profile gives access to a WET-based desktop. When assigning a WET to a profile, you define the desktops associated with the WET Profile Assignment, thanks to a Desktop Manager definition.

> For example, **ITPM Functional Administrator** and **Application Owner Lite** profiles are both connected to **IT Portfolio Management** WET, but each of them gives access to a specific Desktop:
>
> - **ITPM Functional Administrator** profile is assigned the **IT Portfolio Management** WET with "Universal Desktop - Default Desktop".
>
> - **Application Owner Lite** Profile is assigned the **IT Portfolio Management** WET with the "Universal Explorer" Desktop Manager", which includes both Universal Explorer Desktop and Universal Explorer Tablet Desktop.

Thanks to this Desktop Manager, you can define a device-specific desktop, so that when a user connects to the application from a tablet or from computer, the desktop layout is adapted to the device.

> For example, you can connect to **HOPEX Explorer** application from a tablet or from a computer.

For specific purposes, you may need to assign several WETs to the profile.

To assign a WET to the profile:

1. Connect to HOPEX Studio.

2. Access the Profile properties.

3. Select the **Working Environment Template Assignments** tab.

4. Click **New** ⊕.

5. In the **Assigned WET** field, select the WET you want to assign to the profile.

6. (If you want to manage several desktops) Select **Desktop selected via Desktop Manager**
   else keep **Direct selection of desktop** and in **Assigned Desktop** field, select the unique desktop.

7. Keep **Create New Desktop Manager**.

   ➢ You can also reuse an existing one.

8. Click **Next**.

9. (Optional) In the **Name** field, modify the “Desktop Manager” default name, this can be useful if you need to reuse the Desktop Manager for another WET assignment.

10. Click **Connect** 🔗 and connect the desktop(s) you want to define for the profile.

    For example, if you want a specific layout for your desktop according to the device used, you can add a tablet-specific desktop and a computer-specific desktop.

    The WET associated with the Desktop Manager is defined. You must define each desktop usage context.

11. In the list of desktops you have connected, for each desktop, in the corresponding **Device** column select the desktop device type.

    E.g.: Desktop or Tablet.

12. Click **OK**.

    The WET selected is assigned to the profile and its associated desktops are defined with their usage context.

12.10.3 Profiles sharing the same WET

The same WET can be connected to several profiles.

Only the profiles sharing the same WET Assignment give access to the same desktops regarding:

- the toolbar
- the Working Environment Groupe Template (Navigation menus)
- the Working Environment Topic Template (Topics) and its associated MetaCommand items (actions) or defined desktops.

Although most of the Desktop display is similar you can customize the desktop associated with each profile:

- the homepage can be customized for each profile:

  You must define which of the available WET homepage is associated with each profile. Profiles can share the same Homepage or not.
- actions might not be available according to the permissions (CRUD defined for each profile).

12.10.4 Defining the profile homepage

HOPEX homepage shows four customizable blocks:

- Header, with Principles and Tips
- Scope
- Quick access
- Favorite report

  ➢ To customize the HOPEX Homepage, see Customizing the Homepage for a WET.

Prerequisite: the Homepage you can define for the profile must be available in the list of **Homepages** defined for the WET assigned to the profile.

To define a profile homepage:

1. Access the profile properties.
2. Select the **Characteristics** tab.
3. In the **Homepage** field, select the Homepage with the drop-down menu.

12.11 Customizing a desktop for certain users

In some cases, you might need to restrict access to part of a desktop for specific persons or person groups.

> For example, you might need to restrict access to part of the **Business Architecture** desktop for two users (Alex and Camille).

In that case the user connects to the Working Environment desktop instead of the standard desktop.

> For example, with the Business Architecture profile, users Alex and Camille connect to the <Working Environment name> desktop, which is a restricted **Business Architecture** desktop.

To do so you need to:

- define a MetaClass used as entry point

- define a Working Environment instance
- assign the Working Environment to the required persons and/or person groups

12.11.1 Creating a Working Environment

To create a Working Environment:

1. Access your entry point.

   `For example: Enterprise Plan.`

   a. Access your HOPEX Solution as a functional administrator.

   `For example: Business Architecture Functional Administrator.`

   b. Access the **Environment** Navigation menu (Working Environment Group Template) and select the topic concerned.

   `For example: Enterprise Plan.`

2. Click **New** ⊕ and create an instance of the topic: the entry point.

   `For example: "Enterprise plan TEST".`

3. Access the topic instance properties and display its **Assignment** page.

4. In the **Working Environment** field, click the right-oriented arrow and select **New** ⊕.

5. Create the Working Environment:

   a. In the **Name** field enter a Working Environment name.

   `For example: "Working Environment TEST".`

   b. In the **Owner** field: your topic instance is already selected.

   `For example: "Enterprise plan TEST".`

   c. In the **Working Environment template**, select the solution Working Environment Template

   `For example: Business Architecture.`

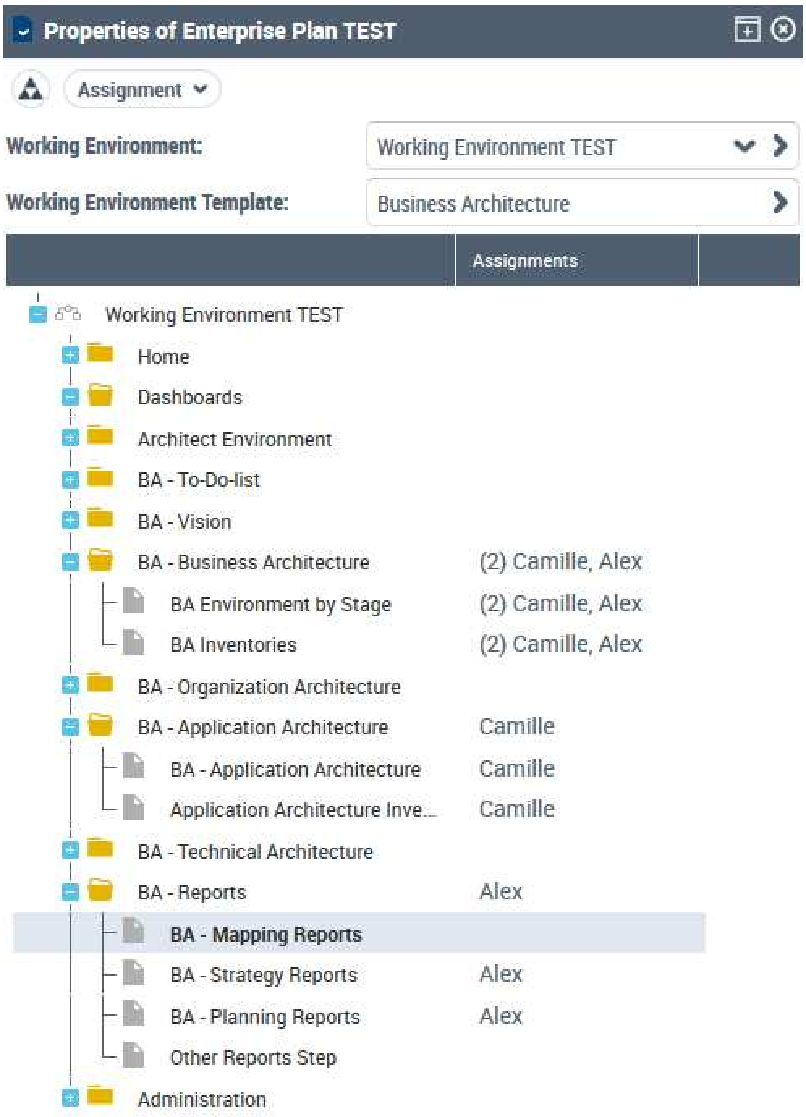
   d. Click **OK**.

6. For each required desktop element row (group or topic), click the corresponding **Assignments** cell and assign the required desktop elements to each required person(s) or person group(s):

   - the required Navigation menus (Groups)

When you assign a Navigation menu (group), by default, all its topics are also assigned. If needed, disconnect the topic assignment concerned or assign directly the specific topic(s) instead of the group.

- the required topics.



Properties of Enterprise Plan TEST

Working Environment: Working Environment TEST

Working Environment Template: Business Architecture

	Assignments
Working Environment TEST	
Home	
Dashboards	
Architect Environment	
BA - To-Do-list	
BA - Vision	
BA - Business Architecture	(2) Camille, Alex
BA Environment by Stage	(2) Camille, Alex
BA Inventories	(2) Camille, Alex
BA - Organization Architecture	
BA - Application Architecture	Camille
BA - Application Architecture	Camille
Application Architecture Inve...	Camille
BA - Technical Architecture	
BA - Reports	Alex
BA - Mapping Reports	
BA - Strategy Reports	Alex
BA - Planning Reports	Alex
Other Reports Step	
Administration	

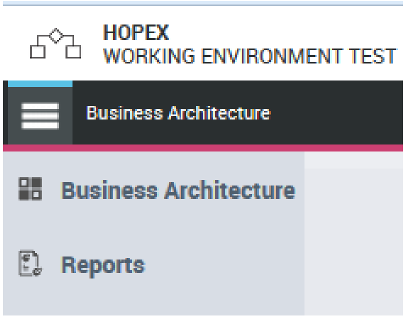
For example, you can define the "Enterprise Plan TEST" Working Environment so that Alex has access:

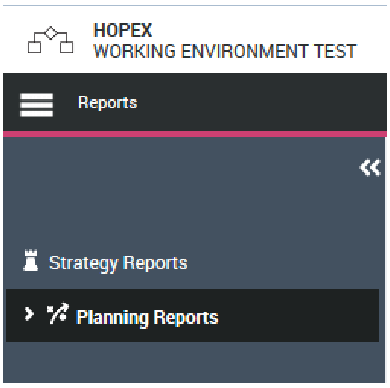
- to all **BA-Business Architecture** topics and
- to specific **BA-Reports** topics: Strategy Reports and Planning Reports

12.11.2 Connecting to a customized desktop

When a person is assigned a Working Environment, at connection the Working Environment desktop is displayed instead of the standard WET desktop.

For example, when Camille or Alex connects to HOPEX with the **Business Architecture** profile, they access the "Enterprise Plan TEST" desktop instead of Business Architecture desktop.



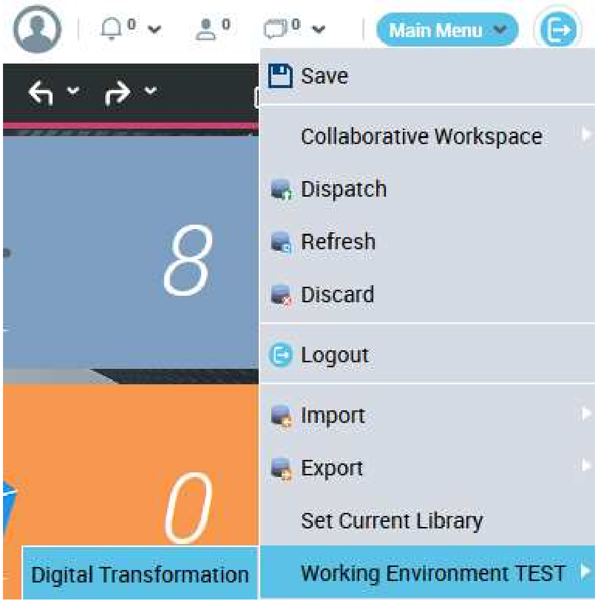


12.11.3 Switching from a Working Environment to another one

When a person is assigned several Working Environments, the person can switch from a Working Environment to another one.

To switch to another Working Environment:

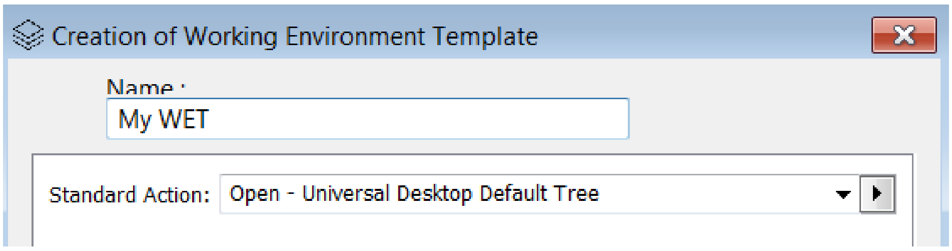
1. From the HOPEX toolbar, select Main menu > <Current Working Environment name> > <Target Working Environment name>.



12.12 WET creation example

To create a WET

1. Connect to **HOPEX** with the HOPEX Customizer profile.
2. Create a WET:
   - **Name**: "my WET".
   - **Standard Action**: "Open - Universal Desktop Default Tree"
     - ➢ See Creating a WET.

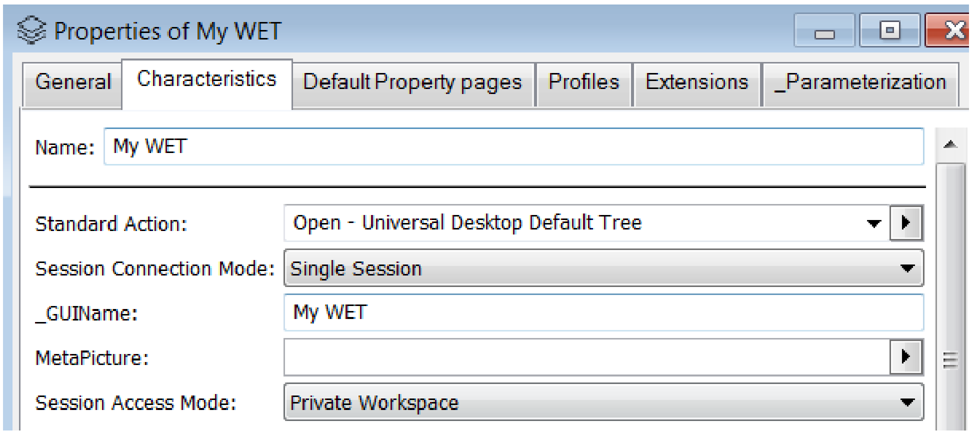


Creation of Working Environment Template

Name : My WET

Standard Action: Open - Universal Desktop Default Tree

3. Define the WET characteristics:
   - **_GUIName**: "my WET".
   - **MetaPicture**:
     - ➢ See Defining the WET characteristics.



Properties of My WET

General | Characteristics | Default Property pages | Profiles | Extensions | _Parameterization

Name: My WET

Standard Action: Open - Universal Desktop Default Tree

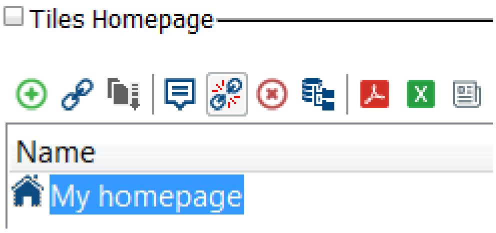
Session Connection Mode: Single Session

_GUIName: My WET

MetaPicture:

Session Access Mode: Private Workspace

4. Create a **Tiles Homepage** for your WET:

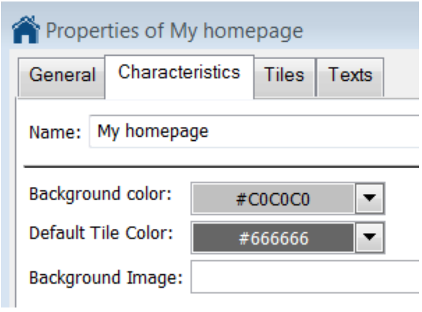


□ Tiles Homepage

Name

My homepage

   - Define a Background color
   - Define a Default Tile color


Properties of My homepage

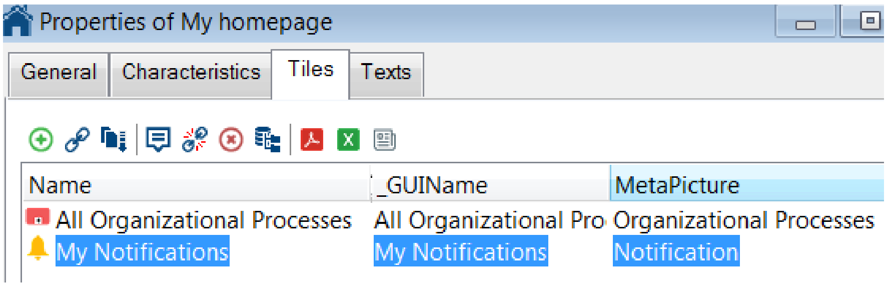
Name: My homepage

Background color: #C0C0C0

Default Tile Color: #666666

Background Image:

- Add tiles to your homepage

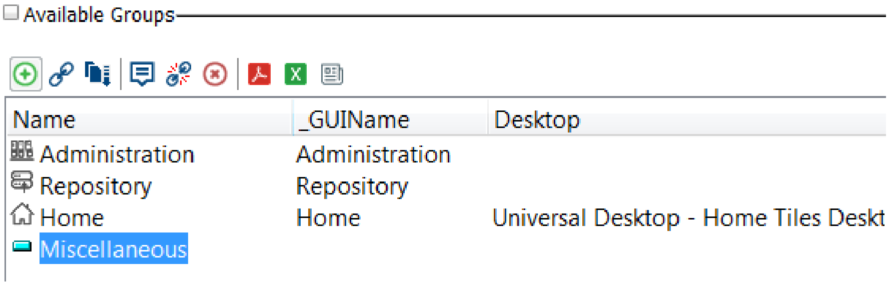

Properties of My homepage

Name	_GUIName	MetaPicture
All Organizational Processes	All Organizational Pro	Organizational Processes
My Notifications	My Notifications	Notification

➢ See **Error! Reference source not found.**.

5. Define your WET Navigation menus:

- Add Navigation menus to your WET: “Home”, “Repository” and “Administration” Working Environment Group Templates
- Create a Navigation menu: Miscellaneous”

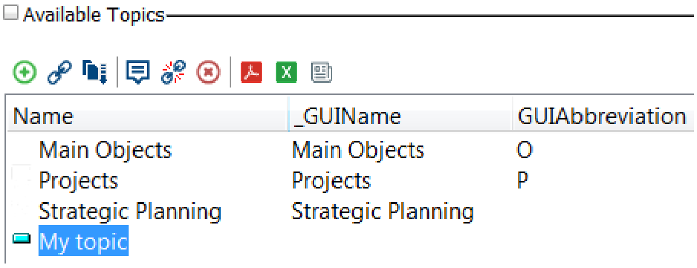

□ Available Groups

Name	_GUIName	Desktop
Administration	Administration	
Repository	Repository	
Home	Home	Universal Desktop - Home Tiles Deskt
Miscellaneous		

➢ See Defining a Navigation Menu (Working Environment Group Template).

6. Define the topics of the Navigation menu you created:

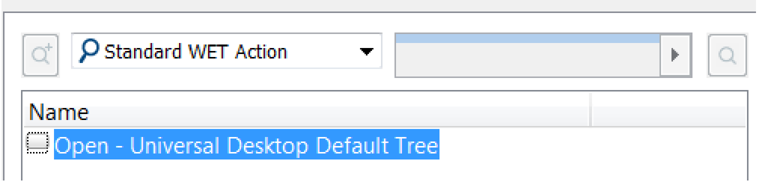
- Add: “Main objects”, “Projects”, “Strategic planning”
- Create: “My topic”


□ Available Topics

Name	_GUIName	GUIAbbreviation
Main Objects	Main Objects	O
Projects	Projects	P
Strategic Planning	Strategic Planning	
My topic		

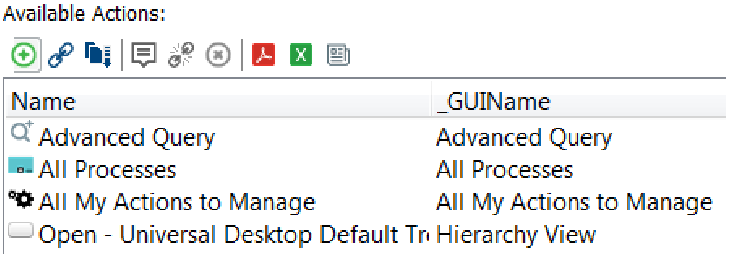
➢ See Defining a Working Environment Topic Template (topic).

2. Define the actions to be included in the topic you created (“Miscellaneous”):
   - the WET standard action: “Open - Universal Desktop Default Tree”



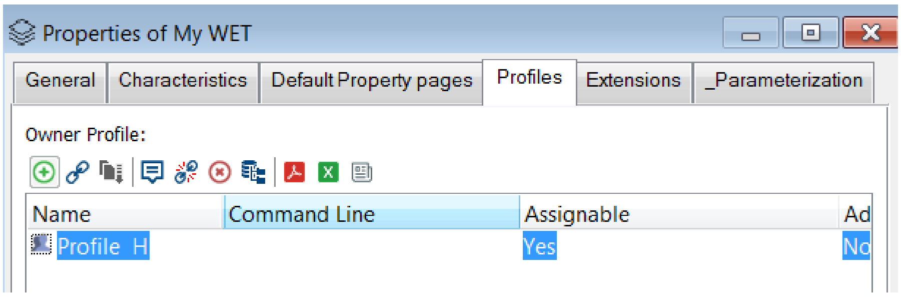
Name
Open - Universal Desktop Default Tree

   - “Advanced Query”, ‘All processes” and “All My Actions to manage” actions.


Available Actions:

Name	_GUIName
Advanced Query	Advanced Query
All Processes	All Processes
All My Actions to Manage	All My Actions to Manage
Open - Universal Desktop Default Tr	Hierarchy View

3. Connect the WET to a profile:



Owner Profile:

Name	Command Line	Assignable	Ad
Profile H		Yes	No

➢ See Assigning a WET to a Profile.

INTEGRATING A POWER BI REPORT IN HOPEX

1. INTRODUCTION

This walkthrough shows how to integrate a Power BI report into the HOPEX web desktop. It focuses mainly on the specifics of external web components embedding in the HOPEX desktop, either through an iFrame or a vendor JavaScript SDK. It will not go into details about the specifics of Power BI or explain all the customization possibilities offered by Hopex Versatile Desktop.

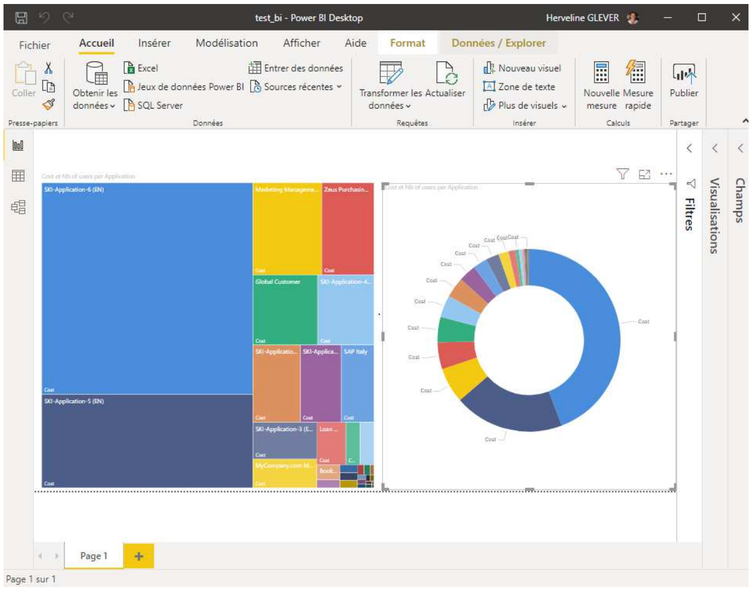
For further options about how to create Power BI reports and embed them, see the official Microsoft documentation https://docs.microsoft.com/en-us/power-bi/

For more information about all the possibilities to configure the HOPEX Versatile Desktop, see HOPEX Power Studio > Customizing the User Interface > Versatile Desktop documentation.

1.1. Preparing the Report

To prepare the report in Power BI Desktop:

1) Access **Power BI Desktop** and create your report.

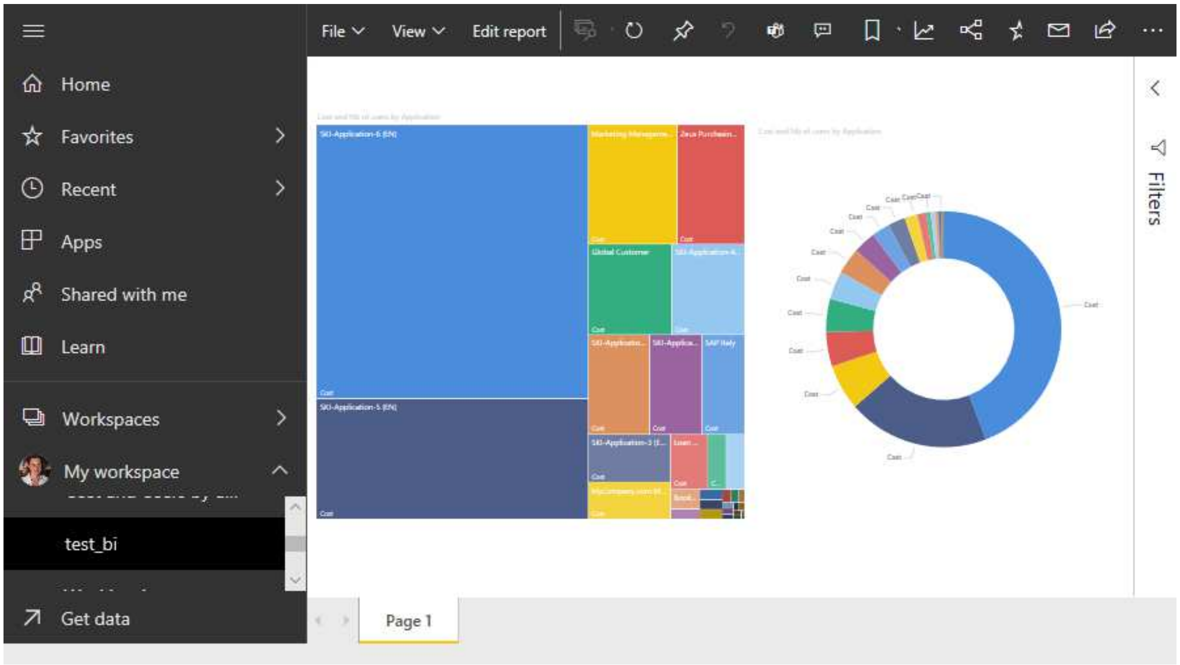


2) In the **Power BI Desktop:**

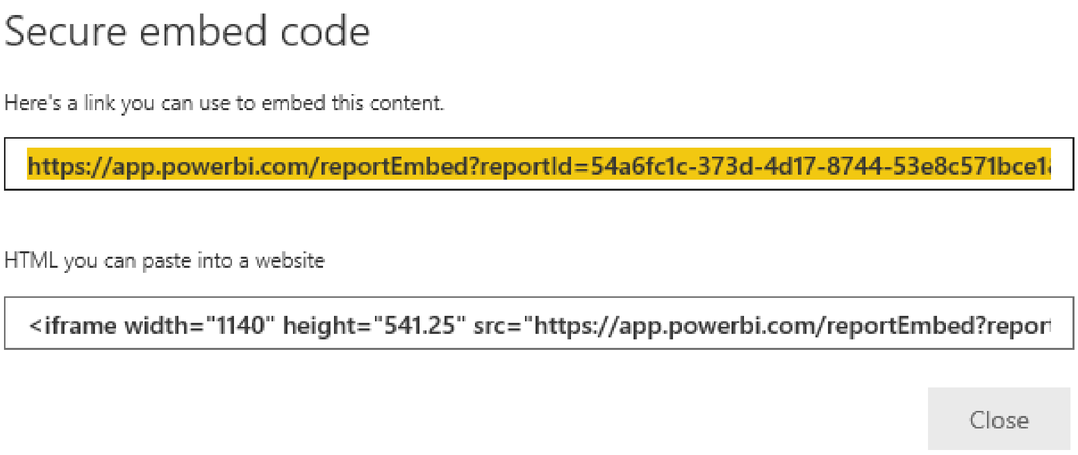
   a) Click **Save** to save your report.

   b) Click **Publish** to publish your report online in the Power BI cloud service.

3) In **Power BI** (powerbi.com), expand **My workspace** pane, and open the report.



4) In the report tool bar, select **File -> Embed** to get a link to securely embed this report in a website or portal.



Secure embed code

Here's a link you can use to embed this content.

https://app.powerbi.com/reportEmbed?reportId=54a6fc1c-373d-4d17-8744-53e8c571bce1

HTML you can paste into a website

<iframe width="1140" height="541.25" src="https://app.powerbi.com/reportEmbed?repor

5) Take note of the **link**.

You will need it in the next steps.

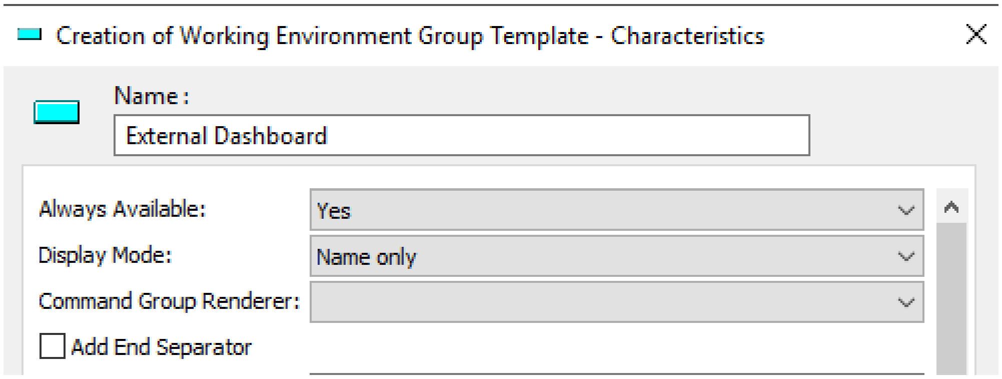
1.2. Adding an Entry Point in HOPEX User Interface

You need to customize the HOPEX desktop, to which you want to add your Power BI report, i.e. you need to add a Navigation pane (a Working Environment Group Template), and its required topic and action.

To customize HOPEX desktop:

1) Connect to HOPEX Customization desktop (windows Front-End, profile: HOPEX Customizer).
2) Display the **MetaStudio** tab (**View > Navigation Windows > MetaStudio**).
3) In the **Working Environment Template** folder, right-click the Working Environment Template to which you want to add a navigation pane and select **New > Working Environment Group Template**.

   `For example: "Enterprise Architecture" Working Environment Template.`
4) Right-click the "Enterprise Architecture" Working Environment Template and select **New > Working Environment Group Template**.
5) Configure the Working Environment Group Template:
   - **Name**: "External Dashboard"
   - **Always Available**: "Yes"
   - **Display Mode**: "Name only"



Creation of Working Environment Group Template - Characteristics

Name : External Dashboard

Always Available: Yes

Display Mode: Name only

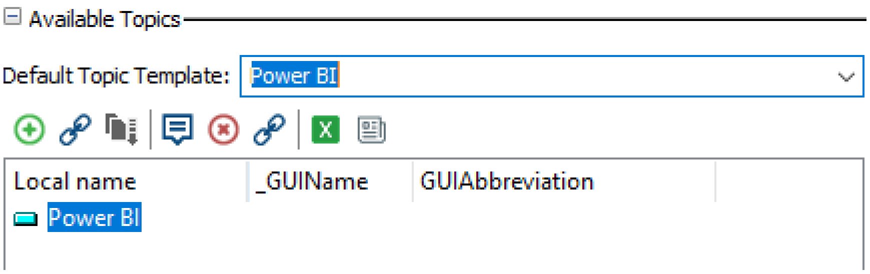
Command Group Renderer:

□ Add End Separator

6) In the **Available Topics** section:

   c) Click **New** ⊕ to create a Topic and name it (e.g.: Power BI).

   d) In the **Default Topic Template** field select the Topic you created.



Available Topics

Default Topic Template: Power BI

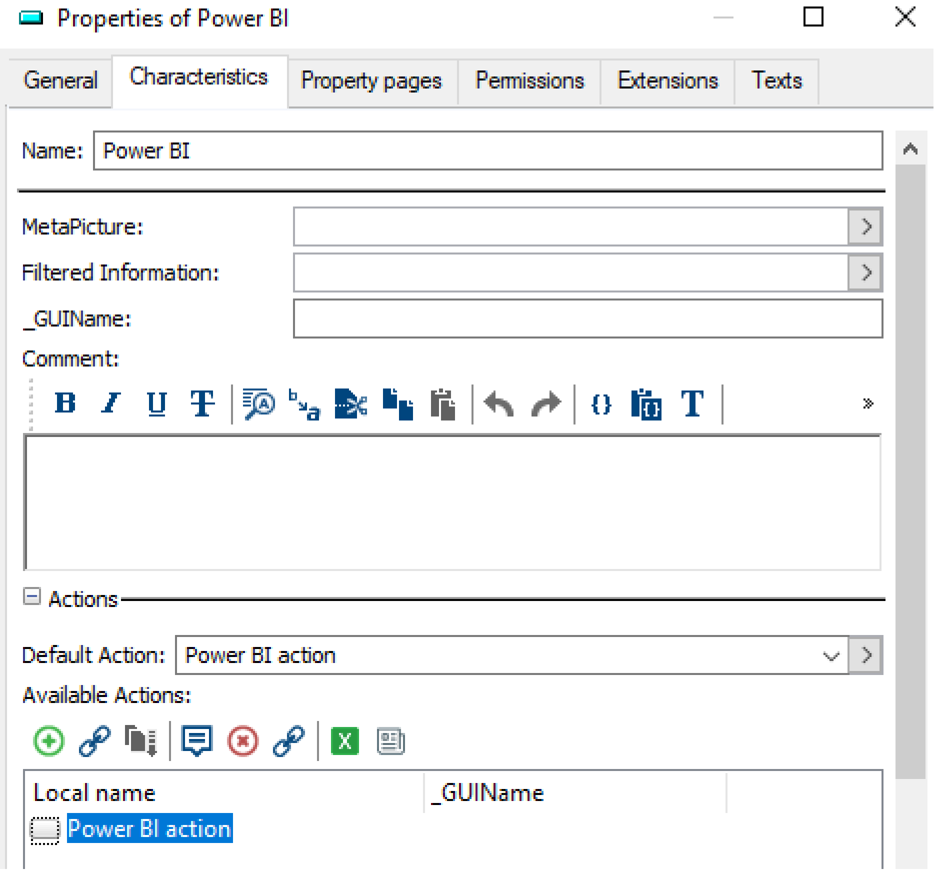
Local name	_GUIName	GUIAbbreviation
Power BI		

7) Access the “Power BI” Topic properties:

a) In its **Actions** section, in **Available Actions**, click **New** ⊕ to create an action

b) Name it “Power BI action”.

c) In the **Default Action** field select the “Power BI action” action.



Properties of Power BI

Name: Power BI

MetaPicture:

Filtered Information:

_GUIName:

Comment:

Actions

Default Action: Power BI action

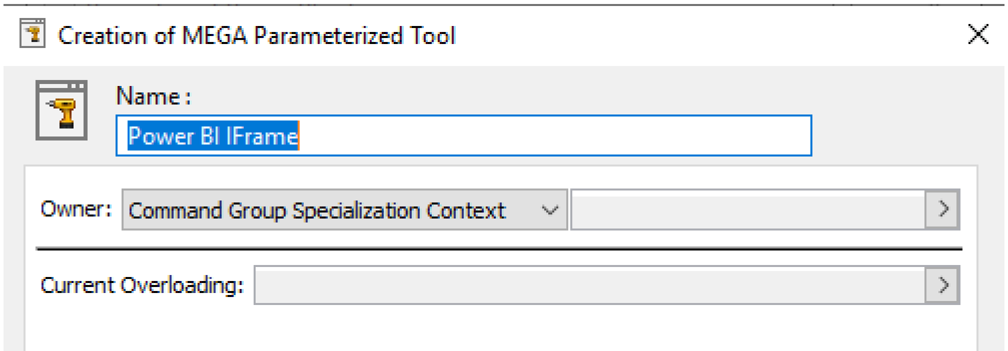
Available Actions:

Local name	_GUIName
Power BI action	

8) Access the “Power BI action” properties:

a) In its **Behavior** section, in the **MEGA Parameterized Tool** field, click the arrow and select **Create MEGA Parameterized Tool**.

b) In the **Name** field enter: “Power BI Iframe”.

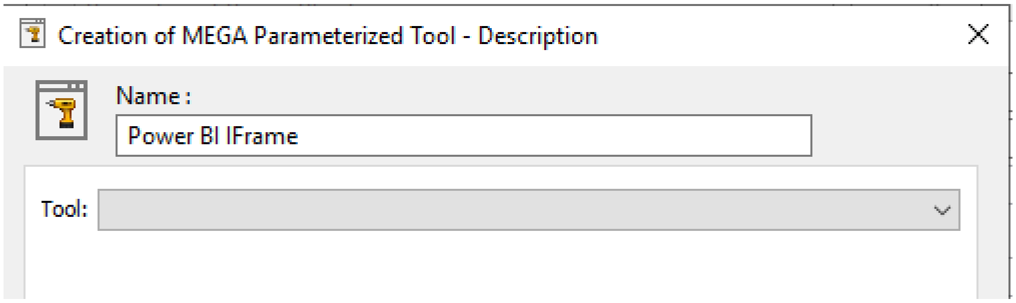

Creation of MEGA Parameterized Tool

Name : Power BI IFrame

Owner: Command Group Specialization Context

Current Overloading:

c) Click **Next**.


Creation of MEGA Parameterized Tool - Description

Name : Power BI IFrame

Tool:

9) Integrate the report in HOPEX, using:

- an iframe (the simplest way), see Integrating the Report with an iFrame.
- the PowerBI JavaScript SDK (a much more complex way, which needs development expertise, but can offer more capabilities and interactions), see Integrating the Report with the PowerBI JavaScript SDK.

2. INTEGRATING THE REPORT WITH AN IFRAME

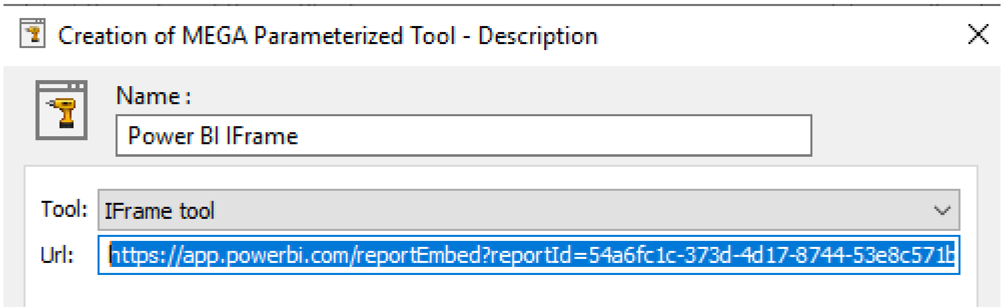
You need to:

- Configure the Mega parameterized tool
- (if needed) Authorize external IFrame

2.1. Configuring the Mega Parameterized Tool

To configure the MEGA Parametrized Tool:

1) In the **MEGA Parameterized Tool – Description** creation wizard, in the **Tool** field, select **IFrame tool**.
2) In the **Url** field, enter the url given by PowerBI in the previous step.



Creation of MEGA Parameterized Tool - Description

Name : Power BI IFrame

Tool: IFrame tool

Url: https://app.powerbi.com/reportEmbed?reportId=54a6fc1c-373d-4d17-8744-53e8c571b

3) Click **Finish**.
4) Click **OK**.
5) Click **Close**.
6) Cick **Close**.
7) In HOPEX tool bar click **Dispatch** to dispatch your changes.
8) Exit HOPEX.

2.2. Authorizing external iFrame

Depending on HOPEX versions, for security reasons, loading of external iFrame might be disabled or generate a warning.

To authorize Power BI:

1) Edit the Web.Config file of the HOPEX IIS application.
2) If present, change both the **Content-Security-Policy-Report-Only** and **Content-Security-Policy** headers to include a frame-src directive specifying the Power BI host source.

```
<configuration>
  <system.webServer>
    <httpProtocol>
      <customHeaders>
        <add name="Content-Security-Policy-Report-Only"
value="block-all-mixed-content; frame-src 'self' https://app.powerbi.com; default-
src 'self'; script-src 'self' 'unsafe-eval' 'unsafe-inline'; style-
src 'self' 'unsafe-inline'; img-src 'self' data:"/>
        <add name="Content-Security-Policy"
value="frame-src 'self' https://app.powerbi.com;"/>
```

2.3. Testing

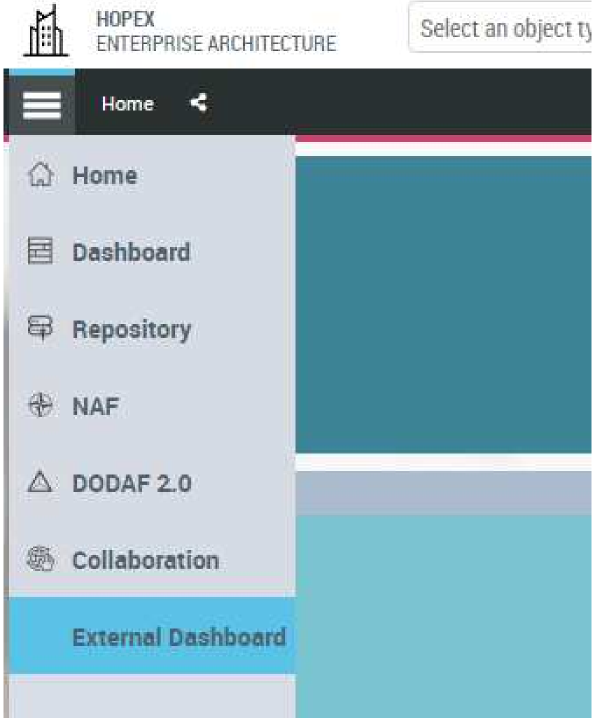
To test your customization:

1) Connect to HOPEX (Web Front-End) with the profile corresponding to the Working Environment Template.

   `E.g.: Enterprise Architect.`

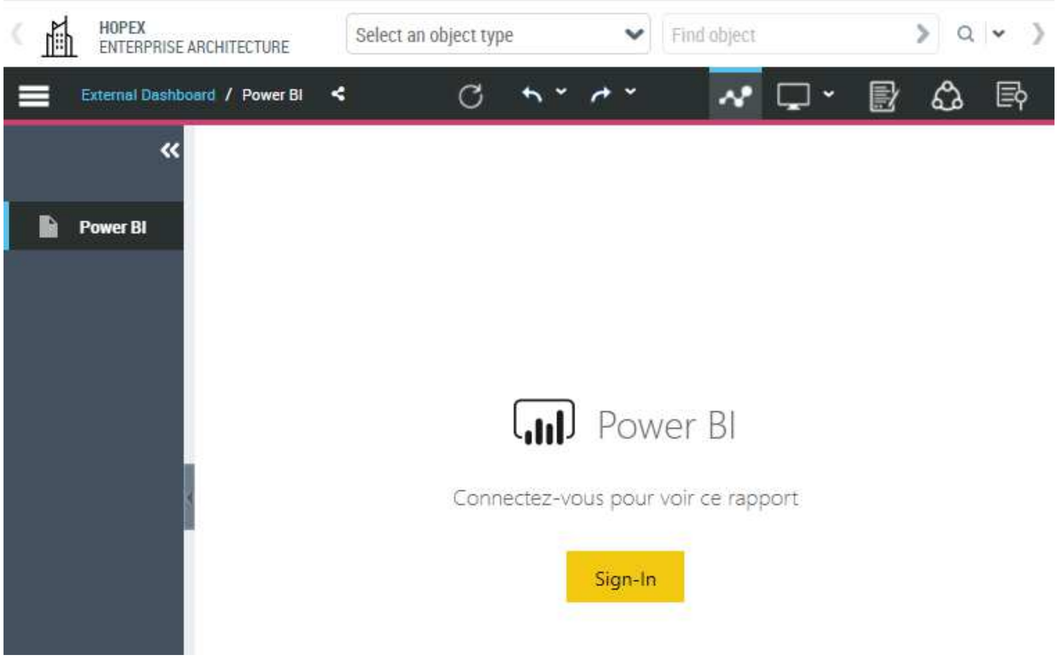
2) Click the **Navigation** menu.

   The **External Dashbord** navigation pane (Environment Group Template) you created is available.



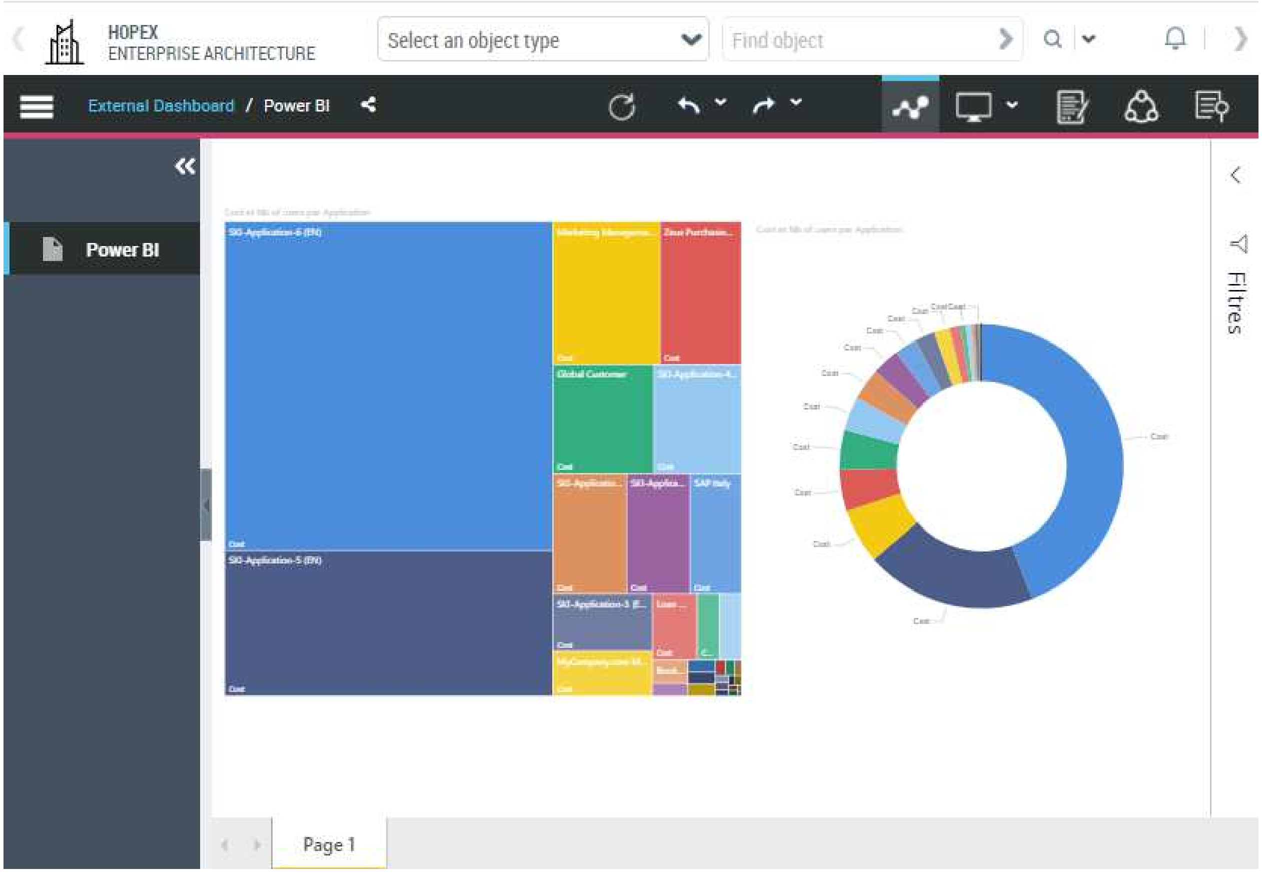
3) Click **External Dashboard**.

The **Power BI** topic you created is available.



4) Power BI might required re-authentication: click **Sign-In** and if required, enter your Power BI credentials.

The report you created in Power BI is displayed.



3. INTEGRATING THE REPORT WITH THE POWERBI JAVASCRIPT SDK

You need to:

- Configure the Mega parameterized tool
- Register the HOPEX application in Azure Directory
- Create the JavaScript wrapper
- (if needed) Authorize external IFrame

3.1. Configuring the Mega Parameterized Tool

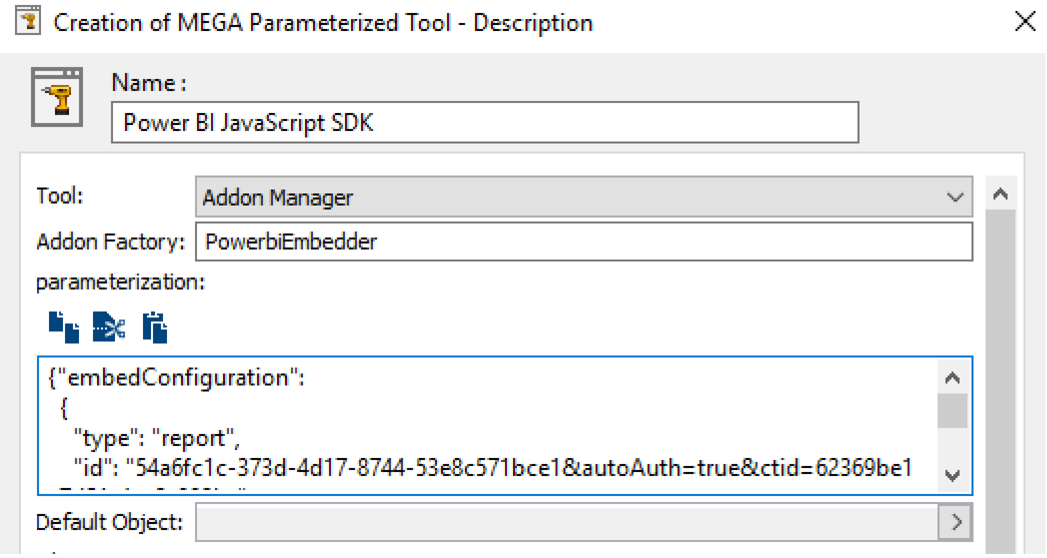
To configure the MEGA Parametrized Tool:

1) In the **MEGA Parameterized Tool – Description** creation wizard, in the **Tool** field, select **Addon Manager** tool.
2) In the **Addon Factory** field, enter “PowerbiEmbedder”.

   Warning: this field is case sensitive.
3) In the **parameterization** pane, use this template to set the parameterization:

   Make sure to replace the sample id shown here by the reportId found in the url given by PowerBI in the first steps.

```
{"embedConfiguration":
  {
    "type": "report",
    "id": "~~8c9923b5-e7f2-4249-b0a7-d376a0c187ce~~",
    "embedUrl": "https://app.powerbi.com/reportEmbed"
  }
}
```



Creation of MEGA Parameterized Tool - Description

Name : Power BI JavaScript SDK

Tool: Addon Manager

Addon Factory: PowerbiEmbedder

parameterization:

```
{"embedConfiguration":
 {
  "type": "report",
  "id": "54a6fc1c-373d-4d17-8744-53e8c571bce1&autoAuth=true&ctid=62369be1
```

Default Object:

4) Click **Finish**.

5) Click **OK.**

6) Click **Close**.

7) Click **OK**.

8) In HOPEX tool bar click **Dispatch** to dispatch your changes.

3.2. Registering the HOPEX application in Azure Active Directory

These steps might require the help or approval of your IT staff.

To register the HOPEX application in Azure Directory:

1) Go to your Azure portal (https://portal.azure.com) and log-in.

2) In the **Azure services**, select **Azure Active Directory**.

3) In the **Manage** pane, select **App registrations**.

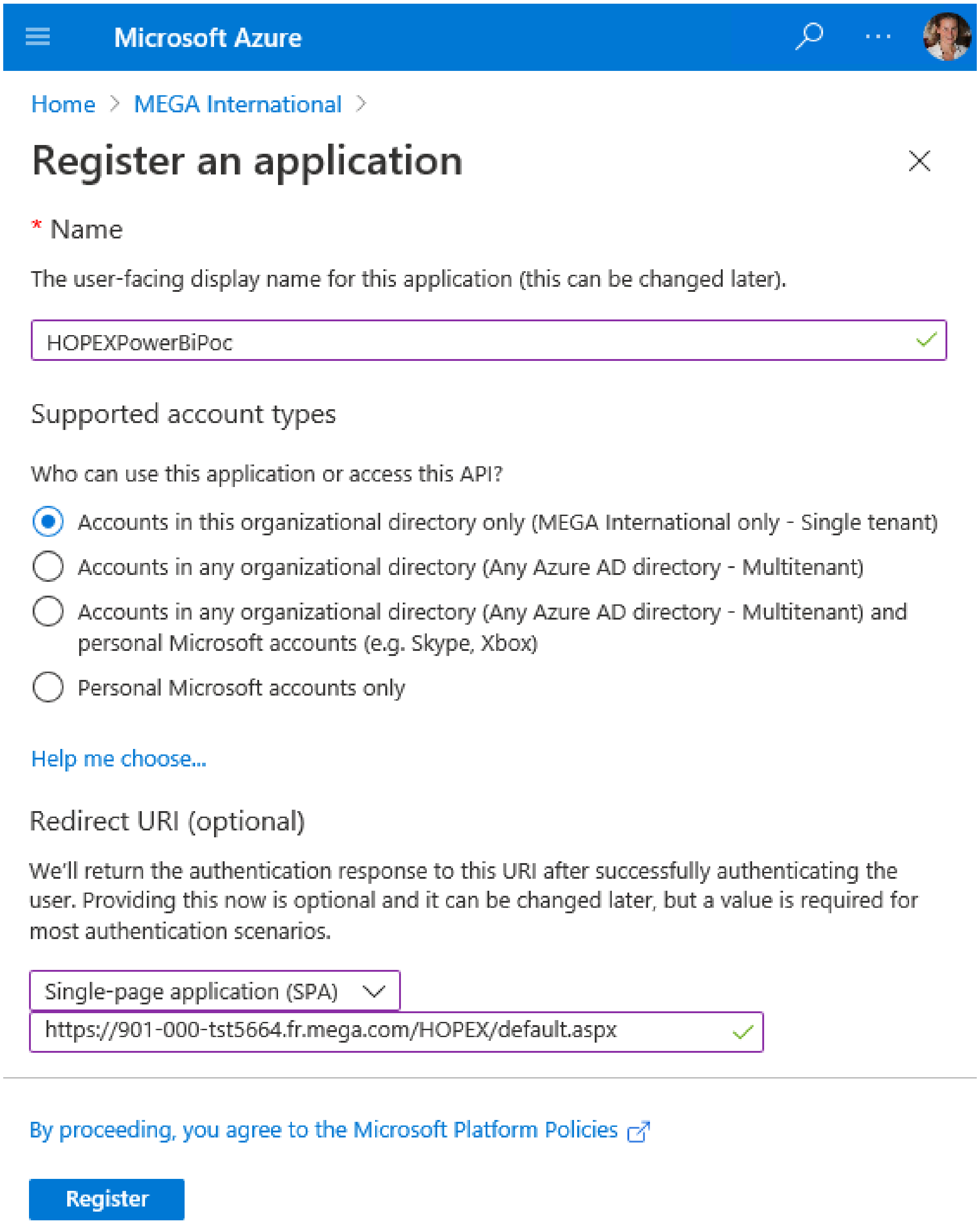
4) In the right pane, click **New registration**.

5) In the **Register an application** page:

   a) In the **Name** field, enter a name: HOPEXPowerBiPoc.

   b) In the **Supported account types** section, keep the Single tenant account type.

   c) In the **Redirect URI (optional)** section, select “Single-Page application (SPA)”, and in the field enter the URL of the HOPEX installation, including the default.aspx. The installation **MUST** use HTTPS.



Register an application

* Name

The user-facing display name for this application (this can be changed later).

HOPEXPowerBiPoc ✓

Supported account types

Who can use this application or access this API?

● Accounts in this organizational directory only (MEGA International only - Single tenant)

○ Accounts in any organizational directory (Any Azure AD directory - Multitenant)

○ Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)

○ Personal Microsoft accounts only

Help me choose...

Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Single-page application (SPA)

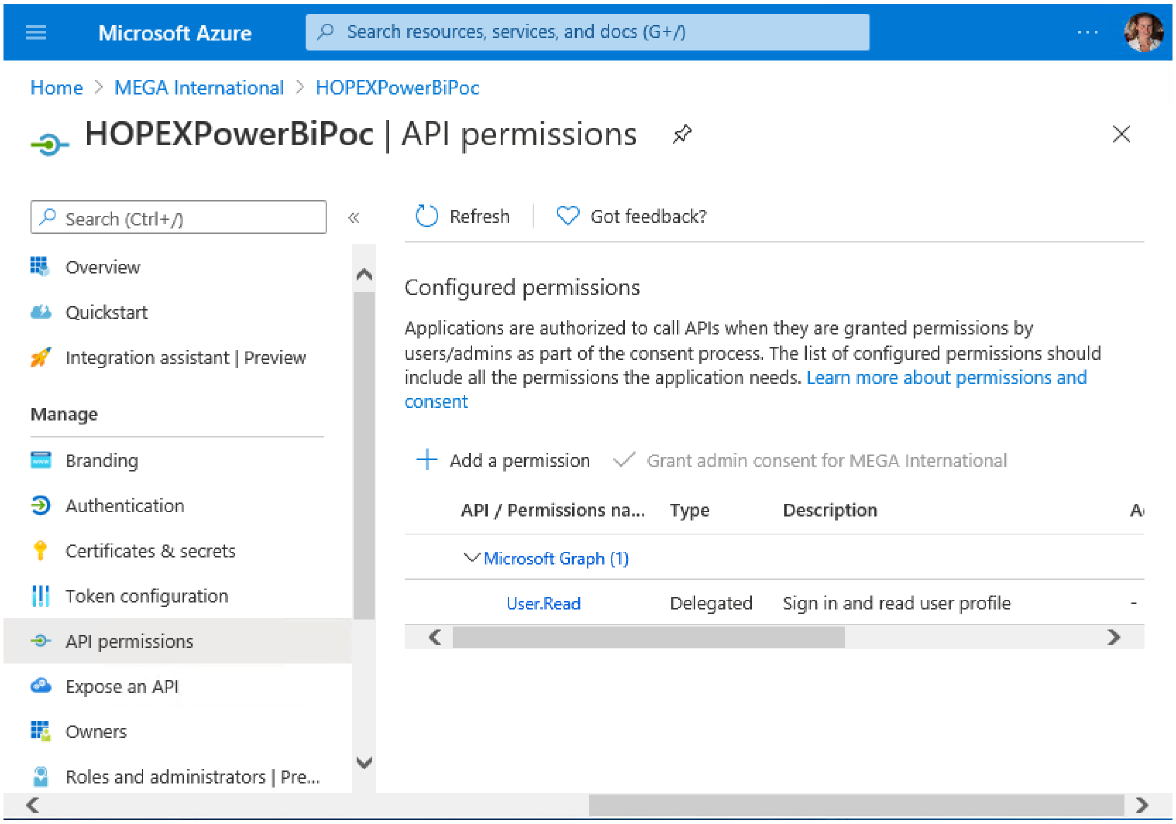
https://901-000-tst5664.fr.mega.com/HOPEX/default.aspx ✓

By proceeding, you agree to the Microsoft Platform Policies

d) Click **Register**.

The new App page appears.

6) In the **Manage** section, select **API permissions**.

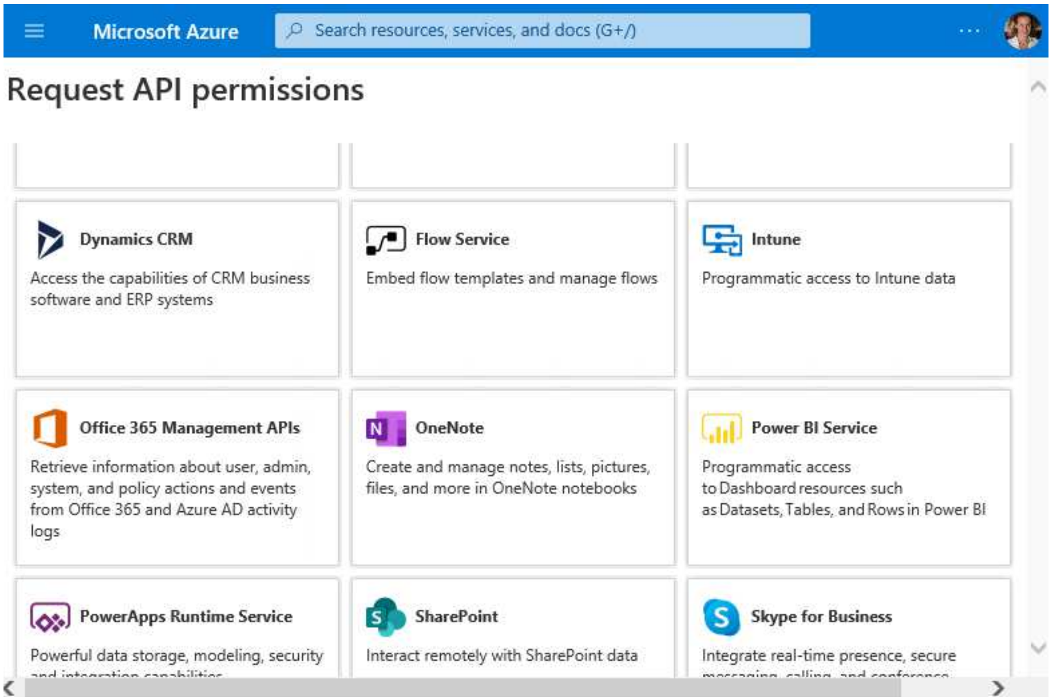

Microsoft Azure
Search resources, services, and docs (G+/)
Home > MEGA International > HOPEXPowerBiPoc
HOPEXPowerBiPoc | API permissions
Search (Ctrl+/)
Overview
Quickstart
Integration assistant | Preview
Manage
Branding
Authentication
Certificates & secrets
Token configuration
API permissions
Expose an API
Owners
Roles and administrators | Pre...
Refresh
Got feedback?
Add a permission
Grant admin consent for MEGA International

Configured permissions

Applications are authorized to call APIs when they are granted permissions by users/admins as part of the consent process. The list of configured permissions should include all the permissions the application needs. Learn more about permissions and consent

API / Permissions na...	Type	Description	A
Microsoft Graph (1)			
User.Read	Delegated	Sign in and read user profile	-

7) In the right pane, click **Add a permission**.



Request API permissions

Dynamics CRM
Access the capabilities of CRM business software and ERP systems

Flow Service
Embed flow templates and manage flows

Intune
Programmatic access to Intune data

Office 365 Management APIs
Retrieve information about user, admin, system, and policy actions and events from Office 365 and Azure AD activity logs

OneNote
Create and manage notes, lists, pictures, files, and more in OneNote notebooks

Power BI Service
Programmatic access to Dashboard resources such as Datasets, Tables, and Rows in Power BI

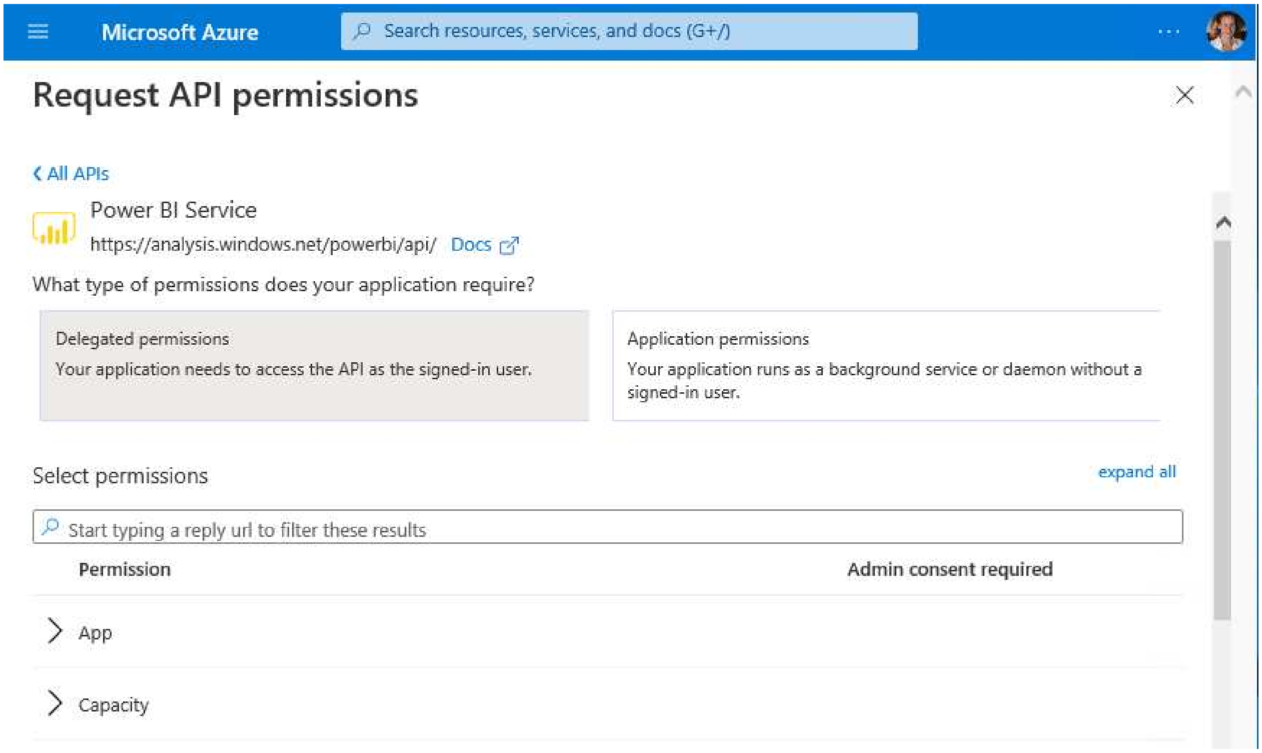
PowerApps Runtime Service
Powerful data storage, modeling, security and integration capabilities

SharePoint
Interact remotely with SharePoint data

Skype for Business
Integrate real-time presence, secure messaging, calling, and conference

a) Click **Power BI Service**.

b) Click **Delegated permissions**.



Request API permissions

< All APIs

Power BI Service
https://analysis.windows.net/powerbi/api/ Docs

What type of permissions does your application require?

Delegated permissions
Your application needs to access the API as the signed-in user.

Application permissions
Your application runs as a background service or daemon without a signed-in user.

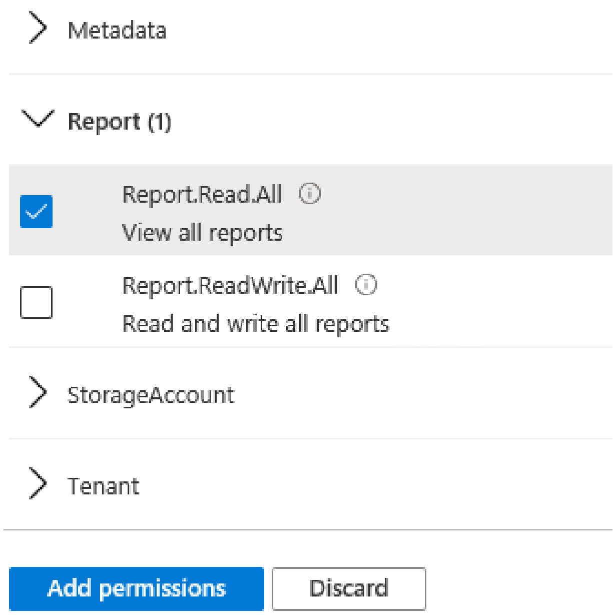
Select permissions

expand all

Start typing a reply url to filter these results

Permission	Admin consent required
> App	
> Capacity	

c) Select the required permissions: in **Report**, select **Report.Read.All** permissions



Metadata

Report (1)

☑ Report.Read.All ⓘ
View all reports

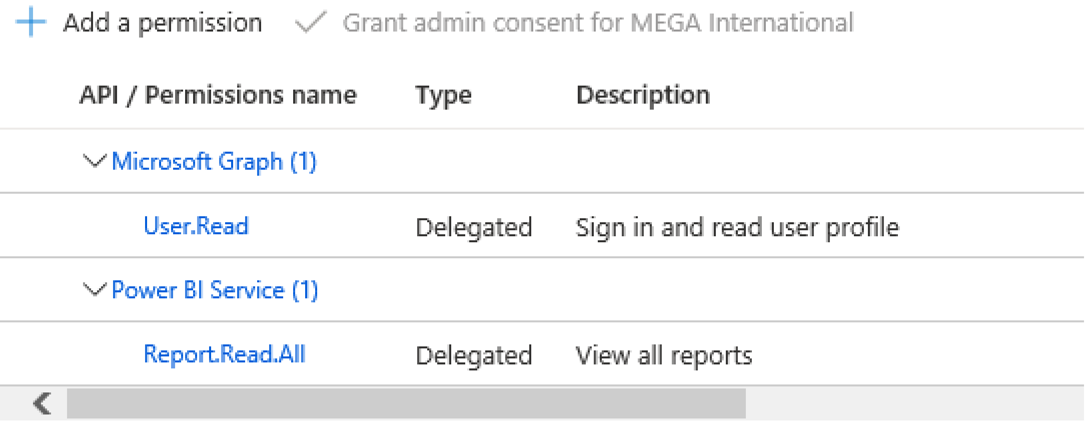
□ Report.ReadWrite.All ⓘ
Read and write all reports

StorageAccount

Tenant

Add permissions Discard

8) Click **Add permissions**.

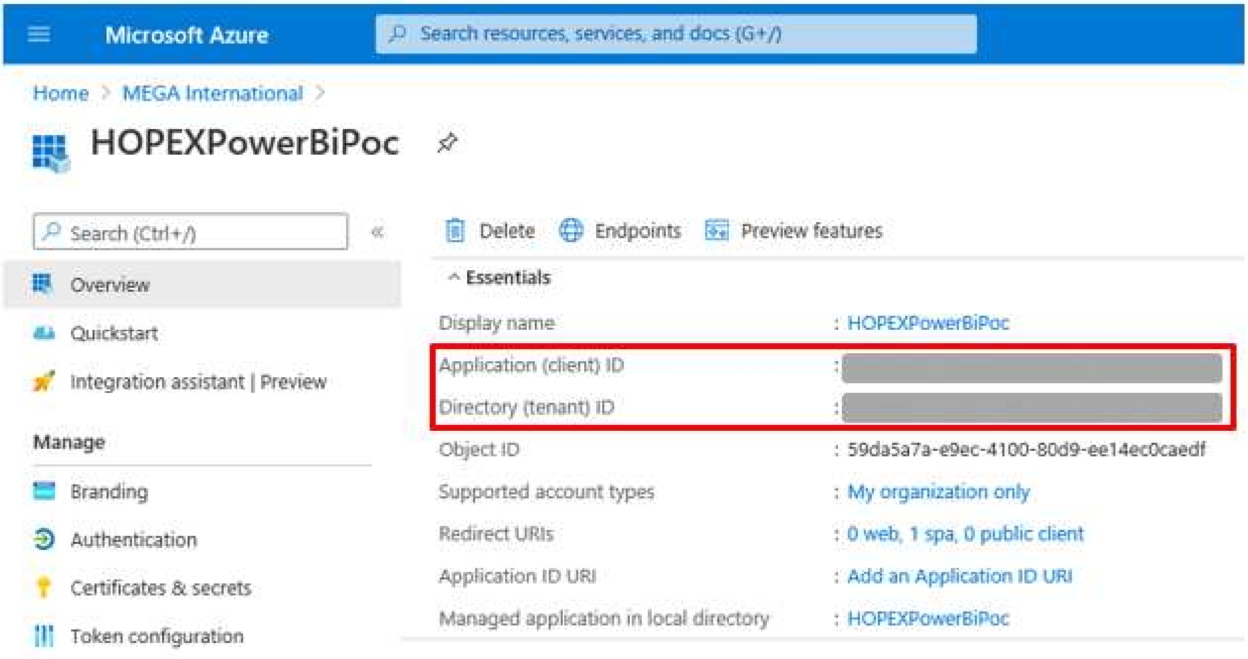


+ Add a permission ✓ Grant admin consent for MEGA International

API / Permissions name	Type	Description
Microsoft Graph (1)		
User.Read	Delegated	Sign in and read user profile
Power BI Service (1)		
Report.Read.All	Delegated	View all reports

9) In the left pane, select **Overview** pane and in the rigth pane take note of both IDs for later steps:

- the **Application (client) ID**
- the **Directory (tenant) ID**

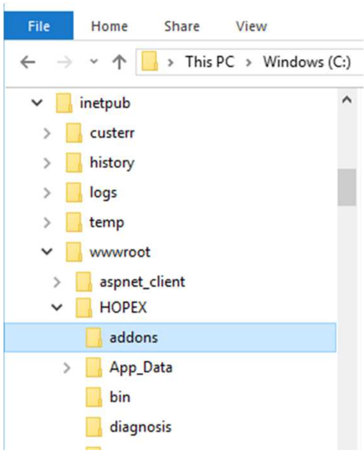


Display name : HOPEXPowerBiPoc
Application (client) ID : [redacted]
Directory (tenant) ID : [redacted]
Object ID : 59da5a7a-e9ec-4100-80d9-ee14ec0caedf
Supported account types : My organization only
Redirect URIs : 0 web, 1 spa, 0 public client
Application ID URI : Add an Application ID URI
Managed application in local directory : HOPEXPowerBiPoc

3.3. Creating the JavaScript wrapper

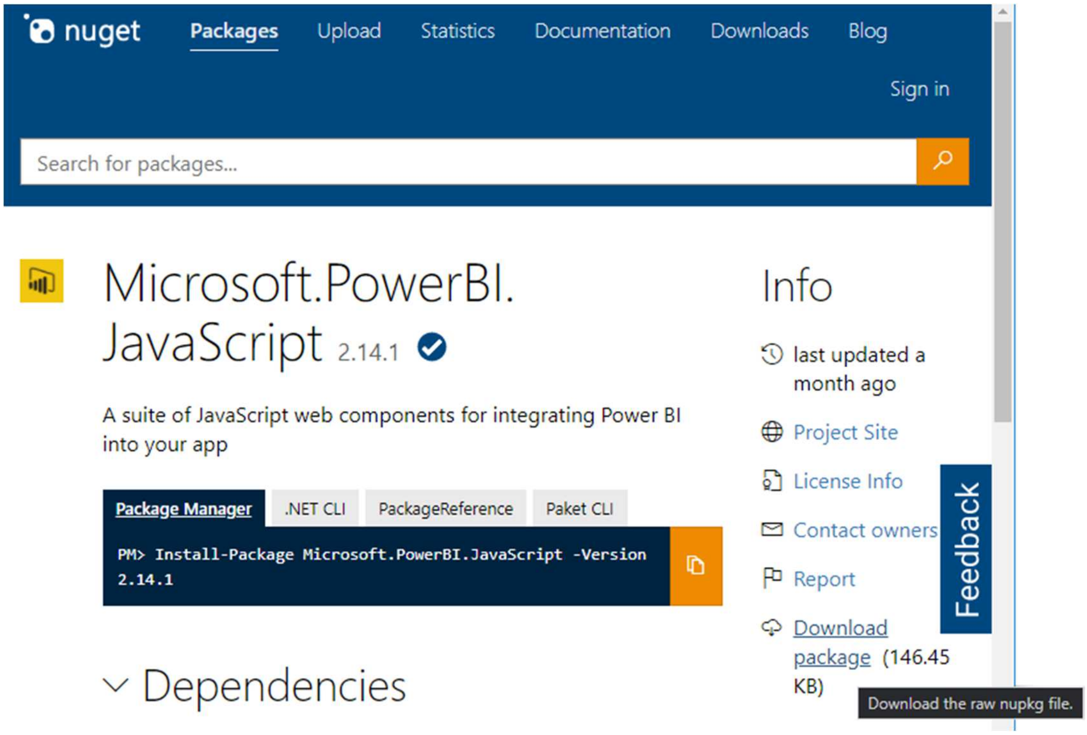
To create the JavaScript wrapper:

1) Access the **addons** folder: in the Hopex web application installation folder (usually c:\inetpub\wwwroot\HOPEX), select the **addons** folder (or create one if it does not already exist).



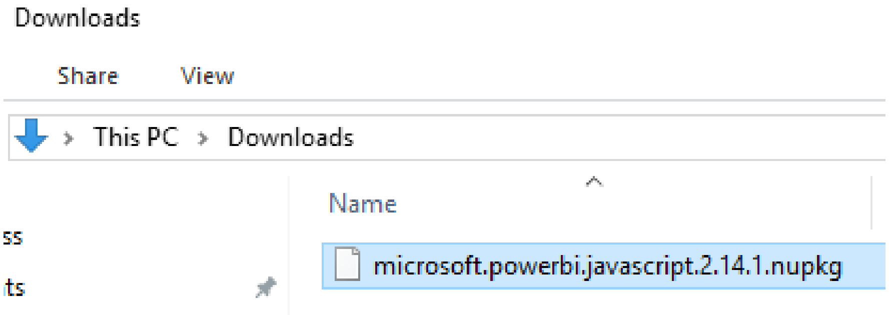
2) Retrieve the PowerBI JavaScript SDK:

a) Go to https://www.nuget.org/packages/Microsoft.PowerBI.JavaScript/.

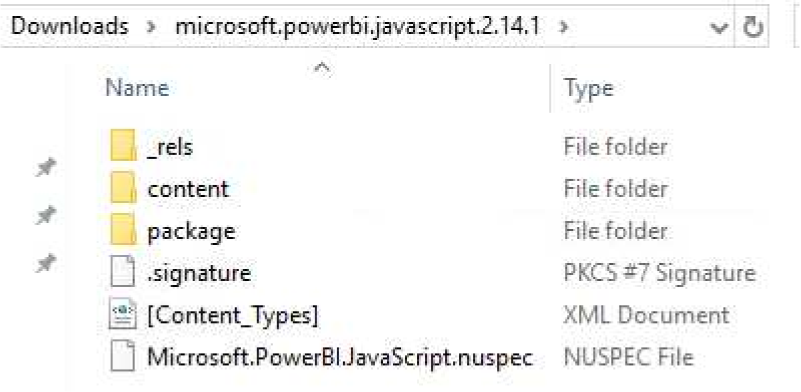


```
PM> Install-Package Microsoft.PowerBI.JavaScript -Version
2.14.1
```

b) In the **Info** pane, click **Download package**.



c) Open the downloaded file with WinZip (or similar utility). Rename the file with a zip extension if needed.


Downloads › microsoft.powerbi.javascript.2.14.1 ›

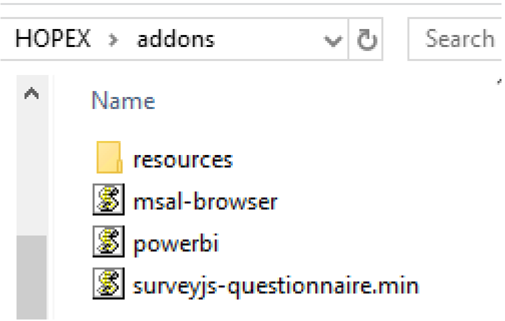
Name	Type
_rels	File folder
content	File folder
package	File folder
.signature	PKCS #7 Signature
[Content_Types]	XML Document
Microsoft.PowerBI.JavaScript.nuspec	NUSPEC File

d) Access the **content > scripts > powerbi.js** file and copy it in the **addons** folder.

3) Retrieve the Microsot Authentication Library for JS:

a) Download the file from https://alcdn.msauth.net/browser/2.1.0/js/msal-browser.js.

b) Copy it in the **addons** directory.


HOPEX › addons

Name

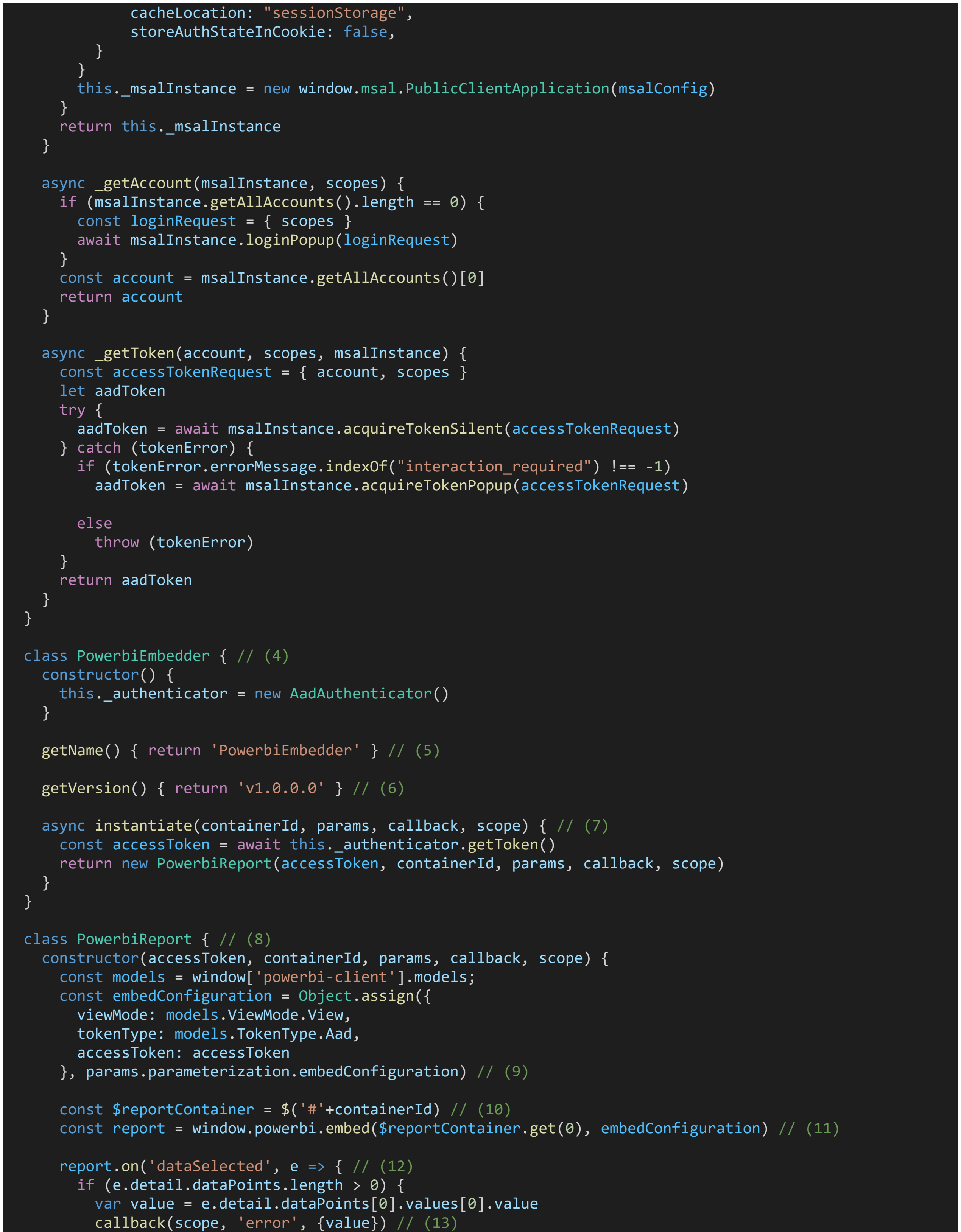
- resources
- msal-browser
- powerbi
- surveyjs-questionnaire.min

4) Create the Addon Factory:

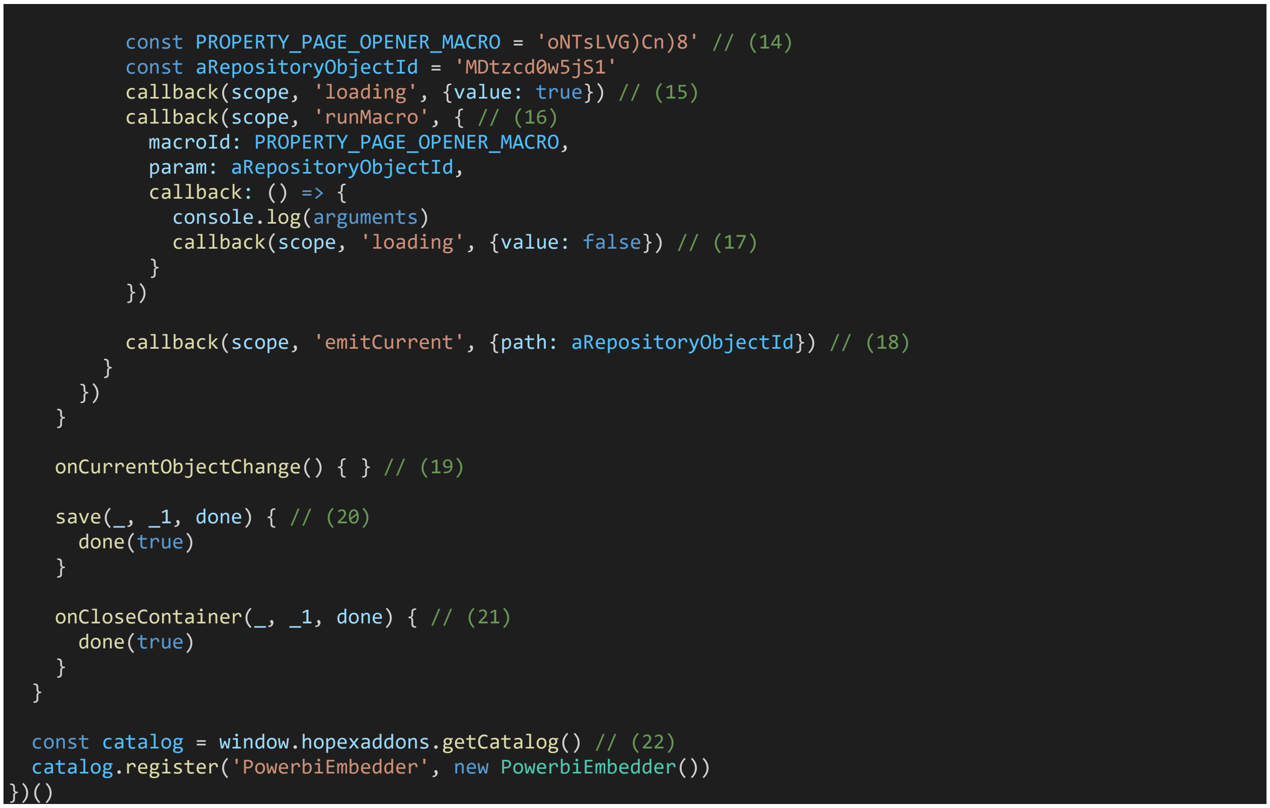
a) In the **addons** folder, create a file named “powerbi-hopex.js”.

b) Paste the following code.

```
(function(){
  class AadAuthenticator {  // (1)
    async getToken() {
      const msalInstance = this._getMsalInstance()
      const scopes = ['https://analysis.windows.net/powerbi/api/Report.Read.All']
      const account = await this._getAccount(msalInstance, scopes)
      let aadToken = await this._getToken(account, scopes, msalInstance)
      return aadToken.accessToken
    }

    _getMsalInstance() {
      if (!this._msalInstance) {
        const msalConfig = {
          auth: {
              clientId: '<your Application (client) ID>', // (2)
              authority: 'https://login.microsoftonline.com/<your directory (tenant) ID>/' // (3)
          },
          cache: {
```



```
            cacheLocation: "sessionStorage",
            storeAuthStateInCookie: false,
        }
      }
      this._msalInstance = new window.msal.PublicClientApplication(msalConfig)
    }
    return this._msalInstance
  }

  async _getAccount(msalInstance, scopes) {
    if (msalInstance.getAllAccounts().length == 0) {
      const loginRequest = { scopes }
      await msalInstance.loginPopup(loginRequest)
    }
    const account = msalInstance.getAllAccounts()[0]
    return account
  }

  async _getToken(account, scopes, msalInstance) {
    const accessTokenRequest = { account, scopes }
    let aadToken
    try {
      aadToken = await msalInstance.acquireTokenSilent(accessTokenRequest)
    } catch (tokenError) {
      if (tokenError.errorMessage.indexOf("interaction_required") !== -1)
        aadToken = await msalInstance.acquireTokenPopup(accessTokenRequest)

      else
        throw (tokenError)
    }
    return aadToken
  }
}

class PowerbiEmbedder { // (4)
  constructor() {
    this._authenticator = new AadAuthenticator()
  }

  getName() { return 'PowerbiEmbedder' } // (5)

  getVersion() { return 'v1.0.0.0' } // (6)

  async instantiate(containerId, params, callback, scope) { // (7)
    const accessToken = await this._authenticator.getToken()
    return new PowerbiReport(accessToken, containerId, params, callback, scope)
  }
}

class PowerbiReport { // (8)
  constructor(accessToken, containerId, params, callback, scope) {
    const models = window['powerbi-client'].models;
    const embedConfiguration = Object.assign({
      viewMode: models.ViewMode.View,
      tokenType: models.TokenType.Aad,
      accessToken: accessToken
    }, params.parameterization.embedConfiguration) // (9)

    const $reportContainer = $('#'+containerId) // (10)
    const report = window.powerbi.embed($reportContainer.get(0), embedConfiguration) // (11)

    report.on('dataSelected', e => { // (12)
      if (e.detail.dataPoints.length > 0) {
        var value = e.detail.dataPoints[0].values[0].value
        callback(scope, 'error', {value}) // (13)
```



```
          const PROPERTY_PAGE_OPENER_MACRO = 'oNTsLVG)Cn)8' // (14)
          const aRepositoryObjectId = 'MDtzcd0w5jS1'
          callback(scope, 'loading', {value: true}) // (15)
          callback(scope, 'runMacro', { // (16)
            macroId: PROPERTY_PAGE_OPENER_MACRO,
            param: aRepositoryObjectId,
            callback: () => {
              console.log(arguments)
              callback(scope, 'loading', {value: false}) // (17)
            }
          })

          callback(scope, 'emitCurrent', {path: aRepositoryObjectId}) // (18)
        }
      })
    }

    onCurrentObjectChange() { } // (19)

    save(_, _1, done) { // (20)
      done(true)
    }

    onCloseContainer(_, _1, done) { // (21)
      done(true)
    }
  }

  const catalog = window.hopexaddons.getCatalog() // (22)
  catalog.register('PowerbiEmbedder', new PowerbiEmbedder())
})()
```

c) **(Important)** Replace the following IDs with the ones retrieved in the App registration steps:

- **ClientId** (2) with the **Application (client) ID**
- part of the **authority** url (3) with the **Directory (tenant) ID**

The following sections describe what are the several parts of this sample code.

Registration of an Addon Factory

All the js files placed in the addons folder will be automatically loaded with the HOPEX desktop. The loading order is not guaranteed.

You need to register in Hopex a factory object which will be called by the Mega Parameterized tool we have defined. This is done by calling hopexaddons.getCatalog() (22) ; hopexaddons is available globally. Then, call register() on the catalog. The first parameter must match exactly, in a case-sensitive manner, the name put in the factory settings of the Mega Parameterized tool in page 12.

The second parameter of the register method must be an object implementing several methods. In our case, we pass an instance of the class PowerBiEmbedder we defined at (4).

Implementation of the Addon Factory

The PowerBiEmbedder implements getName (5), getVersion (6) and instantiate (7)

The most important method is instantiate (7):

- It must return a Promise (or be async)
- It Is responsible for doing the integration of the report in the DOM (11). It is given as parameters :
  - params: an object containing all the settings made on the "Mega Parameterized Tool" (9)
  - containerId: the id of the div which can be used to mount the content (10)
- Note that in our sample code, we have delegated this responsibility to another class constructor
  - A callback and a scope to perform action on the rest of the Hopex desktop
- It must return another object representing the embedded content. This is our PowerBiReport class (8). This allows the HOPEX to callback this component to trigger some lifecycle events.

Implementation of the Addon Instance

This returned object (instance of our PowerBIReport class) must implement:

- onCurrentObjectChange (19) to be notified when the user a repository object anywhere else in the desktop
- save (20) to be notified when the desktop requires the component to save their states. The component must acknowledge they have done so by calling done(true), even if they have no state to save.
- onCloseContainer (21) to be notified when the desktop wants to remove the component. Like for save, the component must acknowledge it is ready to be removed from the DOM by calling done(true)

Interactions with the Hopex Desktop

To perform actions on the Hopex desktop, you must use the callback and scope given by the instantiate methods. In this sample, we are reacting to an event emitted the Power BI SDK when a data point is clicked in the report (12) and performs 3 actions:

- Sending a notification to the desktop (13)
- Opening a property page in a popup (14)
  - As the action takes some times to complete, we first add a "Loading" mask on top of the report (15)
  - We then run a Macro defined in the back-end which is responsible for triggering the display of the property page (16)
  - When the property page is closed, we removed the "Loading" mask (17)

- We change the “Current Object” of the desktop (18). This triggers the refresh of other components in the desktop (like a docked property page) to display data about this objects

For more information on Microsoft specific SDKs

- For authentication (1): https://github.com/AzureAD/microsoft-authentication-library-for-js/tree/dev/lib/msal-browser
- For Power BI Embedding (11): https://github.com/Microsoft/PowerBI-JavaScript/wiki

3.4. Content-Security-Policy tuning

Depending on HOPEX versions, for security reasons, loading of external iframe and connecting to external APIs might be disabled or generate a warning.

To authorize Power Bi to edit the Web.Config file of the HOPEX IIS application:

1) Accces the Web.config file.
2) If present, change both the **Content-Security-Policy-Report-Only** and **Content-Security-Policy** headers, add:
    - a frame-src directive specifying the Power BI host source
    - a connect-src directive for the authentication service.

```
<configuration>
  <system.webServer>
    <httpProtocol>
      <customHeaders>
        <add name="Content-Security-Policy-Report-Only"
value="block-all-mixed-content; frame-src 'self' https://app.powerbi.com;
connect-src 'self' https://login.microsoftonline.com; default-src 'self'; script-src 'self' 'unsafe-eval' 'unsafe-inline'; style-src 'self' 'unsafe-inline'; img-src 'self' data:"/>
        <add name="Content-Security-Policy"
value="frame-src 'self' https://app.powerbi.com;
connect-src 'self' https://login.microsoftonline.com;"/>
```

3.5. Testing

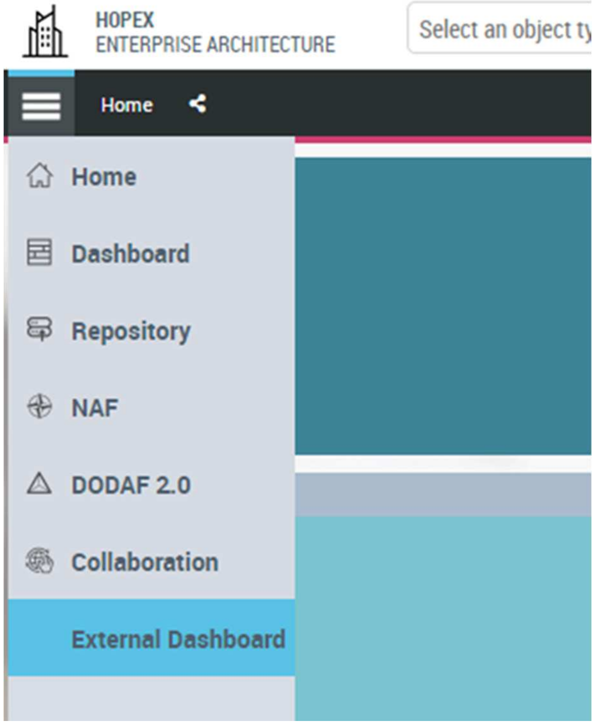
To test your customization:

1) Connect to HOPEX (Web Front-End) with the profile corresponding to the Working Environment Template.

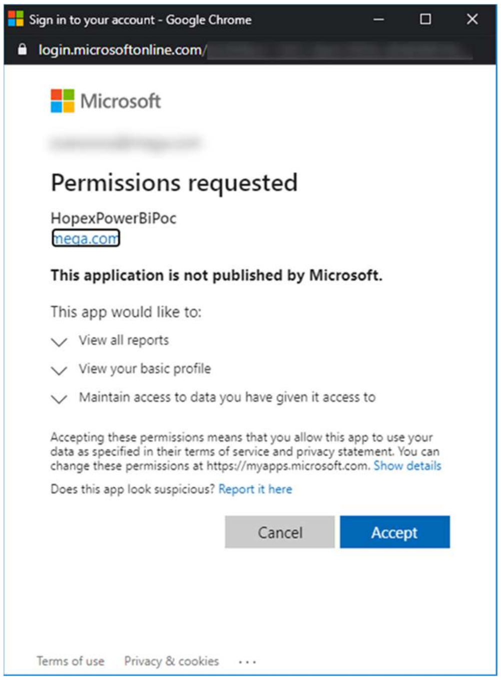
```
E.g.: Enterprise Architect.
```

2) Click the **Navigation** menu.

The **External Dashbord** navigation pane (Environment Group Template) you created is available.



You might then be asked to login with you AD account, as well as for consent to give HOPEX access to the report.



Sign in to your account - Google Chrome
login.microsoftonline.com/
Microsoft

Permissions requested

HopexPowerBiPoc
mega.com

This application is not published by Microsoft.

This app would like to:

- View all reports
- View your basic profile
- Maintain access to data you have given it access to

Accepting these permissions means that you allow this app to use your data as specified in their terms of service and privacy statement. You can change these permissions at https://myapps.microsoft.com. Show details

Does this app look suspicious? Report it here

The report is displayed in the center panel.

CONFIGURING NAVIGATION TREES

The **HOPEX** workspace is organized around navigation windows, such as **Home**, **Projects** and **Main Objects**. These windows enable access via a navigation tree to information relevant in a specific context.

Studio enables creation and modification of navigation trees.

The following points are covered in this chapter:

CONCEPT AND DEFINITIONS

Concepts used for creation of navigation trees are introduced in the following sections:

- Navigation Window Content
- Navigation Tree Structure
- Concepts Overview

Navigation Window Content

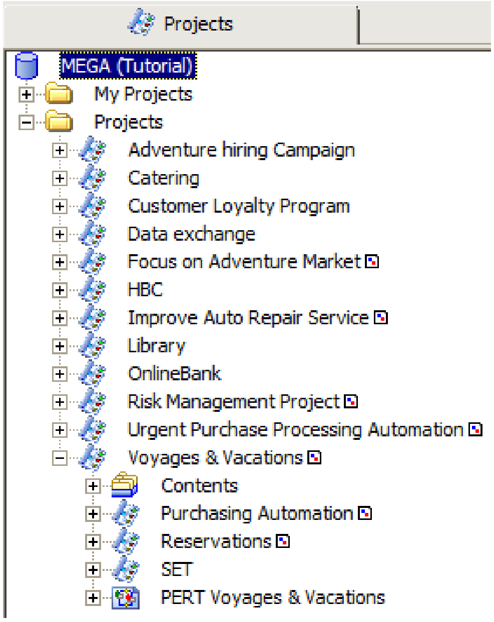
Navigation windows proposed in the **HOPEX** workspace enable hierarchical access to repository objects via a navigation tree.

A navigation tree comprises navigable branches which can group:

- **HOPEX** objects
- Folders
- Classification folders

To view these possibilities:

1. In the **HOPEX** workspace, select **View > Navigation Windows > Projects**.
   The following window appears on the left of your workspace.



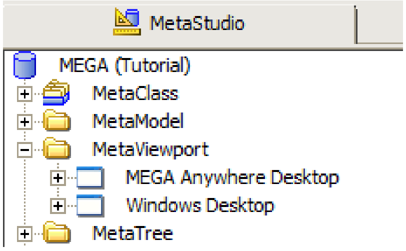
2. Expand the branch of the **Projects** folder.
   Note that sub-branches are objects of **Project** type.

   ☛ *The sub-branches of a navigation tree can contain folders or classification folders.*

Navigation Tree Structure

To access the structure of a navigation tree:

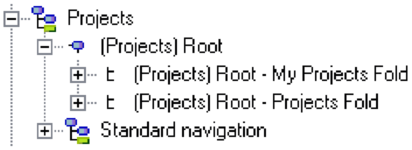
1. In **HOPEX** menu bar, select **View > Navigation Windows > MetaStudio**.
   Most of existing navigation trees are grouped in folders accessible from the **MetaViewport** folder. Trees not attached to any context are in the **MetaTree** folder.


MetaStudio
MEGA (Tutorial)
MetaClass
MetaModel
MetaViewport
MEGA Anywhere Desktop
Windows Desktop
MetaTree

2. Expand the **MetaViewport** folder.
   A list of the different contexts in which existing navigation trees can be used appears.

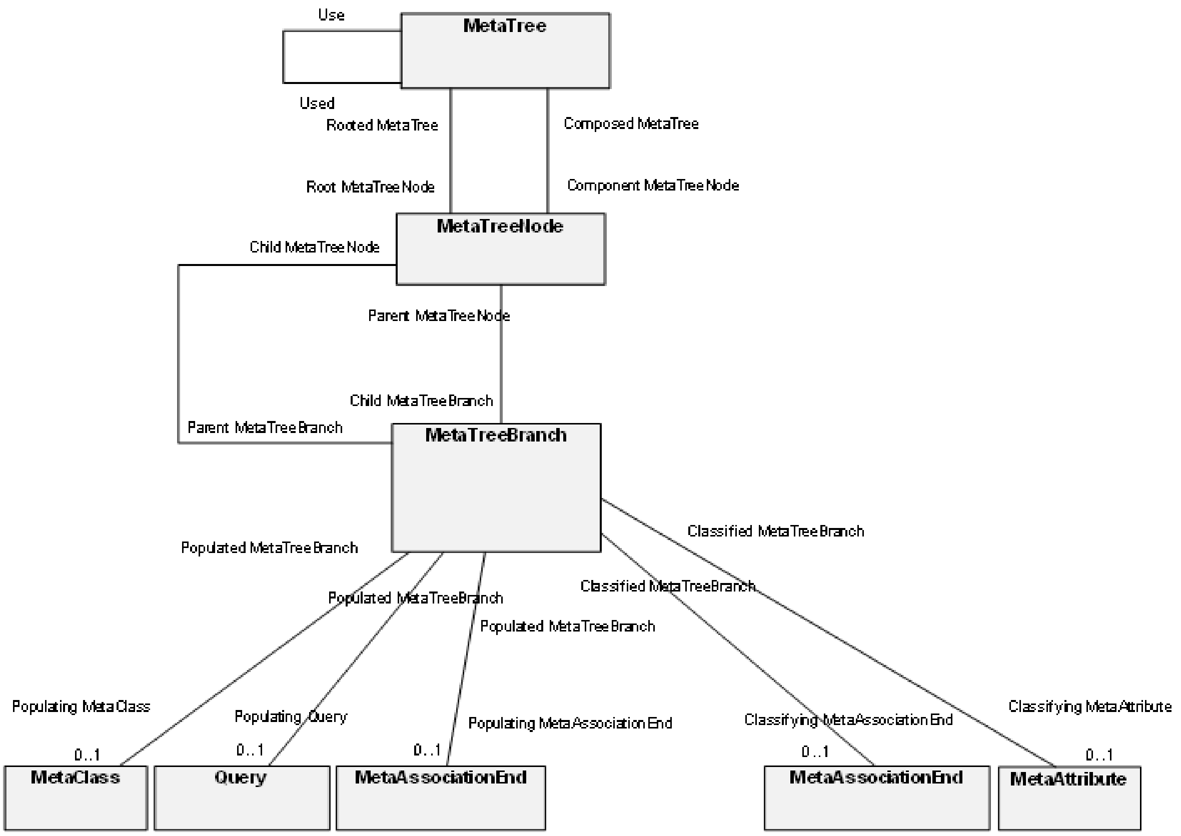
   ☛ *MetaViewport is a MetaClass. It is installed and filtered at extraction of products.*

3. Expand the **Windows Desktop** folder.
   The list of existing navigation trees in your **HOPEX** workspace appears.
4. Expand for example the **Projects** tree, then the **(Projects) Root** node.
   The main branches of the **Projects** navigation window appear:
   - "(Projects) Root - My Projects Fold"
   - "(Projects) Root - Projects Fold"


Projects
(Projects) Root
(Projects) Root - My Projects Fold
(Projects) Root - Projects Fold
Standard navigation

Concepts Overview

Navigation tree configuration uses concepts presented in the following metamodel.



The "MetaTree" MetaClass represents the navigation tree.

The "MetaTreeNode" MetaClass represents the tree nodes. Since a node is required for creation of a branch, each MetaTree has a root node from which main branches are created.

The "MetaTreeBranch" MetaClass represents the tree branches. A branch is systematically connected to a node by the "Parent MetaTreeNode" MetaAssociationEnd. If it includes sub-branches, it is associated with a child node by the "Child MetaTreeNode" MetaAssociationEnd.

The "Populating MetaClass", "Populating Query" and "Populating MetaAssociationEnd" MetAssociationEnds enable definition of content of the branch.

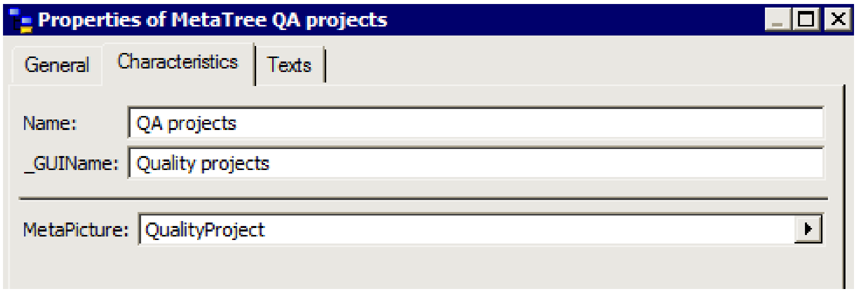
The "Classifying Attribute" and "Classifying MetaAssociationEnd" MetaAssociationEnds enable definition of classification criteria of branches associated with classification folders.

CREATING A NAVIGATION TREE

Creating the object associated with the navigation tree

To create a navigation tree:

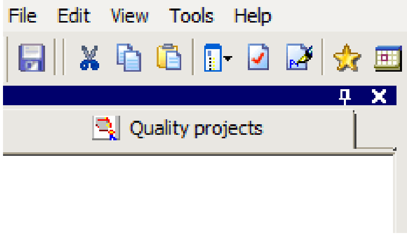
1. In the **MetaStudio** navigation window, expand the **MetaViewport** folder.
2. In the **MetaViewport** folder, right-click **Windows Desktop** and select **New > MetaTree**.
3. In the creation dialog box, enter the name of the new **MetaTree** and click **OK**.
4. From the Properties window of the newly-created MetaTree, specify the name and picture characteristics that will be associated with the tree in the navigation window.



Name: QA projects
_GUIName: Quality projects
MetaPicture: QualityProject

To see the result:

1. Exit and restart **HOPEX.**
2. In the **HOPEX** workspace, select **View > Navigation Windows >** "Name of new MetaTree"**.**



Creating the root node

The root node of a navigation tree is required for creation of main branches of the tree.

To create the root node:

1. Right-click the MetaTree and select **New > MetaTreeNode**.
2. In the dialog box that opens, enter the name of the new node and click **OK**.

   ☛ *The convention adopted for naming the root node is: (<MetaTree Name>) Root.*

   Example: (Project) Root.

   You can create branches from this root node.

CREATING NAVIGATION TREE BRANCHES

Studio enables creation of branches of which content can be:

- Objects
- Folders
- Classification folders.

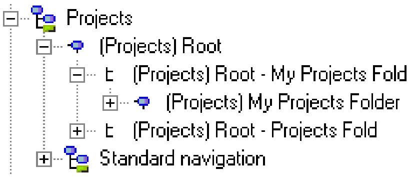
This section covers creation of the different branch categories and introduces the filter concept.

- Creating a Branch Corresponding to a Folder
- Creating a Branch Containing an Object List
- Creating Classification Folder Branches
- Filtering Branch Content.

In extended mode, you can sort objects. This functionality is described in section Sorting Content of a Branch.

Creating a Branch Corresponding to a Folder

The **My Projects** branch of the **Projects** navigation window corresponds to the "(Project) Root - My Projects Fold" branch of the "Projects" tree. This branch itself contains other branches, of which certain are folders.


Projects
(Projects) Root
(Projects) Root - My Projects Fold
(Projects) My Projects Folder
(Projects) Root - Projects Fold
Standard navigation

Creating a branch of folder type

To add a new branch to a navigation tree:

1. In the **MetaStudio** navigation window, expand the **MetaViewport** folder.
2. Position on the navigation tree concerned, for example "Projects".
3. Right-click the parent node of the branch, and select **New > MetaTreeBranch**.
4. In the dialog box that appears, enter the name of the new branch and click **OK**.

☛ *The convention adopted for naming a MetaTreeBranch of folder type is:*

(<MetaTree Name>) <Parent Node Name> - <Folder Name> Fold.

Example: (Project) Root - My Projects Fold.

Creating a node from a branch

A branch containing a child node carries the name of this node.

To create a child node from a branch:

1. Right-click the branch that interests you and select **New > MetaTreeNode**.
2. In the dialog box that opens, enter the name of the new node and click **OK**.

   ☛ *The convention adopted for naming a MetaTreeNode in a branch of folder type is:*

   (<MetaTree Name >) <Folder Name > Folder.

   Example: (Project) Root My Project Folder.
3. Create branches describing the content of your new branch of folder type, see Creating a Branch Containing an Object List.

   ☛ *If no sub-branch is defined for a branch of folder type, no sub-branch will appear in the navigation window.*

Creating a Branch Containing an Object List

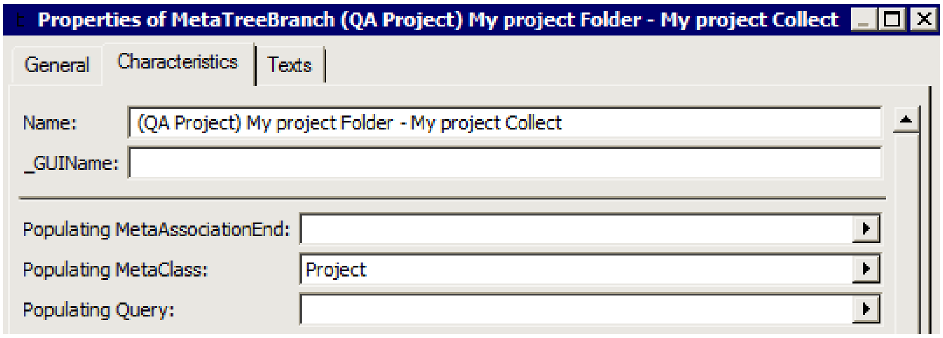
To create a branch containing a collection of objects of a given MetaClass:

1. Right-click the parent node and select **New > MetaTreeBranch**.
2. In the dialog box that appears, enter the name of the new branch and click **OK**.

   ☛ *The convention adopted for naming a MetaTreeBranch containing an object list is:*

   (<MetaTree Name >) <Parent Node Name > - <Category Name > Collect.

   Example: (Project) My Project Folder - My Project Collect.
3. Expand the node corresponding to the folder and in the Properties window of the newly-created MetaTreeBranch, specify the MetaClass of the objects it contains.
   In the example below, this is the "Project" MetaClass.



Properties of MetaTreeBranch (QA Project) My project Folder - My project Collect

Name: (QA Project) My project Folder - My project Collect

_GUIName:

Populating MetaAssociationEnd:

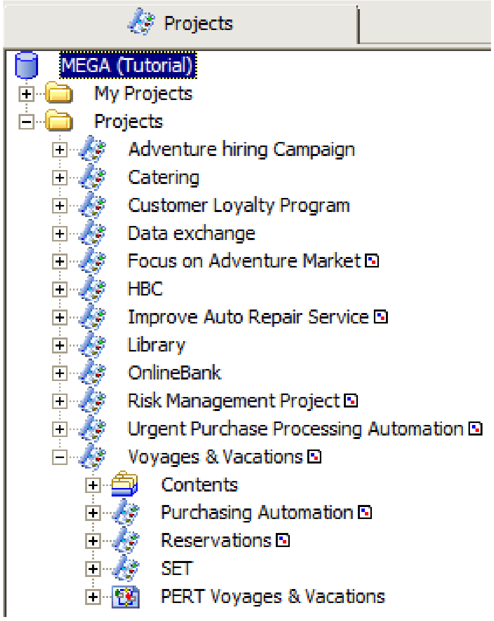
Populating MetaClass: Project

Populating Query:

To see the result:

1. From **View > Navigation Windows** , select the **MetaStudio** navigation window.

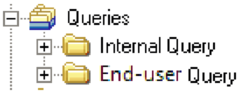
2. Expand the folders **MetaViewport**, **Windows Desktop**, then the tree you have created.
   Note that in this example, the branch created contains the list of objects of Project MetaClass.



Creating Classification Folder Branches

The **Utilities** navigation window contains a **Queries** branch.

- Expand this branch.
  Two folders corresponding to query stereotype values appear:
  - Internal Query
  - End-user Query



Queries can also be classified in specific folders.

Creating a classification folder

Classification criteria are specified from the value of an attribute or a MetaAssociationEnd.

For example, to classify your projects as a function of their type, you can use the "Project Type" MetaAssociationEnd.

Names assigned to the different folders correspond with the different values taken by the attribute or MetaAssociationEnd.

To classify objects of a folder:

1. Create a MetaTreeBranch from the MetaTreeNode corresponding to the parent folder, enter its name and click **OK**.

   ☛ *The convention adopted for naming a MetaTreeBranch intended for classification is:*

   (<MetaTree Name >) <Folder Name > - <Classification Criterion Name > Classify.

   Example: (Object) Root - Project Type Classify.

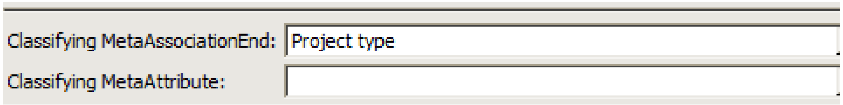
2. Create a MetaTreeNode from the new branch and click **OK**.

   ☛ *The convention adopted for naming a MetaTreeNode enabling classification of objects is:*

   (<MetaTree Name >) <Folder Name > - <Classification Criterion Name > Classify.

   Example: (Object) Root - Project Type Classify.

3. Open the Properties window of the branch created for the classification and specify the classification criteria.
4. In the Classifying MetaAssociationEnd box, enter for example "Project Type".



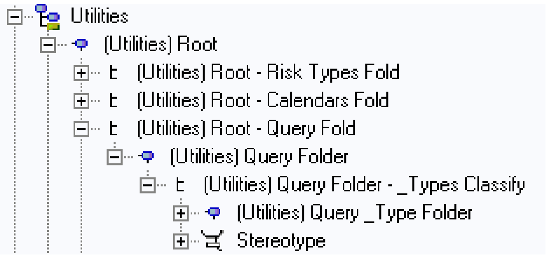
Classifying MetaAssociationEnd: Project type

Classifying MetaAttribute:

Defining content of classification folder branches

This step consists of defining branches associated with each of the classification folders as a function of values taken by the MetaAttribute or MetaAssociationEnd.

New branches are created from the "xxx Classify" node using the same principle as described in section Creating Navigation Tree Branches.



```
Utilities
  (Utilities) Root
    (Utilities) Root - Risk Types Fold
    (Utilities) Root - Calendars Fold
    (Utilities) Root - Query Fold
      (Utilities) Query Folder
        (Utilities) Query Folder - _Types Classify
          (Utilities) Query _Type Folder
          Stereotype
```

Storing classification folders in a main folder

In the **Utilities** navigation window, queries are classified by the value of their "Stereotype" attribute value only.

If you want to add another classification folder, for example as a function of query implementation mode, the **Queries** folder becomes "main folder" of the two classification folders:

- query by stereotype.
- query by implementation mode.

To store classification folders in a main folder:

1. Create a MetaTreeBranch from the MetaTreeNode corresponding to the parent folder, enter its name and click **OK**.

   ☛ *The convention adopted for naming a MetaTreeBranch intended for grouping classification folders is:*

   (<MetaTree Name >) <Folder Name > - <Classification Criterion Name > Classify.

   Examples:

   (Utilities) Query Folder - Stereotype Classification.

   (Utilities) Query Folder - Implementation Mode Classification.

2. Create a MetaTreeNode from the new branch and click **OK**.

   ☛ *The convention adopted for naming a MetaTreeNode enabling classification of objects is:*

   (<MetaTree Name >) <Folder Name > - <Classification Criterion Name > Classify.

   Examples:

   (Utilities) Query Folder - Stereotype Classification.

   (Utilities) Query Folder - Implementation Mode Classification.

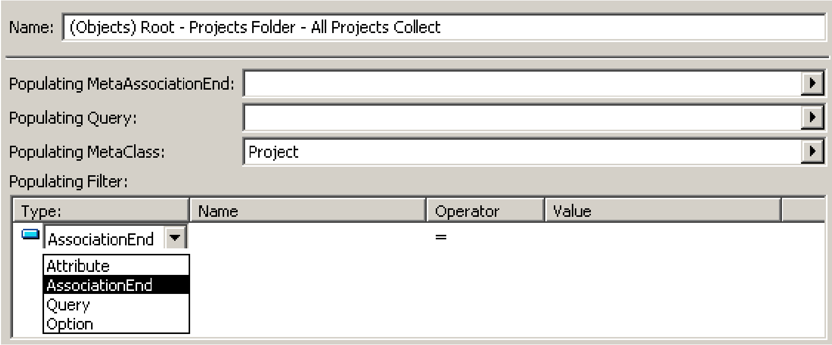
3. Build classification sub-folders from this node.

Filtering Branch Content

A filter enables reduction of the number of objects presented in a branch using criteria defined from a MetaAttribute, a MetaAssociationEnd or a query.

To create a filter on a MetaTreeBranch:

1. Open the Properties window of the MetaTreeBranch that interests you, for example "(Projects) My Project Folder - My Projects Collect".
2. Right-click in the **Populating Filter** space and select **Add** to create a new filter.
3. In the **Type** field of the filter created, select the type of filter you want to use.
   For example, if you want to create a filter on projects that do not have a parent project, use the "Parent Project" MetaAssociationEnd.



Name: (Objects) Root - Projects Folder - All Projects Collect

Populating MetaAssociationEnd:

Populating Query:

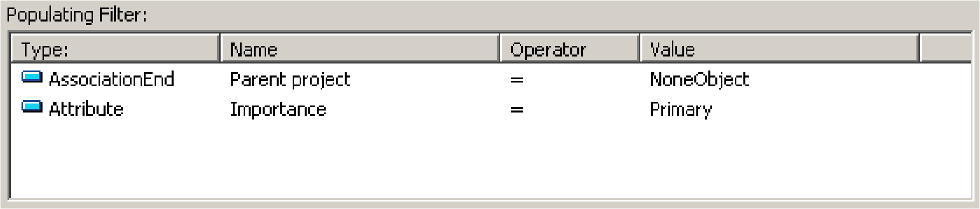
Populating MetaClass: Project

Populating Filter:

Type:	Name	Operator	Value
AssociationEnd		=	

4. In the **Name** field, select the identifier that interests you, for example "Parent Project".

5. Select an **Operator**, then a **Value**.
   In the same way you can add a second filter criterion, for example the "Importance" attribute of the project.



Populating Filter:

Type:	Name	Operator	Value
AssociationEnd	Parent project	=	NoneObject
Attribute	Importance	=	Primary

To see the result:

- Close and reopen the navigation window.

Configuring Navigation

Navigation in a tree depends on configuration defined at the level of:

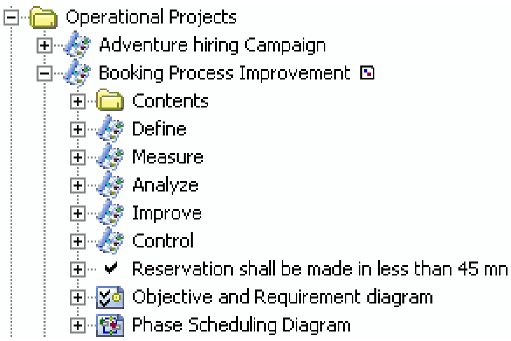
- Navigation possibilities of the objects presented in the tree
- The nature of the tree itself
- Navigation conditions defined in advanced parameters.

This section describes mechanisms provided for:

- Configuring Standard Navigation in a MetaClass
- Navigating from Tree Objects.

Configuring Standard Navigation in a MetaClass

In the absence of specific configuration for a tree, navigation from a given MetaClass always respects the same presentation.



- Operational Projects
  - Adventure hiring Campaign
  - Booking Process Improvement
    - Contents
    - Define
    - Measure
    - Analyze
    - Improve
    - Control
    - Reservation shall be made in less than 45 mn
    - Objective and Requirement diagram
    - Phase Scheduling Diagram

Standard navigation from an object is configured by creation of:

- Branches associated with object collections
- Filters defined on these object collections
- Branches associated with folders.

Configuring navigation for a MetaClass

The list of MetaClasses accessible from an object of a given type is fixed based on the MetaAssociationEnds defined for the MetaClass of the object.

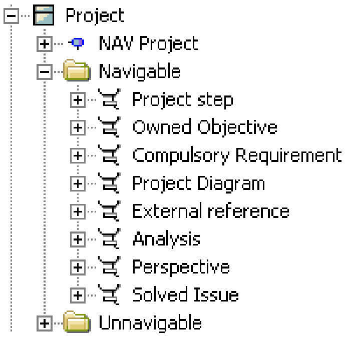
To obtain the list of navigable objects:

1. In the **MetaStudio** navigation window, expand the **MetaTree** folder
2. Under the **MetaTree** folder, expand the **MetaClass Navigation** folder.

3. Expand the folder corresponding to the MetaClass that interests you.

   `Example: "Project".`

   - The "Navigable" folder contains the list of navigable MetaAssociationEnds.
   - The "Unnavigable" folder contains the list of MetaAssociationEnds that are not navigable.



To specify that a MetaAssociationEnd is navigable, there are two possible solutions:

- In the "Unnavigable" folder, select the MetaAssociationEnd that interests you and drag this into the "Navigable" folder.
- From the Properties window of the MetaAssociationEnd concerned, select the **Characteristics** tab and in the **Navigable** box, select "Navigable".

Defining a branch from a MetaClass

To show branches in the navigation tree of an object:

1. In the **MetaStudio** navigation window, expand the **MetaTree** folder
2. Under the **MetaTree** folder, expand the **MetaClass Navigation** folder.
3. Right-click the MetaClass concerned, for example Project, and select **New > MetaTreeNode**.
4. In the dialog box that appears, enter the name of the new node and click **OK**.

   ☛ *The convention adopted for naming this root node specific to a MetaClass is:*

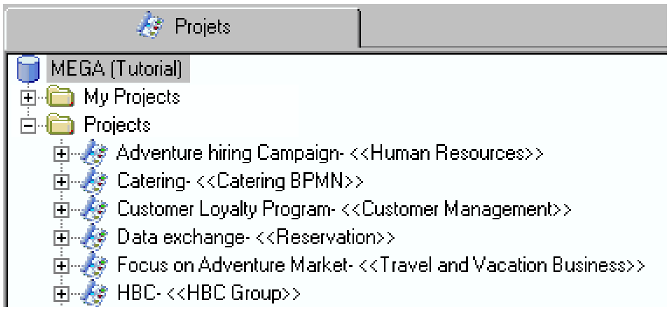
   (<MetaTree Name >) <MetaClass Name > Root.

5. Create branches and filters of this new node according to the procedures described in sections Creating Navigation Tree Branches and Creating Classification Folder Branches.

Managing object titles in navigation trees

Objects in navigation trees are presented by their name. This name generally corresponds to the "Short Name" attribute of the object.

If you wish project titles always to comprise the name of the project followed by the name of its library, you can modify this behavior via the MetaField concept.



MEGA (Tutorial)
My Projects
Projects
Adventure hiring Campaign- <<Human Resources>>
Catering- <<Catering BPMN>>
Customer Loyalty Program- <<Customer Management>>
Data exchange- <<Reservation>>
Focus on Adventure Market- <<Travel and Vacation Business>>
HBC- <<HBC Group>>

Result of change of project titles in a navigation tree

The MetaField MetaClass enables redefinition of titles of objects of a MetaClass:

- from the value of an object attribute, for example its name.
- from the name of the object referenced by a MetaAssociationEnd, for example the owner library.

All components of the new title, attribute or MetaAssociationEnd, are associated with a MetaField. They are referenced by the main MetaField which is attached to the MetaTreeNode of the MetaClass.

Modifying title of a MetaClass

To modify titles of objects of a MetaClass, you must start by creating the main MetaField.

To modify title of a MetaClass:

1. In the **MetaStudio** navigation window, expand the **MetaTree** folder, then the **MetaClass Navigation** folder.
2. Expand the folder of the MetaClass that interests you (for example: Project).
3. Right-click the MetaTreeNode of the MetaClass (for example: NAV Project) and select **Explore**.
   The explorer window opens.
4. Display empty collections.
5. Right-click the "MetaTreeNodeName" folder and select **New**.
   The MetaField creation dialog box opens.

   ☛ *You must be authorized to modify MEGA data.*
6. Specify the name of the MetaField and click **OK**.

   ☛ *The convention adopted for naming the main MetaField is:*

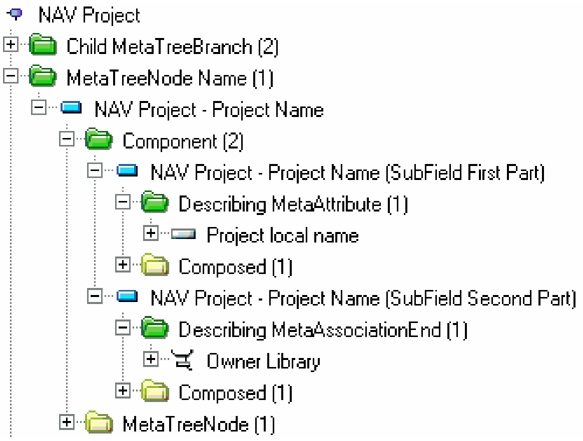
   <MetaTree Name >) <MetaClass Name > Name.

   Example: NAV Project - Project Name

If your new title is built from a unique attribute or a single MetaAssociationEnd, you need only reference it in the MetaField you have just created. For more details, see Defining content of a title.

Creating a composite title

If the new title is composed of several attributes or references, you must create a MetaField per component and add a reference to its content. Each component MetaField is created from the main MetaField.



```
NAV Project
⊞ Child MetaTreeBranch (2)
⊟ MetaTreeNode Name (1)
   ⊟ NAV Project - Project Name
      ⊟ Component (2)
         ⊟ NAV Project - Project Name (SubField First Part)
            ⊟ Describing MetaAttribute (1)
               ⊞ Project local name
            ⊞ Composed (1)
         ⊟ NAV Project - Project Name (SubField Second Part)
            ⊟ Describing MetaAssociationEnd (1)
               ⊞ Owner Library
            ⊞ Composed (1)
      ⊞ MetaTreeNode (1)
```

Result of exploring a standard MetaTreeNode with modification of its title

To create a component MetaField:

1. Right-click the main MetaField (for example: "NAV Project - Project Name") and select **Explore**.
   An explorer window opens.
2. Display empty collections.
3. Right-click the "Components" folder and select **New**.
   The MetaField creation dialog box opens.
4. Specify the name of the sub-MetaField and click **OK**.

   ☛ *The convention adopted for naming a MetaField component is:*

   <MetaTree Name >) <MetaClass Name > Name (SubField <order> Part).

   Example: NAV Project - Project Name (SubField First Part).
5. Similarly create a MetaField per component of the new title.

Defining content of a title

The new title can be built:

- from the value of an attribute.
- from the name of an object referenced by a MetaAssociationEnd.

To specify that a MetaField is associated with an attribute:

1. Right-click the MetaField that interests you (for example: "NAV Project - Project Name (SubField First Part)"), and select **Explore**.
   An explorer window opens.
2. Open the Properties window of the MetaClass associated with the MetaTreeNode and select the **MetaAttribute** tab.

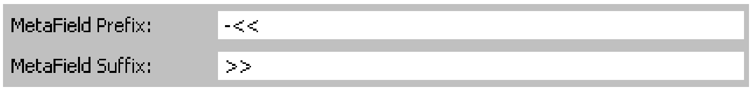
3. Select the attribute you wish to use for the title of objects of the MetaClass, and drag it to the **Describing MetaAttribute** folder of the MetaField explorer window.

   ☛ *To specify that a MetaField is associated with a MetaAssociationEnd, proceed in the same way, dragging the MetaAssociationEnd to the* ***Describing MetaAttribute*** *folder of the MetaField explorer window.*

Adding text within a MetaField

To add text within a MetaField:

1. Open the Properties window of the MetaField.
2. Complete boxes **MetaField Prefix** and **MetaField Suffix** with character strings that you wish to add before or after referenced content.



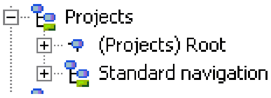
MetaField Prefix: -<<
MetaField Suffix: >>

Navigating from Tree Objects

By default, the objects presented in navigation tree branches are not navigable.

So that all objects presented in a tree will be navigable:

1. In the **MetaStudio** navigation window, expand the **MetaTree** folder.
2. Right-click the MetaTree concerned and select **Connect > MetaTree**.
3. In the dialog box that opens, enter the name of the standard MetaTree **Standard Navigation** and click **OK**.
   As an example, the result is presented below.



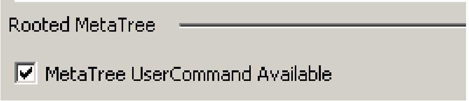
Using Advanced Functions

The following configuration functions are available to a user with extended access to the metamodel.

Associating Commands with the Tree Root Node

So that commands will be available form the top of the tree in the navigation window:

- Open the Properties window of the new node and select the option **MetaTree User Command Available**.



Rooted MetaTree

☑ MetaTree UserCommand Available

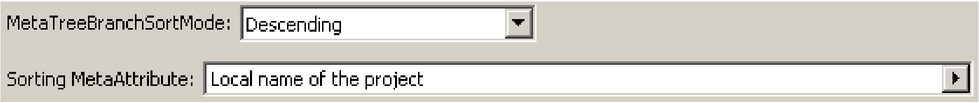
Sorting Content of a Branch

By default, objects contained in a branch are presented in ascending alphabetical order.

Studio allows you to modify this presentation. To do this, you must specify the main attribute for sorting, as well as order of presentation (ascending or descending) of elements of the list.

For example, so that a list of "Project" type objects will be displayed in descending alphabetical order:

1. Open the Properties window of the branch concerned.
2. In the **Sorting MetaAttribute** field, select the MetaAttribute you want to use for sort.
3. In the **MetaTreeBranchSortMode** box, select the order of presentation.



MetaTreeBranchSortMode: Descending

Sorting MetaAttribute: Local name of the project

Conditioning Navigation in a Tree

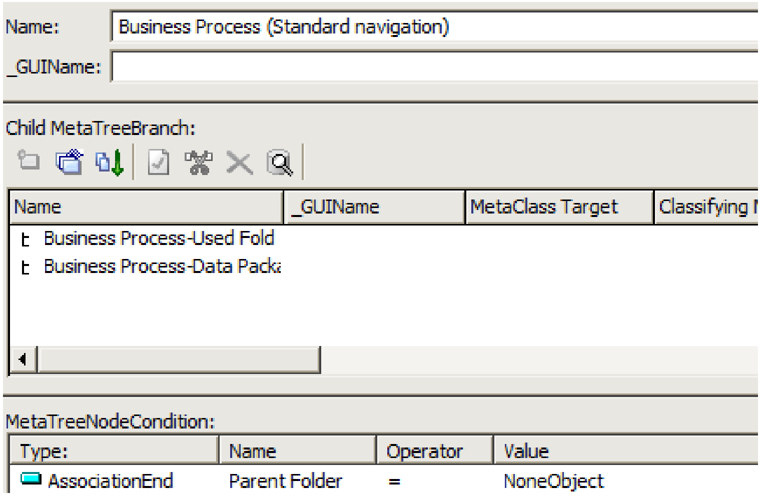
Studio offers the possibility of creating specific branches for each node of a tree. You can define navigation conditions on:

- A node
- A branch
- A classification folder

Defining a navigation condition on a node

To condition navigation on a node:

1. Open the Properties window of the MetaTreeNode that interests you, for example "Business Process (Standard Navigation)".
2. Right-click in the **MetaTreeNodeCondition** space and select **Add**.
3. In the **Type** field of the filter created, select the type of filter you want to use.
   For example, if you want to create a filter on business processes that do not have a parent folder, you use the "Parent Folder" MetaAssociationEnd.



Name: Business Process (Standard navigation)

_GUIName:

Child MetaTreeBranch:

Name	_GUIName	MetaClass Target	Classifying M
Business Process-Used Fold			
Business Process-Data Packa			

MetaTreeNodeCondition:

Type:	Name	Operator	Value
AssociationEnd	Parent Folder	=	NoneObject

4. In the **Name** box, select the identifier that interests you (for example: "Parent folder").
5. Select an **Operator**, then a **Value**.

☛ If the node is not connected to a MetaClass, no condition is proposed.

*An **Option** can indicate that a condition has already been defined.*

Defining a navigation condition on a branch

Given that a branch can also correspond to classification folder, there are two types of conditioning on a branch:

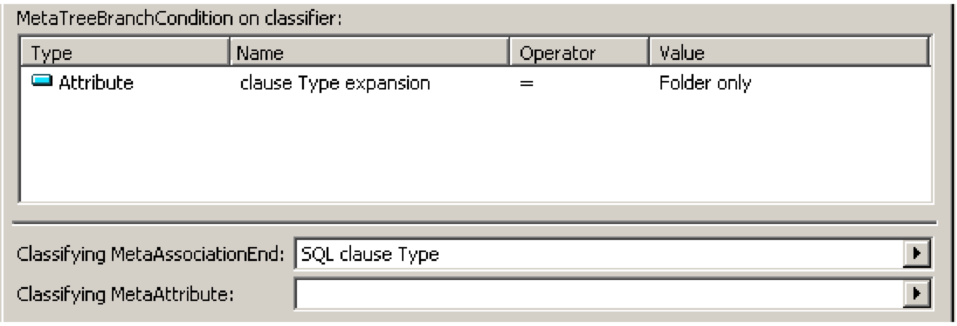
- one relating to the main branch.
- the other relating to the classification branches.

To condition navigation in a branch:

1. Open the Properties window of the MetaTreeBranch that interests you.
2. To define conditioning on the main branch, right-click in the **MetaTreeBranchCondition on Master** space and select **Add**.
3. In the **Type** field of the filter created, select the type of filter you want to use.
4. In the **Name** field, select the identifier that interests you, for example "Parent Folder".
5. Select an **Operator**, then a **Value**.

Defining a navigation condition on a classification folder

You can define a condition on a branch of the classification folder. To do this, use the **MetaTreeBranchCondition on classifier** area of the Properties window of the MetaTreeBranch that interests you, following a procedure as described in Defining a navigation condition on a branch.



MetaTreeBranchCondition on classifier:

Type	Name	Operator	Value
Attribute	clause Type expansion	=	Folder only

Classifying MetaAssociationEnd: SQL clause Type

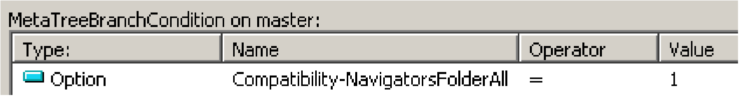
Classifying MetaAttribute:

Using Options

Options can be used to define a filter, classification criterion or conditioning on a node or a MetaTreeBranch as well as on a MetaAssociationEnd or query.

You can for example specify that a branch is only navigable if the user has selected in the **Options** window, from the **Workspace** object, the option 'Display of "All..." folders in the navigator'.

This navigation limit on option appears in the Properties window of the node or branch concerned.



MetaTreeBranchCondition on master:

Type:	Name	Operator	Value
Option	Compatibility-NavigatorsFolderAll	=	1

Customizing Steering Calendars

STEERING CALENDAR

A **Steering calendar** enables to define and launch reminders regarding actions to be performed at predefined due dates.

The following points are covered here:

- ✓ "Introduction to Steering Calendars"
- ✓ "Creating a Steering Calendar"
- ✓ "Steering Calendar Properties"
- ✓ "Customizing Steering Calendars"
- ✓ "Use case: Action Plan workflow"

Introduction to Steering Calendars

A **Steering calendar** includes steering dates, which enable to perform recurrent actions at predefined dates (absolute or relative).

☛ *See "Scheduler Configuration".*

A **Steering date** defines at which (relative or absolute) date the action is to be performed.

☛ *See "Steering Date".*

An **Element with steering calendar** is an element associated with a steering calendar.

☛ *See "Steering Calendar Properties".*

The steering calendar is defined by the following **macros**:

- a parameterization macro, which defines the Scheduler parameterization. It includes the context:
  - current object
  - steering date.
- a job macro, which defines the actions to be performed.

CREATING A STEERING CALENDAR

To create a steering calendar:

1. In **HOPEX Windows Front-End** click **Tools > Explore**.
2. Click **Create** and select the "Steering Calendar" MetaClass.
3. In the wizard that opens, select the **Steering Calendar Type**:
   - Action Plan
   - Control
   - Key Indicator
   - Recommendation

   ☛ *See "Steering Calendar Type".*
4. Connect a **Steering Date** (which corresponds to the execution frequency of interest to you).
5. Open the steering date properties and select the **Scheduling** tab.
6. Specify the information required:
   - start and end date

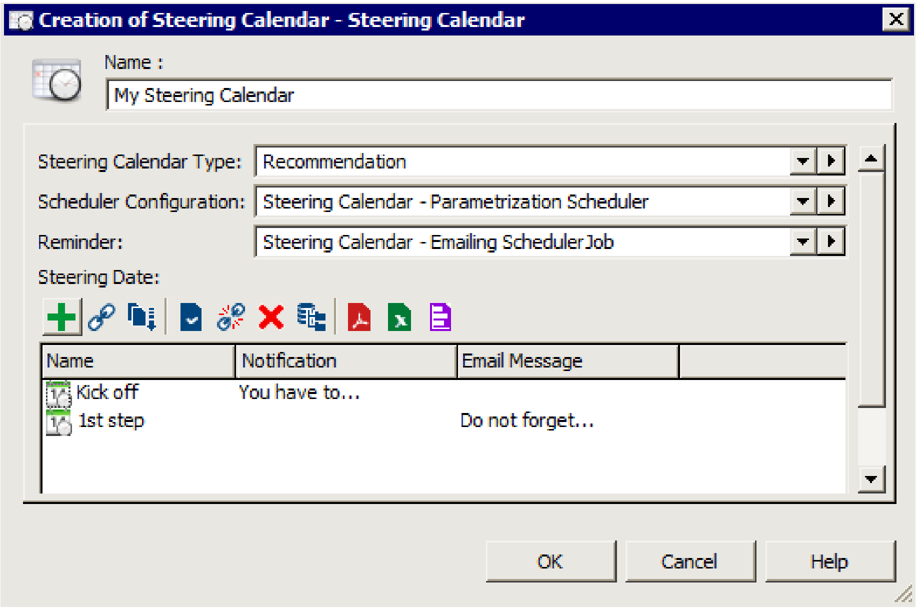
     ☛ *It is recommended to use relative dates on the steering date.*
   - start date and hour (defined in ***UTC*** format, in the user or server time zone).

     ☛ *For details on possible configurations, see the section concerning the scheduler in the "Using the Scheduler" section.*
7. Select **Execute at start date & time** if you wish to launch the execution immediately.

   ☛ *If the check box is deactivated, the scheduler waits for the next recurrent date (and time) to trigger the job.*

See also: "Steering Calendar Properties".

STEERING CALENDAR PROPERTIES



Creation of Steering Calendar - Steering Calendar

Name : My Steering Calendar

Steering Calendar Type: Recommendation

Scheduler Configuration: Steering Calendar - Parametrization Scheduler

Reminder: Steering Calendar - Emailing SchedulerJob

Steering Date:

Name	Notification	Email Message
Kick off	You have to...	
1st step		Do not forget...

A steering calender includes the following parameters:

- "Steering Calendar Type"
- "Scheduler Configuration"
- "Reminder"
- "Steering Date"
- "Scheduling"

Steering Calendar Type

The **Steering Calendar Type** determines the type of steering calendar. **HOPEX** provides you with the following steering calendar types:

- action plan
- control
- key indicator
- recommendation

You can create new steering calendar types.

☛ *See "Creating a Steering Calendar Type".*

Scheduler Configuration

The **Scheduler Configuration** is described by a macro. This macro enables to retrieve:

- the current object
- the steering dates
- the scheduling
- the recording of triggers

HOPEX delivers the following macros:

- "Steering Calendar - Parameterization Scheduler" (default value)
  Macro to be used for action plans or a recommendations.
- "Internal control - Execution Campaign Scheduler Parameterization"
  Macro to be used for controls.

You can create your own macro to customize your scheduler.

☛ *See "Customizing Steering Calendars".*

Reminder

The **Reminder** is described by a macro. This macro details what to do when the job is triggered

MEGA delivers the following macros:

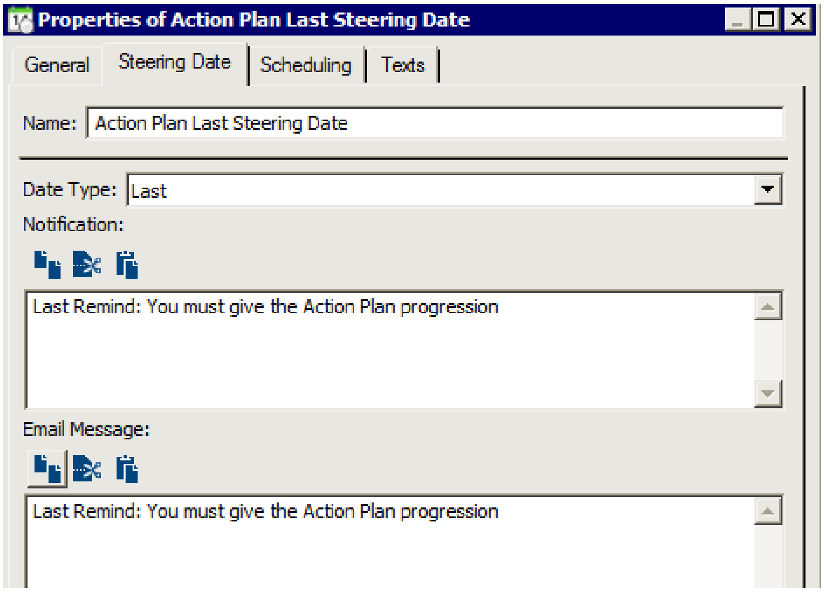
- "Steering Calendar - Emailing Scheduler Job" (default value)
  An email is sent to the person.

  `Use case examples: action plans or recommendations.`
- "Internal control - Execution Campaign Scheduler Job"

  `Use case example: controls.`

Steering Date

A steering date enables the scheduler parameterization: specification of date repeat, start and end date. It also enables specification of the message and/or the notification to be sent (initial, remind or last date).



Name: Action Plan Last Steering Date

Date Type: Last

Notification:

Last Remind: You must give the Action Plan progression

Email Message:

Last Remind: You must give the Action Plan progression

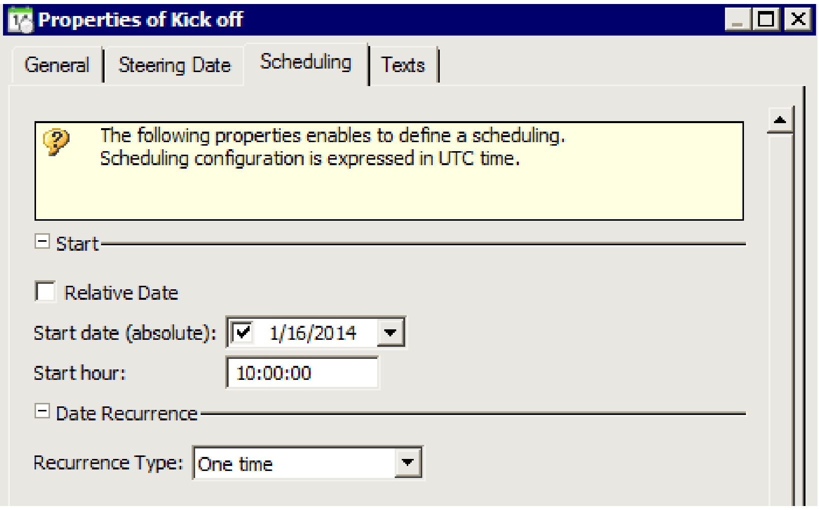
A steering date is defined by:

- its **Name**
- its type (**Date Type**):
  - "Initial": defines the start of the job (i.e.: action plan, control, etc.)
  - "Remind": enables to remind the person responsible for a job to complete the job or to report on the job status.
  - "Last": defines the end of the job (i.e.: action plan, control, etc.)
- a **Notification** (optional)
- an **Email Message** (optional)

Scheduling

For detailed information regarding the date configuration (e.g.: reference date, absolute date, date recurrence) see **HOPEX Power Studio > Using the Scheduler**.

☛ *Note that time is defined in UTC format, which means that daylight saving time is not considered.*



The following properties enables to define a scheduling.
Scheduling configuration is expressed in UTC time.

⊟ Start

□ Relative Date

Start date (absolute): ☑ 1/16/2014

Start hour: 10:00:00

⊟ Date Recurrence

Recurrence Type: One time

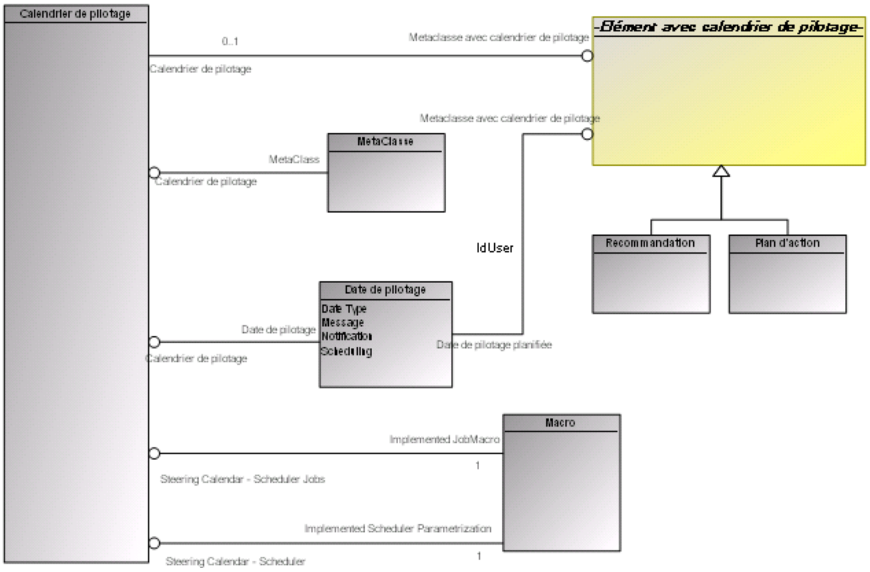
Customizing Steering Calendars

Steering Calendar Metamodel

MEGA provides fully customized steering calendars for:

- action plans
- controls
- recommendations

The steering calendar Metamodel is as follows:



To use the steering calendar on other objects, you need to create a new steering calendar type, see "Creating a Steering Calendar Type".

Creating a Steering Calendar Type

To create a steering calendar type:

1. Link the object related MetaClass to the "Element With Steering Calendar" MetaClass, which creates a new steering calendar type.
2. Create the Scheduler Configuration macro.
   This macro includes the scheduler parameterization.

   The macro while executed retrieves the current object and the steering dates.

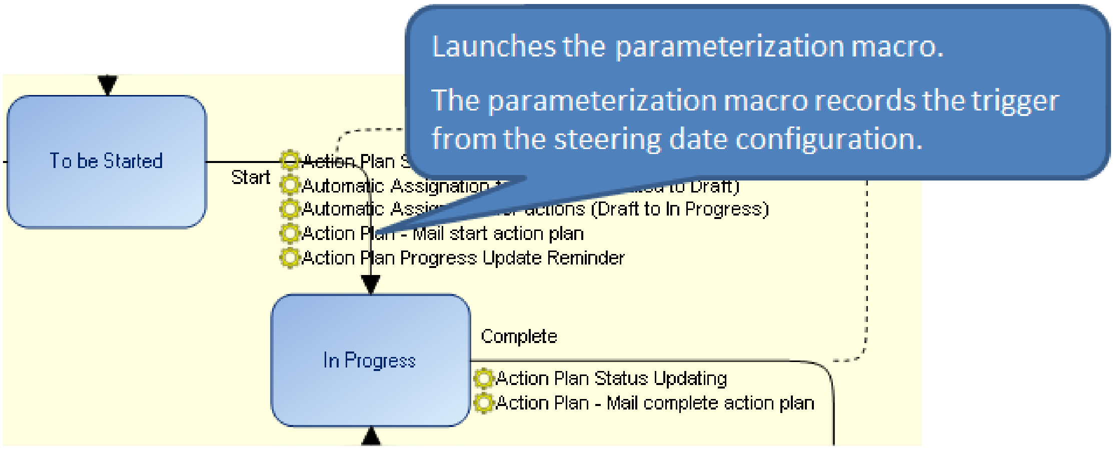
3. Create the job macro.
   This macro includes the job to be executed and the job trigger definition
4. Create the steering date(s).

   ☛ *See "Steering Date".*

USE CASE: ACTION PLAN WORKFLOW

The following is an example of steering calendar use case with an Action Plan workflow.

1. The Workflow transition (i.e.: "To be be started" workflow status - "In progress" workflow status) launches the parameterization macro.

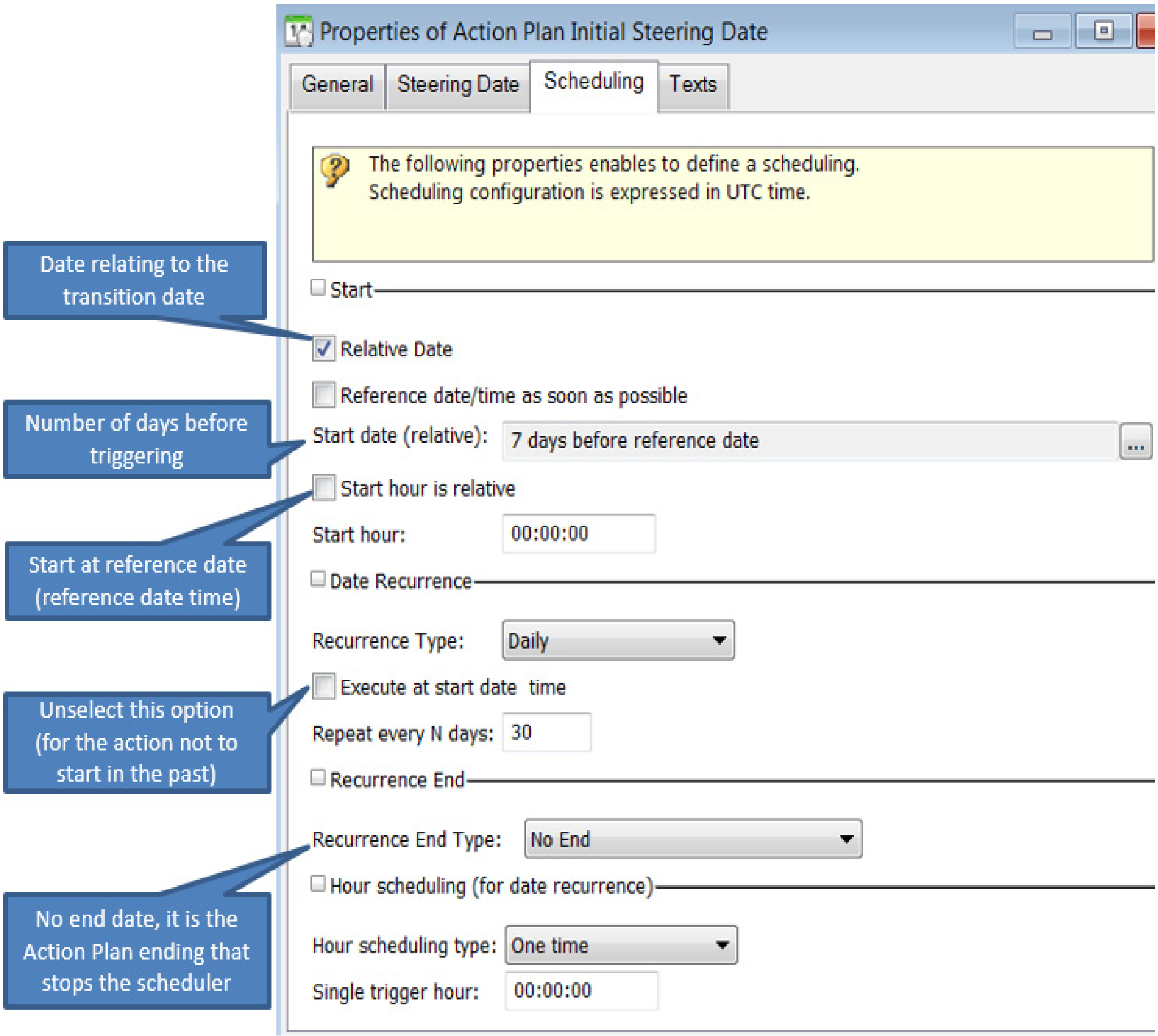


Launches the parameterization macro.

The parameterization macro records the trigger from the steering date configuration.

To be Started
Start
Action Plan S
Automatic Assignation
to Draft)
Automatic Assig
actions (Draft to In Progress)
Action Plan - Mail start action plan
Action Plan Progress Update Reminder
In Progress
Complete
Action Plan Status Updating
Action Plan - Mail complete action plan

2. The parameterization macro records the trigger from the Steering date configuration (**Scheduling** tab).



Properties of Action Plan Initial Steering Date

The following properties enables to define a scheduling.
Scheduling configuration is expressed in UTC time.

□ Start

☑ Relative Date

□ Reference date/time as soon as possible

Start date (relative): 7 days before reference date

□ Start hour is relative

Start hour: 00:00:00

□ Date Recurrence

Recurrence Type: Daily

□ Execute at start date time

Repeat every N days: 30

□ Recurrence End

Recurrence End Type: No End

□ Hour scheduling (for date recurrence)

Hour scheduling type: One time

Single trigger hour: 00:00:00

Date relating to the transition date

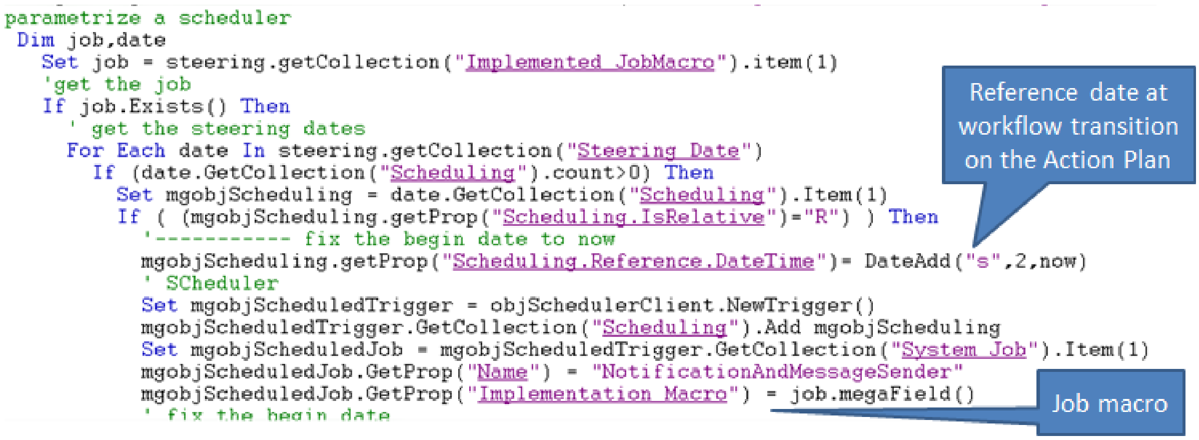
Number of days before triggering

Start at reference date (reference date time)

Unselect this option (for the action not to start in the past)

No end date, it is the Action Plan ending that stops the scheduler

3. The parameterization macro retrieves the current object and the steering dates.
   Parameterization macro: Steering Calendar - Parameterization Scheduler
   Signature: Sub SubscribeSchedulerJobs(currentObject As MegaObject)

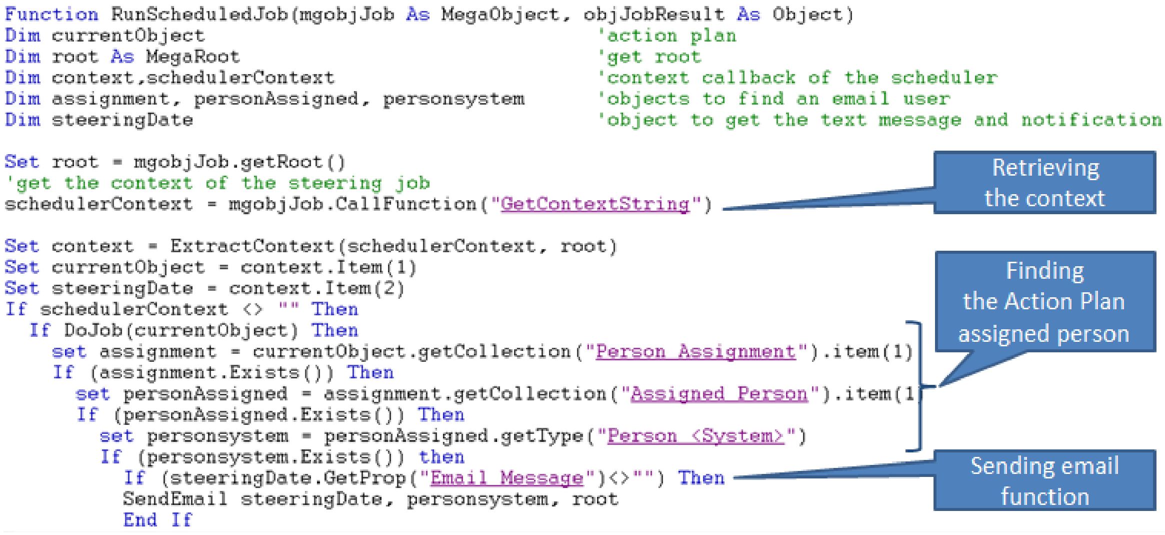


```
parametrize a scheduler
 Dim job,date
   Set job = steering.getCollection("Implemented JobMacro").item(1)
   'get the job
   If job.Exists() Then
     ' get the steering dates
     For Each date In steering.getCollection("Steering Date")
       If (date.GetCollection("Scheduling").count>0) Then
         Set mgobjScheduling = date.GetCollection("Scheduling").Item(1)
         If ( (mgobjScheduling.getProp("Scheduling.IsRelative")="R") ) Then
           '----------- fix the begin date to now
           mgobjScheduling.getProp("Scheduling.Reference.DateTime")= DateAdd("s",2,now)
           ' SCheduler
           Set mgobjScheduledTrigger = objSchedulerClient.NewTrigger()
           mgobjScheduledTrigger.GetCollection("Scheduling").Add mgobjScheduling
           Set mgobjScheduledJob = mgobjScheduledTrigger.GetCollection("System Job").Item(1)
           mgobjScheduledJob.GetProp("Name") = "NotificationAndMessageSender"
           mgobjScheduledJob.GetProp("Implementation Macro") = job.megaField()
           ' fix the begin date
```

Reference date at workflow transition on the Action Plan

Job macro

4. The trigger launches the job macro.
   Job macro: Steering Calendar - Emailing SchedulerJob
   Signature: RunScheduledJob(mgobjJob As MegaObject, objJobResult As Object)



```
Function RunScheduledJob(mgobjJob As MegaObject, objJobResult As Object)
Dim currentObject                               'action plan
Dim root As MegaRoot                            'get root
Dim context,schedulerContext                    'context callback of the scheduler
Dim assignment, personAssigned, personsystem    'objects to find an email user
Dim steeringDate                                'object to get the text message and notification

Set root = mgobjJob.getRoot()
'get the context of the steering job
schedulerContext = mgobjJob.CallFunction("GetContextString")

Set context = ExtractContext(schedulerContext, root)
Set currentObject = context.Item(1)
Set steeringDate = context.Item(2)
If schedulerContext <> "" Then
  If DoJob(currentObject) Then
    set assignment = currentObject.getCollection("Person Assignment").item(1)
    If (assignment.Exists()) Then
      set personAssigned = assignment.getCollection("Assigned Person").item(1
      If (personAssigned.Exists()) Then
        set personsystem = personAssigned.getType("Person <System>")
        If (personsystem.Exists()) then
          If (steeringDate.GetProp("Email Message")<>"") Then
          SendEmail steeringDate, personsystem, root
          End If
```

Retrieving the context

Finding the Action Plan assigned person

Sending email function

Customizing Workflows

CONFIGURING WORKFLOWS

HOPEX proposes different workflows as standard, which you can customize to suit your requirements using **HOPEX Power Studio**.

You can create a workflow and completely configure it, notably using implementation macros. These macros are used on all object types making up a workflow definition diagram.

Sending e-mails and notifications can also be customized.

You can configure workflows from **HOPEX Windows Front-End**.

> ☛ *You must have MEGA APIs to be able to implement macros.*
>
> *See the HOPEX Power Studio - All about starting with APIs guide for more information on using **HOPEX** APIs.*

Defining a Workflow

To create a workflow, you must create a workflow definition and its associated diagram.

The different objects in the workflow definition diagram and their configuration are presented here.

For more details on workflow configuration, see:

- "Workflow Advanced Configuration", page 52
- "Managing E-mails and Notifications", page 56

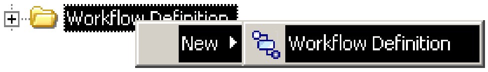
Overview of Workflow Definitions

A workflow definition enables definition of a sequence flow of operations executed by persons. When executed, the workflow assures management of the sequence of operations and notification of the persons involved. The workflow can be applied to a repository object known as the workflow subject. In this case, expected operations are related to this object.

Creating a Workflow Definition

To create a workflow definition in **HOPEX Windows Front-End**:

1. In the **Utilities** navigation window, right-click the **Workflow Definition** folder and select **New > Workflow Definition**.



2. In the dialog box that appears, enter the name of the workflow definition and click **OK**.

The new workflow definition is created.

A workflow must relate to an object type. You can therefore associate an object type with the workflow definition you have just created.

To define the object type to which the workflow relates:

1. In the properties dialog box of the workflow definition, select the **Characteristics** tab.
2. In the **Subject MetaClass** section, click the **Connect** button.
3. In the **Select Query** dialog box, select "Workflow Subject MetaClasses" and click **OK**.
4. Select a MetaClass and click **OK**.

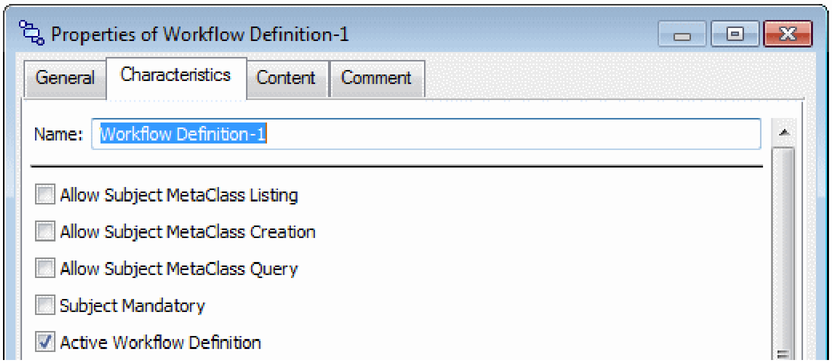
☛ *The connected MetaClass must be a sub-class of the "Workflow Subject" or "System Workflow Subject" abstract MetaClass.*

A subject MetaClass is a MetaClass to which a workflow definition can be applied. Workflows defined by this workflow definition can be executed on instances of this MetaClass.

Several options are available in the workflow definition properties:

- **Allow Subject MetaClass Creation**: allows the user to create the object to which the workflow relates at the moment the workflow is started.
- **Allow Subject MetaClass Listing**: enables the user to list workflow subjects available at the moment the workflow is started.
- **Allow Subject MetaClass Query**: allows the user to start the query tool to select a workflow subject at the moment workflow is started.
- **Subject Mandatory**: obliges the user when executing the workflow to select the object to which the workflow relates.
- **Activate Workflow** : enables workflow activation or deactivation. Users should not access workflows which are in course of definition. For this reason, it is useful to be able to deactivate a workflow.

☛ *If the workflow is deactivated, it can no longer be started. The **Start Workflow** menu of the workflow definition is grayed.*



Name: Workflow Definition-1

□ Allow Subject MetaClass Listing
□ Allow Subject MetaClass Creation
□ Allow Subject MetaClass Query
□ Subject Mandatory
☑ Active Workflow Definition

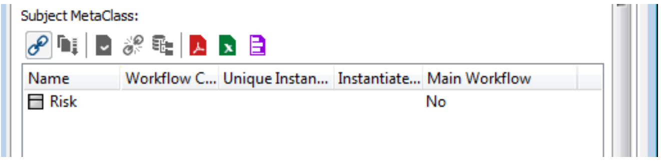
For each subject MetaClass you can specify in the corresponding columns:

- **Workflow Condition**: a condition enables filtering of workflows that can be instanced from a subject MetaClass.

  It is for example possible to start a workflow on an object type that has a particular characteristic.

  ☛ *A condition can be used on workflows instanced at creation of the subject when several workflow definitions exist for a MetaClass.*
- **Unique Instancing**: enables indication that for a subject instance you can have only a single workflow instance in progress.
- **Instance at Creation**: enables creation of a workflow instance at workflow creation that manages the life cycle of the object.
- **Main Workflow**: enables specification, for a given subject MetaClass, that this workflow gives or does not give the main current status.

Several workflow definitions can be defined for the same MetaClass. However, a MetaClass can only have one main workflow definition.



Name	Workflow C...	Unique Instan...	Instantiate...	Main Workflow
Risk				No

To be able to specify the workflow, you must then create the workflow definition diagram.

Accessing workflow definitions

To access workflow definitions:

1. In the Enterprise Architecture desktop, select **View > Navigation Windows > Utilities**.
2. Expand the Workflow Definitions folder.

For each workflow definition, contained elements are displayed in sub-folders:

- workflow definition diagram
  see "Creating a workflow definition diagram", page 34.
- workflow statuses
  see "Workflow Statuses", page 35.
- workflow transitions
  see "Workflow Transitions", page 38.
- workflow participants
  see "Workflow Participants", page 35.
- workflow actions
  see "Workflow Actions", page 50.

Creating a workflow definition diagram

To create a workflow definition diagram:

1. See "Accessing workflow definitions", page 34.
2. Right-click the workflow definition and select **New > Diagram**.
3. In the dialog box that opens, select **Create**.

The diagram is created.

In a workflow definition diagram, you must:

- create a workflow status of **Initial** type.
- create the different workflow statuses and connect these by means of workflow transitions.
- create the last workflow status or statuses of **Final** type.

  ☛ *See "Workflow Statuses", page 35.*
- specify persons that trigger transitions: to do this you connect workflow participants to the different transitions.
  Persons must be associated with workflow participants.

  ☛ *See "Workflow Transitions", page 38.*

Workflow Statuses

To create a workflow status:

1. In the workflow definition diagram, click one of the three workflow status buttons in the diagram insert toolbar to create a workflow status:
   - **Default**
   - **Initial**
   - **Final**

   🕮 *A workflow status corresponds to a step in progress of a workflow defined by a workflow definition.*
2. Click on the diagram.
3. In the dialog box that opens, enter the name of the workflow status and click **OK**.

The new workflow status appears in the diagram.

Workflow Participants

Creating workflow participants enables definition of the persons associated with a workflow transition.

🕮 *A workflow participant enables definition of the set of persons that can be assigned to a workflow transition in the framework of execution of a workflow instance.*

🕮 *A workflow transition connects a source workflow status to a target workflow status. A person associated with a workflow transition for a given workflow status can trigger the workflow transition, passing the workflow instance from the current source workflow status to the target workflow status, which then becomes the current status. By this action, the person informs that the operation expected of them has been executed.*

This set of persons can be defined at workflow configuration or calculated at workflow execution.

You can create a workflow participant:

- directly in the workflow definition diagram.

  ☛ *For more details, see "Creating a participant in a workflow definition diagram", page 35.*
- in the properties of the workflow transition.

  ☛ *For more details on workflow definitions, see "Workflow Transitions", page 38.*
- in the properties of the workflow status.

  ☛ *For more details on workflow statuses, see "Creating a participant from a workflow status", page 36.*

Creating a participant in a workflow definition diagram

To create a workflow participant:

1. In the workflow definition diagram, click the **Workflow Participant** button in the diagram insert toolbar, then click in the diagram.
2. In the dialog box that opens, enter the participant name and click **Finish**.

The workflow participant appears in the diagram.

Creating a participant from a workflow status

Specifying participants on the workflow status enables:

- factorization and avoiding specification of participants on each output transition of the same workflow status.
- on an initial workflow status, indication of who has the right to start the workflow

Associating persons with participants

When the participant has been created, you can connect it to persons. The persons associated with a participant can trigger transitions between two workflow statuses.

To connect persons to a workflow participant:

- In the properties page for the workflow participate, specify an implementation macro.

> ☛ *An implementation macro on a workflow participant enables calculation of a set of persons at workflow execution.*

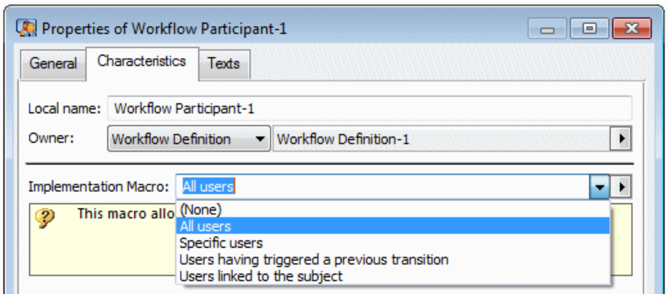
When creating a macro, a wizard will help you initialize its content. Macro content is initialized with the list of parameters that can be used to determine the list of persons.

HOPEX supplies different macros enabling definition of the set of persons likely to trigger workflow transitions.

To use a macro in the framework of a participant:

1. In the properties dialog box of the participant, select a macro from the drop-down list.

2. Click **Apply**.



Properties of Workflow Participant-1
General
Characteristics
Texts
This macro allo
(None)
All users
Specific users
Users having triggered a previous transition
Users linked to the subject

Local name: Workflow Participant-1

Owner: Workflow Definition | Workflow Definition-1

Implementation Macro: All users

A new dialog box appears, depending on the selected macro.

> ☛ *IntelliSense is available here. Enter <Ctrl> + <space> to display possible parameters.*

- **All users**: all persons are selected.
- **Assigned person to subject with specific business role**: enables specification of a list of business roles. Below is an example of parameterization:

  [BusinessRoles]

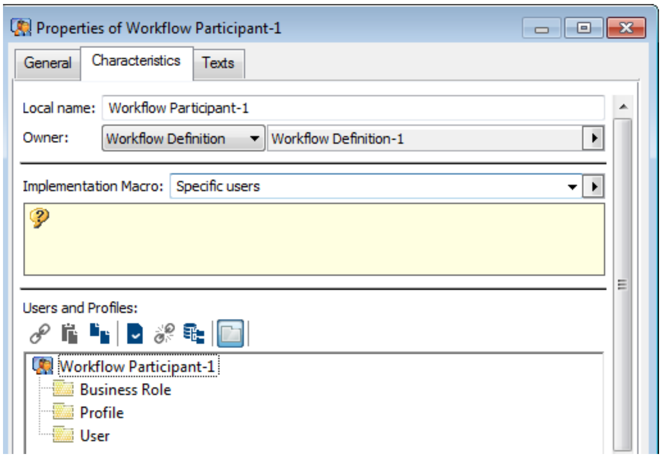
  1=~5tM5w)1JTPoM[Data Owner]

  2=~pcRgRnQkILWE[Data Designer]

  3=~MtM5L42JTb1N[Data Quality Manager]

  4=~0CxUrKPKTToC[Data Steward]

  5=~D5V5GZH5Nbk2[Data Scientist]
- **Specific users**: enables explicit specification of a list of persons. You can connect roles, persons or profiles.



Local name: Workflow Participant-1

Owner: Workflow Definition | Workflow Definition-1

Implementation Macro: Specific users

- **Users linked to the subject**: enables listing of users via a query applied to the workflow subject.

  If the workflow subject is a design task, you can define a query that will find project managers of the project associated with the design task.

The query must be specified in the _Parameterization text, under section [UsersSet], as in the example below:

[UsersSet]

QueryFromSubject = ~llC2RbmyELaC[Action Plan Approvers]

- **Users having triggered a previous transition**: enables specification of person(s) who triggered the previous transition.

The previous transition must be specified in the _Parameterization text, under section [UsersSet], as in the example below:

[UsersSet]

ProvidingTransition = ~n1uG9GqiBD60[Approve]

- **Person Membership of EA Project with specific Business Role**: enables specification of a list of persons from the workflow subject (for EA projects only). The persons involved are those who have a role in the EA project and whose role is specified in the _Parameterization text.
- **Creator of Subject Workflow**: enables specification of the person who created the object which is subject of the workflow.
- **Creator and all Information Asset Responsibility Except Data Chief Officer**: used in **HOPEX Information Architecture**, it enables specification of a list of persons who are assigned to the workflow subject (Design validation). It sends all persons except the Data Chief Officer an email to inform them on the workflow transitions performed for the object they are in charge of.

Workflow Transitions

Creating a workflow transition

To create a workflow transition:

1. See "Accessing workflow definitions", page 34.
2. In the workflow definition diagram, click the **Workflow Transition** button ➞ in the diagram insert toolbar, then draw a link between the two workflow statuses concerned.
3. In the dialog box that appears, enter the name of the transition and click **OK**.
   The transition appears in the diagram.
   You must now define certain properties.

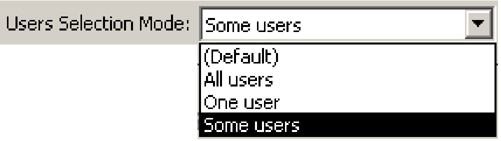
📖 *A workflow transition connects a source workflow status to a target workflow status. A person associated with a workflow transition for a given workflow status can trigger the workflow transition, passing the workflow instance from the current source workflow status to the target workflow status, which then becomes the current status. By this action, the person informs that the operation expected of them has been executed.*

Person selection mode

☛ *Participants can also be specified on the workflow status, in the **Participation** tab.*

The **Participants** tab of a workflow transition enables definition of how persons are selected at assignment of persons to the next workflow status.

Several values are proposed in the **User Selection Mode** field when configuring the workflow:



Users Selection Mode: Some users

- **Some Users**: one or more persons can be selected in a list. This is a list of users authorized to trigger the next workflow transition.
- **All Users**: all the users associated with the corresponding workflow participant are assigned to the next status. No list is proposed.
- **One Person**: only one person can be selected in the list and assigned to the next status (if you select several persons in the list, you cannot trigger the workflow transition).

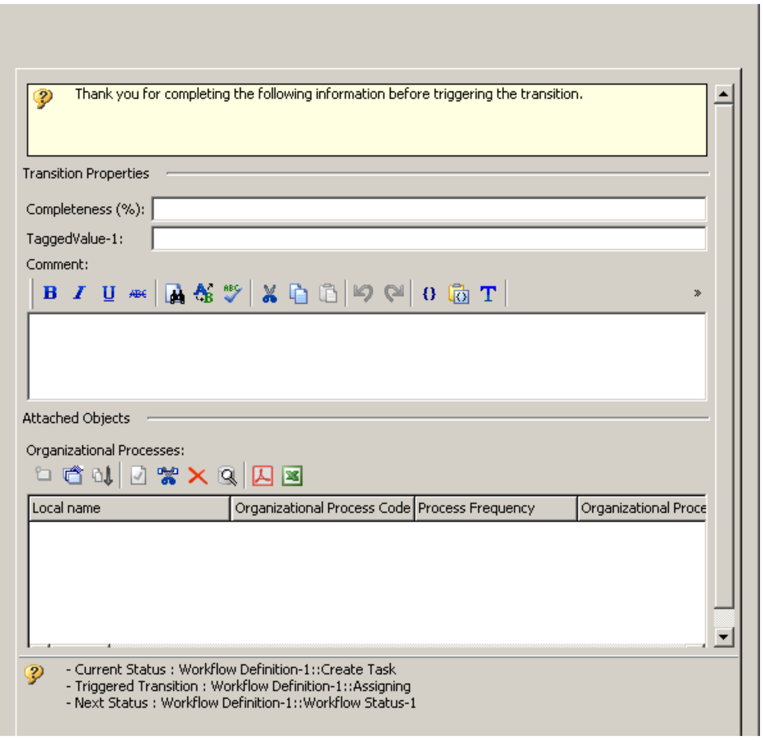
Information associated with workflow transition

You can allow the user who triggers the workflow transition to specify additional information.

You can allow the user to:

- add a comment
- specify one or several properties in the form of fields
- attach objects

☛ *This information can be made mandatory via permissions. For more details on configuration of permissions, see "Configuring Permissions (CRUDs) on Objects", page 68.*



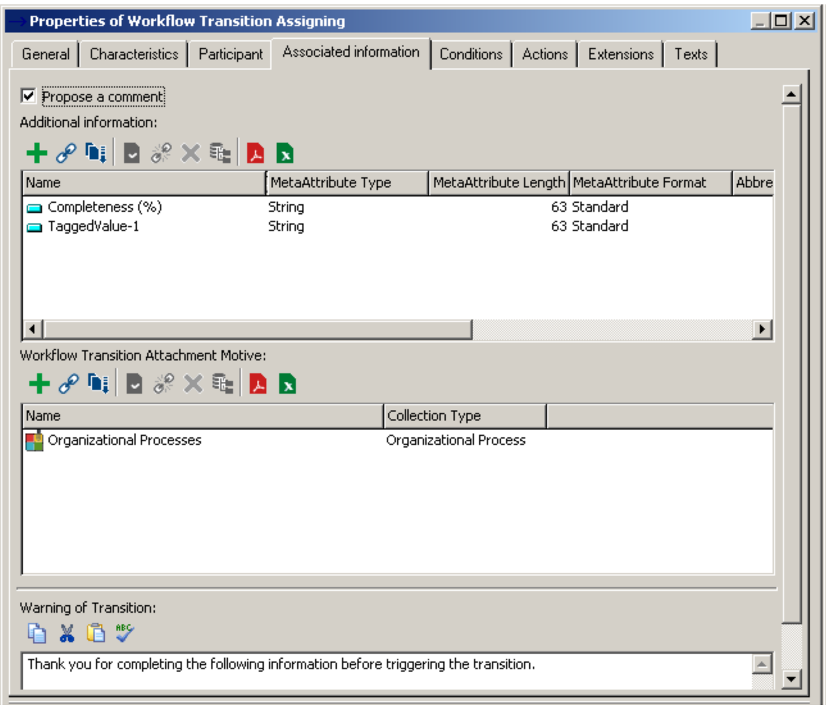
Thank you for completing the following information before triggering the transition.

- Current Status : Workflow Definition-1::Create Task
- Triggered Transition : Workflow Definition-1::Assigning
- Next Status : Workflow Definition-1::Workflow Status-1

Example of information that can be added on a transition

To enable entry of additional information on workflow transitions:

- In the properties dialog box of the workflow transition, select the **Associated Information** tab.



☑ Propose a comment

Additional information:

Name	MetaAttribute Type	MetaAttribute Length	MetaAttribute Format	Abbre
Completeness (%)	String	63	Standard	
TaggedValue-1	String	63	Standard	

Workflow Transition Attachment Motive:

Name	Collection Type
Organizational Processes	Organizational Process

Warning of Transition:

Thank you for completing the following information before triggering the transition.

Example of configurations on a transition

Enabling comment entry

To offer the user the possibility of entering a comment when triggering a workflow transition:

- Select the **Propose a Comment** check box.

Adding properties on transitions

To offer the possibility of specifying properties when triggering a workflow transition:

- Create TaggedValues in the **Additional Information** frame.
  At triggering of the corresponding workflow transition, the dialog box proposed to the user will display the TaggedValues value specified here.

Opening a help tooltip in the transition dialog box

You can display a help tooltip to guide your users at triggering of the workflow transition.

To enter the text to appear in the help tooltip:

- Enter the text to be presented to users in the **Warning of Transition** frame.

Transition attachments

To offer the user the possibility of attaching **HOPEX** objects at workflow transition triggering:

1. Create a motive in the **Workflow Transition Attachment Motive** frame, for example "Items to provide".

   ☛ *An object attachment motive specifies the reason for which the objects are used in the framework of a workflow transition.*

2. Select the motive, and in its properties dialog box select a collection type, which corresponds to a MetaClass.

   ☛ *The same MetaClass can appear in several motives.*

Motives are used particularly:

- in the framework of sending e-mails
  - to list attachments
  - to include objects in the form of attachments in the case of documents

  ☛ *MetaClasses that can be the subject of attachments in e-mails are: documents, reports, books, external references.*

- to associate objects with a notification

  ☛ *For more details on configuration of notifications or e-mails, see:*

  - *"Managing objects in notifications", page 59.*
  - *"Managing attachments in e-mails", page 59*

When the user triggers a transition, a dialog box proposes connection of objects to be attached.

Multiple triggering transitions

Reminder of general case concerning workflow transitions

When a transition has been triggered by a user, the workflow instance passes to the next status.

By default, no other user can now trigger this transition.

Multiple triggering

When the multiple triggering option is activated for a transition, several users can trigger the transition.

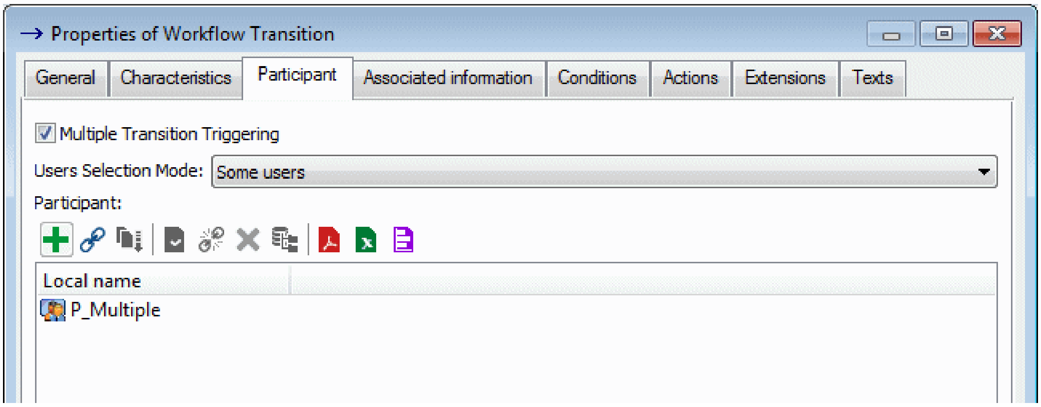
When a user triggers multiple triggering, the workflow instance passes to the next status. Workflow status properties enable identification of:

- users who have triggered the transition
- users who have not yet triggered

To activate the multiple triggering option:

1. In the workflow transition properties dialog box, select the **Participant** tab.

2. Select the **Multiple Participation** option.



☑ Multiple Transition Triggering

Users Selection Mode: Some users

Participant:

Local name
P_Multiple

☛ *The multiple transition should not be connected to initial or to final status.*

Await user intervention

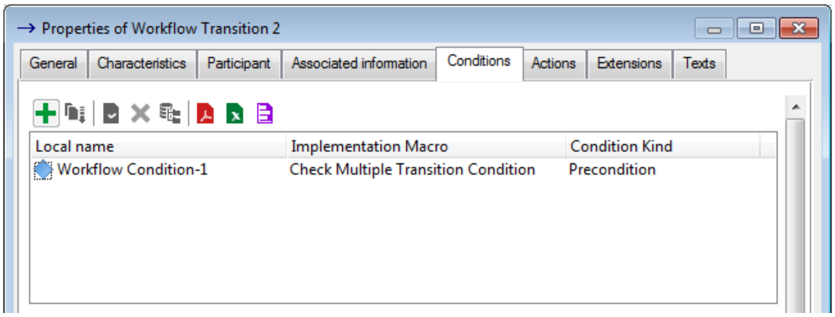
You can arrange that intervention of users of a multiple transition is mandatory for the workflow to continue and pass to the next status.

To do this:

1. On the workflow transition that follows the multiple transition, implement the macro "Check Multiple Transition Condition".

   ☛ *You can duplicate and customize this macro to indicate for example that at least 50% of users must have intervened to be able to pass to the next workflow status.*

2. Modify the value by default for the **Condition Kind** and specify that it is a "Precondition".



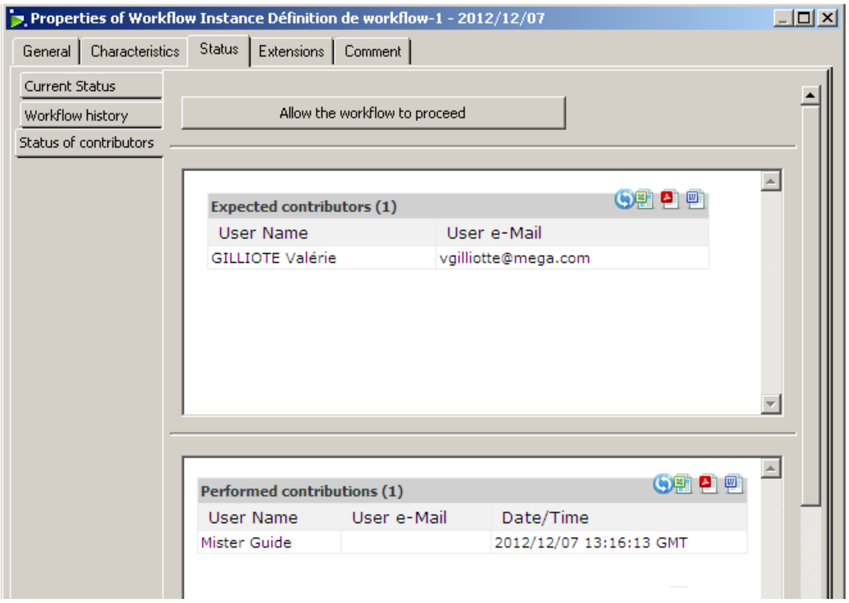
Local name	Implementation Macro	Condition Kind
Workflow Condition-1	Check Multiple Transition Condition	Precondition

With implementation of this macro, and if all users have not yet intervened, the **Next Status** command of the workflow definition pop-up menu is grayed for the user authorized to trigger the next workflow.

Viewing users contributing a workflow

To view contributors awaited in the workflow who have not yet triggered a transition:

- In the properties dialog box of the workflow instance, select the **Status** tab and the **Status of Contributors** subtab.


Properties of Workflow Instance Définition de workflow-1 - 2012/12/07

Expected contributors (1)

User Name	User e-Mail
GILLIOTE Valérie	vgilliotte@mega.com

Performed contributions (1)

User Name	User e-Mail	Date/Time
Mister Guide		2012/12/07 13:16:13 GMT

You can view in this subtab:

- users who have already participated in the workflow (**Performed Contributions** frame)
- users whose contribution is awaited preventing workflow progress (**Expected Contributors** frame)

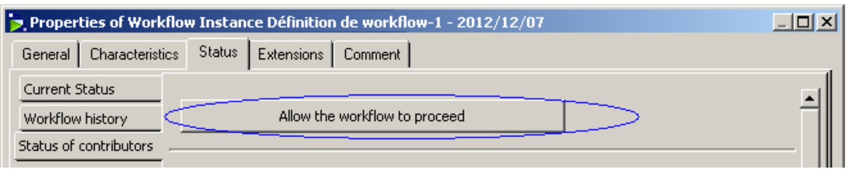
Progressing the workflow

HOPEX allows to bypass the "Check Multiple Transition Condition" macro and authorizes progress without awaiting intervention of other users.

To continue the workflow despite the fact that some users have not yet triggered a transition:

1. In the properties dialog box of the workflow instance, select the **Status** tab.

2. In the **Status of Contributors** subtab, click the **Allow the workflow to proceed** button.


Properties of Workflow Instance Définition de workflow-1 - 2012/12/07
General
Characteristics
Status
Extensions
Comment
Current Status
Workflow history
Status of contributors
Allow the workflow to proceed

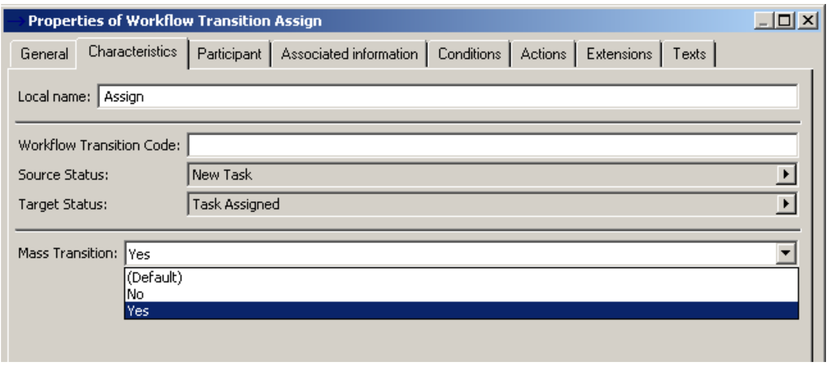
The **Next Status** command of the workflow instance is no longer grayed. The workflow can continue. Other contributors no longer intervene in the workflow.

Mass transitions

You may need to trigger several transitions simultaneously, that is to pass several workflow instances from one status to another.

To be able to authorize transition mass triggering:

- In the properties dialog box of a workflow transition, select the value "Yes" in the **Mass Transition** field.



Local name: Assign

Workflow Transition Code:

Source Status: New Task

Target Status: Task Assigned

Mass Transition: Yes

When the workflow is instanced, you can trigger mass transition from:

- subject objects
- workflow instances

To trigger a transition simultaneously on several workflow subjects in the same current status:

- Right-click the subject objects, for example in a navigation tree, and select command **Mass Transition**.

  ☛ *The instances must be in the same workflow status to enable a mass transition.*

Implementing scheduled transitions

Workflow transitions can be triggered on a given date without user intervention.

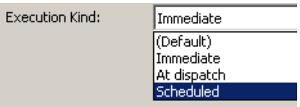
Example: a workflow transition can be triggered if the workflow remains in the same status for more than ten days.

The triggering date can be relative to a reference date, for example the triggering date of a workflow transition or instancing of a workflow.

Example: a reminder is sent one week after passage to a particular workflow status.

To define a transition with relative triggering date:

1. In the dialog box of creation of an action, select "Scheduled" execution kind.


Execution Kind: Immediate

2. Click **Next**.
3. In the **Reference Date** field, two macros are proposed:
   - "Reference date from Subject property"

   Macro configuration text can be specified as follows:

   [SchedulingDate]

```
MetaAttribute = [Planned End Date]
```

☛ *Configuration data entry help is available. To do this, enter the character "[" then simultaneously press keys Ctrl + Space.*

   - "Reference date is transition triggering"

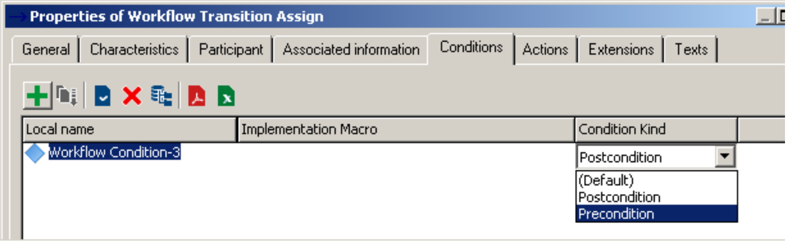
☛ *This macro does not require additional configuration, since it takes the transition triggering date as reference date.*

4. In the **Start** section, specify recurrence conditions and hourly planning.

Conditioning a transition

To condition a workflow transition:

1. In the properties dialog box of a workflow transition, select the **Conditions** tab.
2. Create a condition and select a **Macro**.
3. In the **Condition Kind** field, specify if it is a:
   - **Precondition**
   - **Post-condition**


Properties of Workflow Transition Assign

Local name	Implementation Macro	Condition Kind
Workflow Condition-3		Postcondition

Precondition

The pre-condition first checks that pre-conditions are met for the transition to be proposed.

If conditions are not met, the menu corresponding to the transition is not proposed to the user.

☛ *Specifying a condition on the first workflow transition of a workflow definition is equivalent to defining a condition on the subject. For more details, see "Overview of Workflow Definitions", page 32 (paragraph concerning workflow conditions).*

Post-condition

Possible transition choices are presented to the user.

if required conditions are not met, a message warns that the transition will not be executed.

> ☛ *In batch mode, conditions are systematically assessed.*

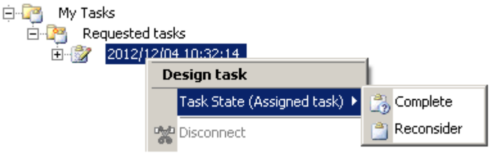
Configuring the workflow transition triggering menu

Workflow subject objects can propose a pop-up menu enabling workflow transition triggering.

> ☛ *The "Design Task" MetaClass for example has this configuration as standard.*

This pop-up menu comprises a set of menus:

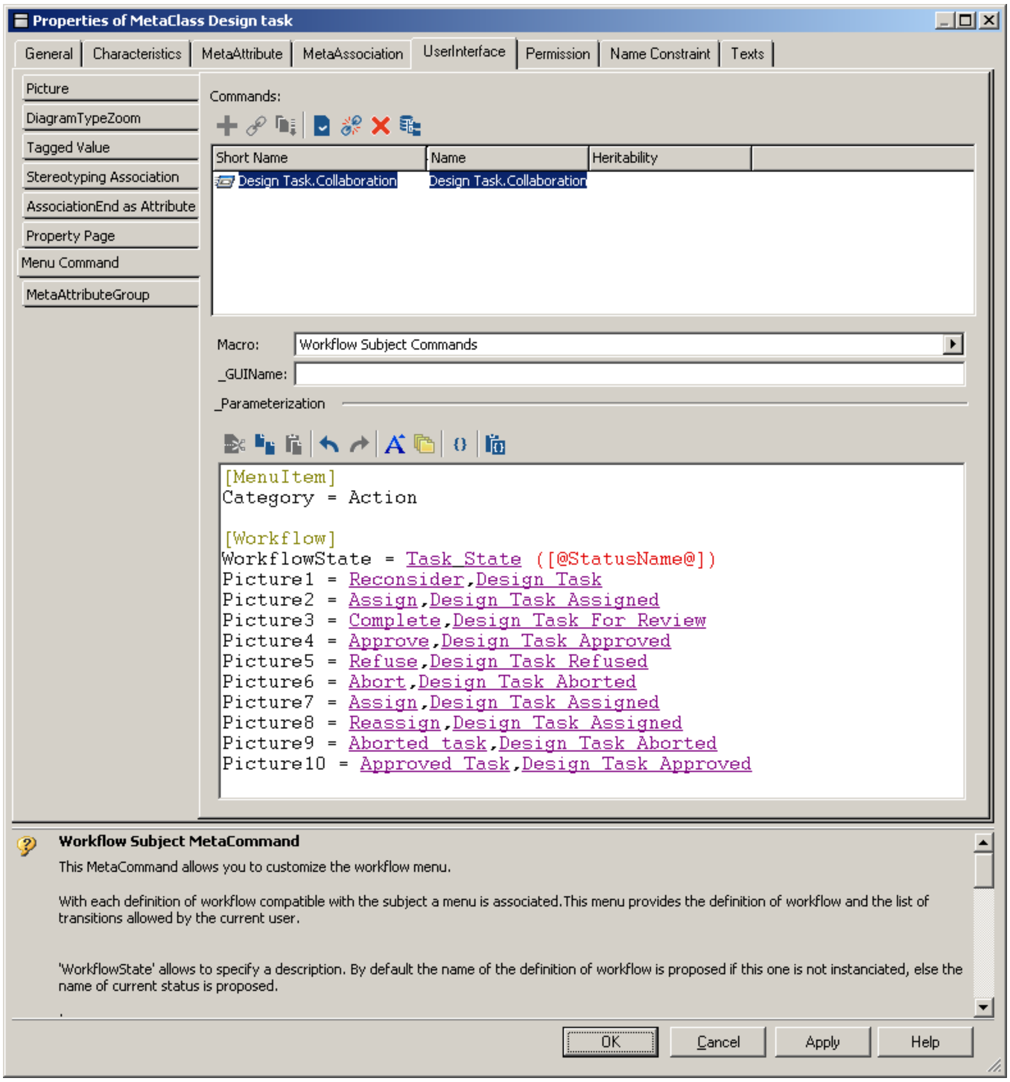
- a menu displaying the state of the object
- several sub-menus presenting the list of workflow transitions that the user can trigger



To create a menu enabling triggering of a transition on an object:

1. In the properties dialog box of the MetaClass concerned, select the **User Interface** tab, **Menu Commands** subtab.
2. Create a **Command** and click its name.
3. In the **Macro** field, associate the "Workflow Subject Command" macro.

4. Configure menu display in the **_Parameterization** section.



Short Name	Name	Heritability
Design Task.Collaboration	Design Task.Collaboration	

Macro: Workflow Subject Commands

_GUIName:

_Parameterization

```
[MenuItem]
Category = Action

[Workflow]
WorkflowState = Task State ([@StatusName@])
Picture1 = Reconsider,Design Task
Picture2 = Assign,Design Task Assigned
Picture3 = Complete,Design Task For Review
Picture4 = Approve,Design Task Approved
Picture5 = Refuse,Design Task Refused
Picture6 = Abort,Design Task Aborted
Picture7 = Assign,Design Task Assigned
Picture8 = Reassign,Design Task Assigned
Picture9 = Aborted task,Design Task Aborted
Picture10 = Approved Task,Design Task Approved
```

Workflow Subject MetaCommand

This MetaCommand allows you to customize the workflow menu.

With each definition of workflow compatible with the subject a menu is associated.This menu provides the definition of workflow and the list of transitions allowed by the current user.

'WorkflowState' allows to specify a description. By default the name of the definition of workflow is proposed if this one is not instanciated, else the name of current status is proposed.

.

☛ *Detailed help is available at the bottom of the properties dialog box after clicking in the configuration area.*

Naming the menu

The "WorkflowState" key enables menu naming. It can be defined via:

- code templates
- tags [@StatusName@] and [@WorkflowName@].

This configuration enables structuring of all menus in the same way.

WorkflowState = ~8bjDw1bEFTtR[Task State] ([@StatusName@])

Naming the menu for a given workflow definition

The key "WorkflowState%n" enables naming of a menu for a given workflow definition. You must specify the workflow definition.

```
WorkflowState1 = ~9nWjkOxmE95G[Action Plan Workflow], [@StatusName@]
```

Associating an image with a workflow transition

To associate an image with a workflow transition:

- With the key "Picture%n", specify a workflow transition and an image.

```
Picture1 = ~X0O81(okB5S2[Reconsider],~Zimg8ZhWBrn0[Design Task]
Picture2 = ~R3O827okB550[Assign],~(f3u5nUnBvH1[Design Task Assigned]
```

Hiding the menu of a workflow definition

The key "FilterMode" enables hiding of the menu of a workflow definition. Two values are possible:

- "All"
  The value "All" is default behavior. All workflow menus are presented.
- "List"
  The value "List" enables display of only those menus associated with the workflow definitions you want to propose. Only those menus for which the key " WorkflowState%n " was specified are presented.

The configuration below enables proposal of only the menu for the "Action Plan Workflow" workflow definition. No menus are proposed for other possible workflows.

```
WorkflowState1 = ~9nWjkOxmE95G[Action Plan Workflow], [@StatusName@]
FilterMode = List
```

Hiding the state of the object

The first level in the menu concerns the state of the object. You can choose to hide it and display only the workflow transitions.

Managing transitions at reassignment

When a transition is triggered, the responsible of the next transition is calculated.

If assignment of a person is modified, it may be necessary to re-trigger the previous workflow transition to update the next transition responsible.

Two cases can arise:

- case of back transitions
- case of loop transitions

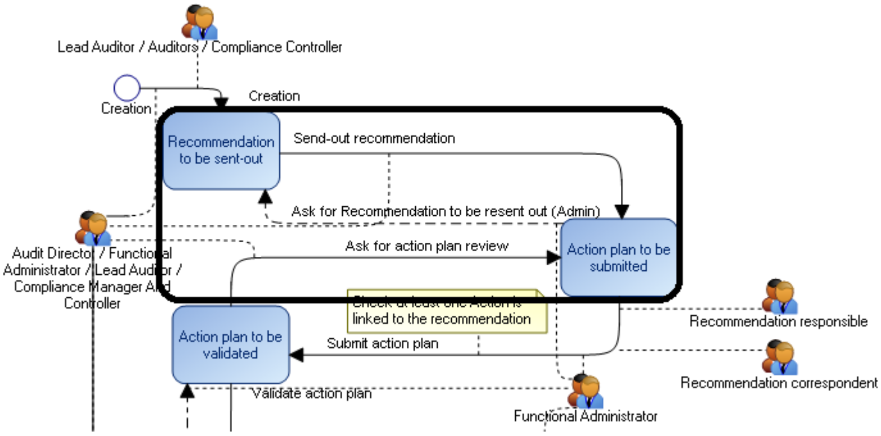
> ☛ *In workflows supplied as standard, back and loop transitions are not used in normal workflow processing. They are reserved for the administrator to process exceptional cases.*

Back transition

When assignment modification infrequent, it may be sufficient to use back transitions.

In the example below on the Recommendation workflow, if the recommendation is in status "Action Plan to be submitted" and the recommendation owner changes, the administrator can:

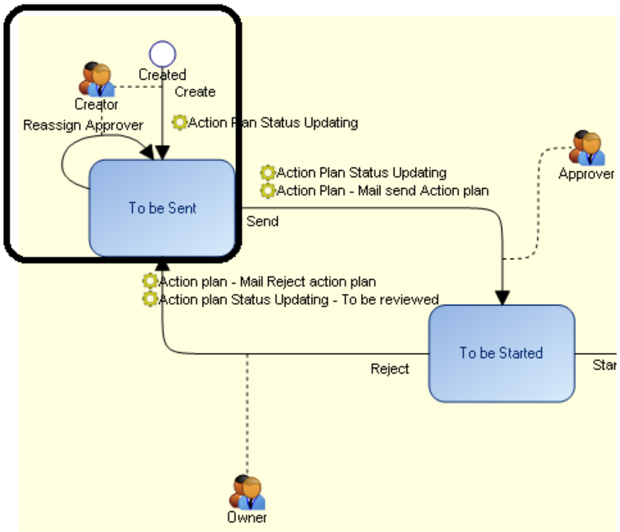
- execute the "Ask for recommendation to be resent (Admin)" transition, and
- re-execute the "Send out recommendation" transition to update the recommendation responsible of the next transition


Lead Auditor / Auditors / Compliance Controller
Creation
Creation
Recommendation to be sent-out
Send-out recommendation
Ask for Recommendation to be resent out (Admin)
Audit Director / Functional Administrator / Lead Auditor / Compliance Manager And Controller
Ask for action plan review
Action plan to be submitted
Check at least one Action is linked to the recommendation
Recommendation responsible
Action plan to be validated
Submit action plan
Recommendation correspondent
Validate action plan
Functional Administrator

Loop transition

When person assignment changes are frequent, you can add a loop workflow transition on the status of the workflow concerned.

For example, in the "Top-Down Action Plan" workflow, a loop transition has been added on the "To be sent" state. In this way the action plan creator can easily reassign the action plan approver.


Created
Creator
Create
Reassign Approver
Action Plan Status Updating
Action Plan Status Updating
Action Plan - Mail send Action plan
Approver
To be Sent
Send
Action plan - Mail Reject action plan
Action plan Status Updating - To be reviewed
To be Started
Reject
Star
Owner

Workflow Actions

📖 *A workflow action enables association of a processing execution with triggering of a workflow transition (processing execution can be postponed until dispatch or a later scheduled date).*

You can create a workflow action:

- on a workflow transition

  ☛ *For more details, see "Workflow Transitions", page 38.*

- on a workflow status

  ☛ *For more details, see "Workflow Statuses", page 35.*

Configuring a workflow action on a workflow status

It can be useful to specify a workflow action on a workflow status:

- to trigger a scheduled transition

  Example: an action is triggered 10 days after the arrival of a workflow transition in the status

  ☛ *For more details, see "Implementing scheduled transitions", page 44.*

- to factorize and avoid avoid specifying the same action on various workflow transitions

  ☛ *When a transition is triggered, it is the action specified on the target (and not source) status that is implemented.*

Creating a workflow action

Execution of a workflow action is by implementation of a macro. The macros proposed as standard enable execution of different types of action.

To create a workflow action:

1. See "Accessing workflow definitions", page 34.
2. In the workflow definition diagram, click the **Workflow Action** button in the diagram insert toolbar, then click in the diagram.
   The action creation wizard opens.
3. Create an implementation macro or select a macro corresponding to the required action:
   - "Automatic triggering of a transition": when the transition is triggered, you can automatically trigger another transition

     ☛ *For more details, see "Implementing scheduled transitions", page 44.*

   - "Automatic triggering of a transition with interaction"

     ☛ *For more details on macros with workflow interaction, see "Implementing Workflow Interactions", page 54.*

   - "Send mail"
   - "Send mail with interaction workflow"
   - "Send notification"
   - "Send notification with interaction workflow"

     ☛ *For more details on macros for sending notifications or e-mails, see "Configuring Actions with Message or Notification", page 56.*

   - "Update attribute for subject"

     ☛ *For more details, see "Managing object status with a specific attribute", page 53.*

4. Select the execution type:
   - Immediate
   - Scheduled

     ☛ *For more details on scheduling transitions, see "Implementing scheduled transitions", page 44.*

   - At dispatch

   Depending on the implementation macro selected and the execution type, different dialog boxes appear.
5. Click **Finish**.
6. In the diagram, connect the action you have just created to the workflow transition (or status) required.

Workflow Advanced Configuration

Managing Workflow Object Statuses

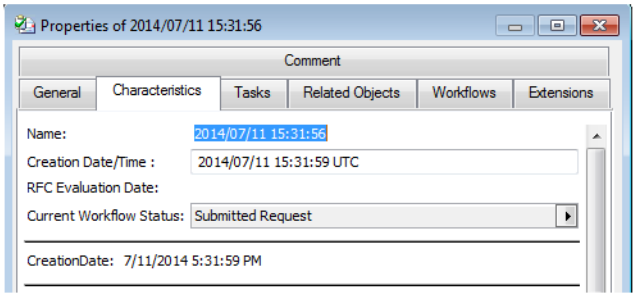
In **HOPEX**, the status of an object can be specified in two ways:

- with a generic attribute
- with a specific attribute

Managing object status with the generic attribute

The workflow engine supplies the generic attribute "Current Workflow Status".

It appears in the **Characteristics** tab of the properties dialog box of the workflow subject (or system workflow subject).



Properties of 2014/07/11 15:31:56

Name: 2014/07/11 15:31:56

Creation Date/Time : 2014/07/11 15:31:59 UTC

RFC Evaluation Date:

Current Workflow Status: Submitted Request

CreationDate: 7/11/2014 5:31:59 PM

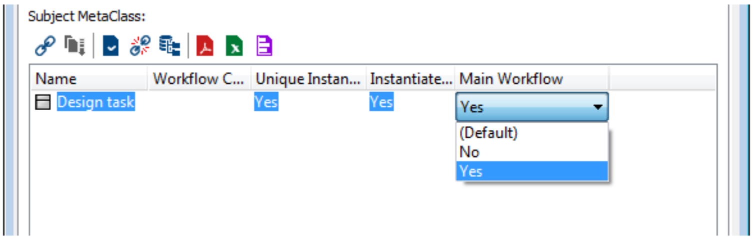
☛ *This attribute is a calculated attribute of identifier type. It contains the identifier of a workflow status.*

Several workflow definitions can be associated with an object. Among these workflow definitions however, only one can supply the object status.

To specify the workflow definition to be taken into account for object status calculation:

1. See "Accessing workflow definitions", page 34.
2. Open the properties dialog box of the workflow definition.

3. In the **Main Workflow** field, select value "Yes".



Several workflow definitions can be defined for the same MetaClass. However, a MetaClass can only have one main workflow definition.

☛ *This is the link between the workflow definition and the subject MetaClass.*

Managing object status with a specific attribute

In certain cases, it can be useful not to use a generic attribute defined at workflow engine level to manage object state. This can be the case when certain workflow statuses are considered too technical and the end user does not wish them to be visible. Here it consists of displaying a workflow state independent of workflow status.

HOPEX enables establishment of mapping between a specific attribute and the workflow status.

The generic macro ***Updating Attribute for Subject*** enables execution of an attribute update according to the current workflow status.

To carry out this configuration, you must:

- define an action.
- associate this action with the generic macro "Updating Attribute for Subject".
- establish mapping between the attribute and the workflow statuses.
  This configuration is carried out in the "_Parameterization" text.
- connect the action to each transition on which you want to update the attribute.

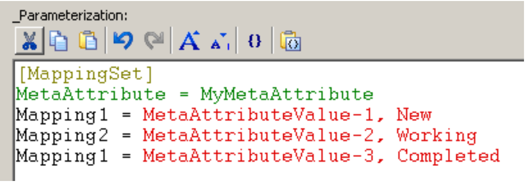
To establish mapping between the specific attribute and the workflow statuses:

1. See "Accessing workflow definitions", page 34.
2. In the action properties dialog box, select the **Characteristics** tab.
3. In the **_Parameterization** frame, insert a section [MappingSet].
4. Indicate the identifier of the specific attribute you previously created.

```
<MappingAttribute>=<AttributId>
```

5. Create the link between:
   - a MetaAttributeValue of the attribute
   - a workflow status

   <Mapping%i%>=<MetaAttributeValueId>,<WorkflowStatutId>



```
[MappingSet]
MetaAttribute = MyMetaAttribute
Mapping1 = MetaAttributeValue-1, New
Mapping2 = MetaAttributeValue-2, Working
Mapping1 = MetaAttributeValue-3, Completed
```

Implementing Workflow Interactions

Workflow interactions principle

From a workflow instance you can act on one or several other workflow instances: this is a workflow interaction.

The workflow interaction calls:

- A source workflow: workflow from which the transition is triggered.
- A target workflow: workflow called from the source workflow.

Example: in the framework of a workflow on action plans, when a workflow is triggered, actions owned by the action plan must be triggered.

You can configure workflow interactions on:

- a workflow action
- a workflow condition

Configuration of a workflow interaction is carried out in two stages. You must use and configure:

- a workflow action macro
- a workflow interaction macro

☛ *Configuration of macros delivered as standard is explained in the **HOPEX** interface, in the tooltip describing the macro.*

Workflow interaction macros

Workflow action macros call workflow interaction macros, which indicate how to access target workflow instances from the source workflow instance.

to create a workflow interaction macro:

❱ Implement the following method:

```
String getWorkflowInstanceTarget

  Context                          as WorkflowContextAction,

  mgcollWorkflowInstanceTarget     as MegaCollection

)
```

where:

- Context is the workflow interaction execution context. This context uses the WorkflowContextAction interface.
- mgcollWorkflowInstanceTarget is the collection of target workflow instances. It is empty: you must fill it.

Workflow interaction examples

With e-mail sending

A user triggers a final workflow transition from a design task. An e-mail is sent to the person responsible for the request for change (if there is a request for change). The e-mail is to be sent only if the request for change is in course of processing.

To configure this workflow interaction:

1. Create an action with the macro "Send mail from transition with interaction workflow", with the following configuration:

   ```
   "Workflow=Target"
   ```

2. Use the workflow interaction macro "Interaction workflow defined with the Subject link on workflow action" with the following configuration:

   ```
   [WorkflowInteraction]
   SubjectLink = ~WceoJb1gEz7R[Motive system of task]
   ```

   ☛ *The link used is [Design task.Motive of task]*

   ```
   [WorkflowTargetCondition]
   WorkflowStatus = ~AwMZq391FDML[Request in progress]
   ```

   The e-mail is sent if the request for change is in "Request in progress" status.

With notification sending

As standard, an interaction exists between:

- the workflow definition of design tasks
- the workflow definition of requests for change

When the last task corresponding to a request for change is completed, a notification is sent to the owner of the request for change.

☛ *For more details on requests for change, see "Using Requests For Change", page 694.*

MANAGING E-MAILS AND NOTIFICATIONS

You can parameterize emails and notifications sent within the framework of workflows.

Configuring Actions with Message or Notification

When you create an action to configure e-mail and notification sending, you can configure fields defining:

- recipients
- subject
- the message body or the notification sent to persons concerned

☛ *For more details on actions, see "Workflow Actions", page 50.*

Defining text

The text of a message or notification can be defined in their properties dialog boxes, or in the workflow action creation dialog box (message or notification).

☛ *See "Workflow Actions", page 50.*

To define the text displayed in a mail or notification, you can:

- enter free text
- use tags

These tags can be used :

- in the subject of the message or notification

  ☛ *The subject corresponds to the first line of the message or notification comment.*

- in the body of the message or notification

Defining recipients

Tags also enable definition of message or notification recipients.

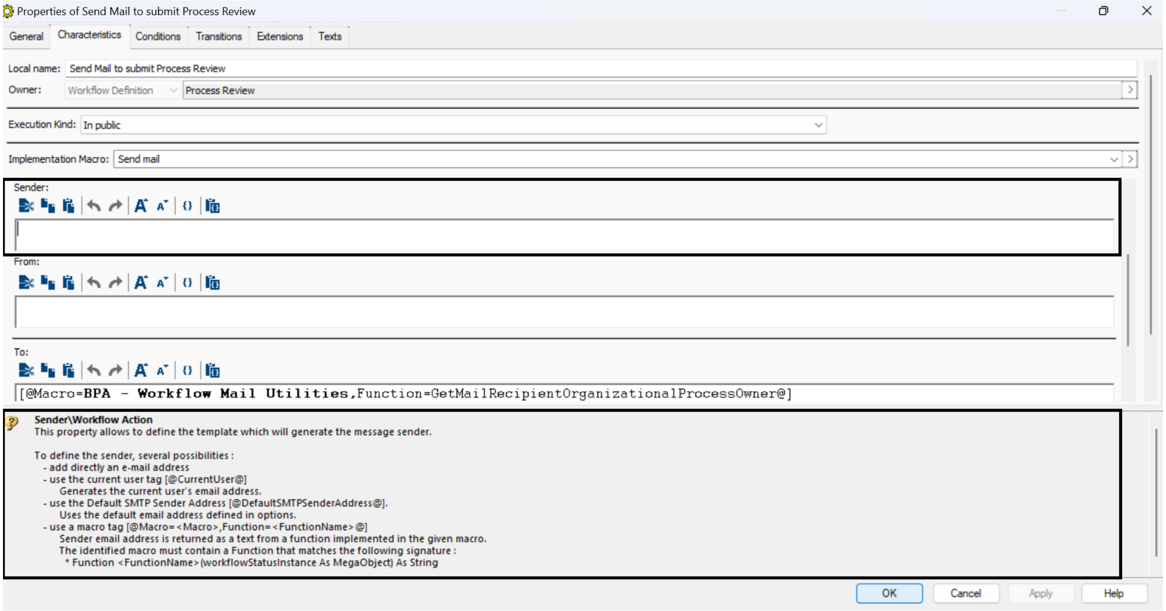
In the case of a message, an e-mail address can be entered directly.

Defining the sender

You can define the sender using one of the following options:

- A permanent email address of your choice
- The email address of the current user
- The default email address specified in the SMTP configuration
- An email address that is returned as a text value

For SaaS solutions: If the SMTP server is managed by MEGA, we recommend using the default SMTP sender address. Otherwise, you may not receive notifications. See "Configuring the SMTP Server".



Local name: Send Mail to submit Process Review

Owner: Workflow Definition Process Review

Execution Kind: In public

Implementation Macro: Send mail

Sender:

From:

To:

```
[@Macro=BPA - Workflow Mail Utilities,Function=GetMailRecipientOrganizationalProcessOwner@]
```

Sender\Workflow Action
This property allows to define the template which will generate the message sender.

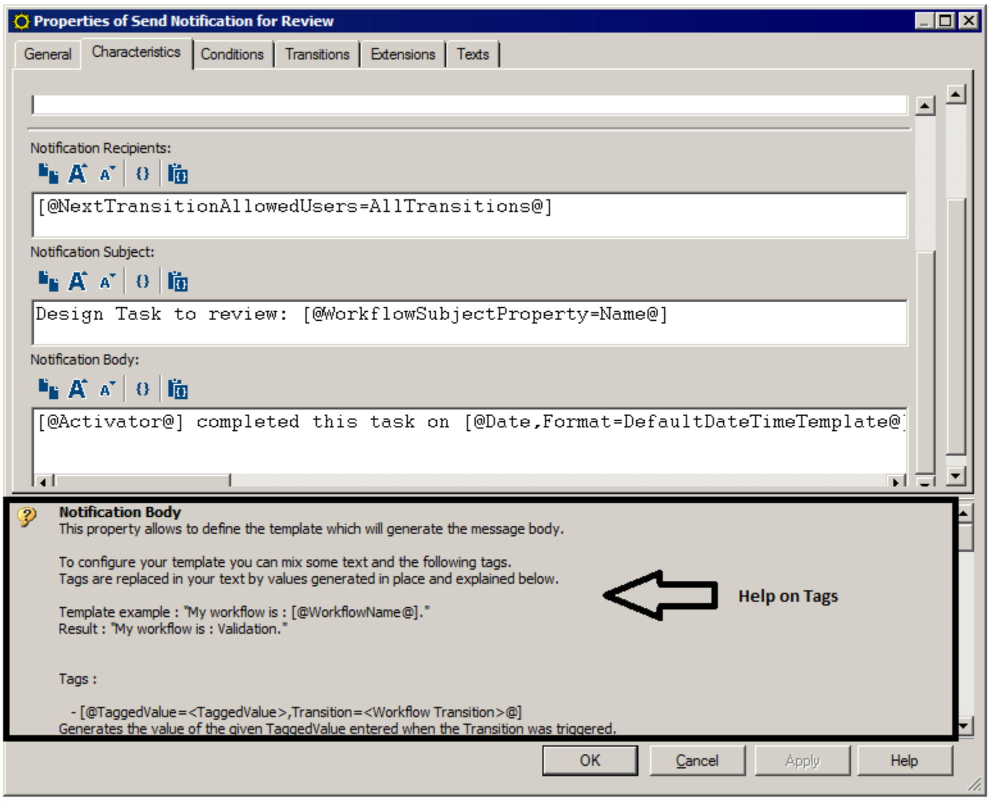
To define the sender, several possibilities :
- add directly an e-mail address
- use the current user tag [@CurrentUser@]
  Generates the current user's email address.
- use the Default SMTP Sender Address [@DefaultSMTPSenderAddress@].
  Uses the default email address defined in options.
- use a macro tag [@Macro=<Macro>,Function=<FunctionName>@]
  Sender email address is returned as a text from a function implemented in the given macro.
  The identified macro must contain a Function that matches the following signature :
  * Function <FunctionName>(workflowStatusInstance As MegaObject) As String

Help on tags

Help on tags used is available in the properties dialog box of a workflow action.

☛ *See also "Workflow Actions", page 50.*

☺ *Enlarge the context-sensitive help pane at the bottom of the dialog box to improve view of the tag description.*



Notification Recipients:

[@NextTransitionAllowedUsers=AllTransitions@]

Notification Subject:

Design Task to review: [@WorkflowSubjectProperty=Name@]

Notification Body:

[@Activator@] completed this task on [@Date,Format=DefaultDateTimeTemplate@]

Notification Body
This property allows to define the template which will generate the message body.

To configure your template you can mix some text and the following tags.
Tags are replaced in your text by values generated in place and explained below.

Template example : "My workflow is : [@WorkflowName@]."
Result : "My workflow is : Validation."

Tags :

- [@TaggedValue=<TaggedValue>,Transition=<Workflow Transition>@]
Generates the value of the given TaggedValue entered when the Transition was triggered.

Help on Tags

Managing Languages

You can specify the language in which you want to send workflow notifications and messages.

Notification languages

All languages specified in the body of the notification are taken in the notification.

E-mail language

E-mails sent following triggering of workflow transitions are in the language of the user.

Managing Objects and Attachments

Managing objects in notifications

The objects you have added on a workflow transition can be added to the notification in the form of a link.

> ☛ *Object types that can form a link in the notification must be associated with the abstract MetaClass "Notification Related Object".*

> ☛ *For more details on objects added on a workflow transition, see "Transition attachments", page 40.*

To add objects to the notification in the form of a link:

1. See "Accessing workflow definitions", page 34.
2. Define a workflow action on a transition or a workflow status and select a macro enabling sending of a notification.

   > ☛ *For more details, see "Creating a workflow action", page 50.*

3. In the properties dialog box of the workflow action on the transition, select according to your requirements:
   - the **Attach the Subject** check box, to add to the notification a link to the workflow subject
   - a macro, in which you must implement the following method:

     Void getAttachments (ActionContext, AttachmentObjectCollection)
   - a query: relating to the subject, if present
   - a motive: all objects corresponding to the motive are added to the notification.

When you execute the workflow, the notification received contains a link to the objects indicated.

Managing attachments in e-mails

In an e-mail, objects you have added on a workflow transition can be the subject of an attachment.

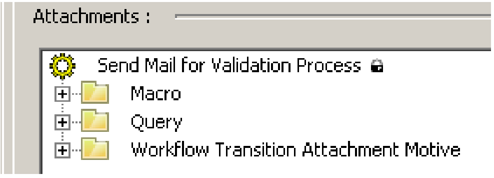
> ☛ *For more details on objects added on a workflow transition, see "Transition attachments", page 40.*

To be able to transform the object into an attachment:

1. See "Accessing workflow definitions", page 34.
2. Define a workflow action on a transition or a workflow status and select a macro enabling sending of an e-mail.

   > ☛ *For more details, see "Creating a workflow action", page 50.*

3. In the properties dialog box of the workflow action on the transition, connect according to your requirements:
   - a macro
   - a query
   - a motive: all objects corresponding to the motive are added to the e-mail.



Attachments :
Send Mail for Validation Process
Macro
Query
Workflow Transition Attachment Motive

Indicating a URL in an E-mail

To be able to add the URL to the **HOPEX** Web Application in an e-mail, you must previously specify access to **HOPEX** in user options.

> ☛ *For more details, see **HOPEX Administration - Supervisor**, chapter "Managing users", paragraph "Managing Users from the Administration Desktop (Web)".*

You must specify the URL in the body of the message. To do this, you can select an object:

- directly via an absolute identifier (IdAbs=),
- by using a query (Query=)
  The query relates to the workflow subject or to the repository root (GetRoot)
- by using a Macro(Macro=)
  The macro must implement the function getUrlObject(WorkflowContextAction) which returns an object.
- by using a workflow subject (Subject).

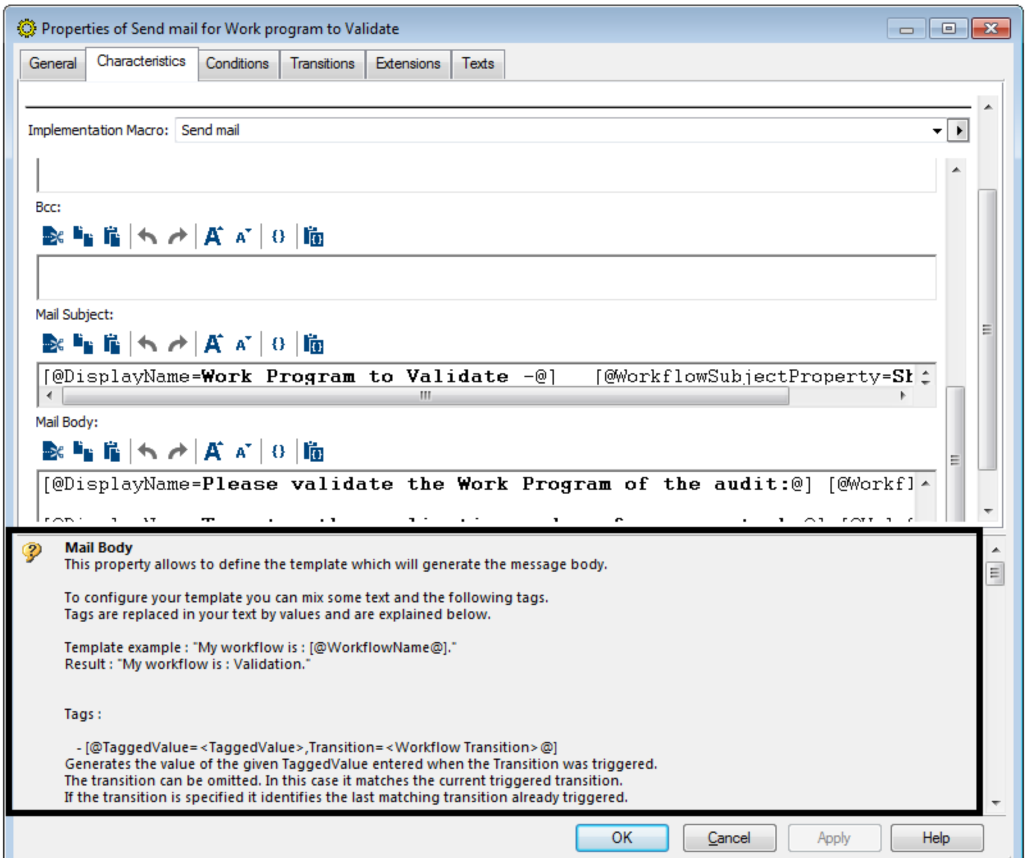
Adding a URL

To add a URL:

1. Open the properties dialog box of the workflow action corresponding to sending of the e-mail.

2. In the **Characteristics** tab, configure the body of the message as indicated in Help.

   ☛ ***To access help, click the Mail Body field. The detail of configuration of body of the message appears at the bottom of the dialog box.***



Implementation Macro: Send mail

Bcc:

Mail Subject:

[@DisplayName=**Work Program to Validate** -@]   [@WorkflowSubjectProperty=Sh

Mail Body:

[@DisplayName=**Please validate the Work Program of the audit:**@] [@Workf]

Mail Body
This property allows to define the template which will generate the message body.

To configure your template you can mix some text and the following tags.
Tags are replaced in your text by values and are explained below.

Template example : "My workflow is : [@WorkflowName@]."
Result : "My workflow is : Validation."

Tags :

- [@TaggedValue=<TaggedValue>,Transition=<Workflow Transition>@]
Generates the value of the given TaggedValue entered when the Transition was triggered.
The transition can be omitted. In this case it matches the current triggered transition.
If the transition is specified it identifies the last matching transition already triggered.

This configuration returns the text of a link of URL type, enabling access to an object from an e-mail.

Opening the application on a specific tab

To allow the user to directly access the appropriate tab in the application, you must carry out an additional configuration, which consists of:

- create an affinity and connect it to the appropriate Desktop Container.
  see "Creating an affinity and connecting it to a Desktop Container", page 62.

  ☛ *A Desktop Container corresponds to a tab in the application.*

- configure the workflow action concerned.
  see "Configuring the workflow action", page 62.

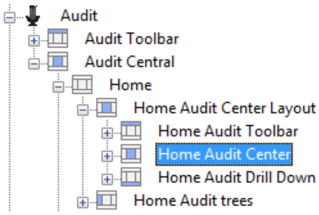
  ☛ *For more details on affinities and Desktop Containers in general, see technical article* ***HOPEX Power Studio - Versatile Desktop****.*

Creating an affinity and connecting it to a Desktop Container

To create an affinity and connect it to a Desktop Container:

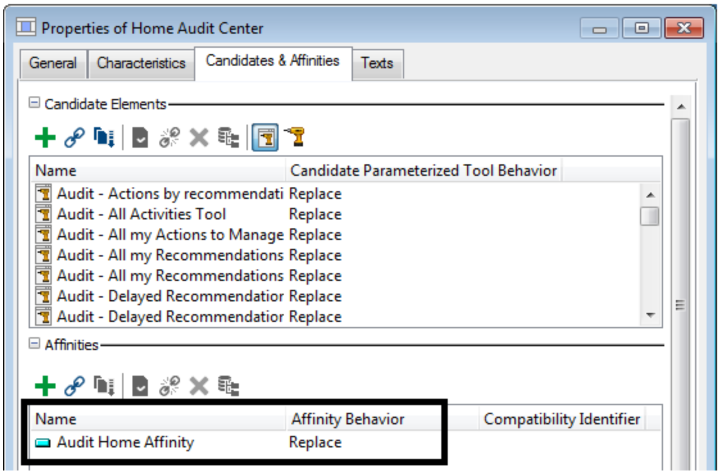
1. In the **MetaStudio** navigation tab, expand the folders to reach the desktop for which you wish to configure access.
2. Select the target Desktop Container.

   ☛ *The Desktop Container concerned must be at the lowest level in the desktop tree.*



Audit
- Audit Toolbar
- Audit Central
  - Home
    - Home Audit Center Layout
      - Home Audit Toolbar
      - Home Audit Center
      - Home Audit Drill Down
    - Home Audit trees

3. In the properties of the Desktop Container, select the **Candidates & Affinities** tab.
4. Create an affinity, and in the **Affinity Behavior** field, select the value "Replace".
   In the example above, an affinity is connected to the "Audit Home" desktop, which corresponds to the Home navigation tab of the **HOPEX** audit solution.



Properties of Home Audit Center

Candidate Elements

Name	Candidate Parameterized Tool Behavior
Audit - Actions by recommendati	Replace
Audit - All Activities Tool	Replace
Audit - All my Actions to Manage	Replace
Audit - All my Recommendations	Replace
Audit - All my Recommendations	Replace
Audit - Delayed Recommendatior	Replace
Audit - Delayed Recommendatior	Replace

Affinities

Name	Affinity Behavior	Compatibility Identifier
Audit Home Affinity	Replace	

This affinity will serve to complete the URL specified in the workflow action corresponding to message sending.

This configuration is supplied by default. If you have customized the standard workflow, you must also perform this customization.

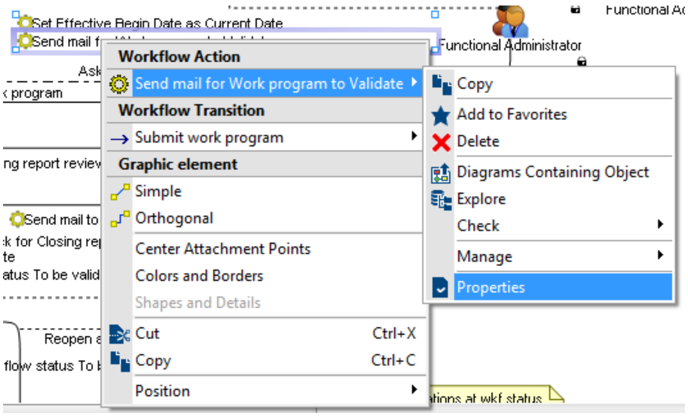
☛ *For your own workflows, you must create and connect an affinity to the appropriate "Desktop container" in the same way.*

Configuring the workflow action

You must now configure the workflow action concerning e-mail sending by referencing the previously created affinity.

To configure the corresponding workflow action:

1. In the workflow definition diagram of the workflow to be modified, right-click the workflow action to be configured and select **Properties**.



2. In the properties of the workflow action, select the **Texts** tab and in the drop-down list select **Mail Body**.
   In the message configuration text, you must complete URL configuration by adding a a string of this type:

   ,Affinity=~OhiFhJ5eJXd9[Audit Home Affinity]

Example of text to be inserted in body of message

[@DisplayName=~htNzOjVgGHTM[To access the application and execute your task,]@][@Url,Subject,Tool=PropertyPage,Application=~2zopd2TnFDcL[GRC Solutions],Desktop=~2hrUiL8DGX58[Audit],Name=~w7LgEpROGXPB[Click here]***,Affinity=~OhiFhJ5eJXd9[Audit Home Affinity]***@].

Result obtained

When the user clicks a link in the e-mail received, the application opens.

When configuring a workflow supplied as standard, you may need to configure the **Collaboration** tree, for example by adding folders in the **Collaboration** navigation window (in the case of adding new workflow statuses).

To do this, see "Configuring Navigation Trees", page 115.

☛ *Configuring a workflow requires expertise in **MEGA APIs**. You can if necessary seek the assistance of a product engineer. See also "All about starting with APIs" for more information on using **HOPEX** APIs.*

DUPLICATING A WORKFLOW DEFINITION

You can initialize a new *workflow definition* by duplicating an existing workflow definition. Similarly, when you wish to configure a workflow supplied as standard, you must duplicate the corresponding workflow definition.

💣 **Do not modify the workflow definitions proposed by HOPEX to avoid loosing modifications when updating to a new version of HOPEX.**

To duplicate a workflow definition:

1. From the **Utilities** navigation window, expand the "Workflow Definitions" folder.
2. From the pop-up menu of the workflow definition, for example "Request For Change", select **Duplicate**.
3. Enter the name of the new workflow definition, as well as the prefix or suffix used to name duplicated objects.
   The new workflow definition is displayed in the "Workflow Definitions" folder.

When the workflow definition has been duplicated, you must:

- deactivate the obsolete workflow definition
- activate the duplicate of the workflow definition when you have completed specification

To activate or deactivate a workflow definition:

❱ In the workflow definition dialog box, select or clear the **Active Workflow Definition** check box.

☛ *If the workflow definition is deactivated the workflow cannot be started.*

MANAGING WORKFLOWS

This chapter describes how to manage workflows.

- ✓ Workflow Administration Options
- ✓ Managing Workflow Users
- ✓ Managing Permissions on Workflows

WORKFLOW ADMINISTRATION OPTIONS

Accessing Workflow Options

To access workflow options:

1. See Modifying options at environment level.
2. Expand **Tools > Collaboration > Workflows**.

Workflow Display Rights

To grant the right to display workflows:

1. See Accessing Workflow Options.
2. In the right pane, select **View workflows of all users**.

Workflow Administration Rights

To grant the right to manage workflows:

1. See Accessing Workflow Options.
2. In the right pane, select **Workflow Administrator**.

Managing Workflow Users

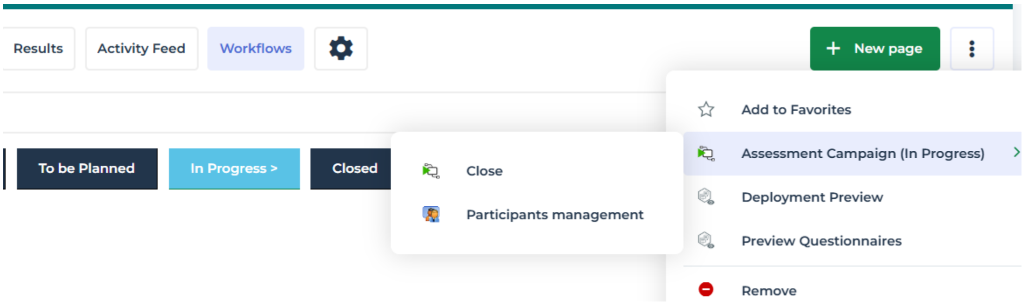
To manage the users of workflow instances you must have administration rights for workflows.

☛ *For more details, see Workflow Administration Rights.*

In this context, "workflow user" corresponds to the list of users that can trigger a workflow transition.

To manage workflow users:

1. In the **Workflows** page of the of the object properties, click the 3 dots and select **Participants management** as follows:



   The users who can trigger the next workflow transition appear.

   ☛ *If different workflow transitions are possible and different participants were defined for each of them in the workflow definition, **HOPEX** provides a list of transitions.*

2. Click on a transition to modify the assigned users.
   The users currently assigned appear.

   You can remove a user and add another in the list provided for this purpose.

Example of how to use this functionality

Mrs. White was assigned to perform a particular transition (Request Evaluation, for example). If she is on holiday and the workflow must take place in her absence, you can add or remove another user here.

This user can connect to MEGA and perform the transition on behalf of Mrs. White.

MANAGING PERMISSIONS ON WORKFLOWS

☛ *You can manage permissions in* ***HOPEX Web Front-End*** *only.*

Configuring Permissions (CRUDs) on Objects

You can configure permissions on objects according to:

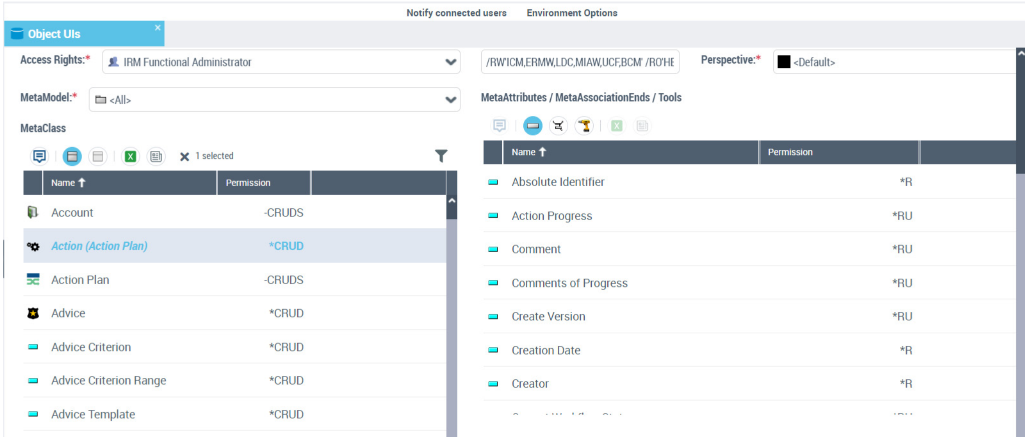
- user profile
- current workflow status.

☛ *Permissions (CRUDs) concerning workflows are not taken into account in the diagram editor.*

To configure permissions:

1. In **HOPEX Administration**, open the environment.
2. Expand the **Permissions** navigation pane.
3. Under **CRUD Management**, select **Object UI Access**.
4. In the **Access Rights** field, select the **Profile** you want to configure.
5. In the **MetaClass** frame, select the workflow subject MetaClass.

   Example: "Action plan"



Access Rights:* IRM Functional Administrator

/RW'ICM,ERMW,LDC,MIAW,UCF,BCM' /RO'HE

Perspective:* <Default>

MetaModel:* <All>

MetaClass

1 selected

Name	Permission
Account	-CRUDS
Action (Action Plan)	*CRUD
Action Plan	-CRUDS
Advice	*CRUD
Advice Criterion	*CRUD
Advice Criterion Range	*CRUD
Advice Template	*CRUD

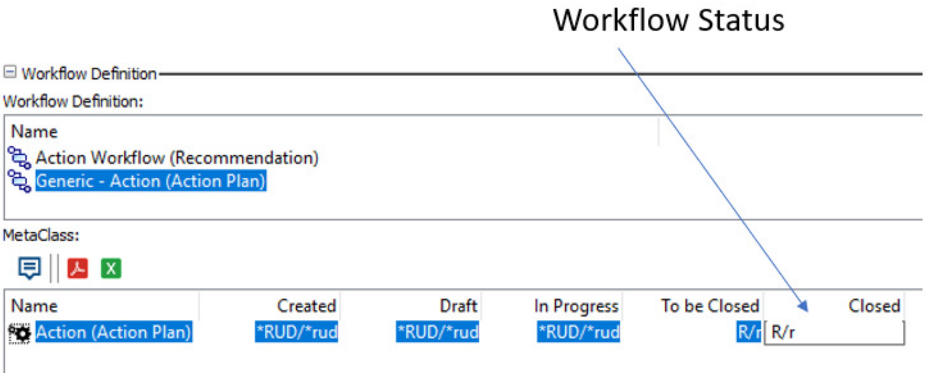
MetaAttributes / MetaAssociationEnds / Tools

Name	Permission
Absolute Identifier	*R
Action Progress	*RU
Comment	*RU
Comments of Progress	*RU
Create Version	*RU
Creation Date	*R
Creator	*R

6. In the **Workflow Definition** section, select the workflow for which you want to define permissions.

7. Configure permissions on the MetaClass subject of the workflow according to the status reached by the workflow.

   Example: The permission "R" (=Read) in status "Closed" means that you cannot modify the action plan when it is in this status.



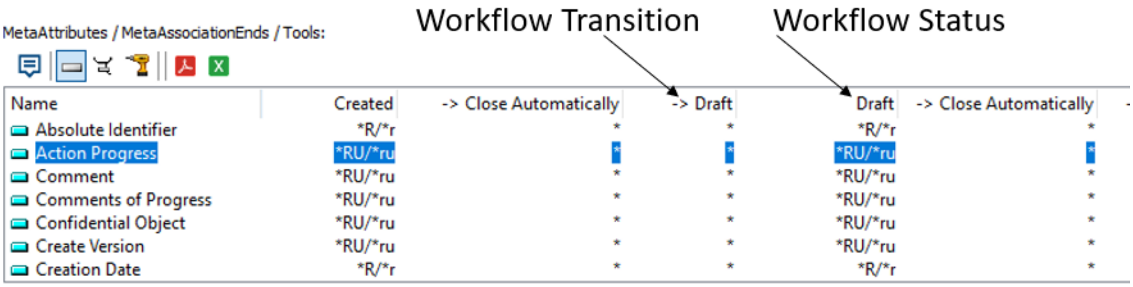
Workflow Definition:

Name
Action Workflow (Recommendation)
Generic - Action (Action Plan)

MetaClass:

Name	Created	Draft	In Progress	To be Closed	Closed
Action (Action Plan)	*RUD/*rud	*RUD/*rud	*RUD/*rud	R/r	R/r

8. Similarly configure attributes and links available on this MetaClass.



MetaAttributes / MetaAssociationEnds / Tools:

Name	Created	-> Close Automatically	-> Draft	Draft	-> Close Automatically	-
Absolute Identifier	*R/*r	*	*	*R/*r	*	
Action Progress	*RU/*ru	*	*	*RU/*ru	*	
Comment	*RU/*ru	*	*	*RU/*ru	*	
Comments of Progress	*RU/*ru	*	*	*RU/*ru	*	
Confidential Object	*RU/*ru	*	*	*RU/*ru	*	
Create Version	*RU/*ru	*	*	*RU/*ru	*	
Creation Date	*R/*r	*	*	*R/*r	*	

Several values are possible on workflow statuses or transitions:

- indicates that the field (attribute or link) is mandatory in this workflow status.
  The user must specify this field to trigger the next workflow transition.
- " RU" (R=Read, U=Update): on a workflow transition, means that the field is proposed in edit mode in the user interface proposing the transition.

> ☛ *"RUM" means that the user must specify the field when triggering the workflow transition.*

Different permissions can be specified for workflow participants and non-participants:

- Values of permissions in upper-case, for example "RU" concern workflow participants.
- Values of permissions in lower-case, for example "ru" concern persons not participating in the workflow.

> ☛ *In the **Transition Attributes** frame, you can specify permissions on TaggedValues created on your workflow transitions. For more details, see Information associated with workflow transition.*

Specifying Persons Authorized to Start a Workflow

To specify persons authorized to start a workflow, see Associating persons with participants.

Generating a Report of Permissions by Workflow Definition

A report allows you to detail permissions for a given workflow.
Parameters are as follows:

- a workflow definition
- a profile

See Generating a Report on Permissions by Profile.

Using the Scheduler

Scheduler

1 INTRODUCTION

1.1 Aim of this documentation

The aim of this documentation is to describe the Scheduler component embedded into HOPEX platform:

- what it is used for
- how to configure it
- how to add scheduled jobs

1.2 Presentation

The Scheduler component enables to execute a macro at a given date and time.

Example: the following features rely on the Scheduler possibilities:

- Steering Calendars
- Alignment: Automatic transfers
- Scheduled Actions in Workflows
- Reminders
- Scheduled transitions
- Assessment Campaign

Macro execution scheduling combinations are the following:

- Trigger at a date and time
- Recurrent triggering (daily, weekly, monthly)
- Trigger relative to another reference date

1.3 Requirements

The Scheduler component meets the following requirements:

- Repository requirement: RDBMS repository
- SOA deployment (see Architecture section)
    - Scheduler is a web service hosted into the MOS, it requires the MOS to be deployed and configured
    - MOS can be deployed either with Windows front end or HOPEX Web front end, so that Scheduler can be used in a full Windows front end environment, in a HOPEX Web front end environment, or in a mixed environment
- Availability: The scheduler is available with products containing features relying on the Scheduler like HOPEX Alignment and HOPEX Assessment. For an exhaustive list of products in which the Scheduler is available, see each specific product description documentation.

1.4 Limitation

Due to date/time internal storage in HOPEX application, the Scheduler does not currently manage date/time after year 2038.

This limitation will be overridden in future versions.

1.5 Architecture

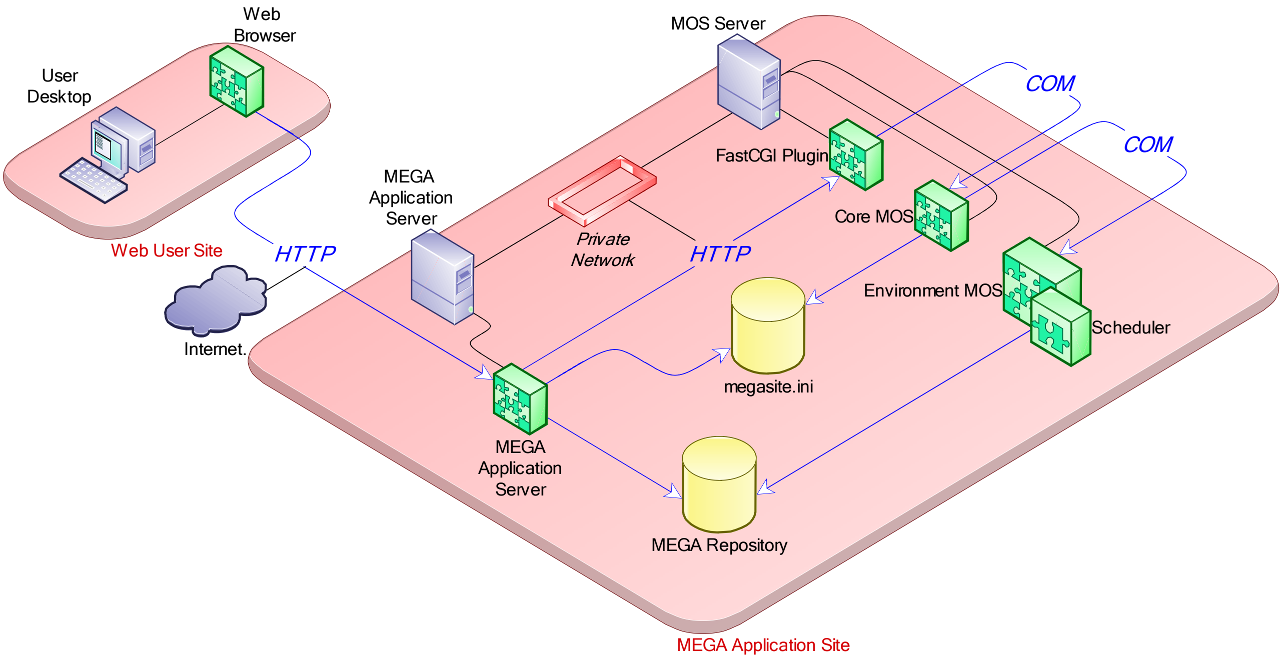
The Scheduler component is hosted into the MOS.

There is one Scheduler instance per environment.

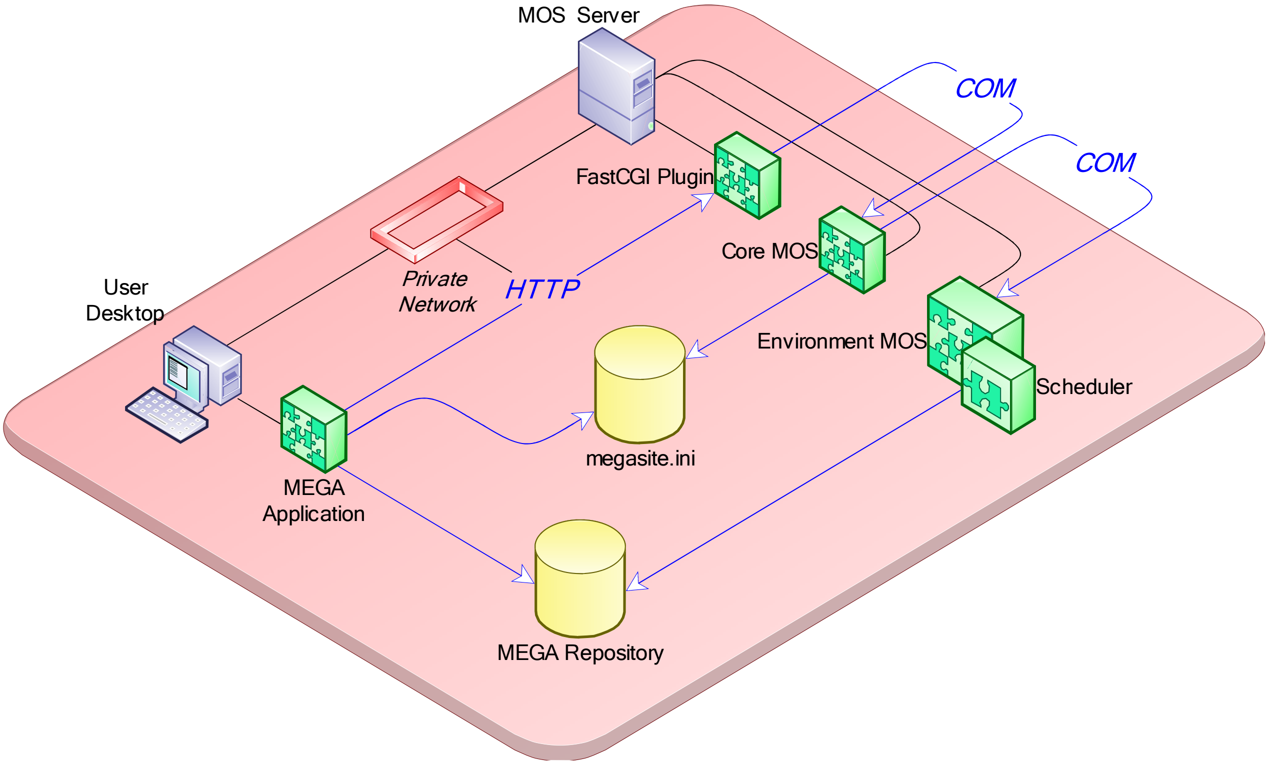
The HOPEX Application client requires an HTTP connection to the Core MOS. The Core MOS application delegates the Scheduler solicitations to the matching Environment MOS.

Scheduler clients have access to the Scheduler using a specific API (see Scheduler Client API section).

1.5.1 Web Front-End architecture

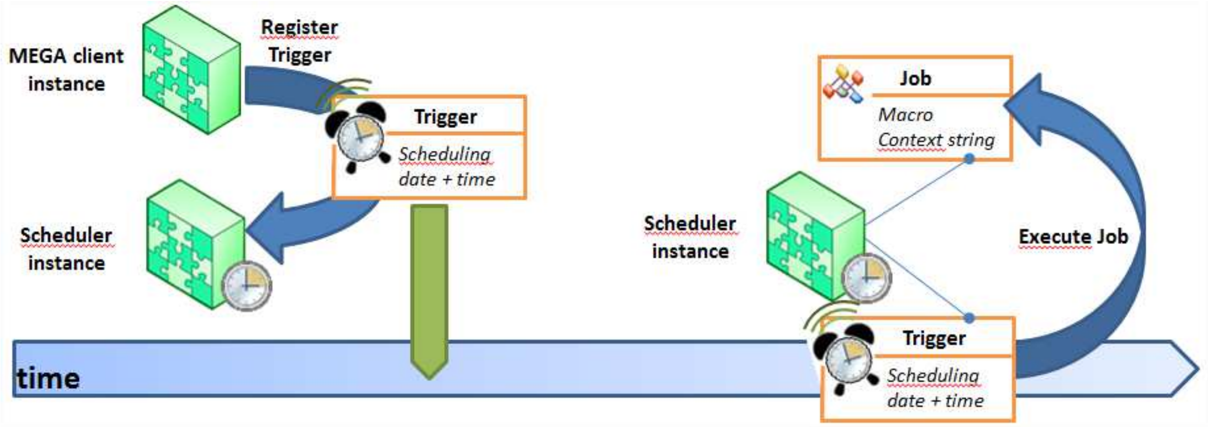


1.5.1 Windows Front-End architecture



2 SCHEDULER PRINCIPLES

2.1 Concepts



2.1.1 Job

The Job is the item identifying the Macro to be executed. It includes a “Context” string used to pass needed information at Macro execution.

2.1.2 Trigger

A Trigger is associated to the Job to define when to execute this Job.

The Scheduling enables to define when (date & time) to execute the Job and which frequency to apply if any.

2.1.3 Scheduler

The Scheduler component provides the following services:

- Add a Trigger (same as register a Trigger)
- Delete a Trigger
- Find existing Triggers
- Modify a Trigger
- Execute Jobs associated to Triggers at the date and time specified in the Scheduling

2.2 Scheduler execution details

2.2.1 Persistance

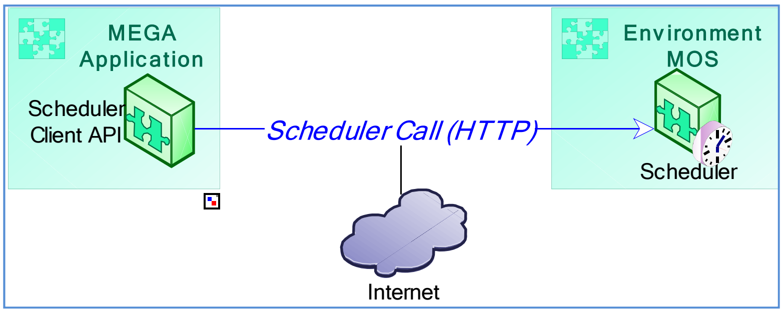
The Scheduler component instantiated into the Environment process loads and manage the Triggers into its memory.

Triggers and Jobs are replicated into the Repository. In case the process hosting the Scheduler dies, all Triggers and Jobs are recovered.

Consistency of the Triggers and Jobs are managed by the Scheduler component and Scheduler Client API. It is forbidden to handle Triggers and Jobs repository objects.

2.2.2 Scheduler API

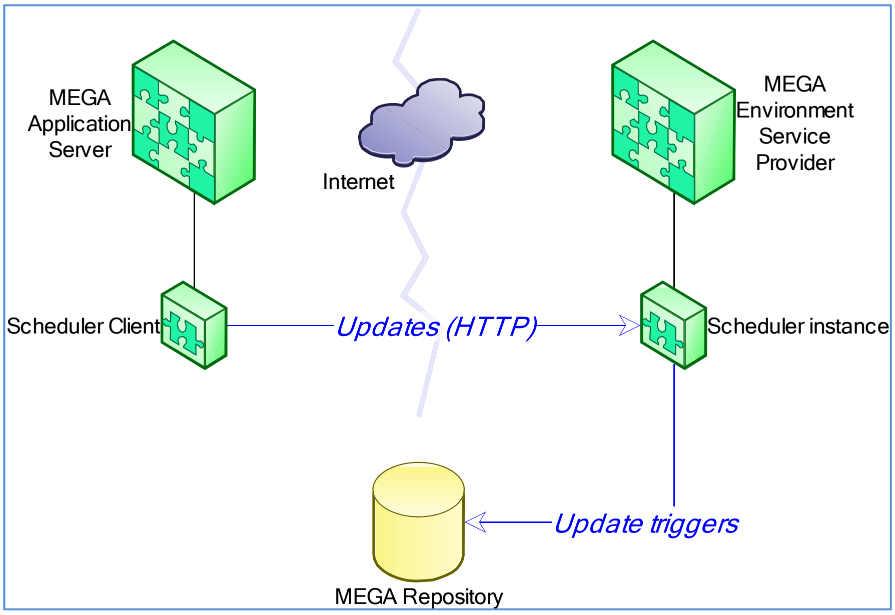
A dedicated API is provided to access Scheduler services. This API is called the Scheduler Client API (see Scheduler Client API section).



You must use the Scheduler Client API for any operation (Add, Remove, Update, Find) performed with the Scheduler. **This API guaranties Triggers and Jobs consistency.**

2.2.3 Scheduler and Workspaces

All operations on the Scheduler are done by default via an HTTP call so that they are executed as soon as the HTTP call is received and processed. The operation is executed without any dependency with the current opened Workspace in the MEGA Application process.



1 WebService mode

Several Scheduler operations update Triggers of are registered into a Scheduler instance:

- Add a Trigger
- Delete a Trigger
- Modify a Trigger

Update mode

If the Scheduler operation depends on changes made into the current Workspace and this operation is executed outside the current Workspace, this could lead to inconsistencies.

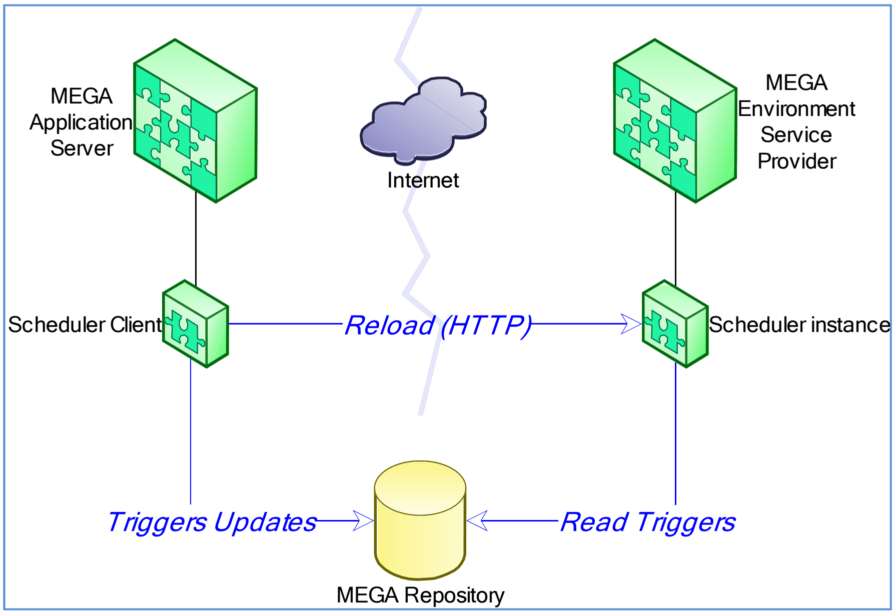
Example:

In a Workspace, an object and a Scheduled Job to process something later on that object are created. If the Workspace is discarded and the Trigger is registered, when the matching Job is executed, it can generate errors, or worse, apply inconsistent updates.

Specify the "*update modes*" for the Scheduler update operations:

- The "Web Service" mode
- The Transactional mode

A Scheduler operation executed in Transactional mode is effective into the Scheduler only once the work in the Workspace has been dispatched. If the work is discarded, the Scheduler operation is also discarded.



2 Tansactional mode

2.2.4 **Job execution: user/profile/repository**

All operations into a MEGA process require a connection (Session & private Workspace):

- A connection is necessary prior to Job execution.
- A Scheduler instance is hosted into an Environment MOS, meaning it is global to an Environment.
- The Job connection implies to know which user connects, to which repository, with which profile.

For security reasons, the user used to execute a Job is always the one who added (registered) the matching Trigger from the MEGA Application Client into the Scheduler instance. Profile and Repository can be specified as parameters of the AddTrigger operation. By default Profile and Repository are the ones specified at the Trigger registering.

3 SCHEDULER CONFIGURATION

Scheduler requires a properly configured MOS. One Scheduler instance is automatically instantiated into each Environment MOS.

3.1 HOPEX installations without Scheduler

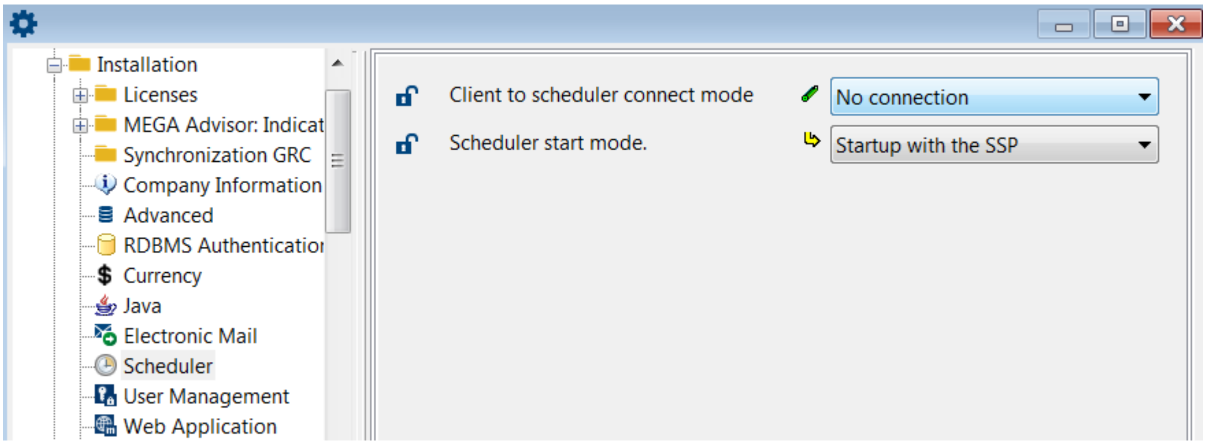
Some standard features (as provided standard Workflow Definitions) require a Scheduler instance. But Scheduler needs MOS and RDBMS repository.

For features relying on Scheduler to be executed, an option enables to deactivate the Scheduler so that all Scheduler operations calls are ignored.

To deactivate dependencies with Scheduler:

1. Go to HOPEX Options: **Installation > Scheduler.**

2. In the right pane, set the **Client to scheduler connect mode** option value to "No connection".



Client to scheduler connect mode: No connection

Scheduler start mode.: Startup with the SSP

3.2 Traceability

The **Scheduler Tracking Level** option enables to log the Scheduler activity into the MEGA error log (ssperrYYYYmmDD.txt file).

All the lines regarding the scheduler activity contain `[Scheduler:` .

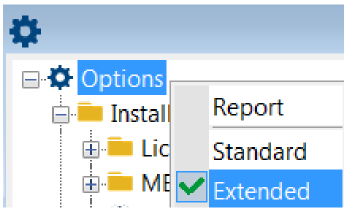
The Scheduler tracking levels are:

- 0: no information logged
- 1: global scheduler actions
  - Scheduler start
  - Scheduler stop
  - Scheduler Triggers reload from repository
- 2: updates and all client side calls
  - Add Trigger
  - Remove Trigger
  - Update Trigger
  - Get Trigger
  - All client side calls
- 3 : job executions
  - Job triggering
  - Job execution

To follow up Scheduler activity:

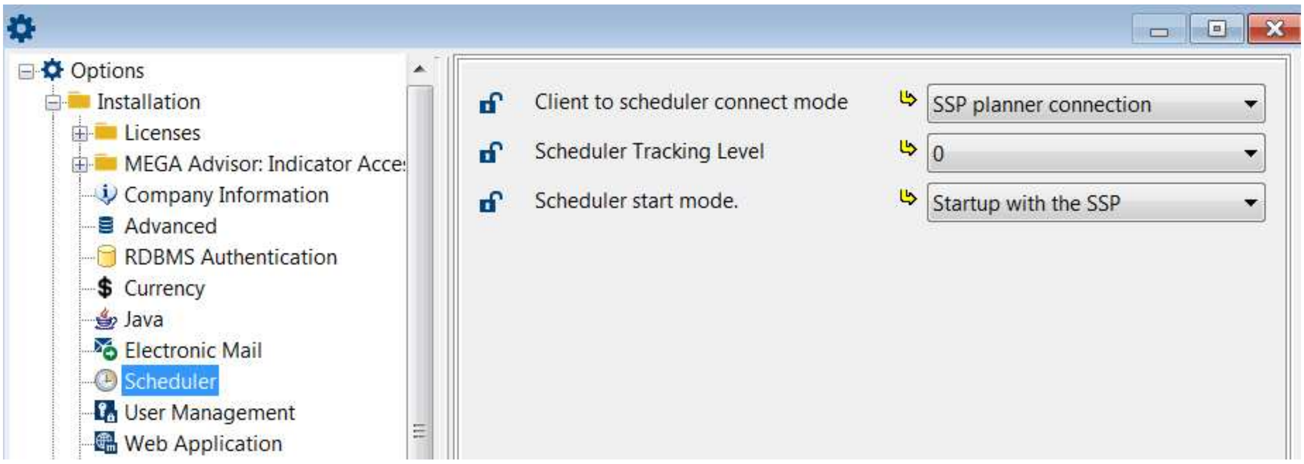
1. Access HOPEX Options.

2. Check that your Option level is "Extended" (right-click **Options** and select "Extended").



3. Expand **Installation** folder and select **Scheduler**.

4. In the right pane, select the **Scheduler Tracking Level** option value.



Client to scheduler connect mode: SSP planner connection

Scheduler Tracking Level: 0

Scheduler start mode.: Startup with the SSP

5. Reboot the SSP for this option to be taken into account.

   The scheduler activity is written into the ssperrYYYYmmDD.txt file.

Example :

```
PID(2764);Thread(4448);Time(00:10:03);Session(SysMA
A);[PID(0x00000ACC),Thread(0x00001160),Time(00:10:03)] [Scheduler:TriggerJob]
[TriggerId(7BBA7C5F528C36D4):JobId(7BBA7C83528C3705):JobMacroId(7BBA7C23528C363F):JobExecUs
erId(0D510D8B2E210068):JobExecProfileId(0000000000000000):JobExecBaseId(0000000000000000):N
extDateTime(2017/06/14 22:10:00):NbRepeat(15)]
PID(2764);Thread(4448);Time(00:10:03);Session(SysMA
A);[PID(0x00000ACC),Thread(0x00001160),Time(00:10:03)]
[Scheduler:CallExternalProcessToExecuteJob] [TriggerName(Indexing
Automaton):TriggerId(7BBA7C5F528C36D4):JobId(7BBA7C83528C3705):JobMacroId(7BBA7C23528C363F)
:JobExecUserId(0D510D8B2E210068):JobExecProfileId(0000000000000000):JobExecBaseId(000000000
0000000):NextDateTime(2017/06/14
22:10:00):NbRepeat(15):ExecutionMode(INDEXATION):ReadWriteMode(W)]
PID(2764);Thread(4448);Time(00:10:03);Session(SysMA
A);[PID(0x00000ACC),Thread(0x00001160),Time(00:10:03)]
[Scheduler:CallExternalProcessToExecuteJob:Result] [OK(1)] [TriggerName(Indexing
Automaton):TriggerId(7BBA7C5F528C36D4):JobId(7BBA7C83528C3705):JobMacroId(7BBA7C23528C363F)
:JobExecUserId(0D510D8B2E210068):JobExecProfileId(0000000000000000):JobExecBaseId(000000000
0000000):NextDateTime(2017/06/14
22:10:00):NbRepeat(15):ExecutionMode(INDEXATION):ReadWriteMode(W)]
PID(9308);Thread(19496);Time(00:10:03);Session(SysMA A (System Base
closed));[PID(0x0000245C),Thread(0x00004C28),Time(00:10:03)] [Scheduler:ExecuteJob]
[TriggerName(Indexing
Automaton):TriggerId(7BBA7C5F528C36D4):JobId(7BBA7C83528C3705):JobMacroId(7BBA7C23528C363F)
:JobExecUserId(0D510D8B2E210068):JobExecProfileId(0000000000000000):JobExecBaseId(000000000
0000000):NextDateTime(2017/06/14
22:10:00):NbRepeat(15):ExecutionMode(INDEXATION):ReadWriteMode(W)]
PID(2764);Thread(4448);Time(00:10:03);Session(SysMA
A);[PID(0x00000ACC),Thread(0x00001160),Time(00:10:03)] [Scheduler:TriggerJob:Result]
[OK(1):NextDateTime(2017/06/14 22:20:00)]
PID(9308);Thread(19496);Time(00:10:03);Session(SysMA
A);[PID(0x0000245C),Thread(0x00004C28),Time(00:10:03)]
[SchedulerClient:HandleScheduledJobEvent(SSP Connection)]
```

4 SCHEDULER CLIENT API

All features relying on the Scheduler **must** use the **SchedulerClient** API to add, remove, update or get Triggers to or from the Scheduler instance.

Direct accesses into the repository to System Triggers, System Jobs are forbidden.

4.1 Detailed description of the SchedulerClient API

The **SchedulerClient** API is detailed into the JavaDoc provided with HOPEX installation.

See "*MegaSchedulerClient*" into either:

- online documentation: ***HOPEX Power Studio > HOPEX APIs > JavaDoc***
- ***java\doc*** folder of HOPEX installation site: ***mj_api.doc.zip*** documentation package.

4.2 Java

4.2.1 Documentation

See "*MegaSchedulerClient*" into either:

- online documentation: ***HOPEX Power Studio > HOPEX APIs > JavaDoc***
- ***java\doc*** folder of HOPEX installation site: ***mj_api.doc.zip*** documentation package.

4.3 VB Script

4.3.1 Documentation

To get access to the **SchedulerClient** API, use the ***SchedulerClient*** Method available on MegaRoot.

Example:

```
Set objSchedulerClient = GetRoot().InvokeFunction("~nNmUTwNTF94K[SchedulerClient]")
```

Members available on the **SchedulerClient** object are defined below (for detailed description see JavaDoc documentation):

```
Class SchedulerClient
  Public Property Get State() As String
  Public Property Get ConnectionMode() As Integer
  Public Property Let ConnectionMode(Integer intConnectionMode)
  Public Property Get DefaultUpdateMode() As Integer
  Public Property Let DefaultUpdateMode(Integer intDefaultUpdateMode)
  Public Sub StartScheduler()
  Public Sub StopScheduler()
  Public Sub ReloadTriggers()
  Public Function NewTrigger() As MegaObject
  Public Sub AddTrigger(mgobjTrigger As MegaObject, strOptions As String)
  Public Sub RemoveTrigger(mgobjTriggerId As Object, strOptions As String)
  Public Sub UpdateTrigger(mgobjTrigger As MegaObject, strOptions As String)
  Public Function GetTrigger(mgidTriggerId As Object, strOptions As String) As MegaObject
  Public Function GetAllTriggers() As MegaCollection
End Class
```

4.3.2 Example 1 : Registering a Trigger

```
Dim mgRoot
Dim objSchedulerClient
Dim mgobjScheduling
Dim mgobjScheduledTrigger
Dim mgobjScheduledJob

Set mgRoot = GetRoot()

' Get the SchedulerClient API object
Set objSchedulerClient = mgRoot.SchedulerClient

' Allocate a "System Trigger" MegaObject to be configured
Set mgobjScheduledTrigger = objSchedulerClient.NewTrigger()
' Get the Scheduling and "System Job" MegaObject to be configured
Set mgobjScheduling =
mgobjScheduledTrigger.GetCollection("~e)PMRueCFbLR[Scheduling]").Item(1)
Set mgobjScheduledJob = mgobjScheduledTrigger.GetCollection("~pWuUJAATF5mD[System
Job]").Item(1)

' Configure the Scheduling
mgobjScheduling.SetProp "~azPMryeCFfRR[Scheduling.IsRelative]", "A"
mgobjScheduling.SetProp "~h)PMS5fCFTXR[Scheduling.RepeatKind]", "S"
' Set the start at Now + 30 seconds
mgobjScheduling.SetProp "~CaW0Vl0PG1pC[Scheduling.Start.AbsDateTime]",
DateAdd("s",30,Now)

' Configure the System Job (with dumy macro and contextstring, just for example)
mgobjScheduledJob.GetProp("~210000000900[Name]") = "My Job"
mgobjScheduledJob.GetProp("~MXuU7x9TFLlD[Implementation Macro]") = "~oHDN0jc0I1p4[My
Macro]"
mgobjScheduledJob.CallMethod "~zdj0fUG4GXRM[SetContextString]", "my informations"

' Register the System Trigger into the Scheduler instance
objSchedulerClient.AddTrigger mgobjScheduledTrigger, "UPDATEMODE=TRANSACTIONAL"
```

4.3.3 Example2 : three ways to setup the scheduling

You can use different ways to setup the scheduling:

- Configure the Scheduling directly using the Scheduling MegaObject properties

```
' Get the Scheduling MegaObject to be configured
Set mgobjScheduling =
mgobjScheduledTrigger.GetCollection("~e)PMRueCFbLR[Scheduling]").Item(1)
' Set the Scheduling date as absolute
mgobjScheduling.SetProp "~azPMryeCFfRR[Scheduling.IsRelative]", "A"
' Set the Scheduling as single start
mgobjScheduling.SetProp "~h)PMS5fCFTXR[Scheduling.RepeatKind]", "S"
' Set the start at Now + 30 seconds
mgobjScheduling.SetProp "~CaW0Vl0PG1pC[Scheduling.Start.AbsDateTime]",
DateAdd("s",30,Now)
```

- Setup the Scheduling from a Scheduling definition stored in an XML string

```
' Get the Scheduling MegaObject to be configured
Set mgobjScheduling =
mgobjScheduledTrigger.GetCollection("~e)PMRueCFbLR[Scheduling]").Item(1)
' Setup the Scheduling from Xml string
mgobjScheduling.CallMethod "~ sJYouhVuFP7S[UpdateFromString]", strXmlSchedulingDefinition
```

- Recover the scheduling from a repository object on which it is defined

The repository object must have a "*~iUhj4likEzJQ[Scheduling]*" MetaAttribute. The Scheduling definition is stored into the standard "*~iUhj4likEzJQ[Scheduling]*" MetaAttribute.

In this example, the Scheduling definition is set as relative. Only the reference date and time have to be set.

```
' Get the repository object on which the Scheduling is defined
Set mgobjSteeringDate = mgobjSteeringCalendar.GetCollection("~qcuD3s3SFXL7[Steering
Date]")
' Get the Scheduling MegaObject to be configured
Set mgobjScheduling =
mgobjSteeringDate.GetCollection("~e)PMRueCFbLR[Scheduling]").Item(1)
' Set the reference date and time (VBScript Date type)
mgobjScheduling.getProp("~V(PMgweCF5OR[Scheduling.Reference.DateTime]")=
dtReferenceDateTime
```

5 SCHEDULING

The Scheduling must be defined when adding (registering) a Trigger into the Scheduler. Scheduling information must be passed as an XML string.

Abstract layers enable to:

- handle the Scheduling information as a MEGA object. In this form, the Scheduling information is the set of properties of this MEGA object and this enables easy update via MegaObject API.
- give access - for reading and updating - to the Scheduling information into reusable MetaPropertyPages.

5.1 Scheduling information: XML Scheduling format

Date and time are interpreted as UTC date and time.

```
<?xml version="1.0" encoding="utf-8"?>
<scheduling>
  <reference                        (defined for relative scheduling)
    date="yyyy/mm/dd"
    hour="hh:mm:ss"
    />
  <start
    date="yyyy/mm/dd"               (defined for absolute scheduling)
    hour="(A/R)hh:mm:ss"            (A=> absolute time, R=> relative: adds to current
time)
    executeatstart="y/n"            (y: launches the trigger at defined start date and
time
                                     n: only launches at recurrent date and time)
    >
    <relativedate                   (defined for relative scheduling)
      daysfromreference="(-)1"      (cannot be used with "dayofweek" or "dayofmonth")
      dayofweek="Su"                (cannot be used with "daysfromreference" or
"dayofmonth")
      dayofmonth="01"               (cannot be used with "daysfromreference" or
"dayofweek")
      weeksfromreference="(-)1"     (in case "dayofweek")
      weekofmonth="1"               (in case "dayofweek")
      monthsfromreference="(-)1"    (in case "dayofmonth" or "weekofmonth")
      month="Jan"                   (in case "dayofmonth" or "weekofmonth")
      />
  </start>
  <dailyrepeat                      (cannot be used with <weeklyrepeat> or
<monthlyrepeat>)
    daysperiod="1"
    >
    <endrepeat
      noend="y/n"                   (cannot be used with "repeatnumber" or "date" and
"time")
      repeatnumber="1"              (cannot be used with "noend" or "date" and "time")
      date="yyyy/mm/dd"             (cannot be used with "repeatnumber" or "noend")
      hour="(A/R)hh:mm:ss"
      >
      <relativedate                 (defined for relative scheduling)
        daysfromreference="(-)1"    (cannot be used with "dayofweek" or "dayofmonth")
        dayofweek="Su"              (cannot be used with "daysfromreference" or
"dayofmonth")
        dayofmonth="01"             (cannot be used with "daysfromreference" or
"dayofweek")
```

```
        weeksfromreference="(-)1"   (in case "dayofweek")
        weekofmonth="1"             (in case "dayofweek")
        monthsfromreference="(-)1"  (in case "dayofmonth" or "weekofmonth")
        month="Jan"                 (in case "dayofmonth" or "weekofmonth")
        />
    </endrepeat>
    <timescheduling
      singlestart="hh:mm:ss"        (cannot be used with "begin", "end" and "repeat")
      begin="hh:mm:ss"              (cannot be used with "singlestart")
      end="hh:mm:ss"                (cannot be used with "singlestart")
      repeat="hh:mm:ss"             (cannot be used with "singlestart")
      />
  </dailyrepeat>
  <weeklyrepeat                     (cannot be used with <dailyrepeat> or
<monthlyrepeat>)
    weeksperiod="1"
    daysofweek="Su,Mo,Tu,We,Th,Fr,Sa"
    >
    <endrepeat
      noend="y/n"                   (cannot be used with "repeatnumber" or "date" and
"time")
      repeatnumber="1"              (cannot be used with "noend" or "date" and "time")
      date="yyyy/mm/dd"             (cannot be used with "repeatnumber" or "noend")
      hour="(A/R)hh:mm:ss"
      >
      <relativedate                 (defined for relative scheduling)
        daysfromreference="(-)1"    (cannot be used with "dayofweek" or "dayofmonth")
        dayofweek="Su"              (cannot be used with "daysfromreference" or
"dayofmonth")
        dayofmonth="01"             (cannot be used with "daysfromreference" or
"dayofweek")
        weeksfromreference="(-)1"   (in case "dayofweek")
        weekofmonth="1"             (in case "dayofweek")
        monthsfromreference="(-)1"  (in case "dayofmonth" or "weekofmonth")
        month="Jan"                 (in case "dayofmonth" or "weekofmonth")
        />
    </endrepeat>
    <timescheduling
      singlestart="hh:mm:ss"        (cannot be used with "begin", "end" and "repeat")
      begin="hh:mm:ss"              (cannot be used with "singlestart")
      end="hh:mm:ss"                (cannot be used with "singlestart")
      repeat="hh:mm:ss"             (cannot be used with "singlestart")
      />
  </weeklyrepeat>
  <monthlyrepeat                    (cannot be used with <dailyrepeat> or
<weeklyrepeat>)
    monthsperiod="1"                (cannot be used with "months")
    months="Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec" (cannot be used with
"monthsperiod")
    daysofmonth="01,02,03,04,05,...,28,29,30,31,L"   (cannot be used with "weeksofmonth")
    weeksofmonth="1,2,3,4,L"                         (cannot be used with "daysofmonth")
    daysofweek="Su,Mo,Tu,We,Th,Fr,Sa"                (cannot be used with "daysofmonth")
    >
    <endrepeat
      noend="y/n"                   (cannot be used with "repeatnumber" or "date" and
"time")
      repeatnumber="1"              (cannot be used with "noend" or "date" and "time")
      date="yyyy/mm/dd"             (cannot be used with "repeatnumber" or "noend")
      hour="(A/R)hh:mm:ss"
      >
      <relativedate                 (defined for relative scheduling)
        daysfromreference="(-)1"    (cannot be used with "dayofweek" or "dayofmonth")
        dayofweek="Su"              (cannot be used with "daysfromreference" or
"dayofmonth")
        dayofmonth="01"             (cannot be used with "daysfromreference" or
"dayofweek")
        weeksfromreference="(-)1"   (in case "dayofweek")
        weekofmonth="1"             (in case "dayofweek")
        monthsfromreference="(-)1"  (in case "dayofmonth" or "weekofmonth")
        month="Jan"                 (in case "dayofmonth" or "weekofmonth")
        />
    </endrepeat>
    <timescheduling
      singlestart="hh:mm:ss"        (cannot be used with "begin", "end" and "repeat")
      begin="hh:mm:ss"              (cannot be used with "singlestart")
      end="hh:mm:ss"                (cannot be used with "singlestart")
      repeat="hh:mm:ss"             (cannot be used with "singlestart")
      />
  </monthlyrepeat>
</scheduling>
```

5.1.1 **<reference>**

Used in case the Scheduling is relative. Defines the date and time in UTC which is used as reference for relative start date and time and relative endrepeat date and time.

5.1.2 **<start>**

Defines the date and time at which the Scheduling starts.

If “executeatstart” is:

- “y”, the Trigger launches the Job at this date and time.
- “n”, the Trigger waits for the next recurrent date and time to launch the Job (see <dailyrepeat>, <weeklyrepeat>, <monthlyrepeat>).

In case the scheduling is:

- relative (see <reference>), the date is defined by the <relativedate> structure and the time “attribute” can be relative (example time="R00:02:00", means +2minutes).
- absolute, the date is defined by the “date” attribute.

5.1.3 **<relativedate>**

Defines a relative date. This structure is used into <start> and <endrepeat>.

The relative date refers to the <reference> date and time.

5.1.4 **"daysfromreference"**

Defines a number of days applied to the reference date, if:

- negative: number of days before
- positive: number of days after the reference date.

5.1.5 **"dayofweek"**

Defines a day of the week at which the Trigger launches the Job.

The value is one of the two characters items: Su=Sunday, Mo=Monday, Tu=Tuesday, We=Wenesday, Th=Thursday, Fr=Friday, Sa=Saturday.

"dayofweek" must be used with "weeksfromreference" or "weekofmonth".

5.1.6 " weeksfromreference "

Defines the number of weeks for the reference date, if:

- zero: the Trigger launches the Job the day of week defined by "dayofweek" in the next 7 days.
- positive: the Trigger launches the Job the day of week defined by "dayofweek" **after** the number of weeks defined
- negative: the Trigger launches the Job the day of week defined by "dayofweek" **before** the number of weeks defined

5.1.7 "weekofmonth"

Defines a week in the month at which the Trigger launches the Job.

The value is one of the following values 1=1st week of the month, 2=2nd week of the month, 3=3rd week of the month, 4=4th week of the month, L=last week of the month (4th or 5th).

"weekofmonth " must be used with "monthsfromreference" or "monthofyear".

5.1.8 "dayofmonth"

Defines a day of the month at which the Trigger launches the Job.

The value can be a number between 1 and 31 or L=last day of month.

"dayofmonth" must be used with "monthsfromreference" or "month".

5.1.9 "monthsfromreference"

Defines the number of months for the reference date, if:

- zero: the Trigger launches the Job the day defined by "dayofmonth" in the next 31 days or the day of the week defined by "dayofweek" and "weekofmonth".
- positive: the Trigger launches the Job the day defined by "dayofmonth" in the next 31 days or the day of the week defined by "dayofweek" and "weekofmonth" **after** the number of weeks defined.
- negative: the Trigger launches the Job the day defined by "dayofmonth" in the next 31 days or the day of the week defined by "dayofweek" and "weekofmonth" **before** the number of weeks defined.

5.1.10 "month"

Defines a month in the year at which the Trigger launches the Job.

The value is one of the three characters items: Jan=January, Feb=February, Mar=March, Apr=April, May=May, Jun=June, Jul=July, Aug=August, Sep=September, Oct=October, Nov=November, Dec=December.

"month" must be used with "dayofmonth" or "weekofmonth".

5.1.11 <endrepeat>

Defines the end of a recurrent scheduling defined by <dailyrepeat>, <weeklyrepeat> or <monthlyrepeat>.

One and only one of the following item can be defined into the <endrepeat> tag:

- "noend": the recurrence never stops
- "repeatnumber": number of time the recurrence occurs (the <start> date and time is not included into this number)
- "date" and "time": absolute date and time
- <relativedate> and "time":
  - date relative to <reference> date
  - time relative to <reference> time if "time" value starts with "(R)"
  - time absolute if "time" value starts with "(A)" or nothing

5.1.12 <timescheduling>

Can only be used into <dailyrepeat>, <weeklyrepeat>, or <monthlyrepeat>. Used to define the hours at which the trigger launches the Job in the recurrent days. Use "singlestart" or the set of attributes "begin", "end", and "repeat".

If used with:

- "singlestart", the Job is launched only once in the day.
- "begin", "end", and "repeat" the Job is launched every "repeat" time between "begin" and "end".

5.1.13 <dailyrepeat>

Used to define a day-based recurrence. The recurrence occurs every "daysperiod" days.

5.1.14 <weeklyrepeat>

Used to define a recurrence based on weeks.

The recurrence occurs every "weeksperiod" weeks on the days defined by "daysofweek". "daysofweek" is a coma separated list of values in the following two characters items : Su=Sunday, Mo=Monday, Tu=Tuesday, We=Wenesday, Th=Thursday, Fr=Friday, Sa=Saturday.

5.1.15 <monthlyrepeat>

Used to define a recurrence based on months. The recurrence occurs every "monthsperiod" months or on "months" of the year.

"months" is a coma separated list of values in the following three characters items : Jan=January, Feb=February, Mar=March, Apr=April, May=May, Jun=June, Jul=July, Aug=August, Sep=September, Oct=October, Nov=November, Dec=December.

The recurrence can occur days of the months using "daysofmonth" or days of week using "weeksofmonth" combined with "daysofweek".

"daysofmonth" is a coma separated list of values that can be a number between 1 and 31 or L=last day of month.

"weeksofmonth" is a coma separated list of values in the following values 1=1st week of the month, 2=2nd week of the month, 3=3rd week of the month, 4=4th week of the month, L=last week of the month (4th or 5th).

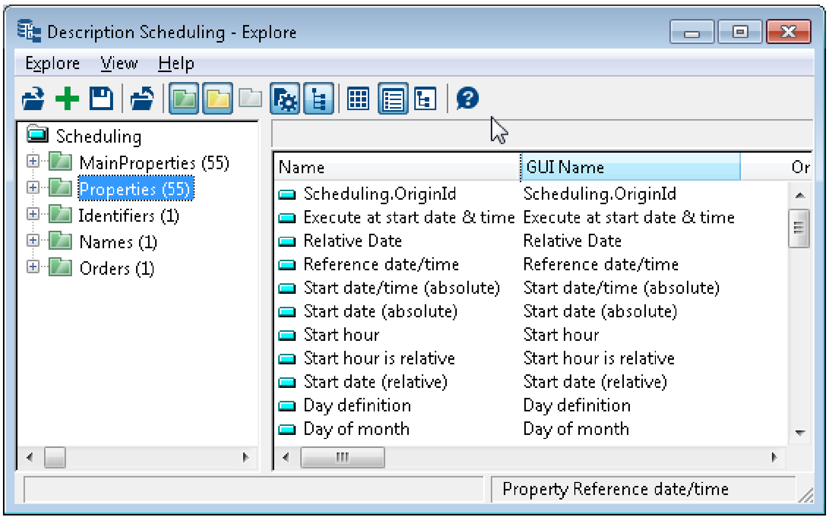
"daysofweek" is a coma separated list of values in the following two characters items : Su=Sunday, Mo=Monday, Tu=Tuesday, We=Wenesday, Th=Thursday, Fr=Friday, Sa=Saturday.

5.2 Scheduling MegaObject interface

The properties of the Scheduling MegaObject available via the Scheduler client API can be listed in the HOPEX Explorer as following:

```
Dim mgRoot
Dim objSchedulerClient
Dim mgobjScheduledTrigger
Dim mgobjScheduling
Dim mgobjSchedulingType

Set mgRoot = GetRoot()

' Get the SchedulerClient API object
Set objSchedulerClient = mgRoot.SchedulerClient

' Allocate a "System Trigger" MegaObject
Set mgobjScheduledTrigger = objSchedulerClient.NewTrigger()

' Get the Scheduling MegaObject
Set mgobjScheduling =
mgobjScheduledTrigger.GetCollection("~e)PMRueCFbLR[Scheduling]").Item(1)

' Get the Scheduling ObjectType MegaObject
Set mgobjSchedulingType = mgobjScheduling.GetTypeObject()

' Explores the Scheduling ObjectType
mgobjSchedulingType.Explore
```

The result is an Explorer window as bellow:



Name	GUI Name	Or
Scheduling.OriginId	Scheduling.OriginId	
Execute at start date & time	Execute at start date & time	
Relative Date	Relative Date	
Reference date/time	Reference date/time	
Start date/time (absolute)	Start date/time (absolute)	
Start date (absolute)	Start date (absolute)	
Start hour	Start hour	
Start hour is relative	Start hour is relative	
Start date (relative)	Start date (relative)	
Day definition	Day definition	
Day of month	Day of month	

Property Reference date/time

All the properties match information elements detailed in "**Error! Reference source not found.**".

A description is available in the comment of each property.

5.3 Scheduling Property Page

5.3.1 Presentation of the Scheduling Property page

Scheduling configuration on a MetaClass on which a scheduling has to be defined is stored into a text property of the MetaClass.

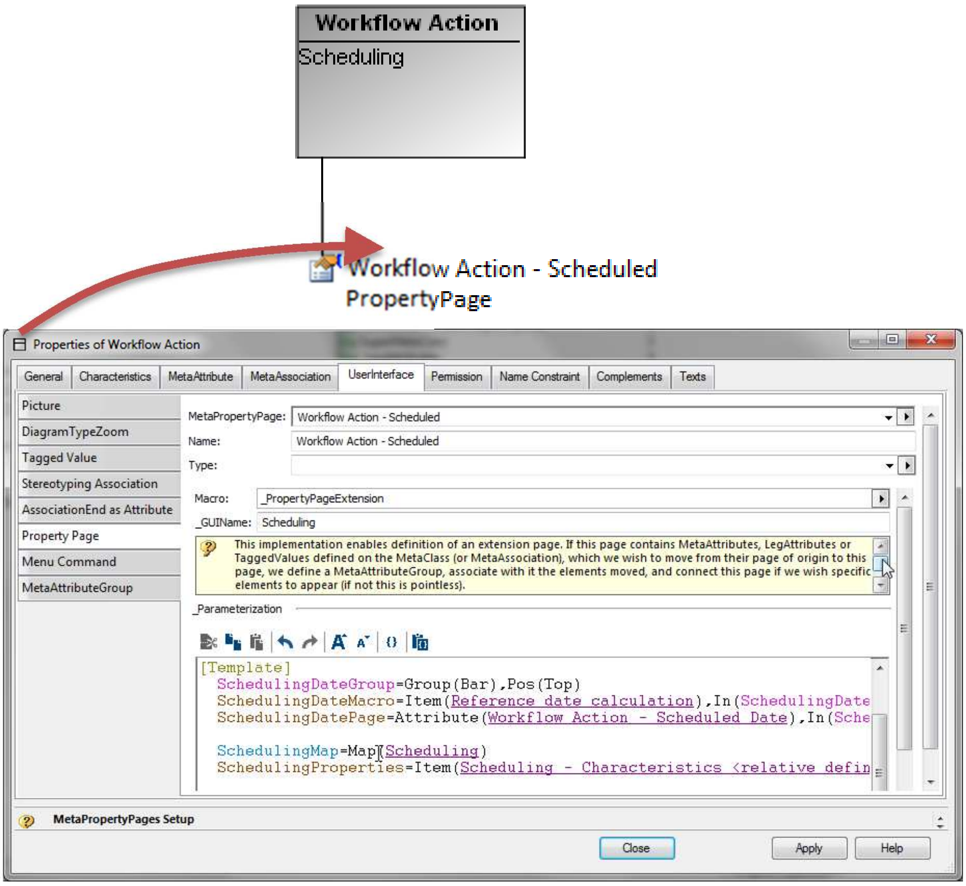
The scheduling property page is a graphical layer over the scheduling XML text format that enables easy edition/display of scheduling configurations.

This property page can be integrated in property pages of custom MetaClasses.

Example: scheduled Workflow Action

The MetaAttribute "Scheduling" is added on "Workflow Action" MetaClass. A MetaPropertyPage is added on "Workflow Action" MetaClass, this MetaPropertyPage includes a platform provided scheduling propertypage using this syntax:

```
SchedulingMap=Map(~e)PMRueCFbLR[Scheduling])
SchedulingProperties=Item(~bwjCkgpSFTAH[Scheduling - Characteristics <relative definition>]),From(SchedulingMap),Control(SubPage),Param(Owner=off,Name=off)
```



MetaPropertyPage: Workflow Action - Scheduled

Name: Workflow Action - Scheduled

Type:

Macro: _PropertyPageExtension

_GUIName: Scheduling

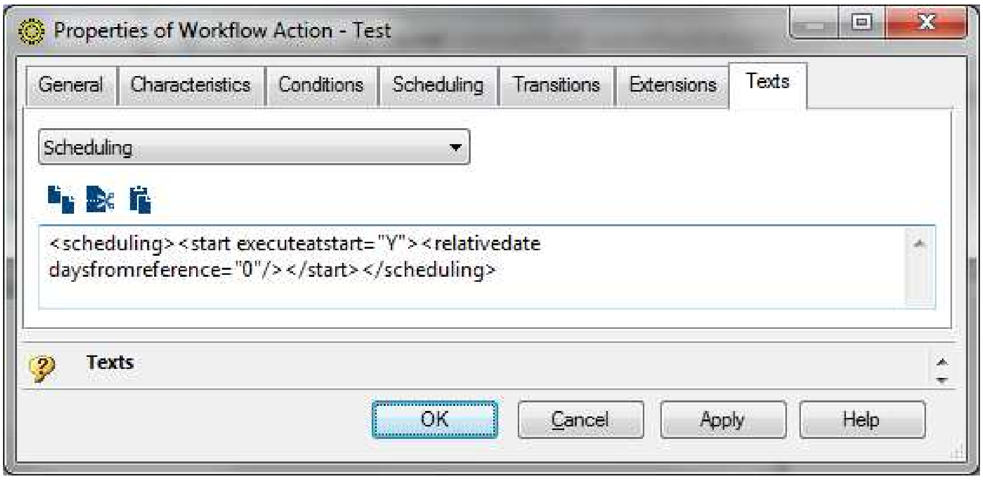
This implementation enables definition of an extension page. If this page contains MetaAttributes, LegAttributes or TaggedValues defined on the MetaClass (or MetaAssociation), which we wish to move from their page of origin to this page, we define a MetaAttributeGroup, associate with it the elements moved, and connect this page if we wish specific elements to appear (if not this is pointless).

_Parameterization

```
[Template]
  SchedulingDateGroup=Group(Bar),Pos(Top)
  SchedulingDateMacro=Item(Reference date calculation),In(SchedulingDate
  SchedulingDatePage=Attribute(Workflow Action - Scheduled Date),In(Sche

  SchedulingMap=Map(Scheduling)
  SchedulingProperties=Item(Scheduling - Characteristics <relative defin
```

The result on "Workflow Action" objects is as follows:

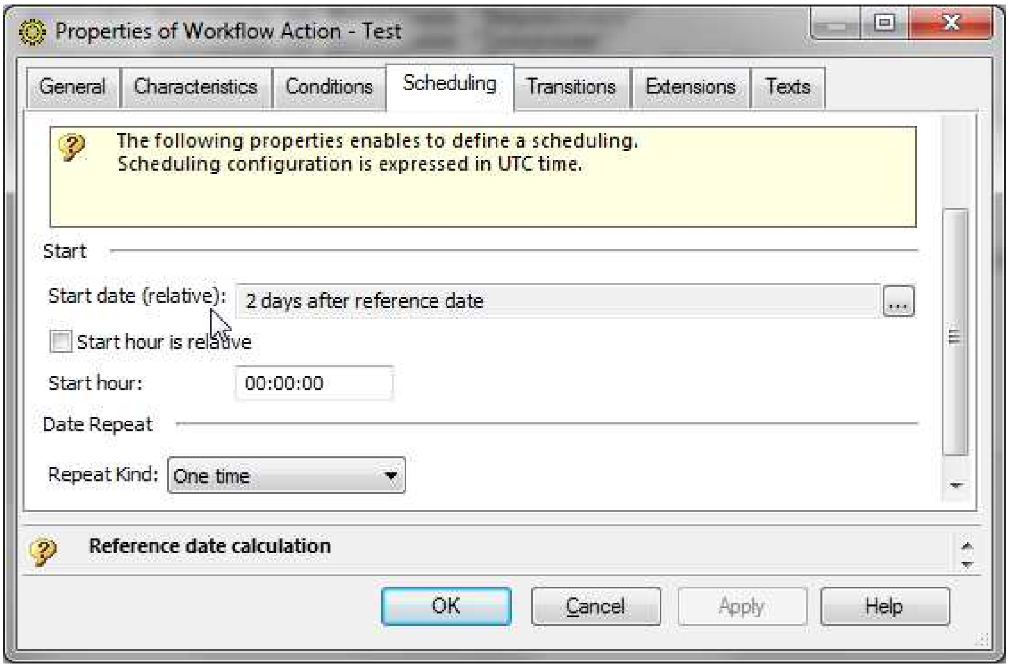
- Physical scheduling XML text format value



Scheduling

```
<scheduling><start executeatstart="Y"><relativedate
daysfromreference="0"/></start></scheduling>
```

- Scheduling graphical layer



The following properties enables to define a scheduling.
Scheduling configuration is expressed in UTC time.

Start date (relative): 2 days after reference date

☐ Start hour is relative

Start hour: 00:00:00

Repeat Kind: One time

5.3.2 **Provided property pages**

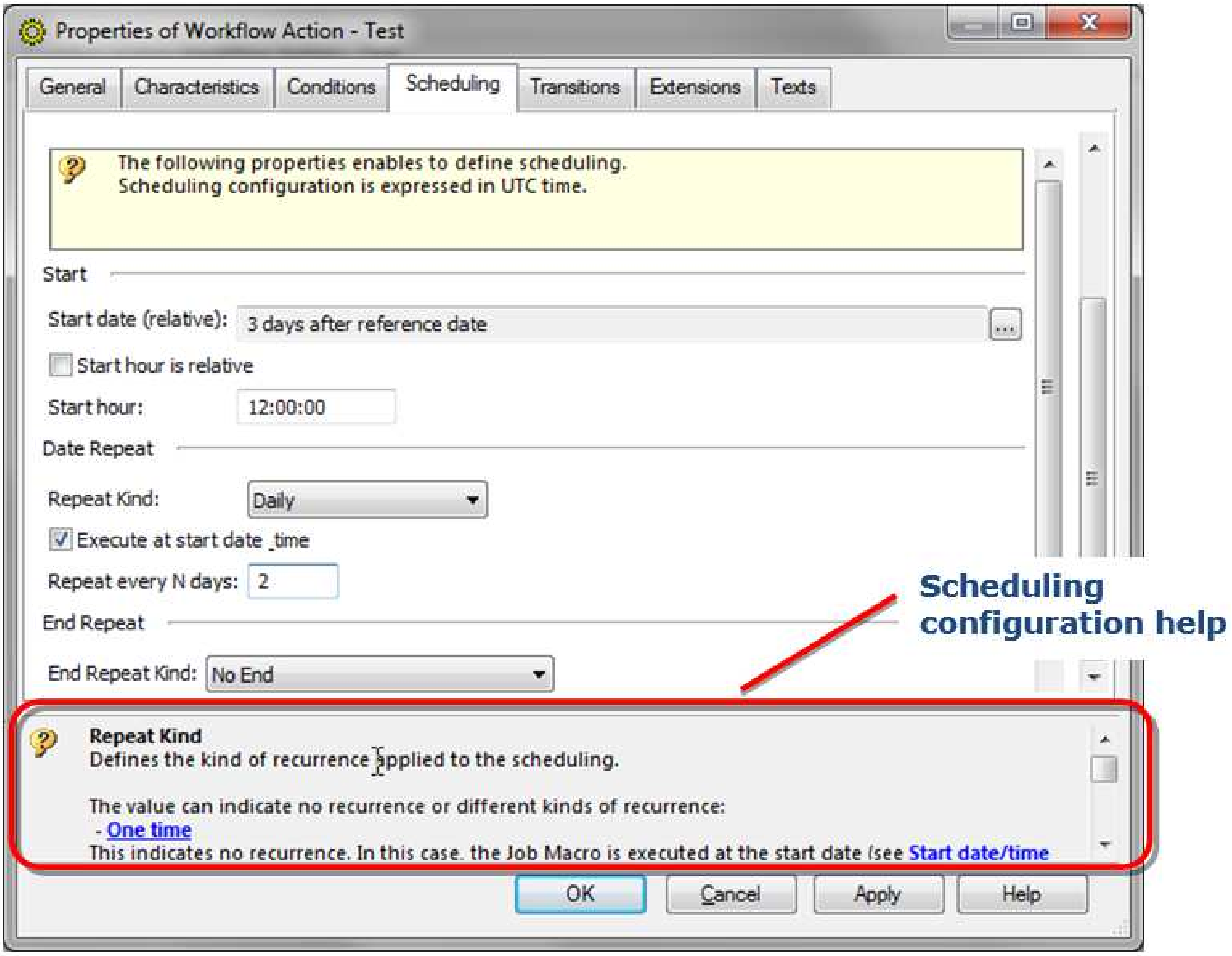
The following MetaPropertyPages can be included into MetaPropertypages of MetaClasses having the "~iUhj4likEzJQ[Scheduling]":

- *~I8QfkFtCF19M[Scheduling - Characteristics]*: this property page enables to configure all possible kind of scheduling
- *~bwjCkgpSFTAH[Scheduling - Characteristics (relative definition)]*: this property page enables to configure scheduling that must be relative
- *~fgkxoz(xH5zN[Scheduling - Characteristics (readonly)]*: this property page enables to display a scheduling configuration

5.3.3 **Property page description help**

All scheduling properties are commented. These properties are the ones that can be listed by exploring the scheduling MegaObject type as described in section Scheduling MegaObject interface.

Thus, help for all properties is available into the contextual help:



Properties of Workflow Action - Test

The following properties enables to define scheduling.
Scheduling configuration is expressed in UTC time.

Start

Start date (relative): 3 days after reference date

□ Start hour is relative

Start hour: 12:00:00

Date Repeat

Repeat Kind: Daily

☑ Execute at start date _time

Repeat every N days: 2

End Repeat

End Repeat Kind: No End

Repeat Kind
Defines the kind of recurrence applied to the scheduling.

The value can indicate no recurrence or different kinds of recurrence:
- One time
This indicates no recurrence. In this case, the Job Macro is executed at the start date (see Start date/time

6 IMPLEMENTING A JOB

6.1 Implemented Function description

When a Trigger is launched the Job Macro is executed. A Job Macro must implement the following function:

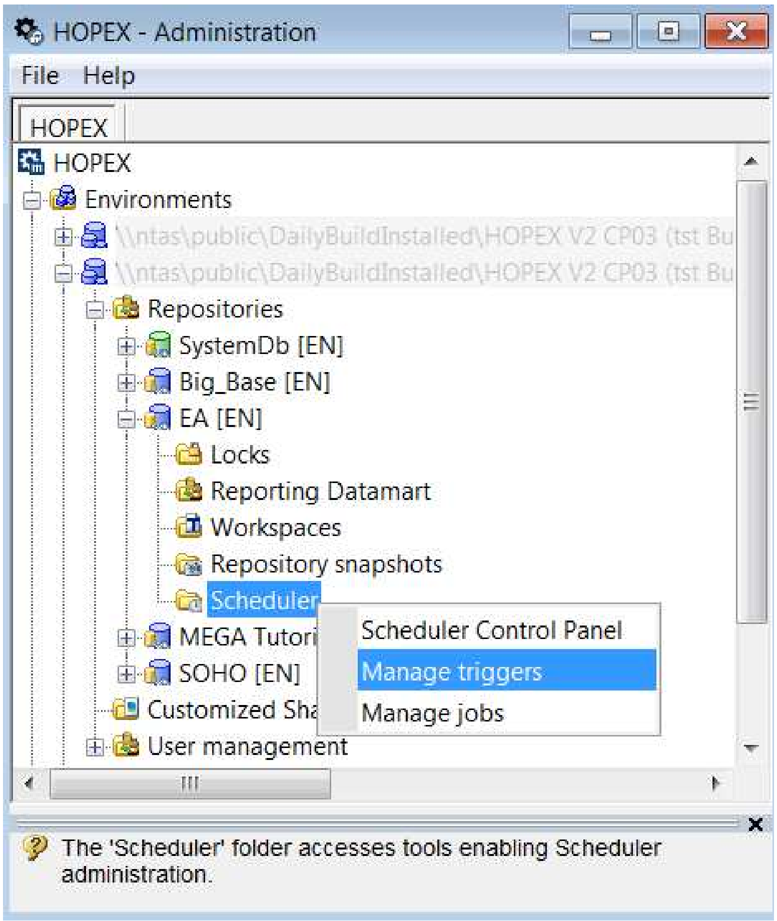
- Function:
  - RunScheduledJob
- Parameters:
  - mgobjJob As MegaObject
    - This parameter is the object describing the Job.
    - Use `GetTypeObject().Explore` on a Job MegaObject to explore the Job type.
    - In particular, use *GetContextString()* Method to recover the string set at Trigger registering (see AddTrigger).
  - objJobResult As Object
    - This parameter is the object enabling to return information needed by the Scheduler after the Job execution.
    - The objJobResult parameter type is composed of the following properties to be set into the Job implementation
      - DiagMessage As String: a diagnosis message used when the job succeed.
      - ErrorMessage As String: an error message in case of job failure (when the Job Function returns False to indicate the job failure).
      - RemoveTrigger As Boolean: in case of job success, set blnRemoveTrigger to True to remove the job trigger so that the job will not be executed anymore.

6.2 Job Function Template

```
'MegaContext(Fields,Types)
Option Explicit

' Function    : RunScheduledJob
' Description : Implement this function to specify what to do when a job is triggered.
' Parameter   :
'   - mgobjJob : object describing the job (this object is not a repository object)
'   - objJobResult : object enabling to return some informations
'       * DiagMessage As String : a diagnosis message used when the job succed
'       * ErrorMessage As String : an error message in case of job failure => return False to indicate job failure
'       * RemoveTrigger As Boolean : in case of job success, set blnRemoveTrigger to True in order to remove the job trigger so that the job will not be executed anymore
' Return      :
'   - Boolean : return True in case of job success or False in case of job failure, in this case, set an error message into strErrorMessage

Function RunScheduledJob (mgobjJob As MegaObject, objJobResult As Object) As Boolean
  ' Initializing variables
  Dim mgRoot
  Set mgRoot = mgobjJob.getRoot()


  ' Write some code here ...


  ' Return True in case of job success
  ' Return False in case of job failure, in this case, set an error message into strErrorMessage
  RunScheduledJob = True
End Function
```

7 TRIGGERS ADMINISTRATION

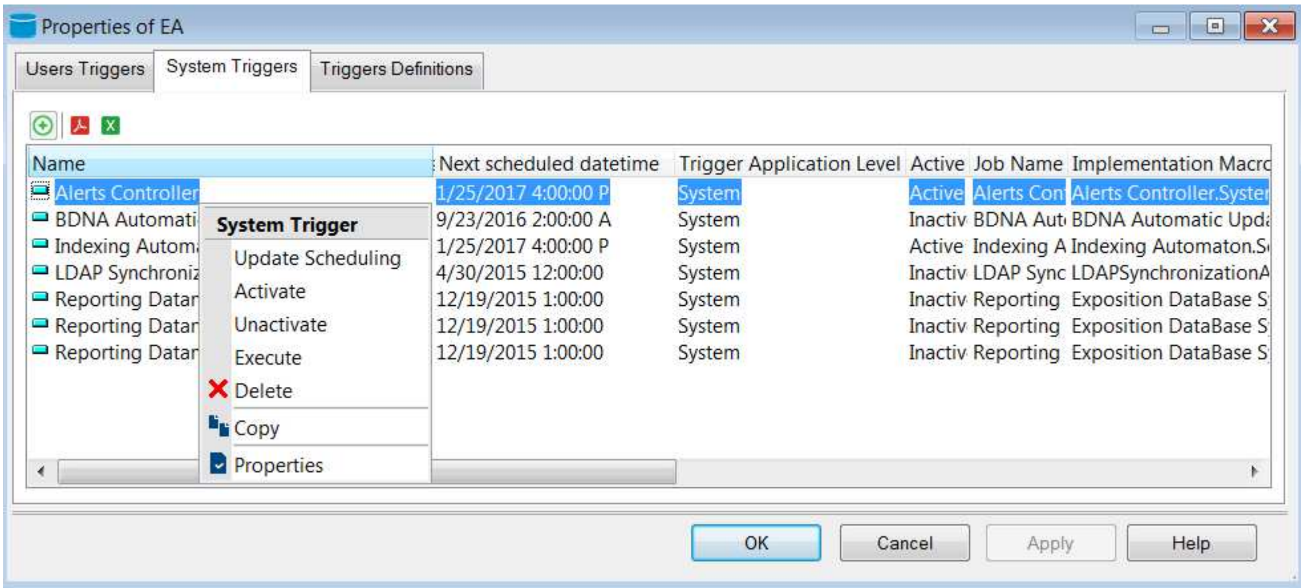
Triggers can be managed from the **Administration** application:



The 'Scheduler' folder accesses tools enabling Scheduler administration.

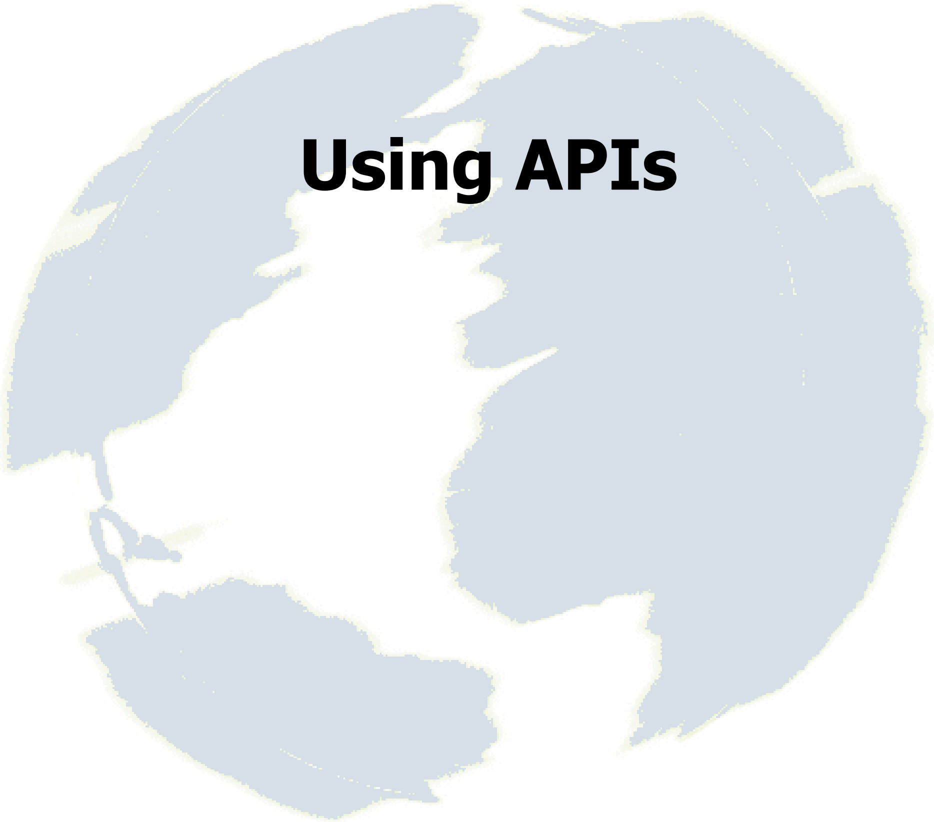
The **Manage triggers** tool enables to:

- access to trigger properties:
  - Next execution dates
  - Scheduling configuration associated
- delete triggers



Name	Next scheduled datetime	Trigger Application Level	Active	Job Name	Implementation Macr
Alerts Controller	1/25/2017 4:00:00 P	System	Active	Alerts Con	Alerts Controller.Syster
BDNA Automati	9/23/2016 2:00:00 A	System	Inactiv	BDNA Aut	BDNA Automatic Upd
Indexing Autom	1/25/2017 4:00:00 P	System	Active	Indexing A	Indexing Automaton.S
LDAP Synchroniz	4/30/2015 12:00:00	System	Inactiv	LDAP Sync	LDAPSynchronizationA
Reporting Datar	12/19/2015 1:00:00	System	Inactiv	Reporting	Exposition DataBase S
Reporting Datar	12/19/2015 1:00:00	System	Inactiv	Reporting	Exposition DataBase S
Reporting Datar	12/19/2015 1:00:00	System	Inactiv	Reporting	Exposition DataBase S

Using APIs



All about starting with APIs

1 INTRODUCTION

HOPEX provides APIs enabling access to the repository and handling actions on the repository.

The aim of this document is to provide information required to start with VB Script and Java APIs in HOPEX.

Note: for information on GraphQL APIs (which are based on Rest APIs), see HOPEX REST APIs – V5.

In HOPEX, to execute code using APIs, always use the macro concept. Macros can be written in VB script or Java. Depending on the implementation type to be added, you must choose the suitable programming language as follows:

- VB Script for simple implementations (few algorithmic code) and when implementing code close to the MetaModel (Command, Property Pages, calculated attributes, etc.).

The advantage of using VB Script is that you can access and easily read your code.

- Java for lengthy processing (including more algorithmic code) and requiring higher performance such as for exports, synchronizations, etc.

The disadvantage of using Java is that the code is directly stored in a jar library and not directly readable.

This document details these two access modes to APIs (VB Script and Java) and provides a toolkit to use in your implementation.

It is important that you bear in mind the recommendations included in this document regarding performance and confidentiality:

- MEGA applications are Web applications, execution time must be optimized to avoid any potential timeout of the application.
- Clients are increasingly demanding regarding security.

Be careful during migration and maintenance, sources must remain available.

Be aware that:

- Advanced customizations may require additional development to work properly after migrating to a HOPEX next major version.
- HOPEX Technical Support does not provide assistance with developing, maintaining or upgrading MEGA advanced customizations.

For information on:

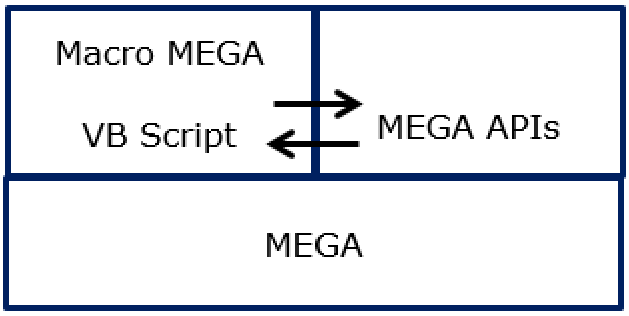
- Java MEGA API documentation, see the ***JavaDoc*** documentation (html format) accessible from the online documentation (**HOPEX Cuqtomization (Windows) > Using APIs > JavaDoc**).

  The ***JavaDoc*** documentation is provided in the HOPEX installation "java\doc" directory.
- Mega Operators – Method, from HOPEX menu bar select **Help > APIs > Methods**.

2 CREATE A VB SCRIPT COMPONENT

2.1 VB Script component architecture

MEGA APIs are natively COM and can be used by COM application. The VB Script language uses the Dispatch protocol to handle objects and access their methods. Therefore using VB Script language to access MEGA objects is completely standard. The following diagram illustrates the way the VB Script component interacts with MEGA APIs:



2.2 Creating a VB Script component

VB Script code can be written and used:

- in an external file
- in a MEGA macro
- directly in the MEGA script editor.

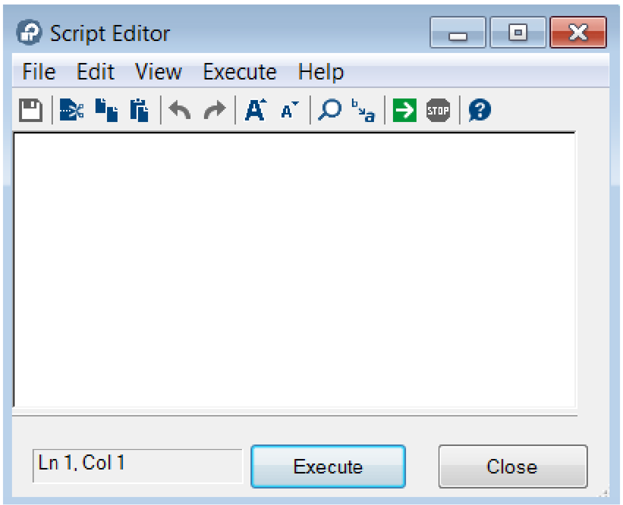
You can use the script editor to write VB scripts using MEGA APIs, which enables you to execute operations on HOPEX repository content.

2.2.1 Creating a VB Script component

To create a VB Script component with the Script Editor:

1. From **HOPEX** menu bar, select **Tools > Script Editor** to launch the script editor.

   The **Script Editor** dialog box opens.



2. From **Script Editor** menu bar, select **File > Save As**.
3. Select either:
   - **a Macro** to save the script as a macro in the repository,
   - **a File** to save the script as an external VB Script file.
4. Enter your VB Script code.

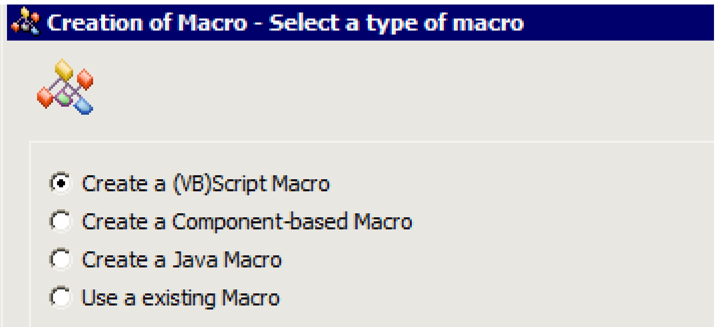
2.2.2 Creating a VB Script macro with the wizard

HOPEX includes a wizard that enables preparation of macro parameters and code according to the use imagine by the user. Code is initialized according to the MEGA concept implemented.

To use the macro creation wizard on an object of TaggedValue type:

1. From HOPEX toolbar, click **Explore** .
2. From **Explore** window, create a new object of the **TaggedValue** MetaClass.
3. Display **Empty Collections**.
4. Right-click **Implementation** folder and select **New**.

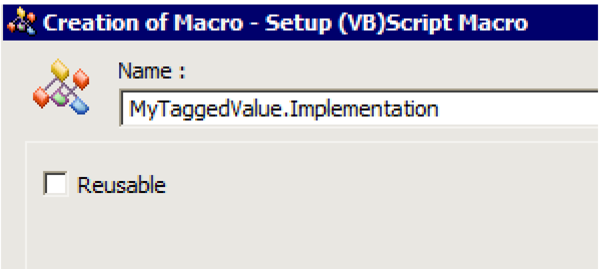
   The **Creation of Macro** wizard appears.



Creation of Macro - Select a type of macro

● Create a (VB)Script Macro
○ Create a Component-based Macro
○ Create a Java Macro
○ Use a existing Macro

5. Select **Create a (VB)Script Macro**.
6. Click **Next**.

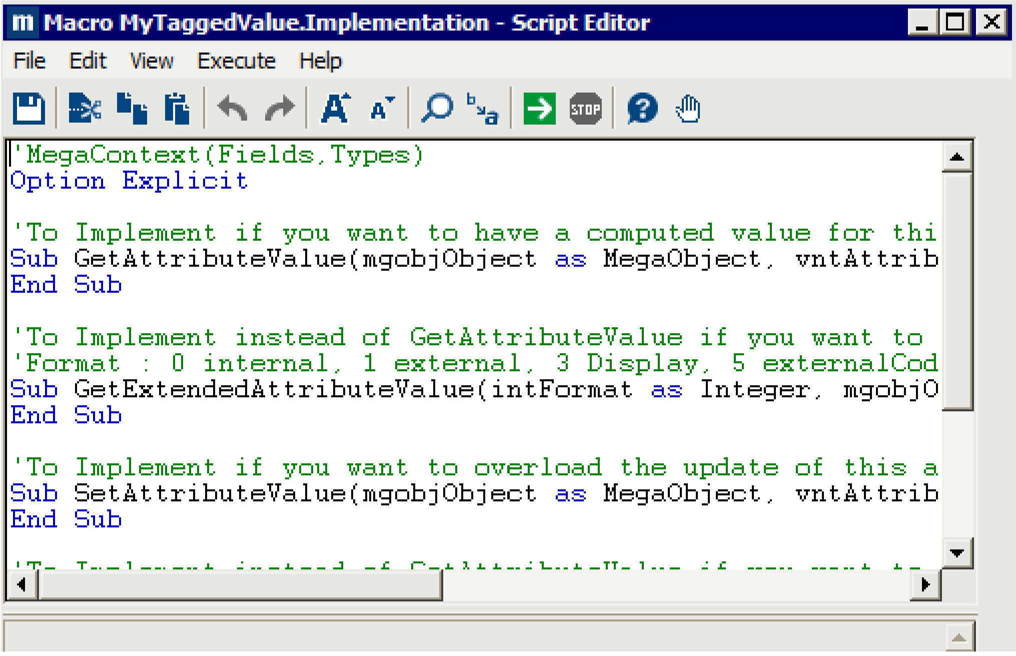


Name : MyTaggedValue.Implementation

□ Reusable

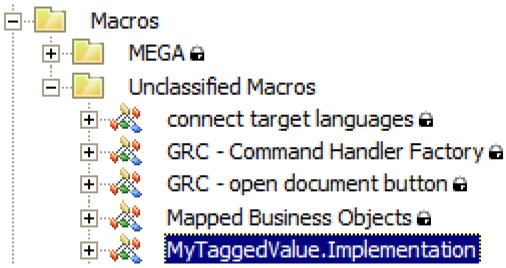
7. (Optional) Modify the **Name** of the macro.
8. (Optional) If the macro you have created can be reused for another TaggedValue, select the **Reusable**.
9. Click **Finish**.

   The macro (e.g.: "MyTaggedValue.Implementation) is created and its code already initialized.



```
'MegaContext(Fields,Types)
Option Explicit

'To Implement if you want to have a computed value for thi
Sub GetAttributeValue(mgobjObject as MegaObject, vntAttrib
End Sub

'To Implement instead of GetAttributeValue if you want to
'Format : 0 internal, 1 external, 3 Display, 5 externalCod
Sub GetExtendedAttributeValue(intFormat as Integer, mgobjO
End Sub

'To Implement if you want to overload the update of this a
Sub SetAttributeValue(mgobjObject as MegaObject, vntAttrib
End Sub
```

The macro (e.g.: "MyTaggedValue.Implementation) is stored in the **Macros** folder, **Unclassified Macros** sub-folder.



- Macros
  - MEGA
  - Unclassified Macros
    - connect target languages
    - GRC - Command Handler Factory
    - GRC - open document button
    - Mapped Business Objects
    - MyTaggedValue.Implementation

2.3 Editing a VB Script code

The **Script Editor** menu bar enables you to access commands that ease your editing actions.

The **Script Editor** tool bar enables you to easily access these commands to simplify your editing work.

Alternatively, you can use the menu shortcuts.

To edit a VB Script file/macro with the Script Editor:

1. In the **Script Editor** menu bar, select **File > Open** and select the script type **a File** or **a Macro**.
2. Select the file/macro.

   The file/macro code is displayed in the **Script Editor**.

To execute the script:

In the **Script Editor** toolbar, click **Execute** .

Alternatively in the **Script Editor** menu bar, select **Execute > Execute**.

Global code execution starts.

To stop your script execution, in the **Script Editor** toolbar, click **Stop** .

To save the script code:

In the **Script Editor** toolbar, click **Save** .

Alternatively in the **Script Editor** menu bar, select **File > Save As.**

To move parts of the code to another position:

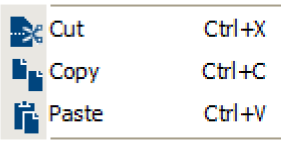
In the **Script Editor** toolbar, select the code part and click either:

Cut to cut the selected code

Copy to copy the selected code

Paste to paste the selected code

Alternatively in the **Script Editor** menu bar, select the **File** menu and then select the corresponding command or use the menu shortcuts:



Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V

To cancel or repeat an action you have carried out:

In the **Script Editor** toolbar, click:

Cancel to cancel your last action

Repeat to repeat you last action

Alternatively in the **Script Editor** menu bar, select the **Edit menu** and then select **File** menu and then select the corresponding command or use the menu shortcuts:

To enlarge or reduce font size:

In the **Script Editor** toolbar, click:

Enlarge Font to enlarge the font size

Reduce Font to reduce the font size

Alternatively in the **Script Editor** menu bar, select the **View** menu and then select the corresponding command.

To query or replace part of your code:

In the **Script Editor** toolbar, click:

Query to find a character string in your code

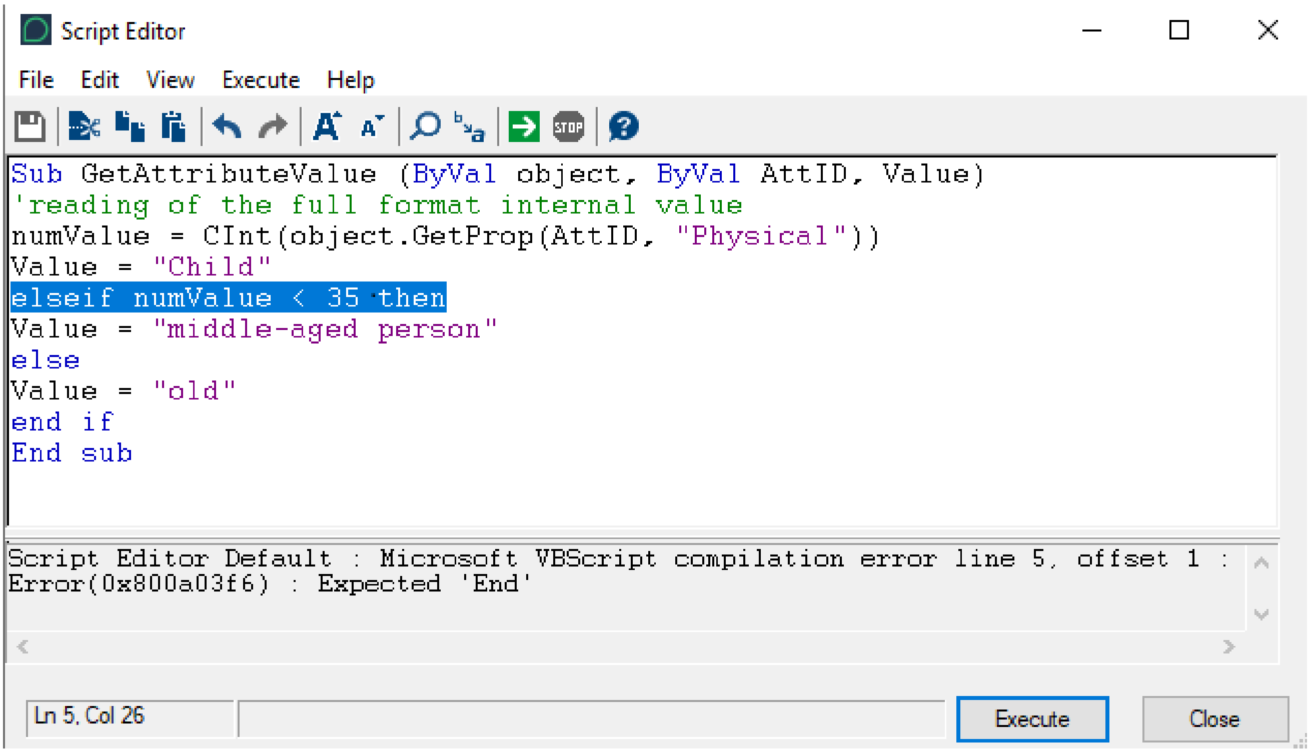
Replace to find a character string in your code and replace it by another one

Alternatively, in the **Script Editor** menu bar, select **Edit > Query** or **Edit > Replace**.

To test your script:

In the **Script Editor** toolbar, click **Execute** to test your script.

If there is an error in your script, the line in error is highlighted and an error message is displayed in the bottom pane. This error message includes the line number in error and the error type.



```
Sub GetAttributeValue (ByVal object, ByVal AttID, Value)
'reading of the full format internal value
numValue = CInt(object.GetProp(AttID, "Physical"))
Value = "Child"
elseif numValue < 35 then
Value = "middle-aged person"
else
Value = "old"
end if
End sub
```

```
Script Editor Default : Microsoft VBScript compilation error line 5, offset 1 :
Error(0x800a03f6) : Expected 'End'
```

To stop your script execution, in the **Script Editor** toolbar, click **Stop** .

Alternatively, in the **Script Editor** menu bar, select **Execute > Execute** or **Execute > Stop**.

To reach a specific point in your code:

1. In the **Script Editor** menu bar Select **Edit > Go to Line**.
2. In the dialog box that appears, enter the line number you want to reach and click **OK**.

3 CREATE A JAVA COMPONENT

This chapter details the creation of a MEGA plug-in in Java.

The Java APIs provided by MEGA enable access to the possibilities offered by the existing VB Script API.

The HOPEX 1.0 Java APIs allow for the creation of plug-ins written in Java. The component written in Java is deployed in the form of a Jar library referenced via a macro.

The following subjects are covered:

- Using Eclipse as the environment for developing a MEGA plug-in written in Java

  See Development environment (Eclipse).

- Creating a MEGA component implemented in Java

  See Calling a component written in Java from HOPEX.

- Implementing the Java MEGA API using the example of creating a MetaCommand

  See Using the MEGA API in a component written in Java.

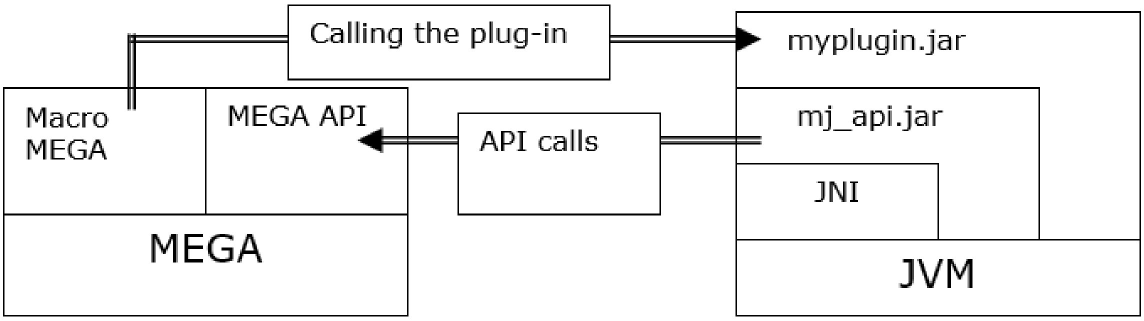
- Debugging a MEGA plug-in written in Java with Eclipse

  See Debugging a Java component called from HOPEXDebugging a Java component called from .

3.1 MEGA and Java general considerations

3.1.1 Architecture of a MEGA plug-in written in Java

The following diagram illustrates the call from MEGA of a plug-in written in Java. The plug-in uses repository content via MEGA APIs.



Versions of the applications used in this chapter are as follows:

MEGA	**HOPEX Aquila**
JDK	17
	Available at the following location: www.AdoptOpenJDK.net
Eclipse	Eclipse IDE 2021-03
	Available at the following location: https://www.eclipse.org/downloads/packages/

Calling the plug-in

A macro enables referencing of the plug-in in HOPEX. It identifies the class to be instanced.

Depending on the macro use case, the instanced class must implement a specific interface. It is via this interface that the plug-in will be called. In HOPEX, the functionalities that can be implemented in VB script can also be implemented in Java.

Examples:

- Meta Wizards
- Methods
- Update Tools
- Commands
- etc.

Using the MEGA API in the plug-in

When called, the plug-in can in turn call the MEGA application via the Java MEGA API.

The Java MEGA API is a set of objects and interfaces enabling, via JNI, calling the existing VB script MEGA API (Dispatch, Automation…).

This set of objects and interfaces is available in the mj_api.jar library delivered with HOPEX.

3.1.2 The JVM

In order to function, the Java MEGA API depends on a JVM (Java Virtual Machine).

This JVM is contained in the JRE (Java Runtime Environment) delivered with HOPEX.

Java plug-ins and applications integrating with HOPEX must be developed using a version of Java compatible with the JRE delivered with HOPEX.

3.1.3 The JDK

While the JRE enables only the execution of Java applications, the JDK also enables their creation: its contents include the Java compiler and all elements required for developing a Java application.

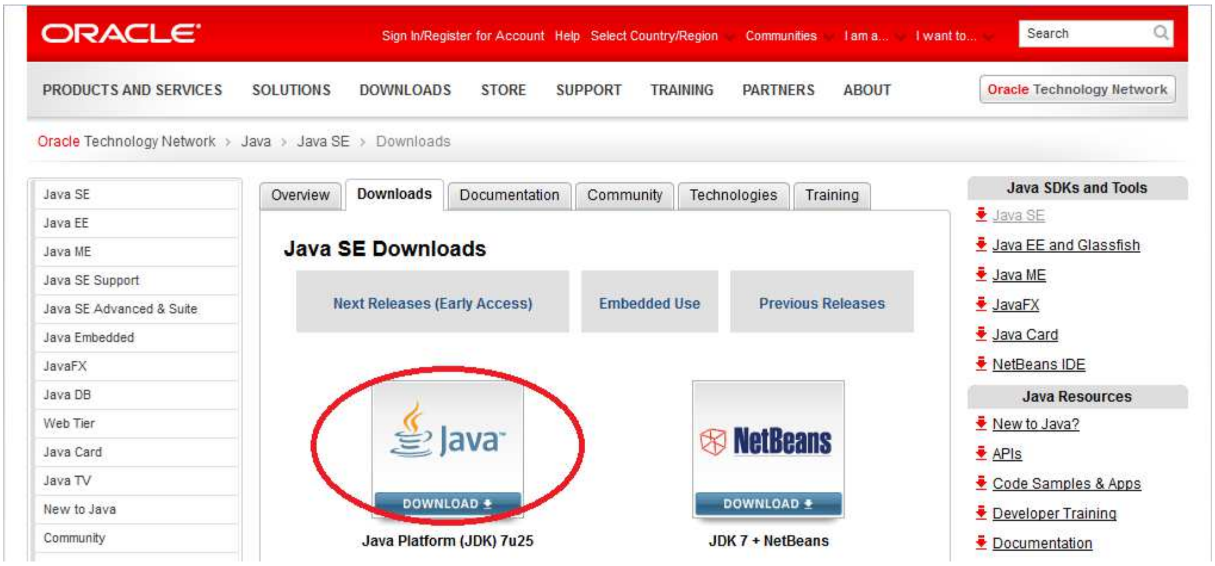
To develop a MEGA plug-in in Java, it is necessary to obtain and install the compatible JDK.

To download and install the JDK:

1. Go to the Oracle Web Site.

   To get the url path see table in Architecture of a MEGA plug-in written in Java section.

2. Download the JDK.



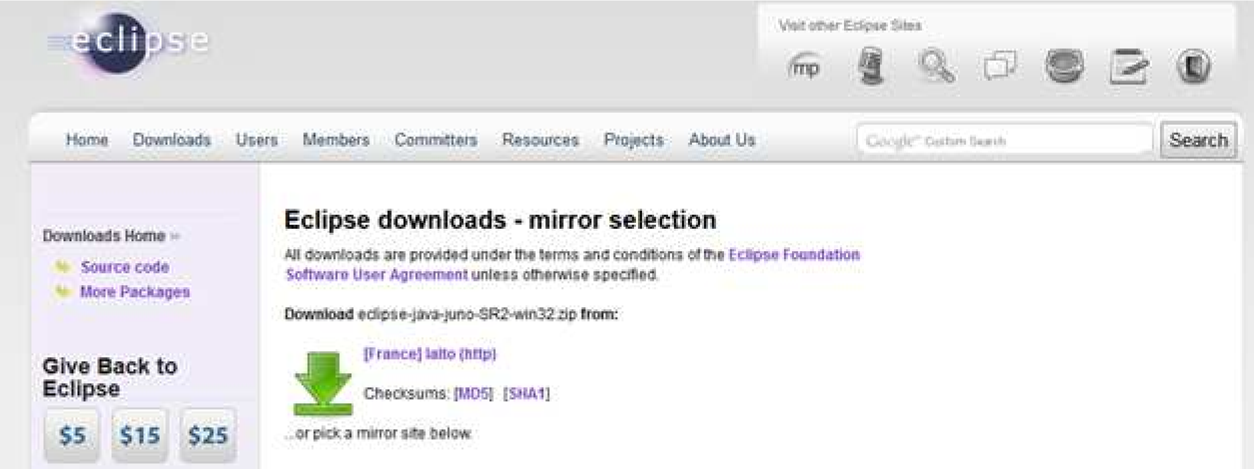
3.2 Development environment (Eclipse)

Eclipse is the Java-oriented development environment preferred for developing Java components integrated in HOPEX or interfacing with HOPEX.

3.2.1 Installing Eclipse

Downloading

MEGA recommends using the Juno release of Eclipse.



Eclipse downloads - mirror selection

All downloads are provided under the terms and conditions of the Eclipse Foundation Software User Agreement unless otherwise specified.

Download eclipse-java-juno-SR2-win32.zip from:

[France] lalto (http)

Checksums: [MD5] [SHA1]

...or pick a mirror site below.

Download **Eclipse** from the url given in the table in the Architecture of a MEGA plug-in written in Java section.

You obtain a zip Archive file **eclipse-java-juno-SR2-win32**.

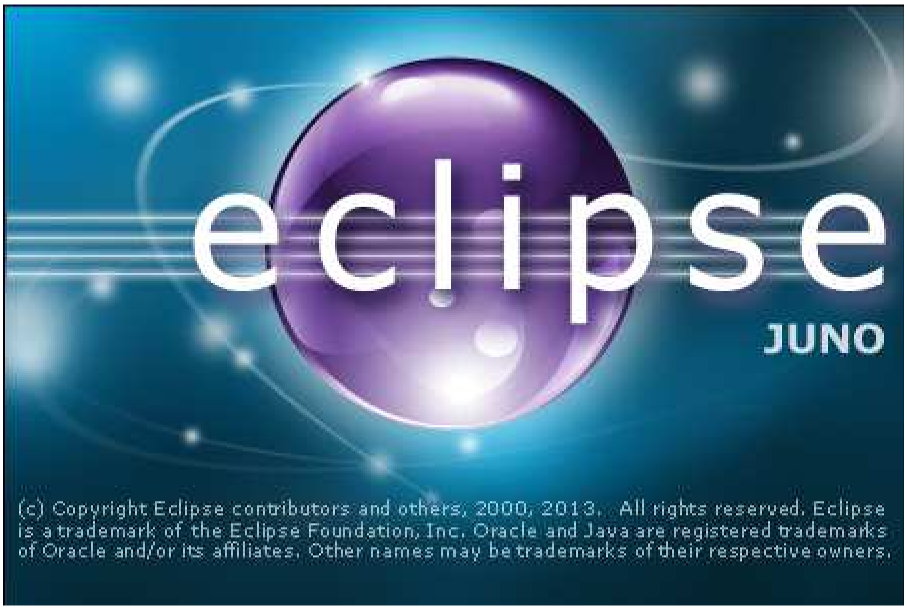
Installing Eclipse

To install Eclipse:

1. Unzip the zip Archive file at the hard drive root or in the **Program Files** folder.

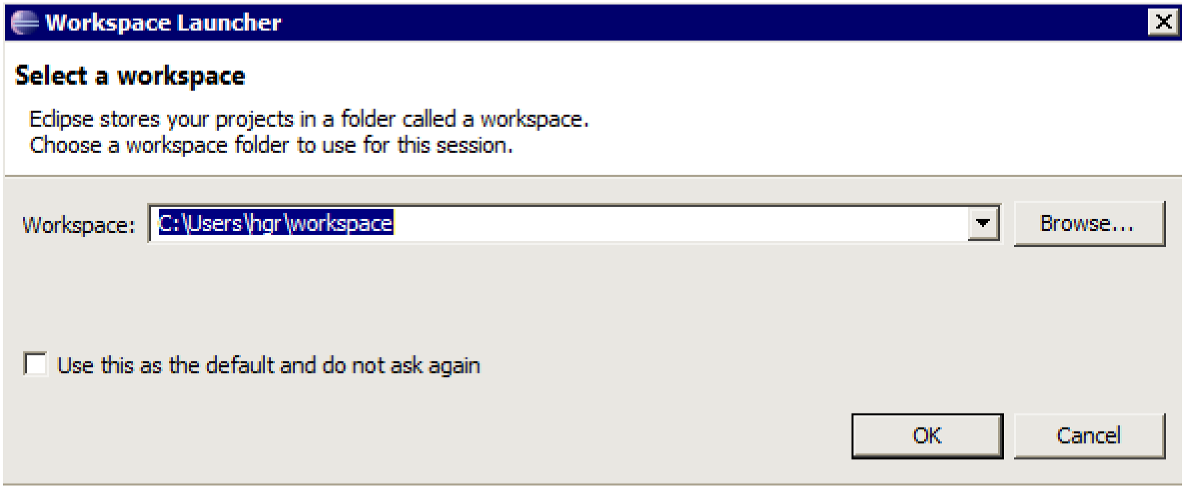
   Note: Certain versions of Eclipse are delivered with a JRE, for others you must obtain it.

2. Launch **Eclipse**.



(c) Copyright Eclipse contributors and others, 2000, 2013. All rights reserved. Eclipse is a trademark of the Eclipse Foundation, Inc. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

3. (optional) In the Workspace Launcher, modify your workspace folder.



Workspace Launcher

Select a workspace

Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\Users\hgr\workspace

□ Use this as the default and do not ask again

4. Click **OK**.

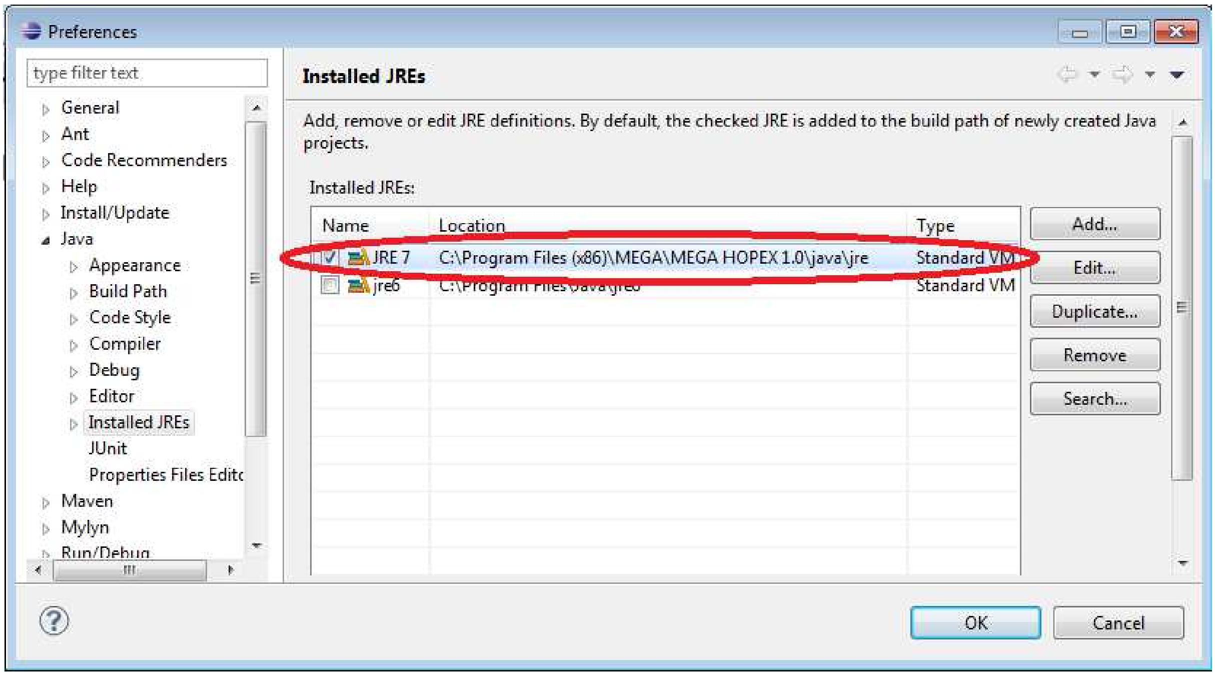
Selecting a JRE compatible with HOPEX in Eclipse

Selecting the JRE to be used by default in Java projects avoids using an incorrect JRE in new Java projects.

Using a JRE version delivered with HOPEX insures total compatibility.

To select in Eclipse a JRE compatible with HOPEX:

1. From the Eclipse menu bar, select **Window** > **Preferences**.
2. From the **Preferences** dialog box, in the left pane, expand **Java** node and select **Installed JREs**.
3. In the **Installed JREs** right pane, add a reference to the HOPEX compatible JRE.



Preferences

Installed JREs

Add, remove or edit JRE definitions. By default, the checked JRE is added to the build path of newly created Java projects.

Installed JREs:

Name	Location	Type
☑ JRE 7	C:\Program Files (x86)\MEGA\MEGA HOPEX 1.0\java\jre	Standard VM
□ jre6	C:\Program Files\Java\jre6	Standard VM

3.2.2 General writing conventions

Compliance with a writing convention is highly recommended.

The Eclipse "CheckStyle" plug-in enables integration in Eclipse of automatic management of Java source style validation.

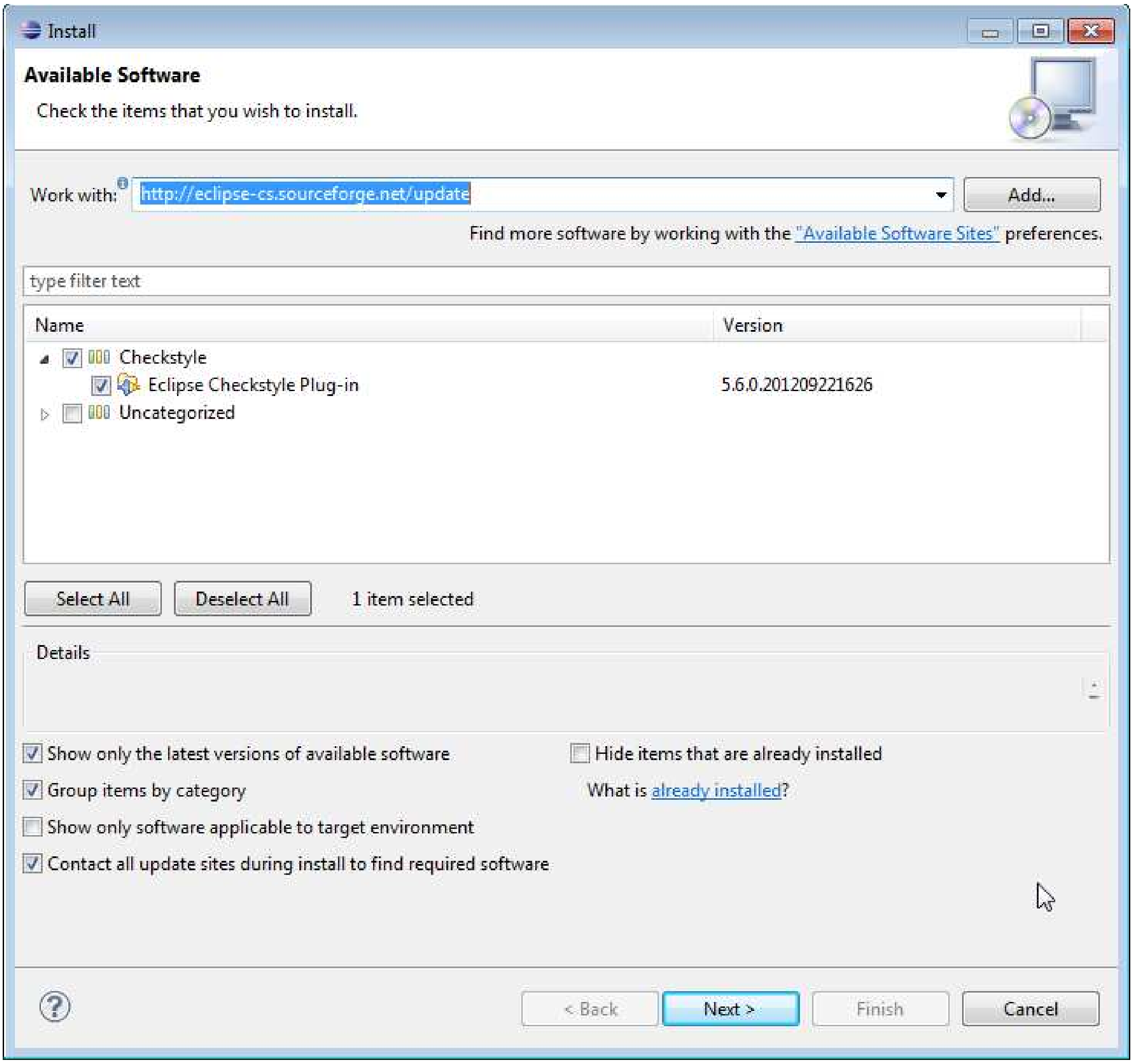
Installing the writing style validation plug-in

To install the writing style validation plug-in:

1. From Eclipse, in the menu bar, select **Help > Install New software**.

   The **Install** wizard appears.

2. In the **Work with** field, enter the site path: `http://eclipse-cs.sourceforge.net/update`.
3. Click **Add**.



Install

Available Software

Check the items that you wish to install.

Work with: http://eclipse-cs.sourceforge.net/update   Add...

Find more software by working with the "Available Software Sites" preferences.

type filter text

Name	Version
☑ Checkstyle	
☑ Eclipse Checkstyle Plug-in	5.6.0.201209221626
□ Uncategorized	

Select All   Deselect All   1 item selected

Details

☑ Show only the latest versions of available software

☑ Group items by category

□ Show only software applicable to target environment

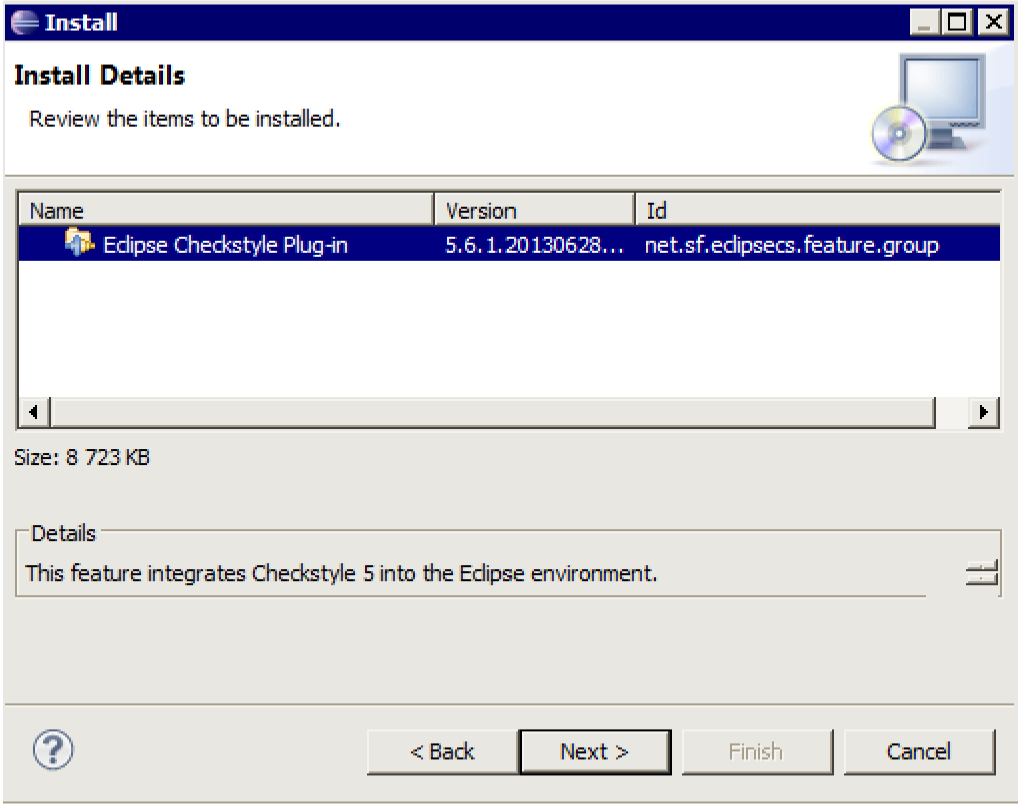
☑ Contact all update sites during install to find required software

□ Hide items that are already installed

What is already installed?

< Back   Next >   Finish   Cancel

4. In the pane, select **Eclipse CheckStyle Plug-in**.
5. Click **Next**.



Install

Install Details

Review the items to be installed.

Name	Version	Id
Eclipse Checkstyle Plug-in	5.6.1.20130628...	net.sf.eclipsecs.feature.group

Size: 8 723 KB

Details

This feature integrates Checkstyle 5 into the Eclipse environment.

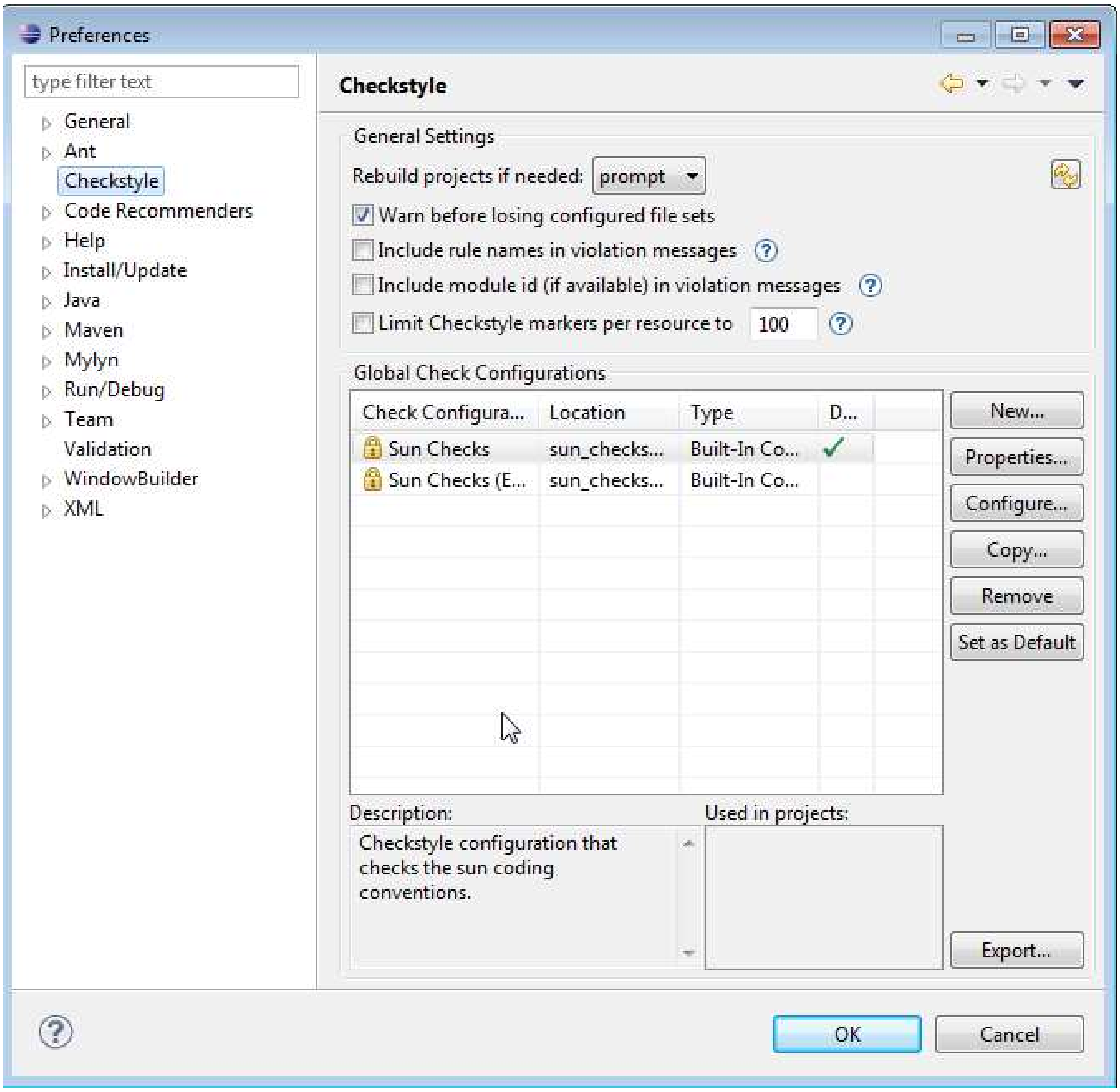
6. Click **Next**.
7. Accept the plug-in installation to execute the plug-in setup.

Configuring writing style check

To configure writing style check:

1. From Eclipse, in the menu bar, select **Window** > **Preferences**.

   The **Preferences** dialog box appears.



General Settings

Rebuild projects if needed: prompt

☑ Warn before losing configured file sets

□ Include rule names in violation messages

□ Include module id (if available) in violation messages

□ Limit Checkstyle markers per resource to 100

Global Check Configurations

Check Configura...	Location	Type	D...
Sun Checks	sun_checks...	Built-In Co...	✓
Sun Checks (E...	sun_checks...	Built-In Co...	

Description:

Checkstyle configuration that checks the sun coding conventions.

Used in projects:

2. From the **Preferences** dialog box, in the left pane, select **Checkstyle**.
3. In the **Checkstyle** right pane, select an encoding standard. Standards supplied with the plug-in are:
   - Sun development conventions.
   - Sun conventions slightly modified, adapted to Eclipse default formatting.

3.3 Calling a component written in Java from HOPEX

This section illustrates a call on a Java component using an example of how to create a MEGA macro implemented in Java.

This macro enables writing of text in a file.

3.3.1 Creating a Java macro in HOPEX

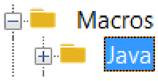
Configuring macros implemented in Java

All macros implemented in Java must reference the target class instanced and called via the macro:

1. From the macro properties dialog box, in the **Characteristics** tab, set the **SystemComponent** attribute value to "Dispatch”.
2. Attribute a value of form "java:*Package/Class*" to the "_ObjectFactory" property (see example in the next chapter). *Package* is the name of the Java package in which the referenced class is defined. Period characters (‘.’) of the package name must be replaced by slash characters (‘/’). The name of the class is separated from the name of the package by the slash character (‘/’).

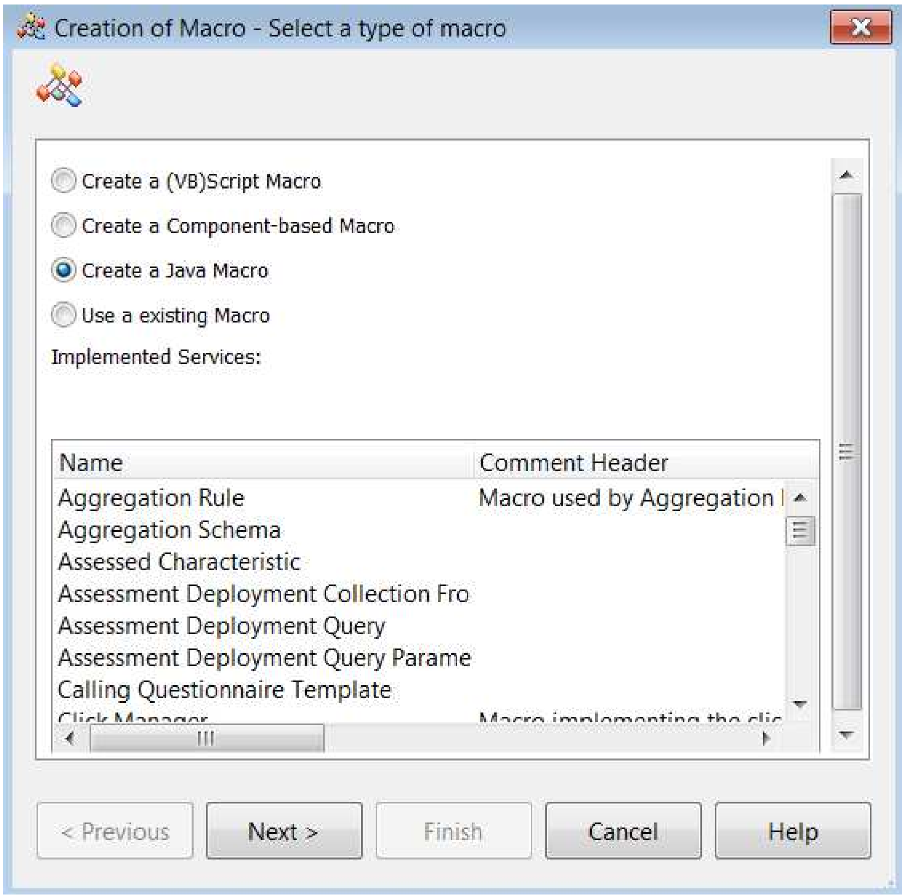
To create and configure a Java macro in HOPEX:

1. From the main HOPEX menu bar, select **View > Navigation Windows > Utilities**.
2. In the **Utilities** tab, to create a new macro folder, select **Macro > New > Folder of Macros**, and name it, for example “JAVA”.



3. From your “JAVA” folder, create a new Java Macro.

The **Creation of Macro** wizard appears.



Creation of Macro - Select a type of macro

○ Create a (VB)Script Macro

○ Create a Component-based Macro

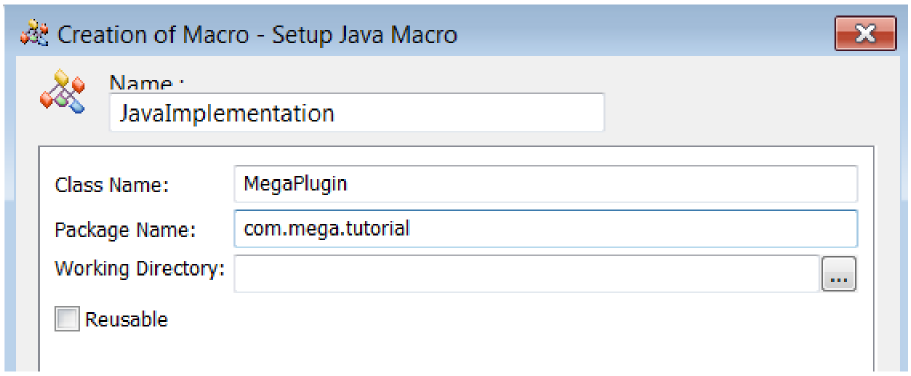
● Create a Java Macro

○ Use a existing Macro

Implemented Services:

Name	Comment Header
Aggregation Rule	Macro used by Aggregation I
Aggregation Schema	
Assessed Characteristic	
Assessment Deployment Collection Fro	
Assessment Deployment Query	
Assessment Deployment Query Parame	
Calling Questionnaire Template	
Click Manager	Macro implementing the clic

4. Click **Next**.
5. In the **Setup Java Macro**:
    a. In the **Class Name** field, enter for example “MegaPlugin”.
    b. In the **Package Name** field, enter for example “com.mega.tutorial”.



Creation of Macro - Setup Java Macro

Name : JavaImplementation

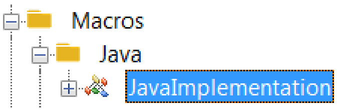
Class Name: MegaPlugin

Package Name: com.mega.tutorial

Working Directory:

□ Reusable

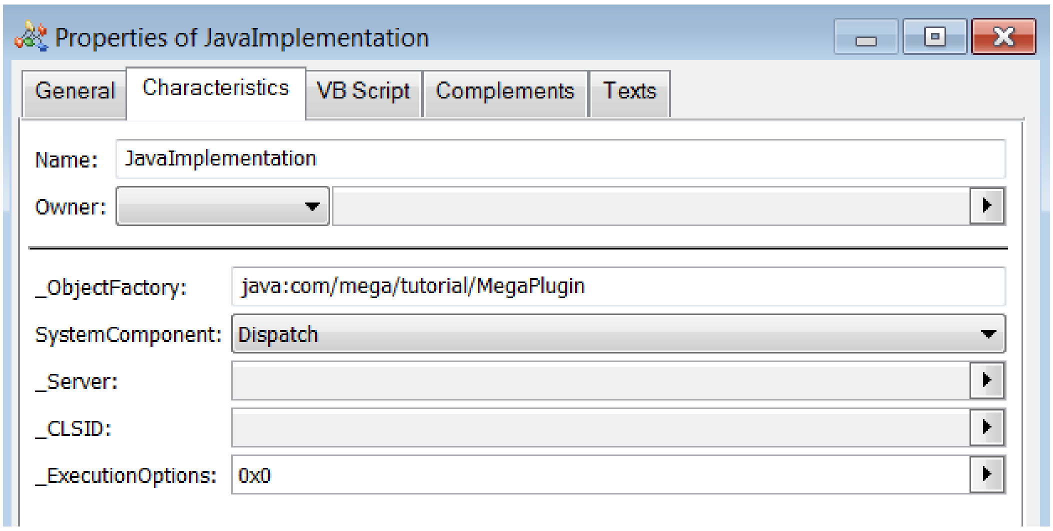
6. Click **Finish** to terminate the wizard.



7. Open the “JAVAImplementation” macro properties, and check that:
    – the **SystemComponent** attribute value is "Dispatch”.

- the **_ObjectFactory** attribute contains the package and class name with the syntax defined in the above section.

  The value here is defined as "java:com/mega/tutorial/MegaPlugin" (identifies the "MegaPlugin" class of the "com.mega.tutorial" package):



Name: JavaImplementation

Owner:

_ObjectFactory: java:com/mega/tutorial/MegaPlugin

SystemComponent: Dispatch

_Server:

_CLSID:

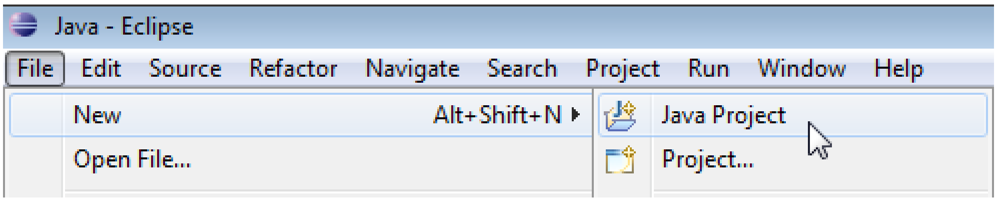
_ExecutionOptions: 0x0

3.3.2 Creating an Eclipse project

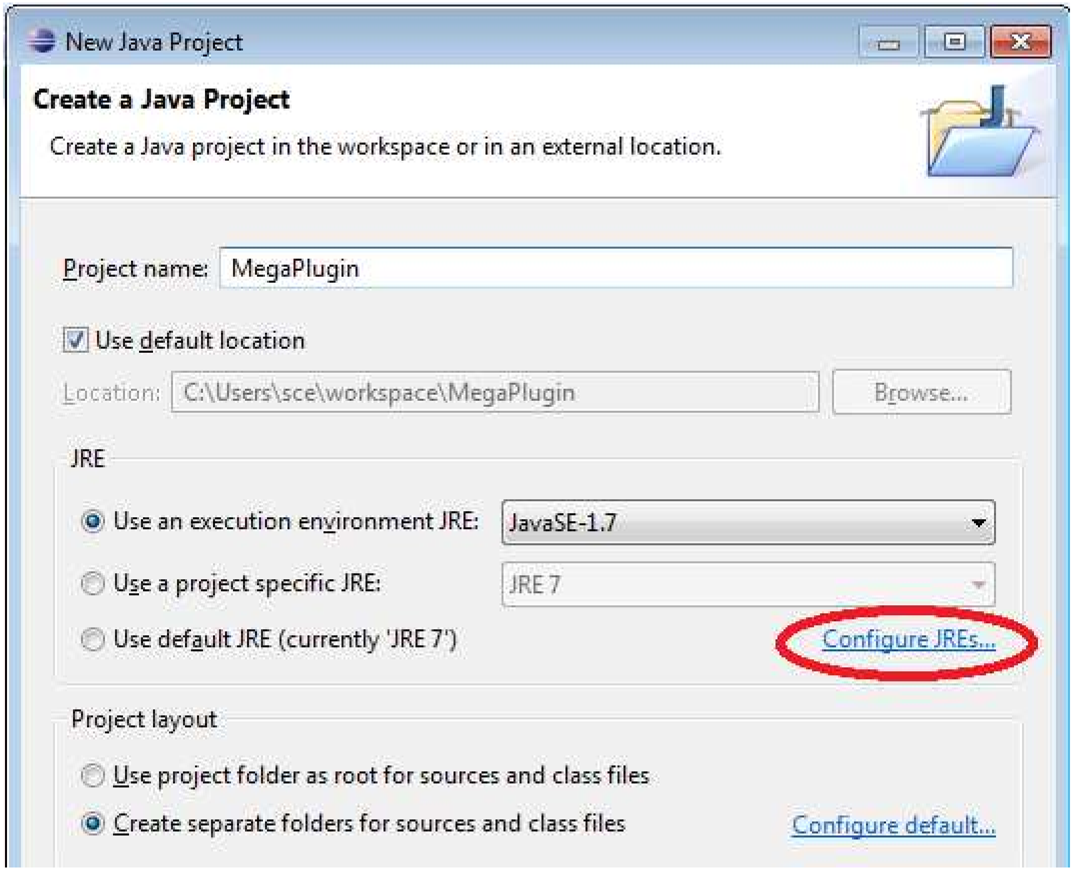
To create the Java component, the "Java" perspective should be used (in Eclipse online Help, see the information relating to use of perspectives).

To create a new Java project:

1. From the Eclipse menu bar, select **File > New > Java project**.



2. Enter the Java **Project name**.
3. Ensure that the JRE used is compatible with the MEGA version for which the Java component is created ("Configure JREs").



New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: MegaPlugin

☑ Use default location

Location: C:\Users\sce\workspace\MegaPlugin

JRE

● Use an execution environment JRE: JavaSE-1.7

○ Use a project specific JRE: JRE 7

○ Use default JRE (currently 'JRE 7')

Project layout

○ Use project folder as root for sources and class files

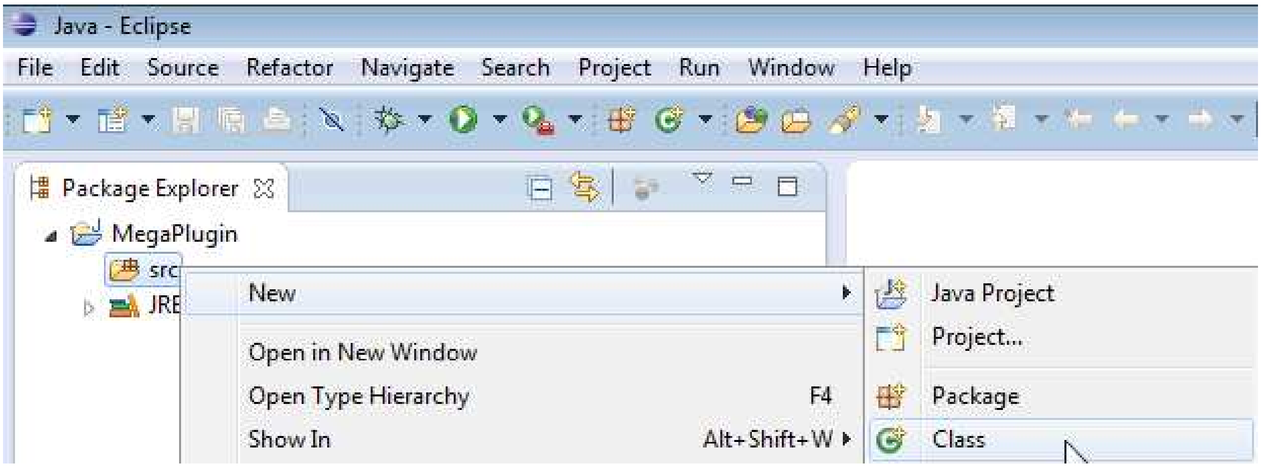
● Create separate folders for sources and class files

4. Click **Finish** to validate the Java project creation.

Creating implementation class of the macro

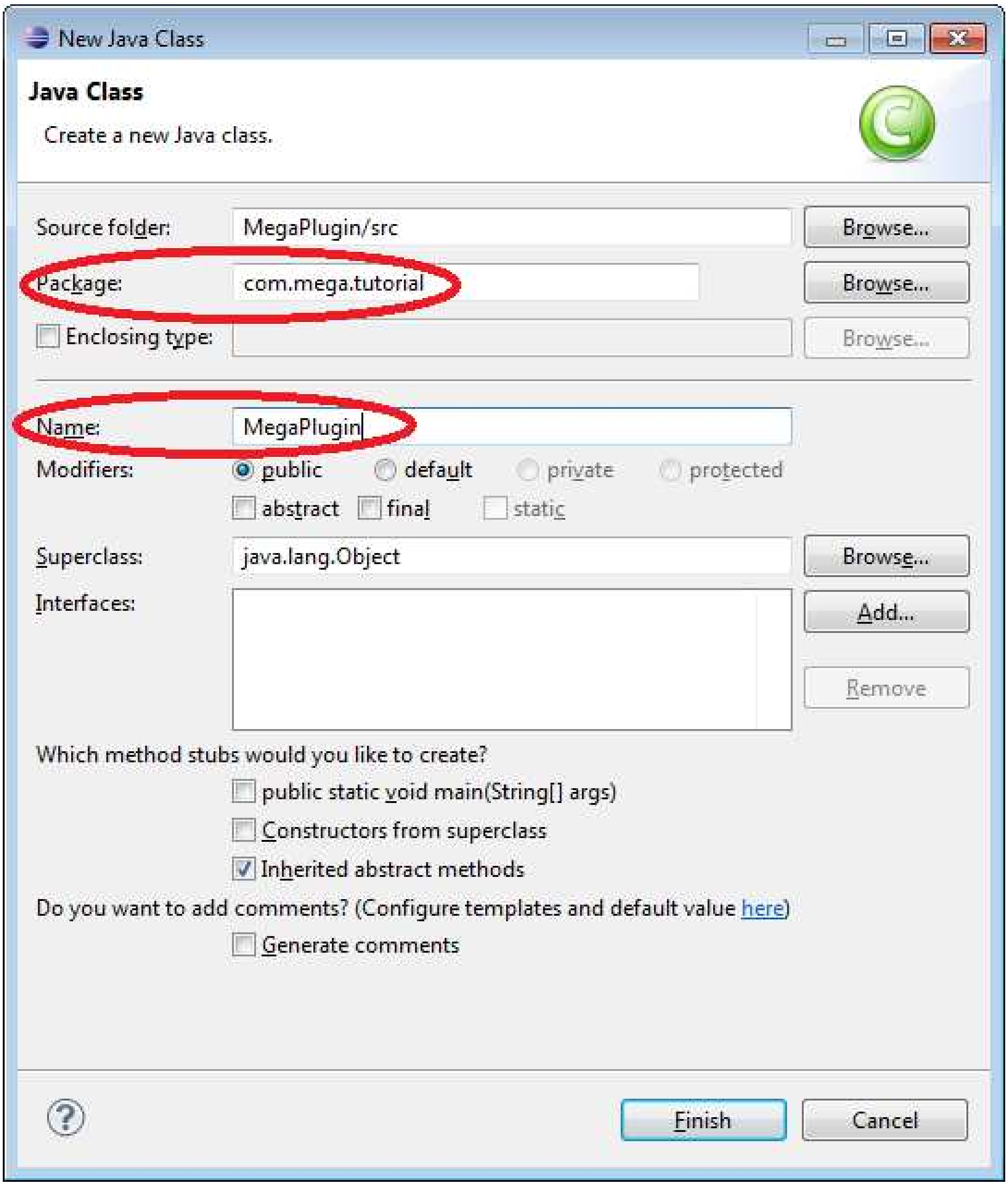
To create an implementation class of the macro:

1. From the Eclipse menu bar, select **Window > Show View > Package Explorer**.
2. In the **Package Explorer** tab, expand the **MegaPlugin** project
3. Right-click the **src** folder and select **New > Class**.



4. In the **Package** field, enter "com.mega.tutorial".

5. In the **Name** field enter the class name "MegaPlugin":



New Java Class

Java Class

Create a new Java class.

Source folder: MegaPlugin/src

Package: com.mega.tutorial

□ Enclosing type:

Name: MegaPlugin

Modifiers: ● public ○ default ○ private ○ protected

□ abstract □ final □ static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

□ public static void main(String[] args)

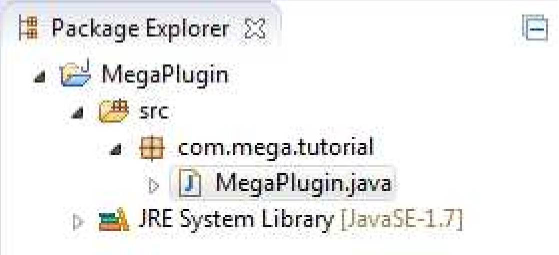
□ Constructors from superclass

☑ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

□ Generate comments

6. Click **Finish**.



Package Explorer

- MegaPlugin
  - src
    - com.mega.tutorial
      - MegaPlugin.java
  - JRE System Library [JavaSE-1.7]

Note: As previously seen, the package and class created are those identified in the _ObjectFactory" parameter of the corresponding MEGA macro.

3.3.3 Implementing the macro

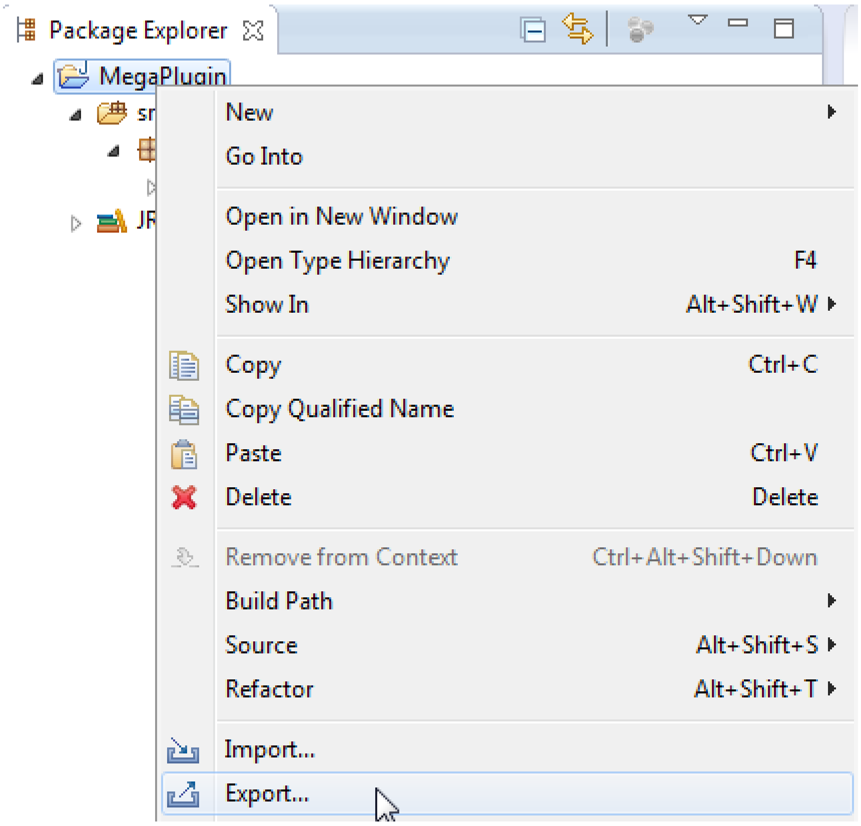
Implementation of the macro consists of a class exposing the writeInFile public method that can be called from HOPEX:

```java
// Package containing the implementation class.
package com.mega.tutorial;

// Need to write some text in a file.
import java.io.FileWriter;
// Need to throw exception managed by MEGA.
import java.io.IOException;

/**
 * MegaPlugin Macro implementation class.
 */
public class MegaPlugin {

      /**
       * MegaPlugin Macro implementation class.
       * @param   filename      Name of the file created.
       * @param   text          Some text that will be passed from MEGA.
       * @throws   IOException  Method called from MEGA must throw
       *                                      IOException exception.
       */
      public void writeInFile(String fileName, String text) throws IOException
      {
            FileWriter writer = null;
            // Open a FileWriter with file name
            writer = new FileWriter(fileName);

            // Write the text in the file
            writer.write(text);

            //Close the file
            if(writer!=null)
            {
                  writer.close();
            }
      }
}
```

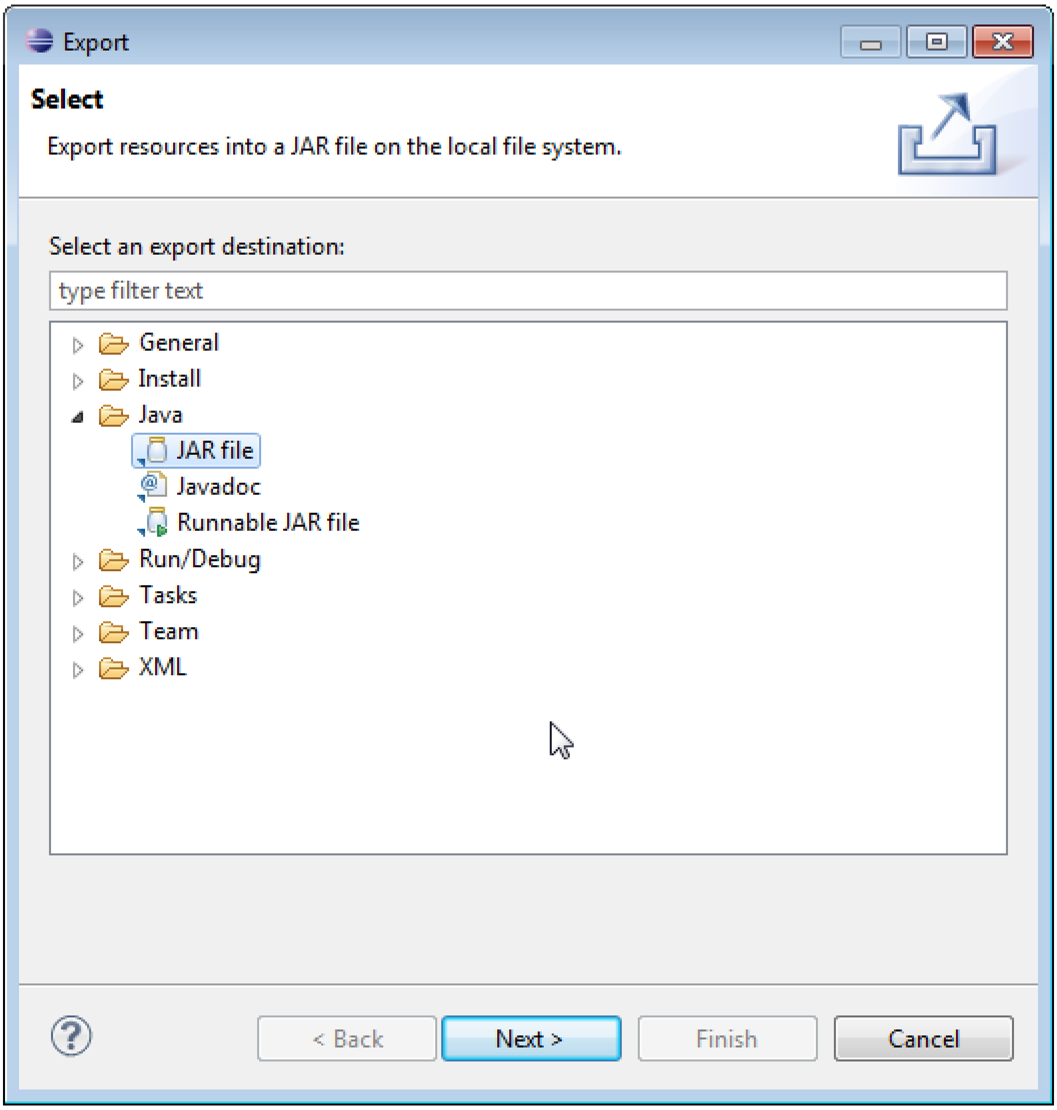
Compiling and deploying the Java component

To compile the Java component in the form of a JAR file

1. In the Java project, right-click MegaPlugin and select **Export**.



2. In the **Export** wizard, expand the **Java** folder and select **JAR file**.



Export

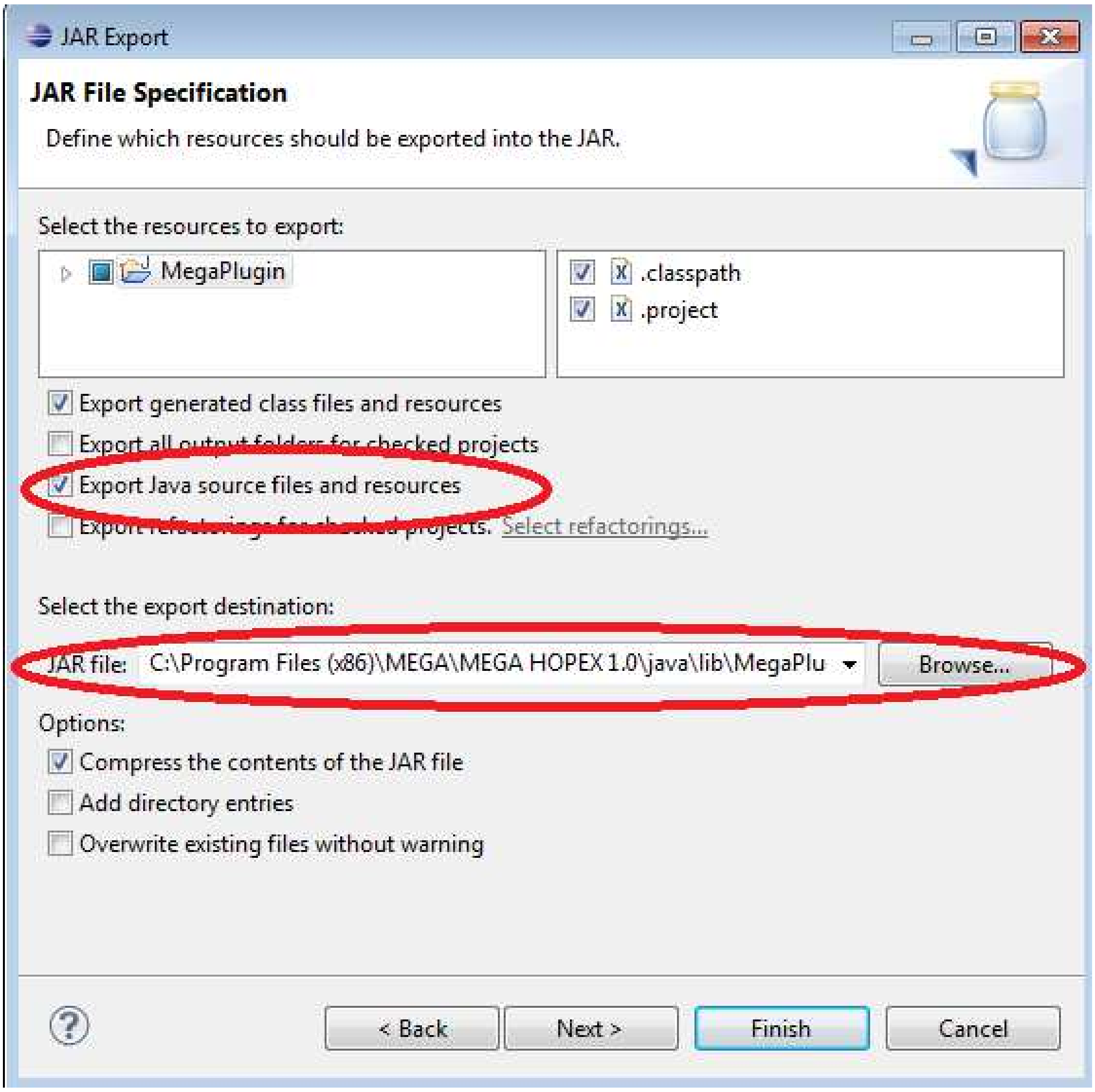
Select

Export resources into a JAR file on the local file system.

Select an export destination:

type filter text

3. Click **Next**.



JAR Export

JAR File Specification

Define which resources should be exported into the JAR.

Select the resources to export:

- MegaPlugin
- ☑ .classpath
- ☑ .project

☑ Export generated class files and resources

☐ Export all output folders for checked projects

☑ Export Java source files and resources

☐ Export refactorings for checked projects. Select refactorings...

Select the export destination:

JAR file: C:\Program Files (x86)\MEGA\MEGA HOPEX 1.0\java\lib\MegaPlu

Options:

☑ Compress the contents of the JAR file

☐ Add directory entries

☐ Overwrite existing files without warning

4. Make sure **Export Java source files and resources** is selected.
5. In the **Select the export destination** field, click **Browse** and indicate the location of the JAR file to be generated.
6. Click **Finish**.

Note 1: The JAR file must be generated (or copied after generation) in the "java\lib" directory of the HOPEX installation site.

Note 2: It is important to export your Java source files. Indeed, once the project is delivered, the end user may encounter problems and call MEGA Technical Support. This will enable the Support to better diagnose the problem. Also, the customer may ask you to add additional customizations. It would be a shame to have to rewrite the full code. With this option, the code is kept and can be reused.

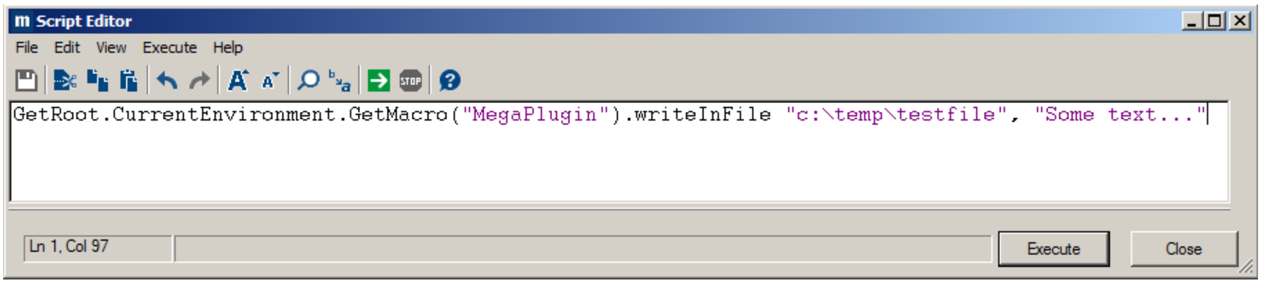
3.3.4 Executing the macro

When the JAR file has been deployed in the HOPEX "java\lib" directory, the macro can be tested.

To test the macro:

1. In the HOPEX script editor, perform the following command:

```
GetRoot.CurrentEnvironment.GetMacro("MegaPlugin").writeInFile
"c:\temp\testfile", "Some text..."
```



```
GetRoot.CurrentEnvironment.GetMacro("MegaPlugin").writeInFile "c:\temp\testfile", "Some text..."
```

3.4 Using the MEGA API in a component written in Java

This chapter is an introduction to implementation of Java MEGA APIs using the example of the creation of a "Command" that can be executed from a Business Process's contextual menu.

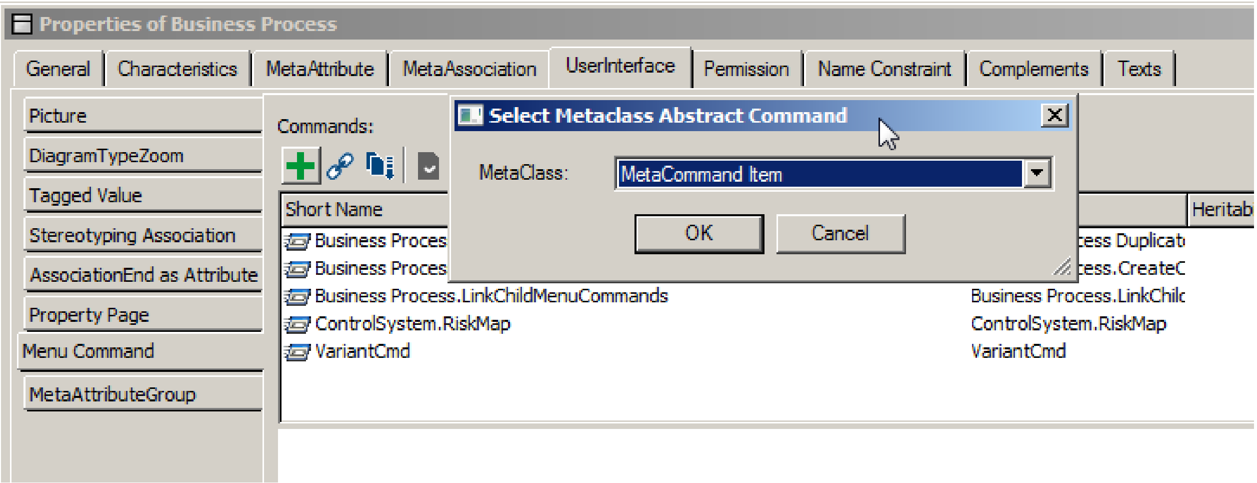
The effect of this Command is to export the hierarchy of Business Processes into a text file.

Note: The Java MEGA API is not detailed in this document. Java MEGA API documentation is provided in the HOPEX installation "java\doc" directory and is also accessible from HOPEX online documentation (**HOPEX Customization (Windows) > Using APIs > JavaDoc**).

3.4.1 Creating the Command

To create the command:

1. From **HOPEX MetaStudio**, open the "Business Process" MetaClass properties.
2. In the **UserInterface** tab, select the **Menu Command** sub tab, and create a new MetaCommand Item named "ExportPlugin":



MetaClass: MetaCommand Item

Business Proces — ess Duplicat

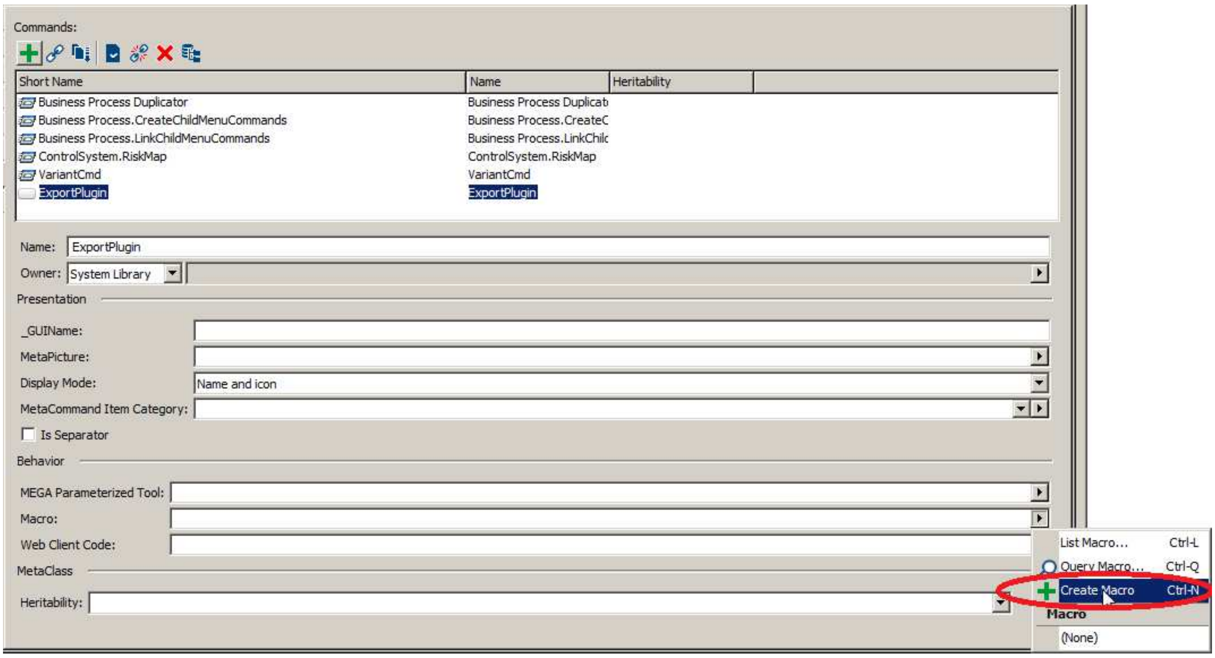
Business Proces — ess.CreateC

Business Process.LinkChildMenuCommands — Business Process.LinkChilc

ControlSystem.RiskMap — ControlSystem.RiskMap

VariantCmd — VariantCmd

3. Create a macro from the **Macro** field of the MetaCommand:



Short Name	Name	Heritability
Business Process Duplicator	Business Process Duplicat	
Business Process.CreateChildMenuCommands	Business Process.CreateC	
Business Process.LinkChildMenuCommands	Business Process.LinkChilc	
ControlSystem.RiskMap	ControlSystem.RiskMap	
VariantCmd	VariantCmd	
ExportPlugin	ExportPlugin	

Name: ExportPlugin

Owner: System Library

_GUIName:

MetaPicture:

Display Mode: Name and icon

MetaCommand Item Category:

□ Is Separator

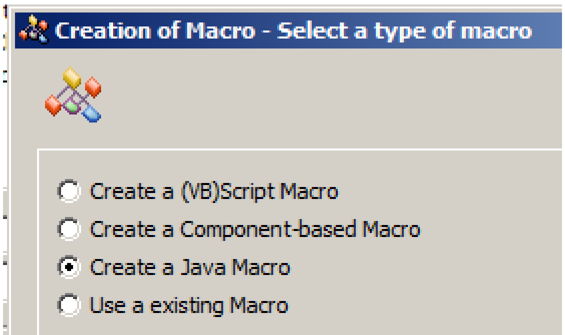
MEGA Parameterized Tool:

Macro:

Web Client Code:

Heritability:

4. Select the creation of a Java macro:



○ Create a (VB)Script Macro

○ Create a Component-based Macro

● Create a Java Macro

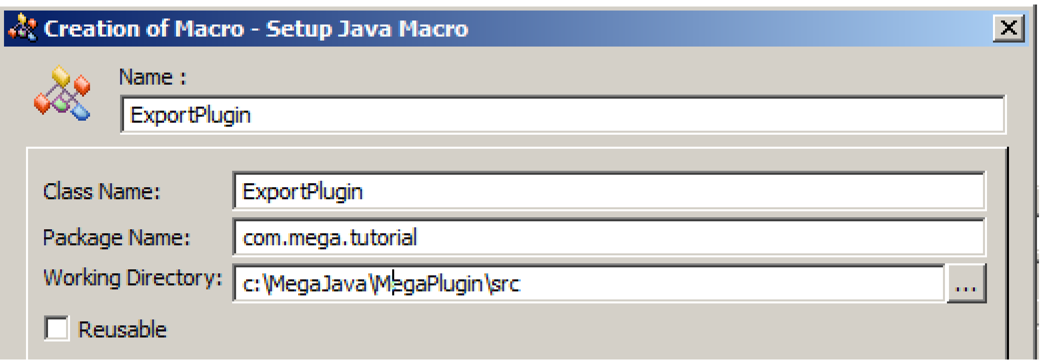
○ Use a existing Macro

5. Name the macro "ExportPlugIn" and specify the following parameters:

   a) “Class Name” identifies the Java class called, specify “ExportPlugin”.

"Package Name" identifies the package of the Java class called, specify "com.mega.tutorial".

b) "Working Directory" identifies the directory in which to generate the Java class initialized with the functions to be implemented.



Name : ExportPlugin

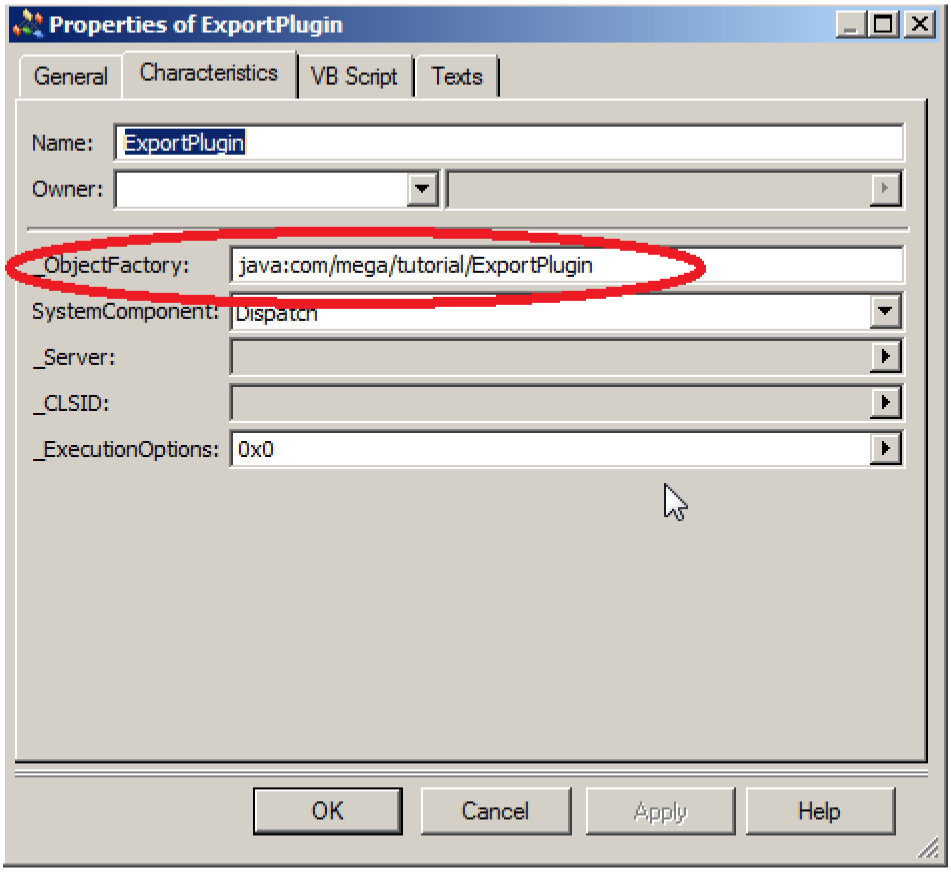
Class Name: ExportPlugin

Package Name: com.mega.tutorial

Working Directory: c:\MegaJava\MegaPlugin\src

□ Reusable

6. Creation of the macro initializes the "ObjectFactory" and "SystemComponent" properties. In the macro properties, the **_ObjectFactory** field is specified with the identifier of the MetaCommand implementation Java class ("java:*Package/Class*"):



Name: ExportPlugin

Owner:

_ObjectFactory: java:com/mega/tutorial/ExportPlugin

SystemComponent: Dispatch

_Server:

_CLSID:

_ExecutionOptions: 0x0

7. The Java class is generated, according to the interface to be implemented, in the directory previously specified:

```
package com.mega.tutorial;
import com.mega.modeling.api.*;
public class ExportPlugin {
/**
```

```
        * Function called when the command is triggered from a MegaObject.
        * Example: click on a menu.
        *
        * @param mgobjSource
        *            MegaObject on which the command is applied.
        */
        public void InvokeOnObject(MegaObject mgobjSource, Object userData) {
        }
    }
```

3.4.2 Configuring the Eclipse project for use of Java MEGA API

In this section, the Eclipse project « MegaPlugin » used in the Creating an Eclipse project section is reused.

Referencing the Java MEGA API access component

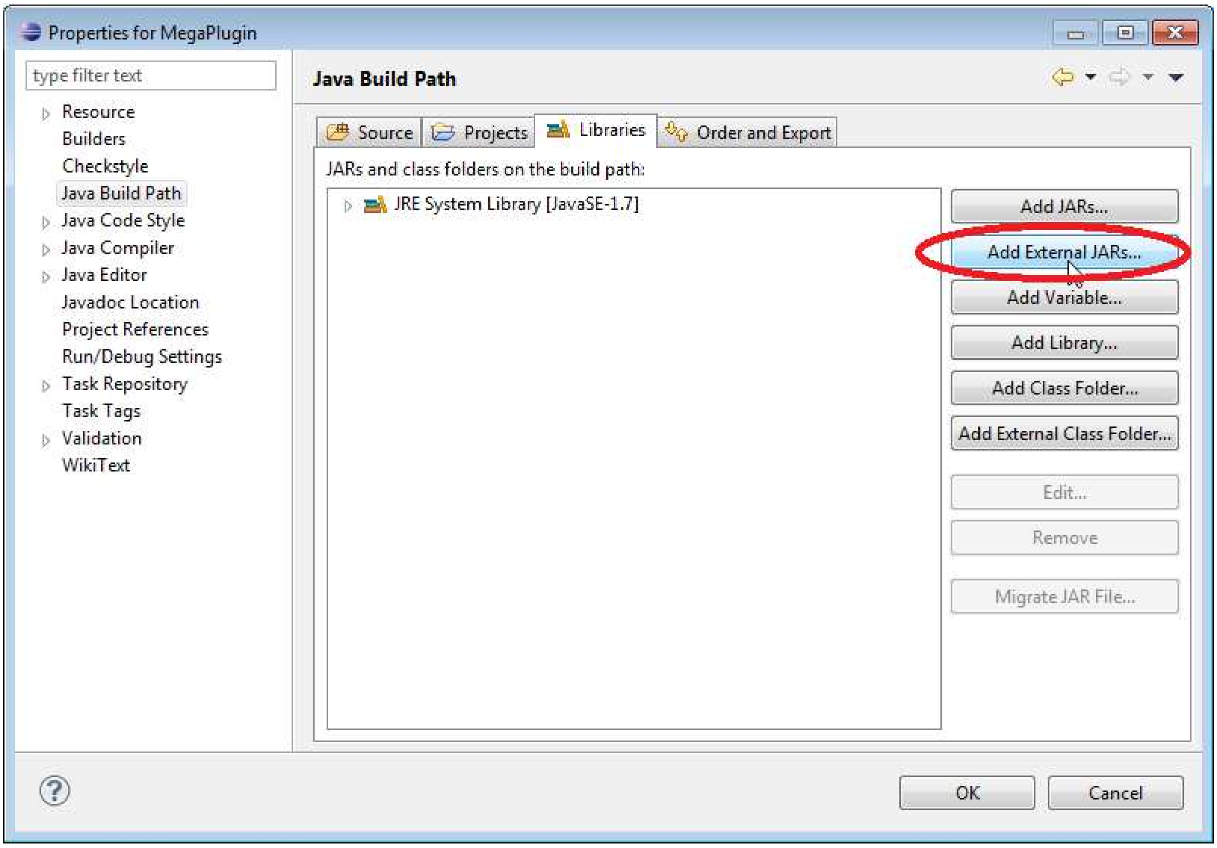
So that implementation of the MetaCommand Item can use Java MEGA APIs, the API access component must be referenced by the “MegaPlugin” Eclipse project.

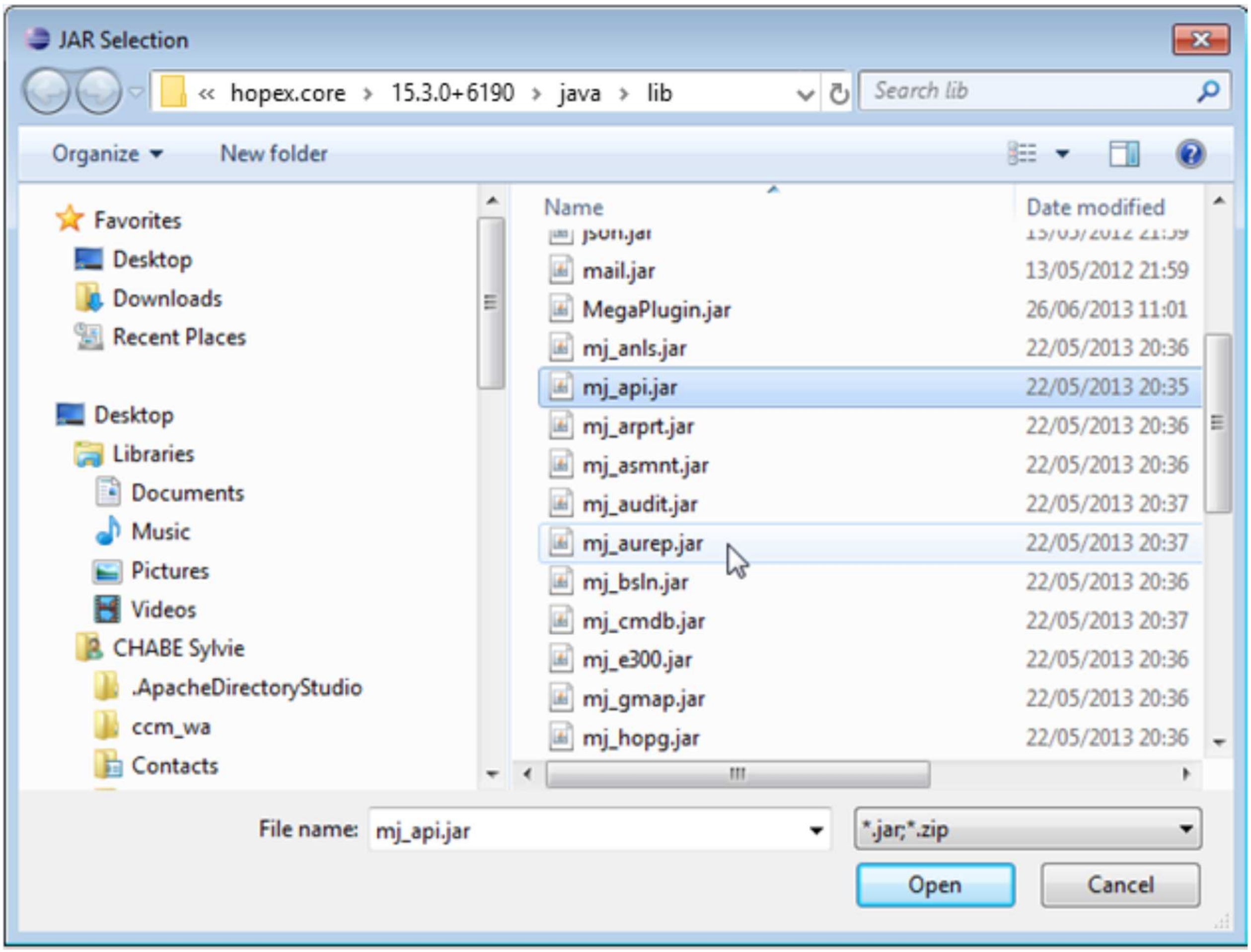
The “java\lib” directory of the HOPEX installation site contains the Java components delivered by MEGA. Among these, file "mj_api.jar" provides access to Java MEGA API.

Note: As previously seen, customized Java components must also be deployed in the "java\lib" directory.

To reference the "mj_api.jar" component:

1. In "Package Explorer" select "Properties" of the "MegaPlugin" project to open its properties.
2. In the **Java Build Path** node, **Libraries** tab, add the reference to file "mj_api.jar":

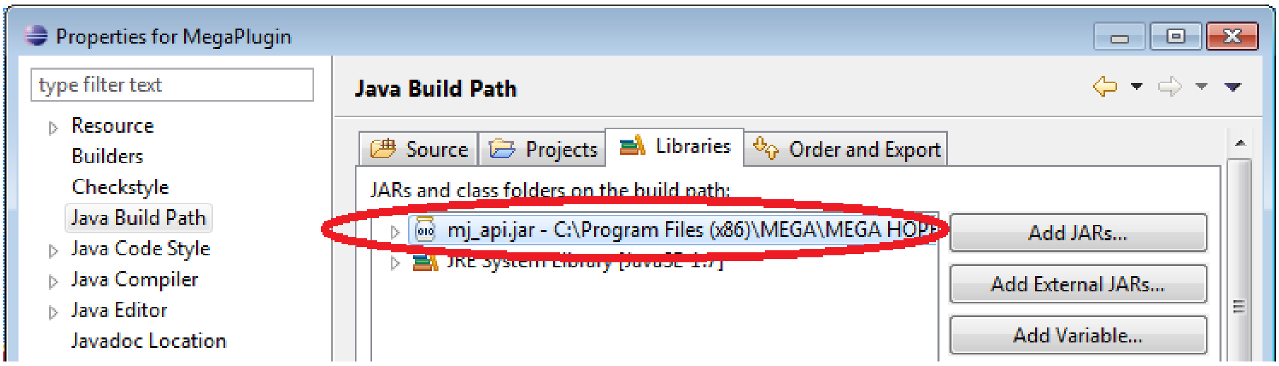




Name	Date modified
json.jar	13/05/2012 21:59
mail.jar	13/05/2012 21:59
MegaPlugin.jar	26/06/2013 11:01
mj_anls.jar	22/05/2013 20:36
mj_api.jar	22/05/2013 20:35
mj_arprt.jar	22/05/2013 20:36
mj_asmnt.jar	22/05/2013 20:36
mj_audit.jar	22/05/2013 20:37
mj_aurep.jar	22/05/2013 20:37
mj_bsln.jar	22/05/2013 20:36
mj_cmdb.jar	22/05/2013 20:37
mj_e300.jar	22/05/2013 20:36
mj_gmap.jar	22/05/2013 20:36
mj_hopg.jar	22/05/2013 20:36

File name: mj_api.jar

.jar;.zip



Integrating API documentation in the development environment

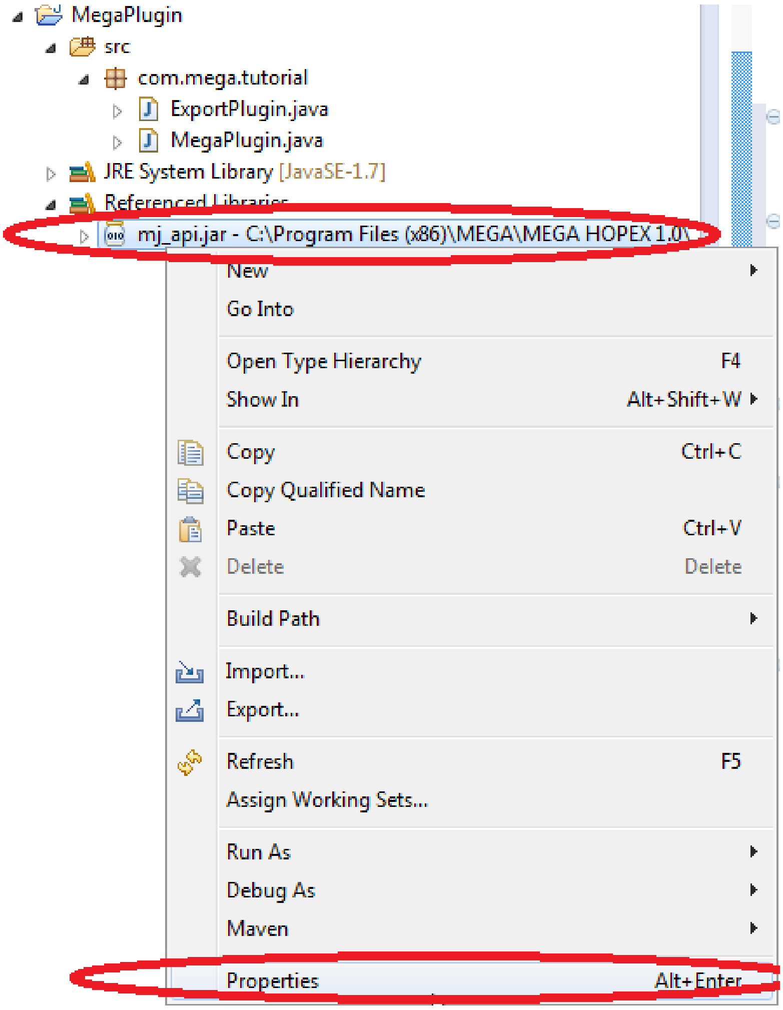
Java MEGA API documentation is delivered in the form of an archive, file "mj_api.zip" in the "java\doc" directory of the HOPEX installation.

The archive contains documentation in HTML format generated by JavaDoc.

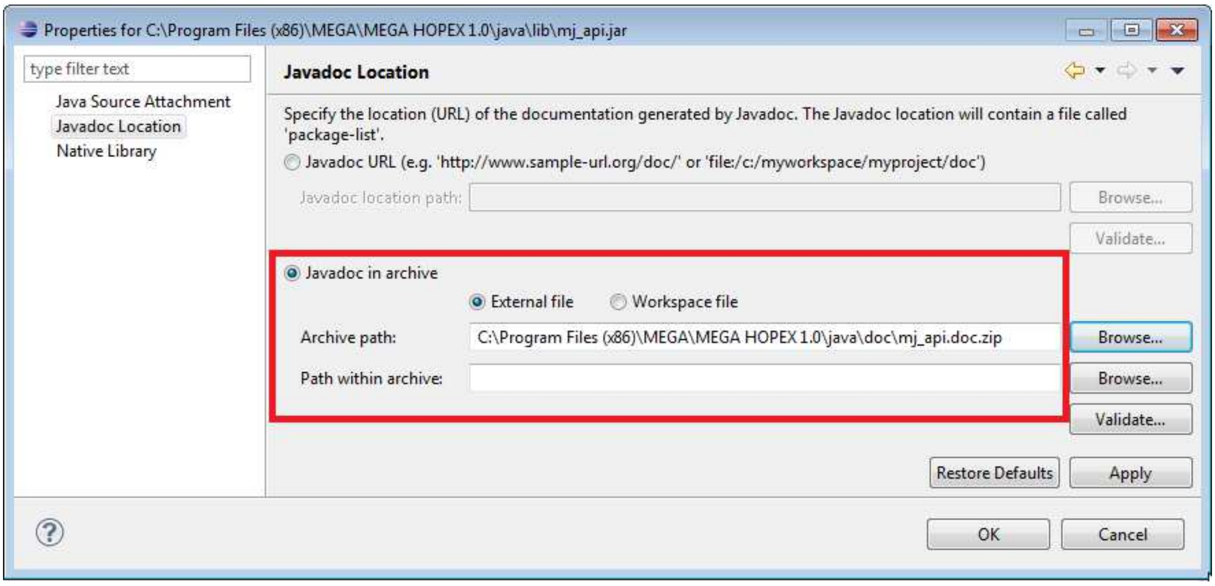
To benefit from total integration of Java MEGA API help, API documentation should be associated by referencing the file "mj_api.jar".

To associate JavaDoc help with the "mj_api.jar" component, specify the location of the archive containing help in the properties of the referenced "mj_api.jar" library:

1. Open the "mj_api.jar" library properties:



2. In the node "Javadoc Location", specify the location of the JavaDoc help (MEGA API JavaDoc help is available as a compressed archive in the "java\doc" directory of the HOPEX installation):


Properties for C:\Program Files (x86)\MEGA\MEGA HOPEX 1.0\java\lib\mj_api.jar

Javadoc Location

Specify the location (URL) of the documentation generated by Javadoc. The Javadoc location will contain a file called 'package-list'.

○ Javadoc URL (e.g. 'http://www.sample-url.org/doc/' or 'file:/c:/myworkspace/myproject/doc')

Javadoc location path:

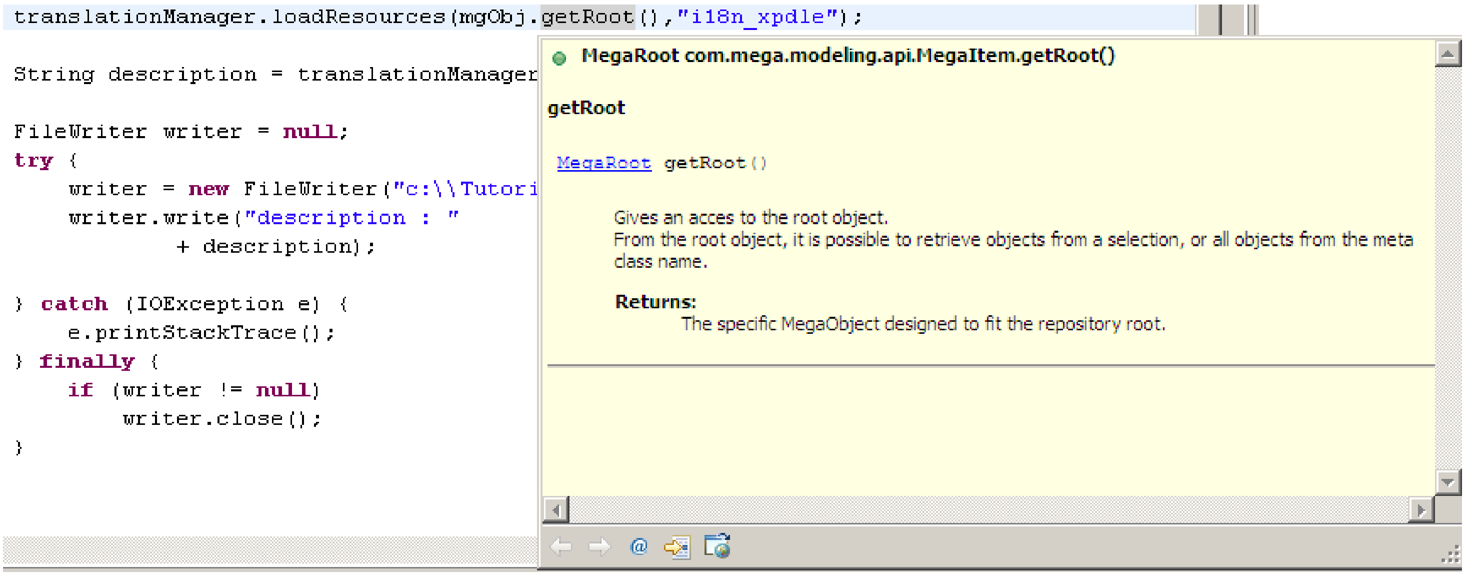
● Javadoc in archive

● External file ○ Workspace file

Archive path: C:\Program Files (x86)\MEGA\MEGA HOPEX 1.0\java\doc\mj_api.doc.zip

Path within archive:

Once the JavaDoc help has been referenced, it may be used by positioning the cursor on documented keywords.



```
translationManager.loadResources(mgObj.getRoot(),"i18n_xpdle");

String description = translationManager

FileWriter writer = null;
try {
    writer = new FileWriter("c:\\Tutori
    writer.write("description : "
            + description);

} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (writer != null)
        writer.close();
}
```

● MegaRoot com.mega.modeling.api.MegaItem.getRoot()

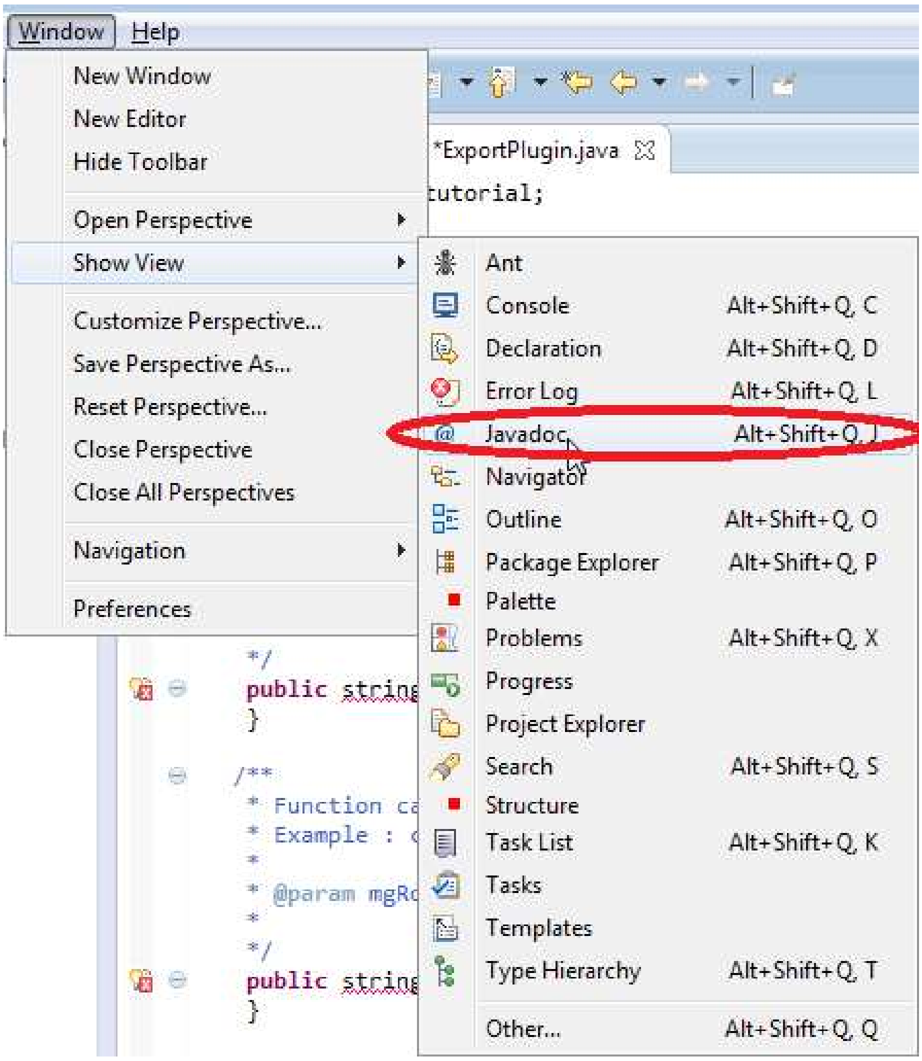
getRoot

MegaRoot getRoot()

Gives an acces to the root object.
From the root object, it is possible to retrieve objects from a selection, or all objects from the meta class name.

Returns:
The specific MegaObject designed to fit the repository root.

Javadoc help can be permanently displayed in a dedicated window. To display this window, select **Window** > **Show View** > **Javadoc**:



```
tutorial;
    */
    public string
    }

    /**
     * Function ca
     * Example : 
     *
     * @param mgRo
     *
     */
    public string
    }
```

3.4.3 Generating identifiers for the metamodel used

MEGA APIs enable exploration and modification of models described in the HOPEX repository.

The concepts (MetaClasses, MetaAssociationEnds, MetaAttributes, etc.) used through an API can be identified by their name or identifier.

Example finding sub-processes of a business process:

```
subProcesses = process.getCollection("Component");
```

It is highly recommended to use the identifiers of concepts to ensure compatibility with languages and potential future metamodel modifications.

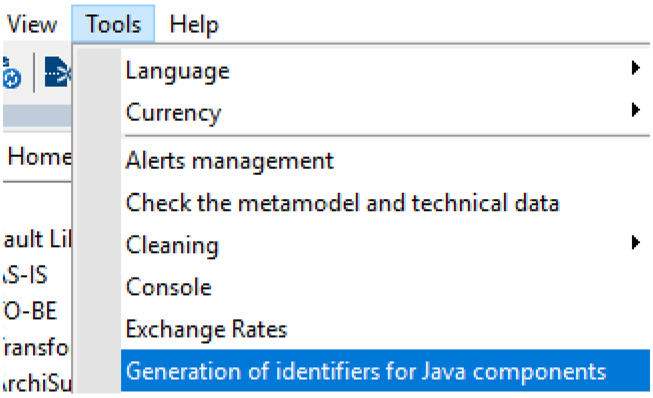
The example above can therefore be written as:

```
// using the identifier in its Absolute Identifier (IdAbs) format
subProcesses = process.getCollection("81)gvmQ9pKE0");
// using the identifier in its hexadecimal (_HexaIdAbs) format
subProcesses = process.getCollection("ABFBAC39332500E5");
// using the identifier in its MEGA field format
subProcesses = process.getCollection("~81)gvmQ9pKE0[Component]");
```

A wizard enables generation of classes defining identifiers to be used for a set of MetaClasses.

To execute this wizard:

1. From HOPEX **Tools** menu, select **Generation of identifiers for Java components**.



2. In the wizard, specify:
   - the Java classes generation path

     Folders managed by HOPEX are:

     - custom module folder and sub-folders

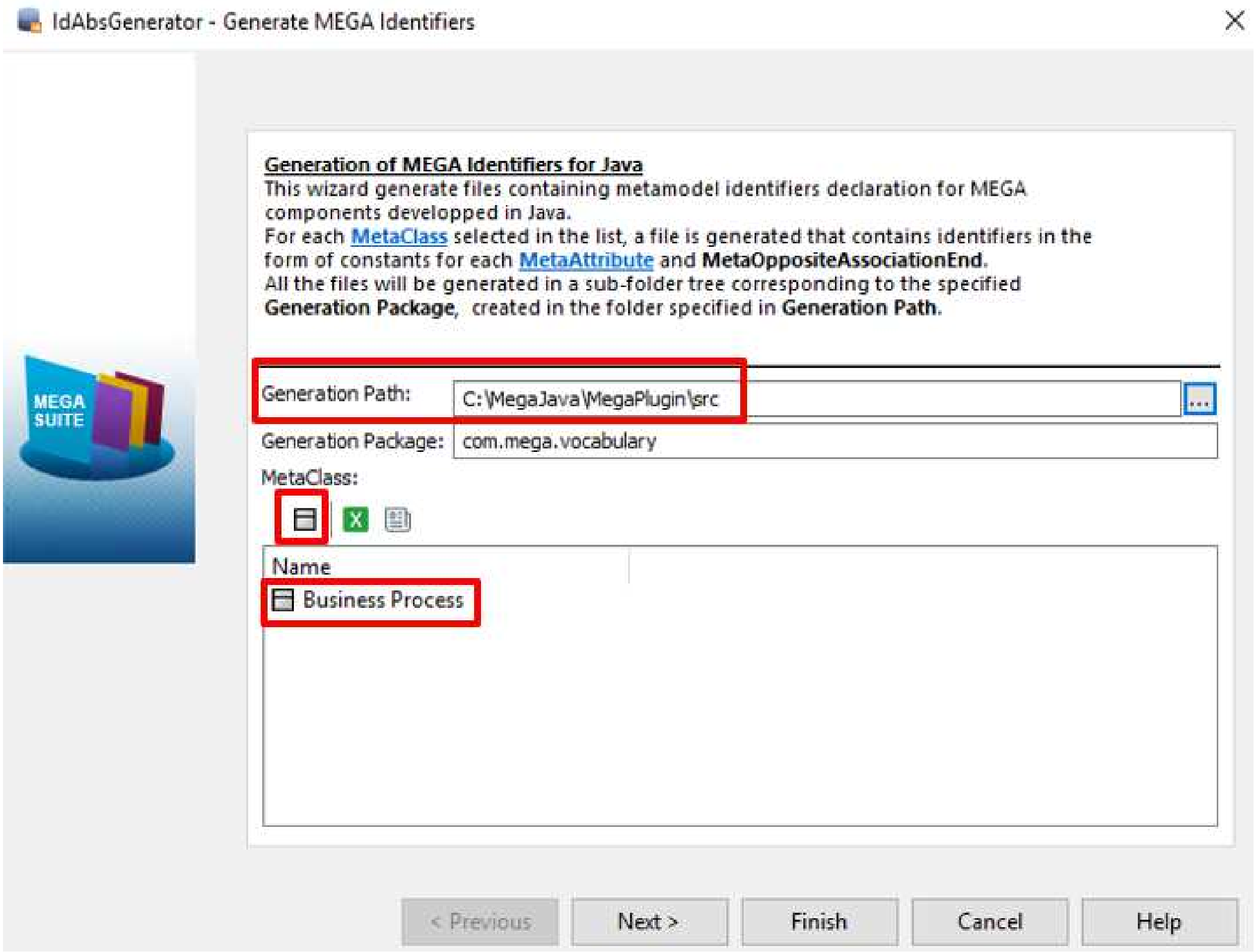
     - current user folder under: [Environment]\SysDb\USER\XXX

     - current user folder under: [Environment]\Db\<Repository name>\USER \XXX

     - WORK folder under: [Environment]\SysDb\WORK

     - WORK folder under: [Environment]\Db\<Repository name>\WORK

     You can also add other folders adding them to the "MacroFSOAllowedFolders.json" file of your custom module.
   - the MetaClasses for which identifiers must be generated.


IdAbsGenerator - Generate MEGA Identifiers

Generation of MEGA Identifiers for Java
This wizard generate files containing metamodel identifiers declaration for MEGA components developped in Java.
For each MetaClass selected in the list, a file is generated that contains identifiers in the form of constants for each MetaAttribute and **MetaOppositeAssociationEnd.**
All the files will be generated in a sub-folder tree corresponding to the specified **Generation Package**, created in the folder specified in **Generation Path.**

Generation Path: C:\MegaJava\MegaPlugin\src

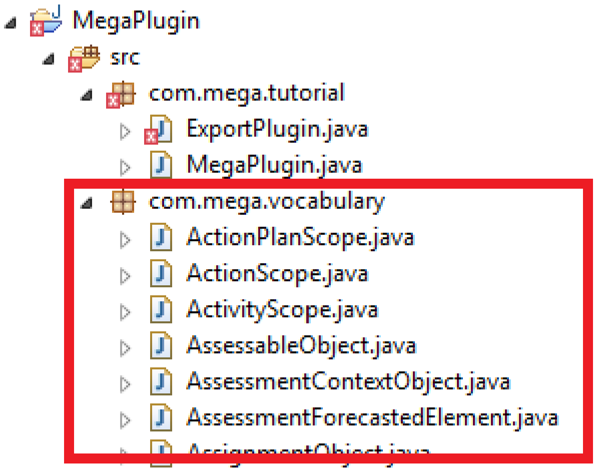
Generation Package: com.mega.vocabulary

MetaClass:

Name
Business Process

In our example, files will be generated in the directory containing the sources: "C:\MegaJava\MegaPlugin\src".

A package "com.mega.vocabulary" is created containing classes corresponding to declarations of selected MetaClasses as well as their inherited MetaClasses:


```
MegaPlugin
  src
    com.mega.tutorial
      ExportPlugin.java
      MegaPlugin.java
    com.mega.vocabulary
      ActionPlanScope.java
      ActionScope.java
      ActivityScope.java
      AssessableObject.java
      AssessmentContextObject.java
      AssessmentForecastedElement.java
      AssignmentObject.java
```

3.4.4 Implementing the MetaCommand Item

```
package com.mega.tutorial;

import com.mega.modeling.api.*;
import com.mega.vocabulary.*;
import java.io.FileWriter;
import java.io.IOException;

public class ExportPlugin {

   private static String filePath = "C:\\Temp\\BusinessProcess.txt";



   /**
    * Function called when the command is triggered from a MegaObject.
    * Example: click on a menu.
    *
    * @param mgobjSource
    *            MegaObject on which the command is applied.
    */
    public void InvokeOnObject(MegaObject mgobjSource, Object userData) throws
IOException
    {
          FileWriter writer = null;
          MegaObject process = null;

          // Source object is a process
          process = mgobjSource;

          // Open a FileWriter with file name defined statically in the class
          writer = new FileWriter(filePath);

          // Writes the source process name
          writer.write(process.getProp(BusinessProcess.MA_GenericLocalName) +
"\r\n");

       // Get the collection of sub-processes
          // Uses: MegaObject.getCollection function with the identifier
          //        of sub-process generated in the com.mega.vocabulary package
          MegaCollection subProcessesCollection = null;
          subProcessesCollection =
process.getCollection(BusinessProcess.MAE_Component);
```

```
       // Search for all the sub-processes
          for (int subProcessesIndex = 1;
                 subProcessesIndex < subProcessesCollection.size() + 1;
                 subProcessesIndex++)
          {
                MegaObject subProcess = null;

           // Get the sub process in the collection at subProcessesIndex
position
           subProcess = subProcessesCollection.get(subProcessesIndex);

                // Writes the sub-process name
                writer.write("   " +
subProcess.getProp(BusinessProcess.MA_GenericLocalName) + "\r\n");
          }
          if(writer!=null)
          {
                writer.close();
          }
    }

}
```

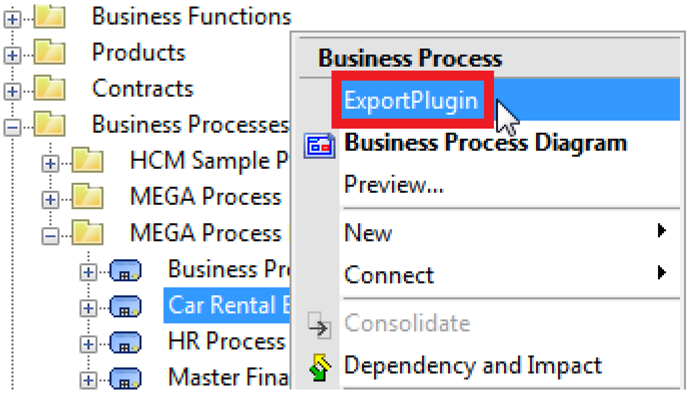
Compiling and deploying

Follow the Compiling and deploying the Java component procedure described in the Calling a component written in Java from section.

If changes are taken into account, after Java code modification, regenerate the Jar file and copy it in the MEGA java/lib directory, then exit and restart HOPEX.

Using the MetaCommand Item

To execute the MetaCommand Item, in the Business Process contextual menu select **ExportPlugin**:



The result is the creation of a file in the directory c:\temp:

```
Car Rental Business
   Provide Rental Cars
```

3.5 Debugging a Java component called from HOPEX

Debugging a Java component called from HOPEX uses remote access mode to the Java virtual machine.

This chapter explains how to use this mode of debugging with HOPEX.

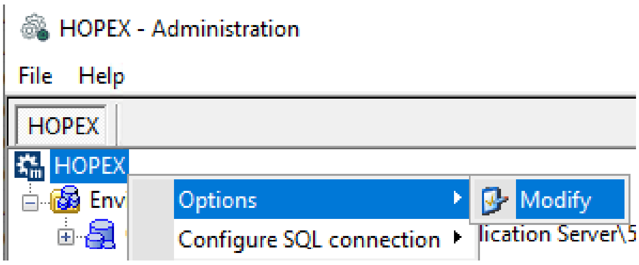
3.5.1 Configuration

HOPEX options

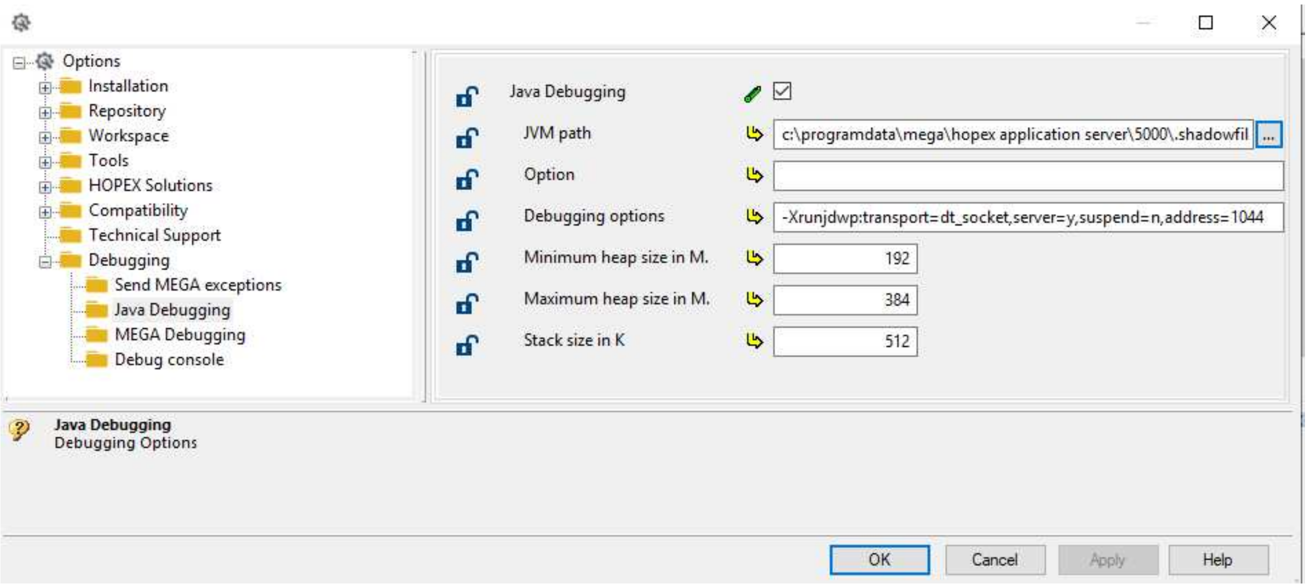
To debug a Java component, you need to activate the **Java Debugging** option.

To activate the Java Debugging option:

1. From HOPEX administration.exe, access the site level options.



2. In the **Options** tree, select **Debugging > Java Debugging** folder.
3. In the right pane select **Java Debugging** option.



Java Debugging ☑

JVM path: c:\programdata\mega\hopex application server\5000\.shadowfil

Option:

Debugging options: -Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=1044

Minimum heap size in M.: 192

Maximum heap size in M.: 384

Stack size in K: 512

Java Debugging
Debugging Options

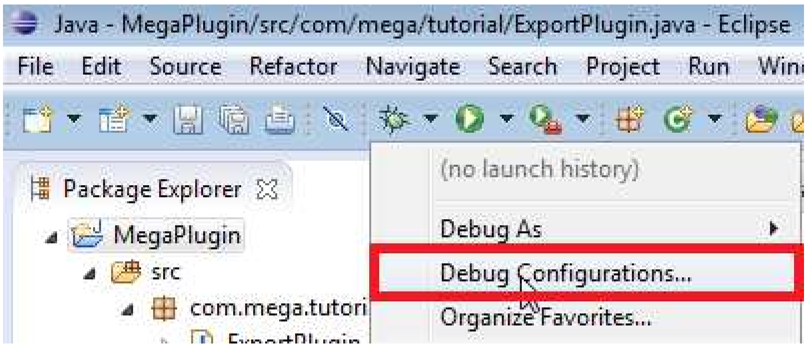
Options are:

- *JVM path*: specify the location of the Java virtual machine to be used.
- *Option*: enables addition of virtual machine execution options.

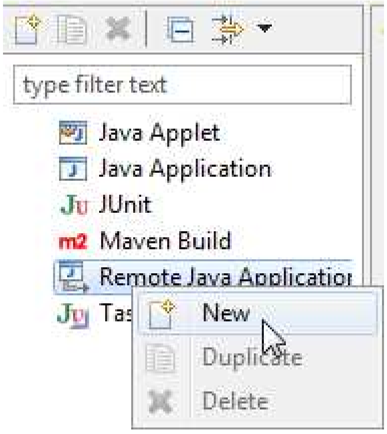
- *Debugging options*: debugging options.
- *Heap and stack size*: enable the specification of available memory sizing.

Debugging configuration in Eclipse

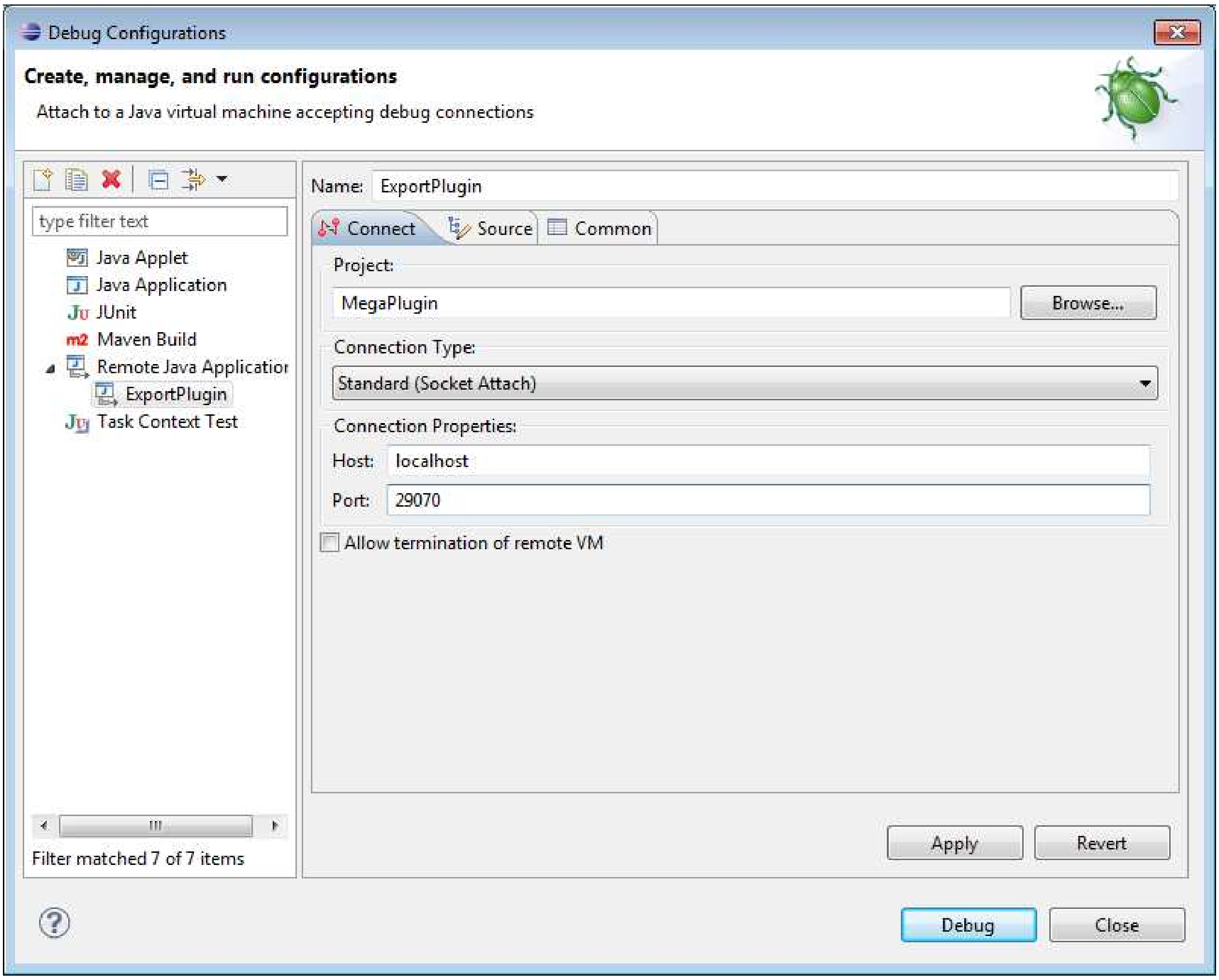
1. From the Eclipse project, open the debugging configuration dialog box (right-click the project and select **Debug As** > **Debug Configurations**):



2. Create a new **Remote Java Application** debug configuration



3. Specify parameters enabling debugging by socket attachment:
   - Connection Type: Standard (Socket Attach).
   - *Connection Properties*: host and port enabling access to the JVM used by HOPEX.



Create, manage, and run configurations

Attach to a Java virtual machine accepting debug connections

Name: ExportPlugin

Project: MegaPlugin

Connection Type: Standard (Socket Attach)

Connection Properties:

Host: localhost

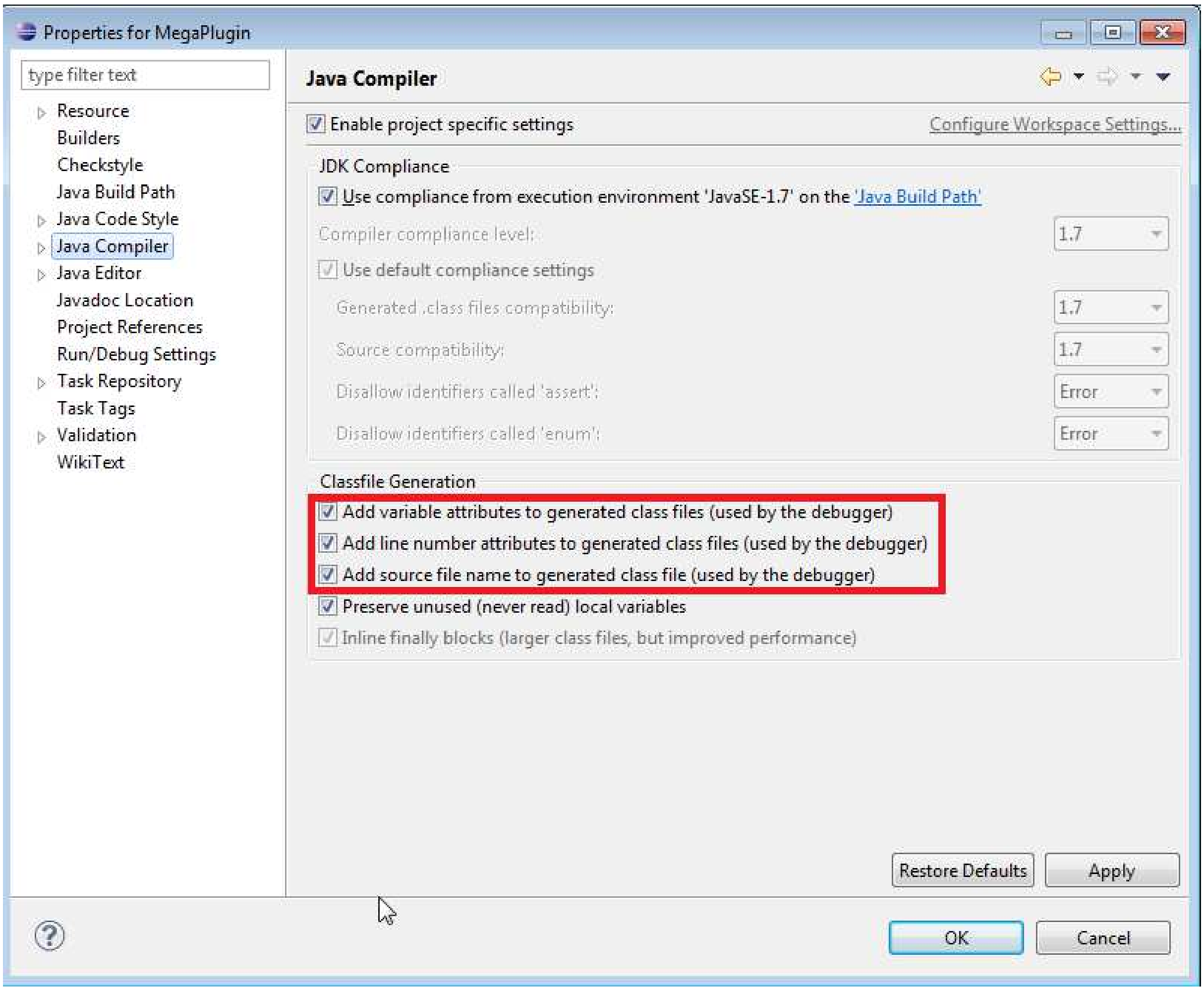
Port: 29070

☐ Allow termination of remote VM

Filter matched 7 of 7 items

Compiling the project

Check that debugging options are activated:



Java Compiler

☑ Enable project specific settings   Configure Workspace Settings...

JDK Compliance

☑ Use compliance from execution environment 'JavaSE-1.7' on the 'Java Build Path'

Compiler compliance level: 1.7

☑ Use default compliance settings

Generated .class files compatibility: 1.7

Source compatibility: 1.7

Disallow identifiers called 'assert': Error

Disallow identifiers called 'enum': Error

Classfile Generation

☑ Add variable attributes to generated class files (used by the debugger)

☑ Add line number attributes to generated class files (used by the debugger)

☑ Add source file name to generated class file (used by the debugger)

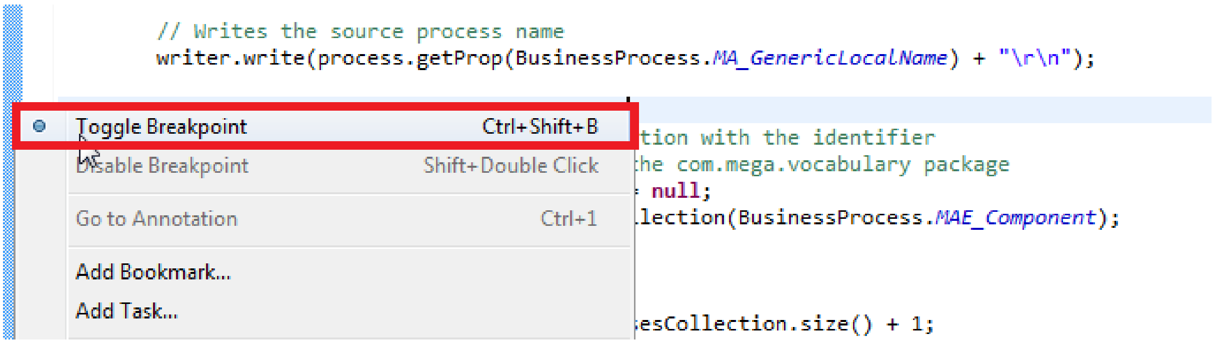
☑ Preserve unused (never read) local variables

☑ Inline finally blocks (larger class files, but improved performance)

3.5.2 Debugging

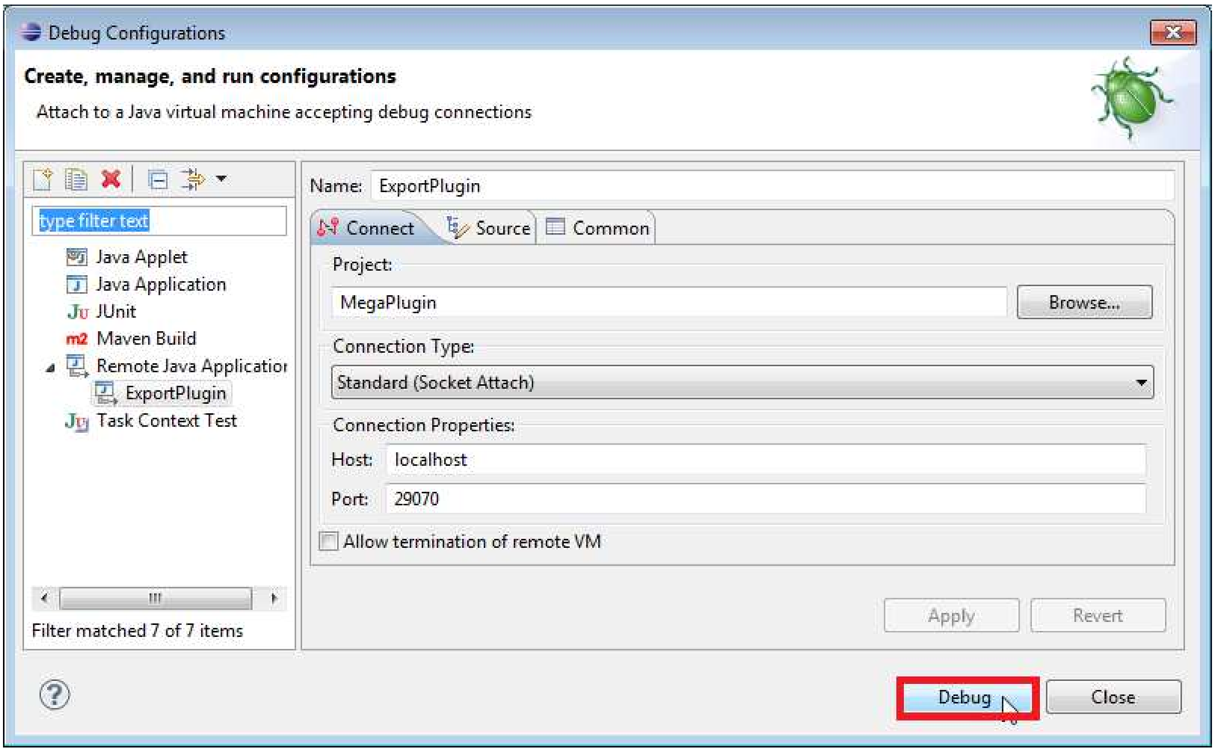
To debug:

1. Add breakpoints where necessary.


Toggle Breakpoint Ctrl+Shift+B
Disable Breakpoint Shift+Double Click
Go to Annotation Ctrl+1
Add Bookmark...
Add Task...

```
        // Writes the source process name
        writer.write(process.getProp(BusinessProcess.MA_GenericLocalName) + "\r\n");

tion with the identifier
the com.mega.vocabulary package
 null;
lection(BusinessProcess.MAE_Component);

esCollection.size() + 1;
```

2. Execute the project in debug mode:



Create, manage, and run configurations

Attach to a Java virtual machine accepting debug connections

Name: ExportPlugin

Project:

MegaPlugin

Connection Type:

Standard (Socket Attach)

Connection Properties:

Host: localhost

Port: 29070

□ Allow termination of remote VM

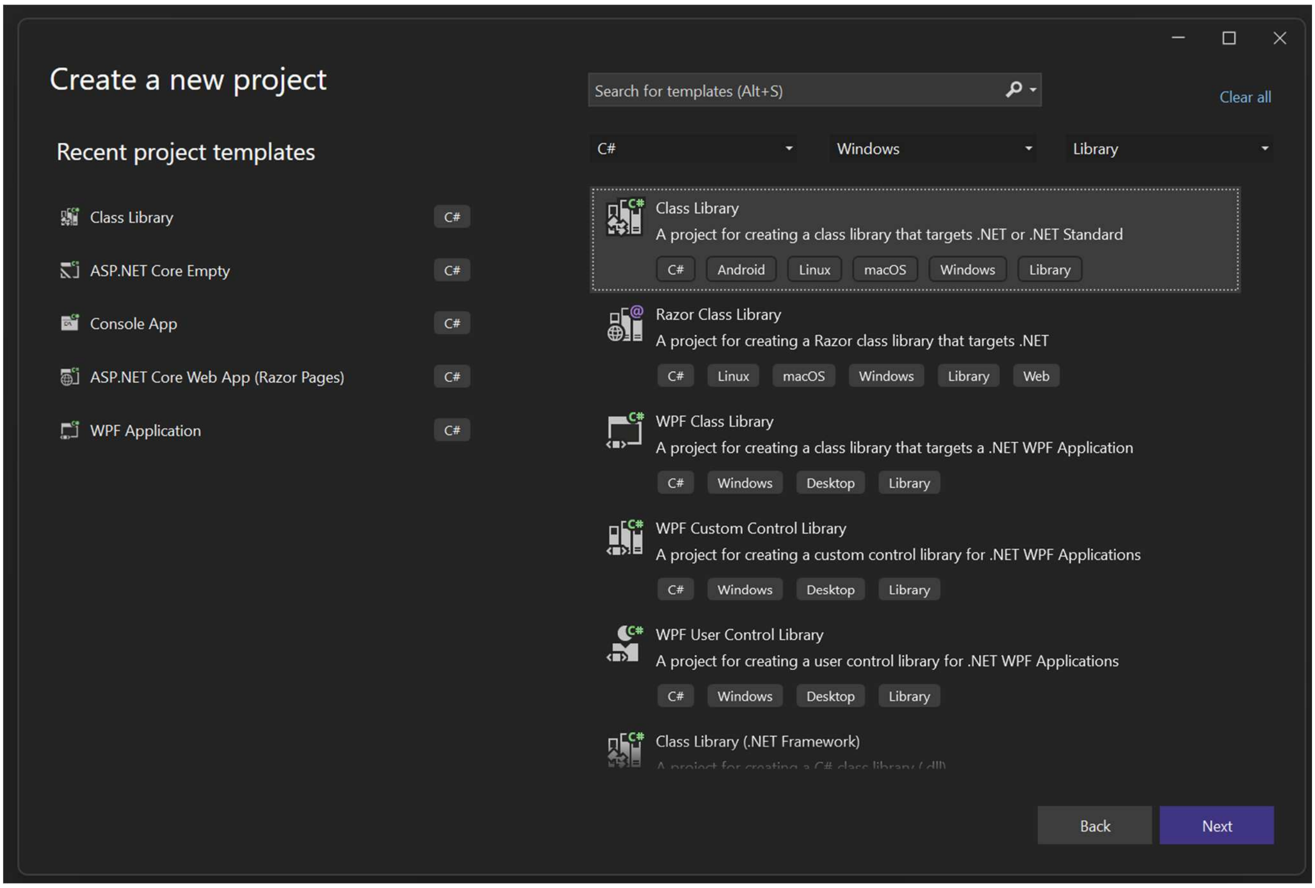
Note: If debugging is not correctly configured, HOPEX can suddenly close. It is necessary to ensure that the correct port is used, and that the Firewall configuration authorizes access to this port (for example, try port 1044 if port 29070 does not permit debugging).

4 Creating a Hopex C# macro

You can create macros used by Hopex using the C# language.

Macros should implement a public class and have public methods that can be called from Hopex.

The C# code must be built as C# DLL (Class Library).



Create a new project

Recent project templates

- Class Library C#
- ASP.NET Core Empty C#
- Console App C#
- ASP.NET Core Web App (Razor Pages) C#
- WPF Application C#

Search for templates (Alt+S)

C# | Windows | Library

Class Library
A project for creating a class library that targets .NET or .NET Standard
C# Android Linux macOS Windows Library

Razor Class Library
A project for creating a Razor class library that targets .NET
C# Linux macOS Windows Library Web

WPF Class Library
A project for creating a class library that targets a .NET WPF Application
C# Windows Desktop Library

WPF Custom Control Library
A project for creating a custom control library for .NET WPF Applications
C# Windows Desktop Library

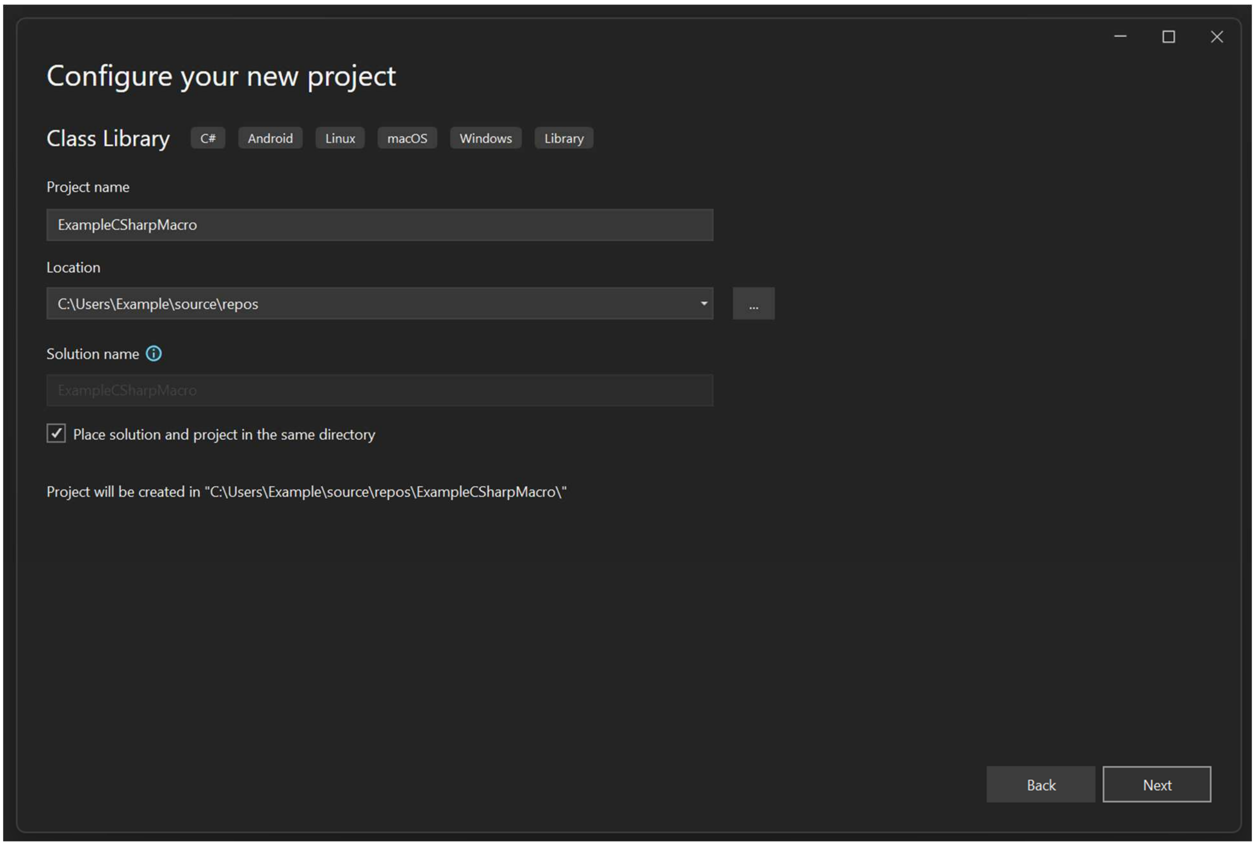
WPF User Control Library
A project for creating a user control library for .NET WPF Applications
C# Windows Desktop Library

Class Library (.NET Framework)
A project for creating a C# class library (.dll)

4.1 Creating a new class library project in Visual Studio

Visual Studio 2022 is strongly recommended, all variants are valid (Community, Professional, Enterprise).

To create a class library project in Visual Studio:



Configure your new project

Class Library

Project name: ExampleCSharpMacro

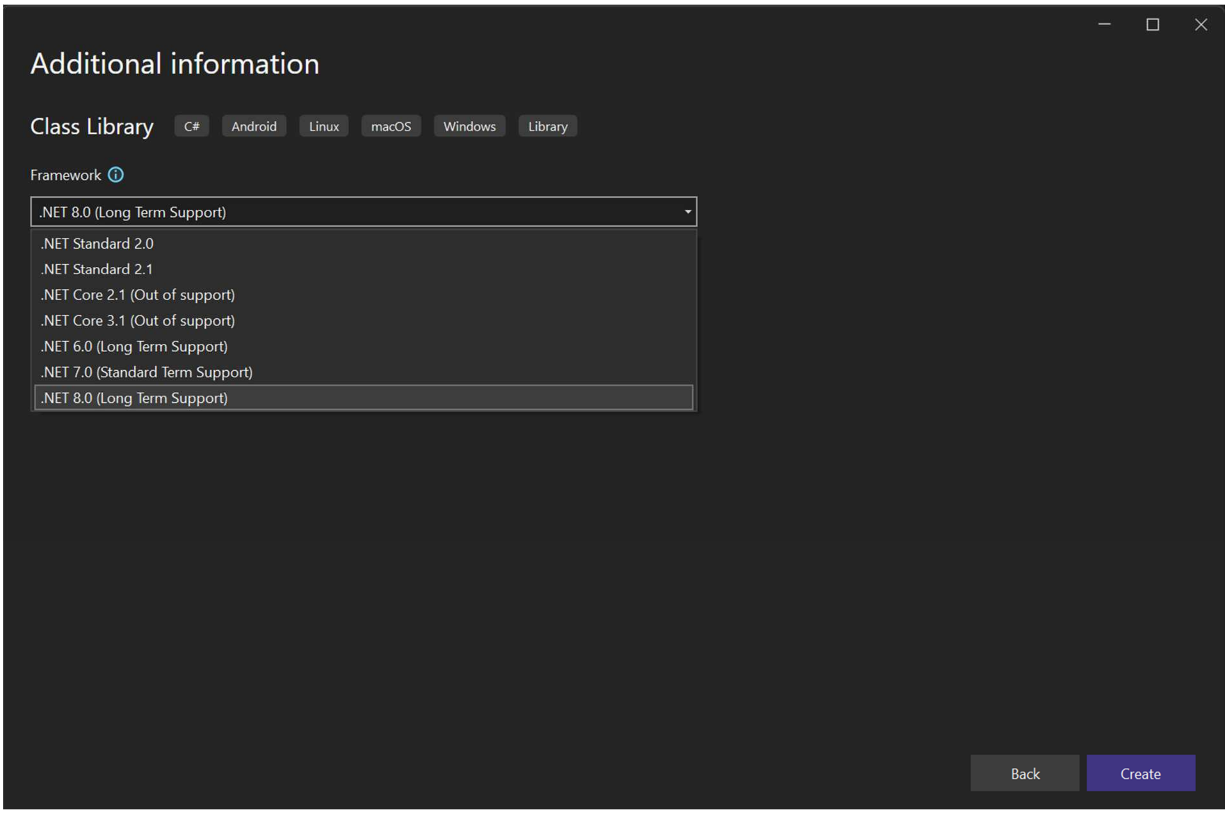
Location: C:\Users\Example\source\repos

Solution name: ExampleCSharpMacro

☑ Place solution and project in the same directory

Project will be created in "C:\Users\Example\source\repos\ExampleCSharpMacro\"

1. Select .NET 8 version (V6 is still supported until end of 2024).



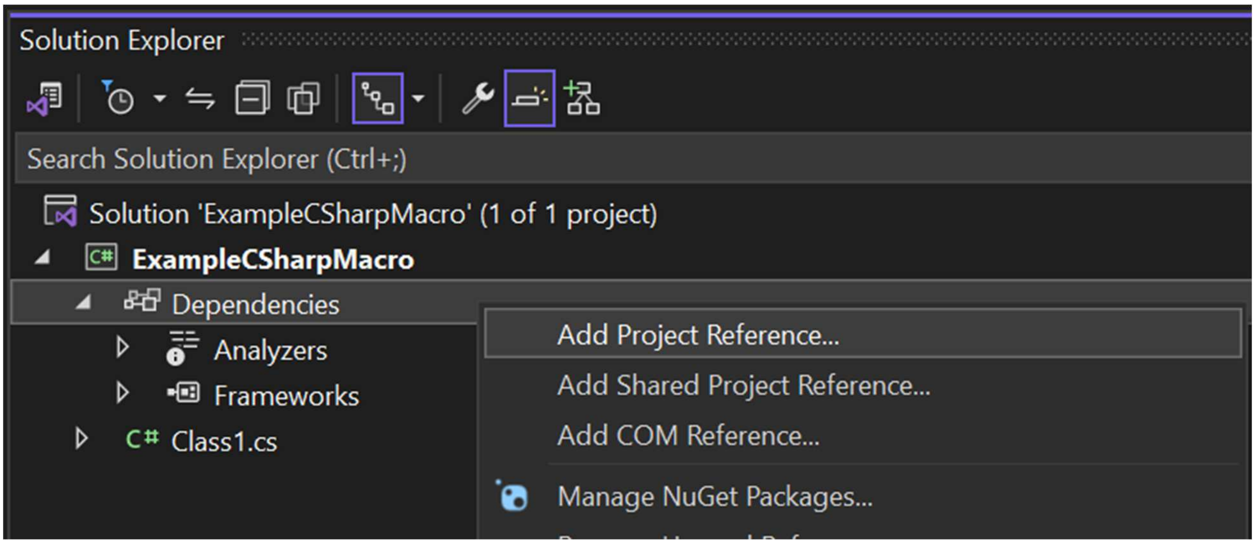
Additional information

Class Library

Framework: .NET 8.0 (Long Term Support)

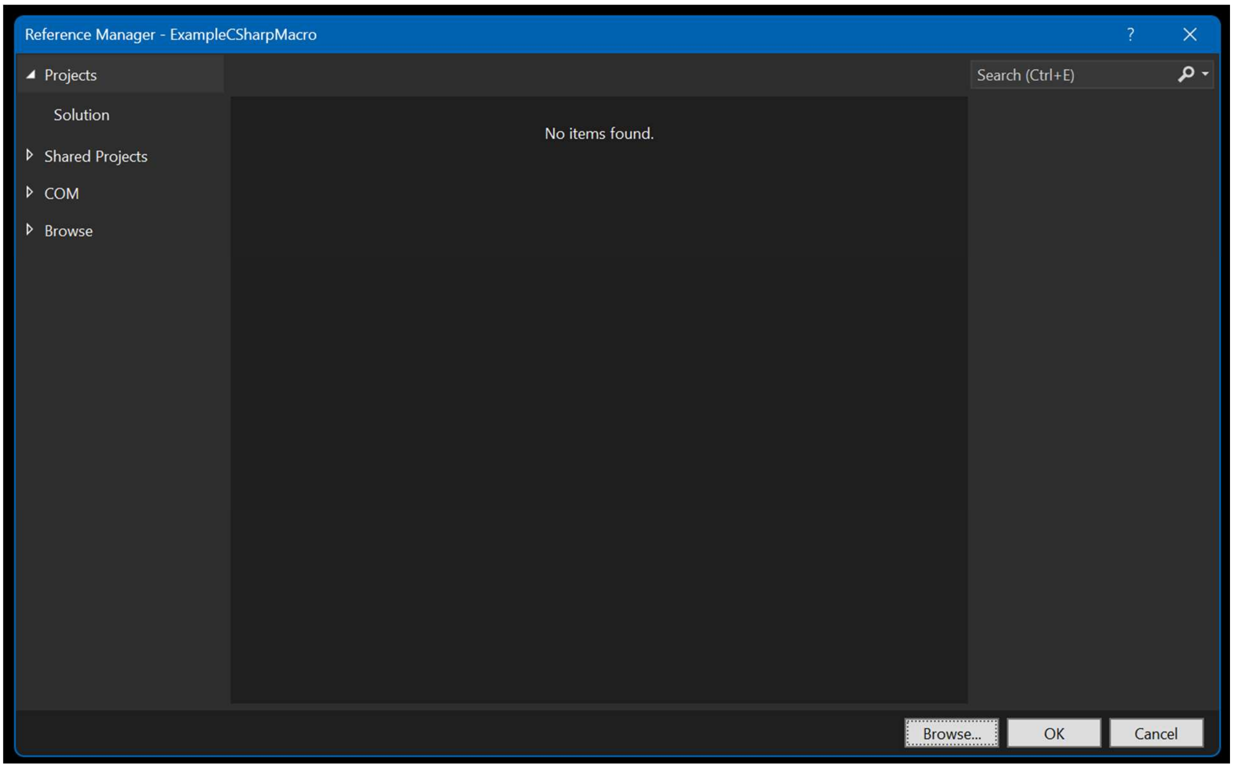
- .NET Standard 2.0
- .NET Standard 2.1
- .NET Core 2.1 (Out of support)
- .NET Core 3.1 (Out of support)
- .NET 6.0 (Long Term Support)
- .NET 7.0 (Standard Term Support)
- .NET 8.0 (Long Term Support)

2. Complete the initialization of the project and go in the **Solution Explorer**.
3. Expand the solution.



4. Right-click **Dependencies** and select **Add Project Reference**.

   A window appears.



5. Click **Browse**.
6. Find the **Mega.Macro.Wrapper.dll**, in the **System** folder, which is inside the Hopex installation directory:

   ```
   C:\ProgramData\MEGA\Hopex Application Server\<port
   number>\.shadowFiles\hopex.core\<version number>\System\
   ```

   Now you can use all the Hopex classes inside your C# macro.

4.2 Building the DLL with the Hopex macro

From Visual Studio, select **Build > Build Solution** menu (short cut Ctrl+Shift+B).

The DLL will be built in your defined output directory.

Note that, if your target output path is the same path where the DLLs should be placed, and Hopex is running, the build will fail because the DLL cannot be replaced while Hopex is active.

If you want to test the DLL in debug, build it in debug mode. When you are done with the debugging, remember to build it in release mode to improve its performance.

4.3 Placing the DLL inside Hopex

The recommended folder where to place the DLLs depends if you want to deploy it as module, inside a customization package, or not.

- Recommended path:

  `<Hopex installation directory>/.shadowfiles/Macros/<Your module name>/<your version number>/hopex.core/dotnet/assemblies/`

- Possible path (useful for the development cycle):

  `<Hopex installation directory>/.shadowfiles/hopex.core/<your version number>/dotnet/assemblies_usr/`

4.4 Try to avoid adding dependencies to the DLL

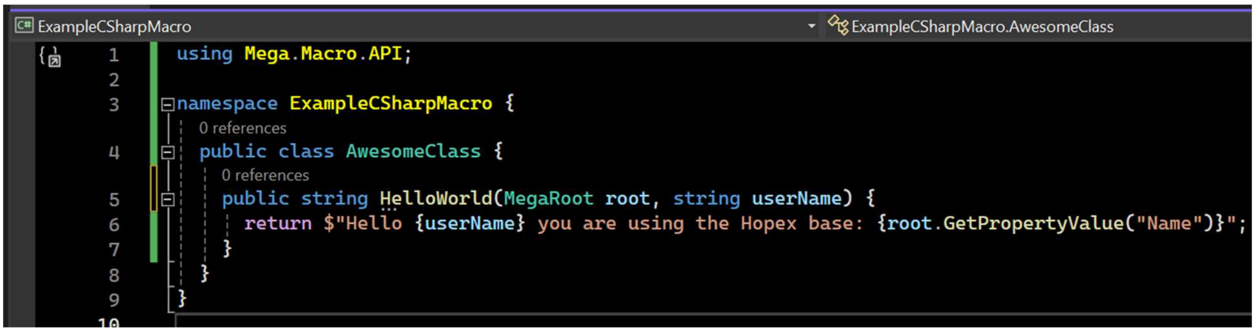
The DLL will not require any standard C# DLL to work, they will be provided by Hopex. Try also to avoid using old DLLs like Newtonsoft.Json because they are fully replaced by better functions native of .NET 6 and 8.

If you really need to add external libraries, be sure they are available as NuGet packages.

4.5 Structure of the code

You can place as many classes as you want inside the DLLs. Only public methods inside public classes will be directly accessible from Hopex.

A typical class will look like:



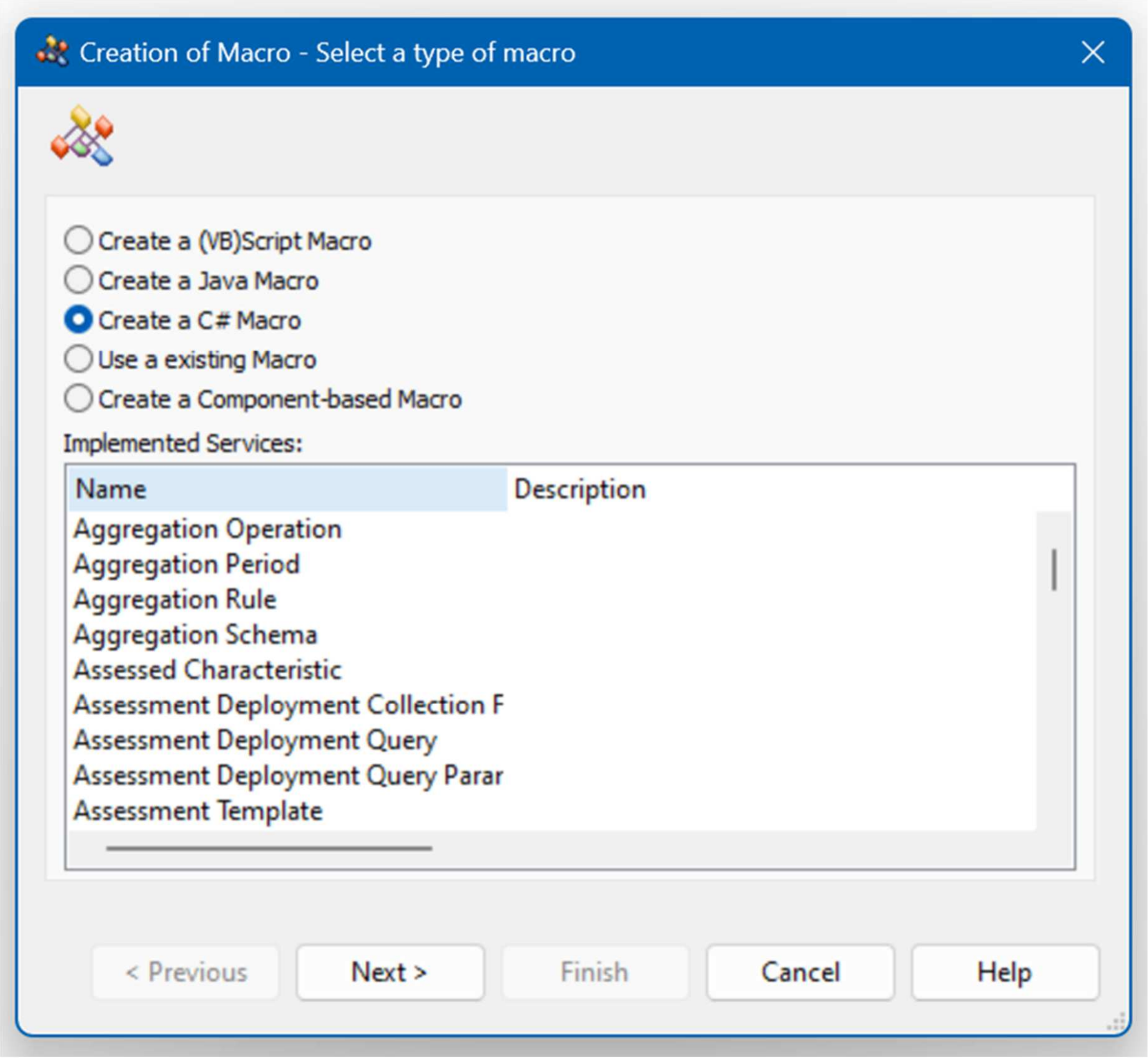
```
using Mega.Macro.API;

namespace ExampleCSharpMacro {
  public class AwesomeClass {
    public string HelloWorld(MegaRoot root, string userName) {
      return $"Hello {userName} you are using the Hopex base: {root.GetPropertyValue("Name")}";
    }
  }
}
```

You can use all the standard Hopex API’s classes like `MegaRoot`, `MegaObject`, `MegaCollection`.

4.6 Defining the Macro in Hopex

Once your DLL is built and deployed, you must define it in Hopex. This must be performed in HOPEX Windows Front-End.

1. In HOPEX, create an item of type macro and follow the wizard.

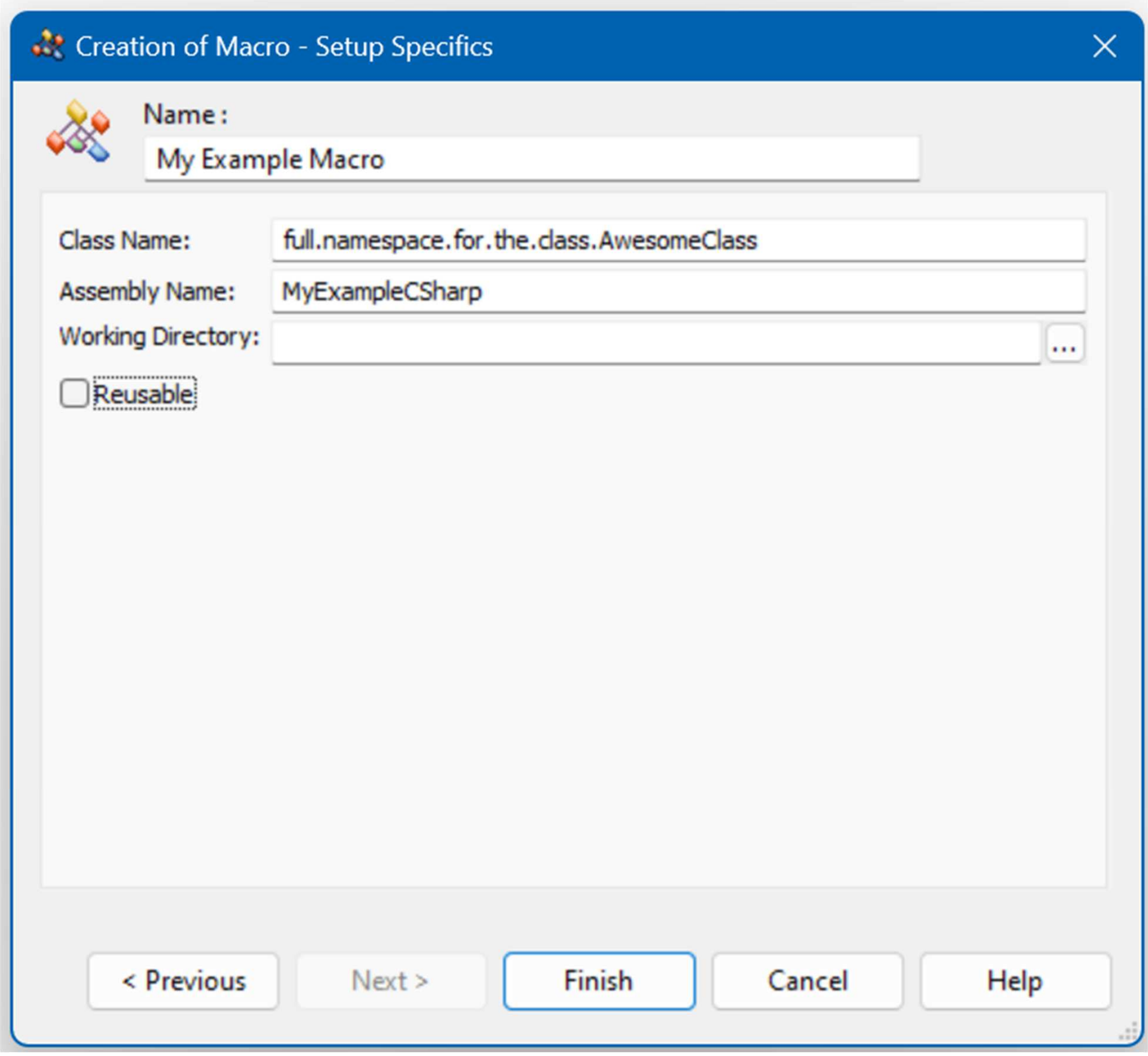


Creation of Macro - Select a type of macro

○ Create a (VB)Script Macro
○ Create a Java Macro
● Create a C# Macro
○ Use a existing Macro
○ Create a Component-based Macro

Implemented Services:

Name	Description
Aggregation Operation	
Aggregation Period	
Aggregation Rule	
Aggregation Schema	
Assessed Characteristic	
Assessment Deployment Collection F	
Assessment Deployment Query	
Assessment Deployment Query Parar	
Assessment Template	



Creation of Macro - Setup Specifics

Name : My Example Macro

Class Name: full.namespace.for.the.class.AwesomeClass

Assembly Name: MyExampleCSharp

Working Directory:

□ Reusable

Make sure to use the full namespace of your class and to write the exact name of the DLL you will build and deploy.

The DLL is expected to be called **MyExampleCSharp.dll** (and placed in one of the previously mentioned folders).

In case the namespace (that is case sensitive as the class name) is wrong or the name of the DLL (not case sensitive) is wrong, the macro will not be loaded and executed.

4.7 Testing your Macro

To test your macro:

1. In Hopex, open the script editor.
2. Enter the following code:

   Note: the second line is an example, your method will have a different name.

   ```
   set m = currentEnvironment.getMacro(‘‘My Example Macro’’)
   print m.HelloWorld(getRoot, ‘‘John Doe’’)
   ```

Make sure you drag and drop the Macro you created before inside the `getMacro` statement to place its ID. You can also open the properties of the macro and copy its ID and then place it as parameter of `getMacro`.

4.8 Debugging your macros with Visual Studio

To debug your macro:

1. In Visual Studio, select **Debug > Attach to process** menu. (Ctrl+Alt+P)
2. Select the processes you want to debug.

   Do not change the connection mode (keep it as automatic) and select all processes with "Hopex" in their names (yes there can be multiple of them).
3. Place a breakpoint in your code and run the macro.

   The breakpoint should be hit, and you can run step by step and inspect variables.

In case you see a warning when setting the breakpoint, be aware that C# DLLs are not loaded at startup but only when needed. The solution is to execute your macro first and then attach Visual Studio to the process and place the breakpoint.

In case your breakpoint says that symbols are approximate be sure you copy not only the DLL of the macro but also the PDB file that is generated in debug mode. This will allow you to debug the code with single instruction precision.

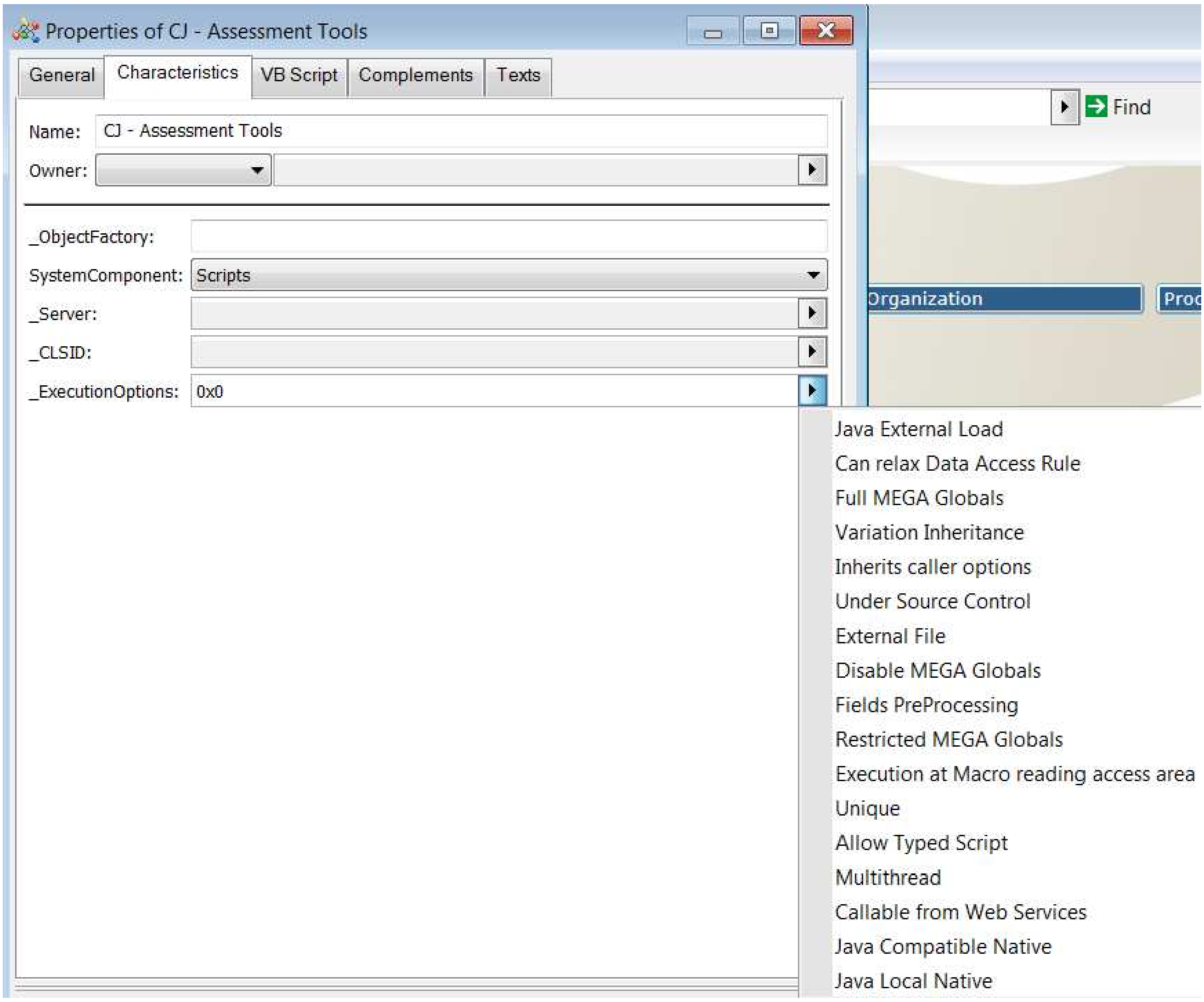
5 MACRO PARAMETERIZATION

To access macros stored in the repository:

1. Display the **Utilities** navigation window.
2. Expand the **Macros** folder.

The **Characteristics** tab of the macro Properties enables definition of macro execution specificities.

The end user can parameterize the macro to define specific behaviors. The parameterization is made on the **_ExecutionOptions** macro attribute.


Properties of CJ - Assessment Tools

Name: CJ - Assessment Tools

Owner:

_ObjectFactory:

SystemComponent: Scripts

_Server:

_CLSID:

_ExecutionOptions: 0x0

This parameterization is important since it may improve performances of the full application.

The following options are available:

Full MEGA Globals: This parameter activates the global variables such as MegaRoot, MegaDB, MegaToolkit, MegaEnv, MegaSite. The access to the MegaRoot is implicit. This parameter is deprecated. It is highly recommended not to use it. Setting this attribute to this property means that your macro is not compatible with HOPEX Web applications.

Variation inheritance: This parameter is set if the GetCollection and GetSelection contained in the macro must retrieve both the direct objects retrieved from a MetaAssociationEnd and the objects obtained from inheritance with the variation mechanism.

Inherits Caller Options: This applies if the current macro has been called by another macro. In that case, the current macro inherits from the variation rights defined on the calling macro. As a sample, if the calling macro has the Variation inheritance value set, then the called macro has the Variation inheritance value set.

External File: The macro is stored in an external VBS file.

Disable MEGA Globals: This option defines that the use of MEGA Globals is forbidden. This is the recommended behavior. The use of Globals does not work with the Web applications.

Fields PreProcessing: This option enables to use directly the fields (MetaAttribute, MetaAssociationEnd) on an object without using GetProp and GetCollection. This option is not really recommended since it is time consumed and decreases the performances. This option is now deprecated.

Restricted MEGA Globals: This option enables the use of two MEGA globals: MegaEnv and MegaToolkit. This parameter is deprecated. It is highly recommended not to use it. The use of Globals does not work with the web applications.

Execution at Macro reading access area: This option defines at which confidentiality level the macro must be executed. The macro contains code and algorithm. The MEGA Philosophy concerning confidentiality is to compute algorithm as if you can access to all the objects in order to have the good result and to hide the confidential objects only for display. Therefore, if the macro is dedicated to display data, this option does not need to be used but as soon as the macro contains algorithm, it is highly recommended to ask for the following question : "do I need to access to all the data to have a coherent result ?". If the answer is "yes", this option must be activated.

For more information, see Confidentiality section.

Unique: This parameter is used for performance purpose. If your script has no global, you may require to instantiate it only once. In that case, the script is loaded only once and executed as much as requested.

Allow Typed Script: This option gives you the availability to type the objects inside your script. For example, in standard, if you declare a function you can write:

```
Function MyFunction( oMegaObject)

Dim myObject

End Function
```

If this option is set, you can type the objects inside your code as the following:

```
Function MyFunction( oMegaObject As MegaObject )

Dim myObject As MegaObject

End Function
```

The interesting point concerning that is that you can use the intellisense on object.

Multithread: This option is available only for Java Macro. If this option is set, it means that the java component works in multhreading that is to say the component may be called at the same time by several threads.

Java Compatible Native:

Java Local Native:

Java External load (until 740): This parameter is for Java debugging purpose. This parameter can be set only if the Macro is a Java plug-in. It launches a new MEGA process which enables to take into account the new compiled version of the Java plug-in without restarting MEGA (Once a Java plug-in is loaded, replacing the jar file has no effect, you must stop and restart MEGA to get the effect of the new java plug-in version).

Under Source Control (until 740): This option is dedicated to MEGA R&D Team and cannot be used by the end user.

6 API Use Principles

6.1 Coding: the right way

6.1.1 MegaRoot

You can use the **Script Editor** to write:

- scripts for testing

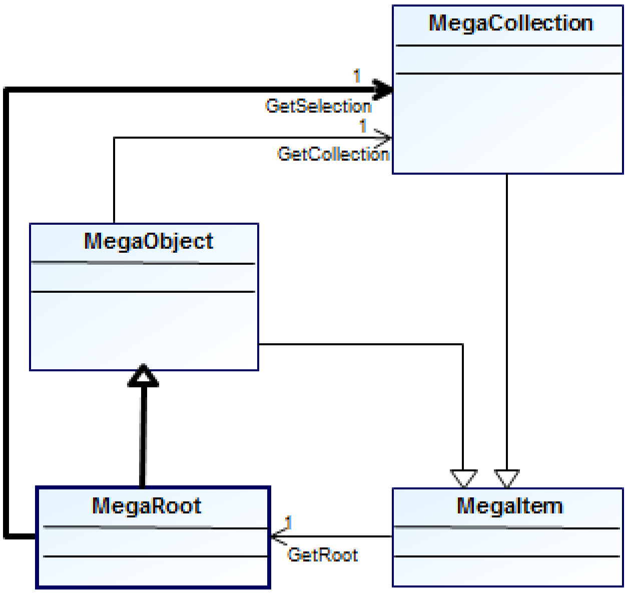
  As these scripts are not stored in HOPEX you can use global variables like MegaDB, MegaRoot or GetRoot to access objects in the repository. This coding way is dedicated to testing use only.

- scripts stored as files or macros in HOPEX.

Do not use MegaDB, MegaRoot, and Get Root in macro scripts stored in HOPEX.

MegaRoot:

- represents the session you are connected with.
- gives access to HOPEX repository data through **GetGollection** or **GetSelection**



In the following sections, **oRoot** is:

- a session with which you are connected
- either set as a parameter of the function (e.g.: `Main(oRoot)`) or you call a function that has as a parameter the MegaObjects.

Here is sample code to access the **MegaRoot** from an object or a MegaEnvironment:

```
Set oRoot = object.GetRoot


Function Sub Main(oRoot)
`code
End Sub
```

6.1.2 MegaFields

You must make reference to an object using its absolute identifier rather than its name: in this case the code is most highly optimized and resists renaming of the instance as well as change of language.

⚠ **To write a query in a script you want to store in HOPEX, use of MegaFields is mandatory.**

An object in **MegaField** format is as follow:

```
"~xxxxxxxxxxxx[Object Name]"
```

Example for Project object:

```
 ~qekPESs3iG30[Project]"
```

Where:

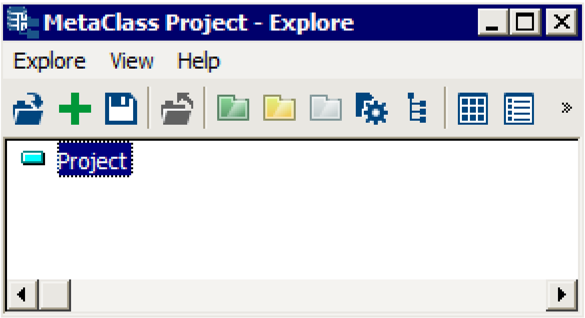
- ~ indicates the start of the MegaField
- `qekPESs3iG30` is the absolute Identifier of the object
- `[Project]` includes the object name

For more information on MegaFields see:

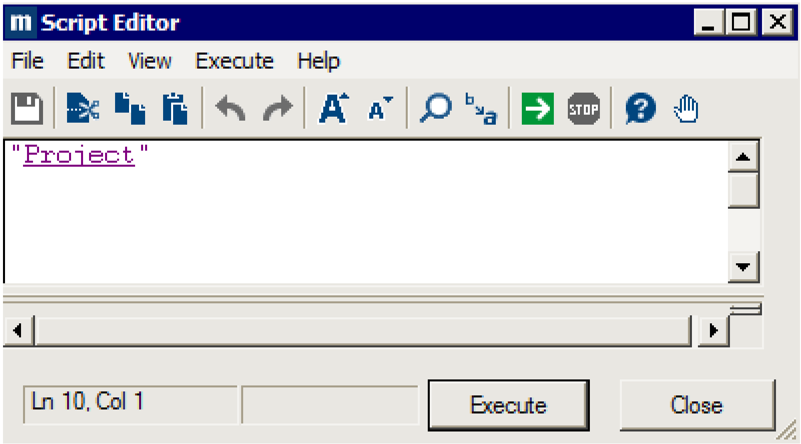
- MegaFields
- Handling Identifiers
- Navigating through data with APIs

To enter Project MetaClass in MegaField format (i.e. with its absolute identifier) in your code:

1. From **Explore** menu bar, select **Explore > object**.
2. In the object list, select **MetaClass**.
3. In the right-pane pane select **Project**.
4. Click **OK**.



5. Right-click **Project** and select **Copy** and **Paste** it in your code.



```
"Project"
```

In **Script Editor**, "Project" includes Object name (Project) and its absolute identifier (qekPESs3iG30):

```
"~qekPESs3iG30[Project]"
```

6.2 Basic Operations

Objects are entered in their MegaField formats, which include their absolute identifiers (e.g.: "~qekPESs3iG30[Project]" instead of "Project", see MegaFields section).

6.2.1 Creating an object

The following script enables creation of a project by applying the **Create** method to the **Project** object.

The name of the project created is indicated between brackets.

Example: Creation of a new project called "P1".

VB Script

```
Set oRoot = object.GetRoot
oRoot.getCollection("~qekPESs3iG30[Project]").Create ("P1")
```

Java

```
mgRoot.getCollection("~qekPESs3iG30[Project]").create("P1");
```

6.2.2 Connecting operations to a project

Example: Creation of new operations that are connected to project "P1".

VB Script

```
Set oRoot = object.GetRoot
Set oProject = oRoot.getCollection("~qekPESs3iG30[Project]").Item("P1")
Set oOperations = oProject.GetCollection("~OsUiS9B5iiQ0[Operation]")
For i = 1 To 10
oOperations.Create "ope-" & i
Next
For Each oOperation in oOperations
oRoot.Print oOperation.getProp("~Z20000000D60[Short Name]")
Next
```

Java

```
MegaObject mgobjProject =
mgRoot.getCollection("~qekPESs3iG30[Project]").get("P1");
MegaCollection mgcolOperations =
mgobjProject.getCollection("~OsUiS9B5iiQ0[Operation]");
for (int j = 1; j <= 10; j++) {
mgcolOperations.create("ope-" + i);
}
for (MegaObject mgobjOperation : mgcolOperations) {
mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]",
mgobjOperation.getProp("~Z20000000D60[Short Name]"));
}
```

Code description:

```
Set oRoot = object.GetRoot
Set oProject = oRoot.getCollection("~qekPESs3iG30[Project]").Item("P1")
```

Retrieves the project called "P1" and its assignment to the variable "oProject".

Use the long name with "item" when manipulating namespaced objects. Example for an entity (DM) "titi" under the datamodel "DM1":

```
oEnt =oRoot.GetCollection("idabs[Entity (DM)]").item("DM1::titi")
```

```
Set oOperations = oProject.GetCollection("~OsUiS9B5iiQ0[Operation]")
```

Retrieves all operations connected to the project "P1".

```
For i = 1 To 10
oOperations.Create "ope-" & i
Next
```

Creates ten operations (called "ope-1"..."ope-10") while adding them to the collection of operations connected to project "P1".

```
For Each oOperation in oOperations
oRoot.Print oOperation.getProp("~210000000900[Name]")
Next
```

Displays the names of the operations in the collection "oOperations".

6.2.3 Modifying a project name

Example: changing the name of project "P1" to "My Project".

VB Script

```
Set oRoot = object.GetRoot
Set oProject = oRoot.getCollection("~qekPESs3iG30[Project]").Item("P1")
oProject.GetProp("~210000000900[Name]") = "My Project"
```

Java

```
MegaObject mgobjProject =
mgRoot.getCollection("~qekPESs3iG30[Project]").get("P1");
                         mgobjProject.setProp("~210000000900[Name]", "My Project");
```

Code description:

```
Set oRoot = object.GetRoot
Set oProject = oRoot.getCollection("~qekPESs3iG30[Project]").Item("P1")
```

Retrieves the project "P1" and assigns it to the variable "oProject".

```
oProject.GetProp("~210000000900[Name]") = "My Project"
```

Assigns the "My Project" value to the "Name" property of project "P1".

Note that `Setprop` is equivalent to `getProp`.

6.2.4 Displaying the names of all repository projects in a window

Example:

VB Script

```
Set oRoot = object.GetRoot
Set oAllProjects = oRoot.getCollection("~qekPESs3iG30[Project]")
For Each oProject in oAllProjects
sText = sText & oProject.getProp(“~Z20000000D60[Short Name]”) & vbCrLf
Next
MsgBox sText
```

Java

```
MegaCollection mgcolAllProjects =
mgRoot.getCollection("~qekPESs3iG30[Project]");
              String strText = "";
              for (MegaObject mgobjProject : mgcolAllProjects) {
                strText = strText + mgobjProject.getProp("~Z20000000D60[Short
Name]") + "\n";
              }
              mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]", strText);
```

Code description:

```
Set oRoot = object.GetRoot
Set oAllProjects = oRoot.getCollection("~qekPESs3iG30[Project]")
```

Retrieves all repository projects and assigns them to the variable oAllProjects.

```
For Each oProject in oAllProjects
sText = sText & oProject.getProp(“~210000000900[Name]”) & vbCrLf
Next
```

Obtains the names of each project inserting a line return between each ("vbCrLf"). After having browsed the collection of repository projects, the variable **sText** contains the names of all projects, each on a new line.

```
MsgBox sText
```

Displays in a window the text contained in sText variable.

6.2.5 Assigning an attribute value to several objects

Example:

VB Script

```
Set oRoot = object.GetRoot

Set oOperations = oRoot.getCollection("~qekPESs3iG30[Project]").Item("My
Project"). getCollection("~OsUiS9B5iiQ0[Operation]")

For Each oOperation in oOperations

oOperation.Importance = "P"

Next
```

Java

```
MegaCollection mgcolOperations =
mgRoot.getCollection("~qekPESs3iG30[Project]").get("My
Project").getCollection("~OsUiS9B5iiQ0[Operation]");

              for (MegaObject mgobjOperation : mgcolOperations) {

                mgobjOperation.setProp("Importance", "P");

              }
```

Code description:

```
Set oRoot = object.GetRoot

Set oOperations = oRoot.getCollection("~qekPESs3iG30[Project]").Item("My Project").
getCollection("~OsUiS9B5iiQ0[Operation]")
```

retrieves the operations connected to the "My Project" project and assigns this collection to the "oOperations" variable.

```
For Each oOperation in oOperations

oOperation.Importance = "P"

Next
```

assigns the internal value "P" (for "Principal") to the importance of each operation of the "oOperations" collection.

6.2.6 Retrieving a query result

Example:

VB Script

```
Set oRoot = object.GetRoot

set myprojects = oRoot.GetSelection~YYxh7XjGIzKE[~YYxh7XjGIzKE[My
projects]]("Select ~qekPESs3iG30[Project] where Not ~KsUiCCB5i0)1[Diagram] ")

 For each oproject in myprojects

 oRoot.print oproject.GetProp("~Z20000000D60[Short Name]")

 Next
```

Java

```
MegaCollection mgcolAllProjects = mgRoot.getSelection("Select
~qekPESs3iG30[Project] where Not ~KsUiCCB5i0)1[Diagram]");

              String strText = "";

              for (MegaObject mgobjProject : mgcolAllProjects) {

                strText = strText + mgobjProject.getProp("~Z20000000D60[Short
Name]") + "\n";
```

```
                            }
                            mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]", strText);
```

Code description:

```
Set oRoot = object.GetRoot

set myprojects = oRoot.GetSelection("Select ~qekPESs3iG30[Project] where Not
~KsUiCCB5i0)1[Diagram] ")
```

assigns the query result to the "myProjects" variable (this query retrieves all projects not described by a diagram).

```
For each oproject in myprojects
 oRoot.print oproject.GetProp("~210000000900[Name]")
 Next
```

displays the names of all projects in "myProjects" collection.

6.2.7 Using a MegaObject or MegaCollection

The MEGA APIs provide the user with two basic types:

- the MegaObject type, which enables access to a MegaObject.
- the MegaCollection type, which enables access to a collection of objects.

An instance is an object, such as a business process, class, database, etc.

A MegaObject enables access to characteristics of an object instance.

Use the **GetProp** function, using the name as parameter.

VB Script

```
MyName = MyObject.Name

MyName = MyObject.GetProp("~210000000900[Name]") ' equivalent to the previous
line
```

Java

```
String strMyName = mgobjMyObject.getProp("~210000000900[Name]");
```

From a MegaObject, you can access each MegaCollection connected to an object instance, using the **GetCollection** function. If "myProject" is an object corresponding to a project instance, to access the collection of operations linked to this project, the code is as follows:

VB Script

```
Set myOperations = myProject.GetCollection("~OsUiS9B5iiQ0[Operation]")
For each anOperation in myOperations
do .... Next
```

Java

```
MegaCollection mgcolOperations =
mgobjMyProject.getCollection("~OsUiS9B5iiQ0[Operation]");
              for (MegaObject mgobjOperation : mgcolOperations) {
               ....
              }
```

The VBScript keyword **For Each** enables browsing of the content of a MegaCollection. However, other forms of search are allowed using the **Item** implicit method for the collection.

If a numeric parameter is passed to **Item**, this is treated as an ordinal position. As the standard VB **Count** function returns the number of items in the collection, you can write:

VB Script

```
numOperations = myOperations.Count
For i = 1 to numOperations
Set anOperation = myOperations.Item(i)
Set anOperation = myOperations(i) ' same thing
Next
```

Java

```
int iNumOperations = mgcolOperations.size();
              for(int j = 1; j<= iNumOperations; j++)
              {
                MegaObject mgobjOperation = mgcolOperations.get(j);
              }
```

6.2.8 Filtering and refining the getCollection API to retrieve objects

The getCollection API enables to retrieve a collection of objects.

You can add parameters to the getCollection API:

`@SKIPCONFIDENTIAL:` confidential objects are filtered and are not included in the collection.

`@TAKECONFIDENTIAL:` deactivates the @SKIPCONFIDENTIAL filter. This command may be necessary if it was defined as default behavior with MegaRoot.SetDefault.

`@IGNOREMETAFILTER:` non-visible metamodel objects (filtered by keys or profile) are not included in the collection.

`@-ACTIVATEMETAFILTER:` deactivates the @ACTIVATEMETAFILTER filter.

`@ACTIVATECONCRETEFILTER:` repository objects with non-visible concrete MetaClass (filtered by keys or profile) are not included in the collection.

`@-ACTIVATECONCRETEFILTER:` deactivates the @ACTIVATECONCRETEFILTER filter.

Refining parameters are:

`@INHERITED:` adds objects inherited for the MetaAssociationEnd in the collection. This parameter also applies to GetSelection.

`@-INHERITED:` deactivates the @INHERITED parameter.

`@INHERITING`: adds inheriting objects for the MetaAssociationEnd in the collection.

`@-INHERITING:` deactivates the @INHERITING parameter.

`@INHERITANCE:` adds inherited or inheriting objects for the MetaAssociationEnd according to its type. This parameter also applies for GetSelection, but only for inherited objects.

`@-INHERITANCE:` deactivates the @INHERITANCE parameter.

`@SUBSTITUTED:` when used alone, includes in the collection only those objects substituted for the MetaAssociationEnd. This parameter is inactive for GetSelection.

`@-SUBSTITUTED:` deactivates the @SUBSTITUTED parameter.

Example:

To get objects inherited from an object use the following API:

```
oTAb.getCollection("source Object MegaField]", "@INHERITED")
```

For example, to get the objects inherited from a column:

```
oTAb.getCollection("~qdFzhq1Bkur1[Column]", "@INHERITED")
```

Note that the corresponding Erql code is:

```
Select [Column] Inherited Where [Table].[Name] Like "#MyTable#"
```

6.2.9 Using Set in a VB Script code

Assignment of MegaObjects by command **Set** results in return of an object.

```
numOperations = myOperations.Count
Set anOper = myOperations.Item(1) 'Returns an object
Print myOperation.Item(1) 'Returns a value
```

Use of the **Set** command leads to reservation of memory space on your workstation. The use of highly recursive functions or tables of high volume can lead to saturation of this memory.

To avoid this type of problem, take care to free memory spaces reserved by unused objects.

This freeing can be carried out by assigning the value **Nothing** to your objects (for more details, see Microsoft documentation on VB Script language).

Example:

```
Set myproject= ...
...
Set myProject = Nothing
```

Use of the "Explicit Option" option is recommended, avoiding use of global variables by explicitly declaring variables in functions (global variables do not operate in HOPEX Web Front-End context).

6.2.10 Accessing MEGA API Public Objects

Public objects of HOPEX are available for external VB Script code, Java code or VBA.

Public objects

Use of MEGA APIs form an external program is based on the three following public components:

MegaApplication

This component enables administration of HOPEX from an external program. It is not accessible from an open session. In particular, it cannot be used to its full extent from the script editor. This is due to the fact that this component totally controls the HOPEX session (connection, repository opening) and is not designed to collaborate with the HOPEX workspace (Windows Front-End). When this component is activated, it is not possible to open an HOPEX workspace (Windows Front-End) or HOPEX Administration.exe.

In VB script, it can be created by the Basic function **CreateObject ("Mega.Application")**.

MegaCurrentEnv

This component enables access to the current HOPEX session. Unlike the **MegaApplication** component, it is used to collaborate with the HOPEX workspace. It enables determination of whether the HOPEX session is open (using the **IsSessionOpen** function) and therefore enables an external program to use HOPEX without explicitly connecting.

However, when the **GetRoot** function of this component is called when the HOPEX workspace is not open, it takes charge of the HOPEX session by proposing a specific connection dialog box. In this case, it is no longer possible to open the HOPEX workspace, since the connection this proposes could be in conflict with that opened from this component.

This component can be used from the script editor and gives access to the current session.

⚠ **This is to be avoided**

In VB script, it can be created by the Basic function: **CreateObject("MegaRepository.CurrentEnv")**.

To obtain the **MegaApplication** public object from a **MegaCurrentEnv** object use the **MegaCurrentEnv.Site** function.

MegaToolkit

This component groups utility functions developed for use indistinctly in internal or external mode. In this way, it does not make reference to characteristics of the session in progress.

In VB script, it can be created by the **CreateObject ("Mega.Toolkit")** Basic function.

6.2.11 Accessing public objects from another MegaObject

This is recommended

To obtain the **MegaCurrentEnv** public object from a **MegaRoot** object use the **MegaRoot.CurrentEnvironment** function.

This is recommended

The **MegaToolkit** public object can be obtained from the following objects:

- **MegaCurrentEnv**, using the **MegaCurrentEnv.Toolkit** function,
- **MegaApplication** using the **MegaApplication.Toolkit** function.

6.3 MEGA API Methods and Functions

HOPEX proposes methods and functions relating to MegaObjects and MegaCollections.

6.3.1 Functions on MegaCollections

Accessing a HOPEX repository

Any MegaObject (instance or collection) can access the repository to which it belongs by means of the **GetRoot** function.

VB Script

```
Dim Root
Set Root = MyObject.GetRoot
```

Java

```
MegaRoot mgRoot;
 mgRoot = mgobjMyObject.getRoot();
```

Collections of isolated objects

Collections obtained from the repository via the name of a MetaClass enable access to all instances of this MetaClass in the repository.

VB Script

```
Dim MyProjects
Set oRoot = object.GetRoot
Set MyProjects = oRoot.getCollection("~qekPESs3iG30[Project]")
' contains all projects in the repository.
```

Java

```
MegaCollection mgcolMyProjects = (MegaCollection)
mgRoot1.invokePropertyGet("~qekPESs3iG30[Project]");
```

You can also select objects using the **GetSelection** function.

VB Script

```
Dim MyProjects
```

```
Set MyProjects = Root.GetSelection _
("Select ~qekPESs3iG30[Project] where Name like 'A#'")
```

Java

```
MegaCollection mgcolMyProjects = mgRoot.getSelection("Select
~qekPESs3iG30[Project] where Name like 'A#'");
```

GetSelection can be called from any instance.

The **GetTypeID** function returns the identifier of the MetaClass of instances in the collection.

Collections of slave objects (connected)

Collections obtained from an instance other than the root are association collections. This is the way to obtain a collection of objects that are connected to an instance by a given association.

The original instance can be retrieved with the **GetSource** function. The **GetTypeID** function returns the identifier for the MetaAssociationEnd used.

VB Script

```
Set theSource = myOperationCollection.GetSource
Set theDuplicate = theSource.GetCollection _
(myOperationCollection.GetTypeID)
```

Java

```
MegaObject mgobjSource = mgcolOperations.getSource();
              MegaCollection mgcolOperationsDuplicate =
mgobjSource.getCollection(mgcolOperations.getTypeID());
```

The objects contained in such collections are slave objects. They can be used to access either the characteristics of the linked object or those of the MetaAssociation traversed.

VB Script

```
For each anOperation In myOperationCollection
' Name is an attribute of the operation, order is a link attribute
oRoot.Print anOperation.GetProp("~210000000900[Name]"), anOperation.
GetProp("~410000000H00[Order]")
Next
```

Java

```
String strText = "";
for (MegaObject mgobjOperation : mgcolOperations) {
  // Name is an attribute of the operation, order is a link attribute
  strText = strText + mgobjOperation.getProp("~210000000900[Name]") + " " +
mgobjOperation.getProp("~410000000H00[Order]") + "\n";
}
mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]", strText);
```

The **GetTypeID** function when used on a slave object returns the identifier of the MetaAssociationEnd used. It returns the same value as **GetTypeID** for the collection that provided access to the instance.

A slave object establishes a relation between three sub-objects:

- The source object, which can be obtained using the **GetSource** function

  (which is the equivalent function for the collection).

- The target object, which can be obtained using the **GetTarget** function.
- The association object, which can be obtained using the **GetRelationship** function.

The slave object is therefore a shortcut to the target and association objects. The above code can also be written as:

VB Script

```
For Each Ope In myOperationCollection

oRoot.Print Ope.GetTarget.GetProp("~210000000900[Name]"),
Ope.GetRelationship. GetProp("~410000000H00[Order]")

Next
```

Java

```
String strText = "";

for (MegaObject mgobjOperation : mgcolOperations) {

// Name is an attribute of the operation, order is a link attribute

strText = strText + mgobjOperation.getTarget().getProp("~210000000900[Name]")
+ " " + mgobjOperation.getRelationship().getProp("~410000000H00[Order]") +
"\n";

}

mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]", strText);
```

The target and source objects are not slave objects. The **GetTypeID** function returns the identifier for the object type and therefore is similar to **GetClassID**. The example below lists the "brothers" of a slave object (including itself).

VB Script

```
For Each Brother In _ MySlave.GetSource.GetCollection(MySlave.GetTypeID)

oRoot.Print Brother.GetProp("~210000000900[Name]")

Next
```

Java

```
String strText = "";

              MegaObject mgobjMySlave = mgcolOperations.get(1);

              for (MegaObject mgobjBrother :
mgobjMySlave.getSource().getCollection(mgobjMySlave.getTypeID())) {

                strText = strText +
mgobjBrother.getProp("~210000000900[Name]");

              }

              mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]", strText);
```

Sorted collections

It is possible to obtain a collection that is sorted by one or more characteristics of the instances in the collection.

VB Script

```
Set MyOperations = MyProj.Operation ("~410000000H00[Order]", "~
Z20000000D60[Short Name]")

Set MyOperations = MyProj.Operation (1, "~ Z20000000D60[Short Name]")

Set MyOperations = MyProj.Operation _

(1, "~510000000L00[Creation Date]", -1, "~410000000H00[Order]")
```

Java

```
MegaObject mgcolMyProj =
mgRoot.getCollection("~qekPESs3iG30[Project]").get(1);

    MegaCollection mgcolMyOps = mgcolMyProj.getCollection("Operation",
"~410000000H00[Order]", "~ Z20000000D60[Short Name]");

    mgcolMyOps = mgcolMyProj.getCollection("Operation", 1,
"~Z20000000D60[Short Name]");

    mgcolMyOps = mgcolMyProj.getCollection("Operation", 1,
"~510000000L00[Creation Date]", -1, "~410000000H00[Order]");
```

A positive value preceding the criterion indicates a sort in ascending order, a negative value means descending order.

Sort is in ascending order by default.

Searching a collection

The implicit **Item** function is used to find a particular instance in a collection. In addition to the name (local if applicable), the identifier, or the sequence number, an instance can be found by one of its characteristics:

VB Script

```
Set MyInstance = MyOperations("Type Operation", "A")

Set MyOcc = MyOperations(TypeOpeID, "A") ’ identical to above,

’ if TypeOpeID contains the identifier of the characteristic
```

If several instances match, the first one is returned.

It is also possible to find a specific position by using an object directly.

VB Script

```
Set MyOcc = MyOperations(MyOtherOcc) ’ equivalent to

Set MyOcc = MyOperations(MyOtherOcc.GetID)
```

6.3.2 Functions and methods on MegaObjects

A function executes processing and returns a result. Unlike a method, parameters specified for a function must be placed between brackets.

Modifying an instance

To modify an instance, the GetProp method must be applied:

VB Script

```
MyOperations.Comment = _

"Here is the comment for my operation..."

MyOperations.GetProp("~f10000000b20[Comment]") = _

"Here is the comment for my operation..."
```

Java

```
mgobjMySlave.setProp("~f10000000b20[Comment]", "Here is the comment for my
operation...");
```

Disconnecting a slave object

To disconnect a slave object:

Apply the **Remove** method to a collection of slave objects specifying the name, identifier, or object to be connected.

⚠ **The object is no longer usable after this operation.**

VB Script

```
MyCollection.Remove MyObject
```

Java

```
remove();
```

Deleting an Object

To delete an object:

- ➢ Apply the **Delete** method to the object.

⚠ **The object is no longer usable after this operation.**

Be careful while using Delete on a collection, as items are shifted and the use of "for each" might not work properly.

VB Script

```
MyObject.Delete ""
```

Java

```
delete("");
```

Connecting an object

To connect an object:

Apply the **Add** method to a collection of slave objects, specifying the name, identifier, or object to be connected.

VB Script

```
MyCollection.Add MyObject
MyCollection.Add MyObject.GetID
MyCollection.Add "Test"
```

"Test" is the name of an existing instance.

Java

```
mgcolMyCollection.add(mgcolMyObject);
mgcolMyCollection.add(mgcolMyObject.getID());
mgcolMyCollection.add("Test");
```

Creating an object

To create an object, apply the Create method to a collection of objects obtained from the root (a simple create) or from a collection of slave objects (create/ connect).

- Without a parameter, the function creates an instance that has as its name the class name followed by a counter.
- A parameter of type String specifies the name (local if applicable) of the instance.

For any other attribute, you must specify the attribute name (or identifier), followed by the value to be assigned.

VB Script

```
MyCollection.Create
MyCollection.Create "Name1"
MyCollection.Create "Name2", "Comment", _
"Comment of Name2"
```

Java

```
mgcolMyCollection.create();
mgcolMyCollection.create("Name1");
mgcolMyCollection.create("Name2", "Comment", "Comment of Name2");
```

Accessing a new object

A new object created using an **Add** or **Create** method can be accessed as follows:

VB Script

```
set MyNewObj1 = MyColl.Create("Name1")
set MyNewObj2 = MyColl.Add ("Test")
```

Java

```
MegaObject mgcolMyNewObj1 = mgcolMyCollection.create("Name1");
MegaObject mgcolMyNewObj2 = mgcolMyCollection.add("Test");
```

Accessing the MetaClass of an object or collection

The **GetType** function allows an object or collection to be considered as an instance of a given MetaClass.

Used from a ***MegaObject***, the **GetType** function enables considering an object as a function of the MetaClass given as parameter.

VB Script

```
MyOrgProc.GetType("BPMN Process").explore runs the explorer from an object of
the Organizational Process concrete MetaClass, considering it as an object of
the "BPMN Owner Element" MetaClass.
```

Java

```
mgobjMyOrgProc.getType("BPMN Process").invokeMethod("Explore");
```

*Used without a parameter, operator **GetType** enables consideration of the current object as an element of the concrete MetaClass to which it belongs.*

Used from a ***MegaCollection*** of objects of different concrete MetaClasses, operator **GetType** allows you to obtain a collection restricted to instances of the MetaClass specified as parameter.

VB Script

```
oCollection.GetType("Sequence flow").explore runs the explorer on objects of
the collection belonging to the "Sequence flow" MetaClass
```

Java

```
mgcolCollection.getType ("~jsV6VsHL7vJ0[Sequence
Flow]").invokeMethod("Explore");
```

If no object of the collection inherits the MetaClass given as parameter, operator **GetType** returns nothing. No error is generated.

Accessing object attributes

The **GetAttribute** function enables access to an object of MegaAttribute type to obtain type characteristics, translations if these exist, as well as the attribute value in its different formats.

VB Script

```
Print myobj.GetAttribute("Operation Type")
' Displays the attribute internal value
Print myobj.GetAttribute("Operation Type").Value("Display")
'Displays the attribute value in the language used
```

Java

```
mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]", mgcolMyObj.getProp("Operation
Type"));
// Displays the attribute internal value
mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]",
mgcolMyObj.getAttribute("Operation Type").getValue("Display"));
// Displays the attribute value in the language used
```

Attribute Format

You can obtain an attribute value in a given format. The default format is ASCII.

Available formats are: "Internal", "Ascii", "External", "Display". This is used as follows:

VB Script

```
Print MyOcc.GetProp("Operation Type", "External")
```

Java

```
mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]", mgobjOcc.getProp("Operation
Type", "External"));
```

A default format can be specified on the instance using the **SetDefault function**.

VB Script

```
MyInstance.SetDefault("Display")
```

Java

```
mgobjOcc.setDefault("Display");
```

When defined for the root, the default format is applied to all instances.

VB Script

```
Root.SetDefault("Display")
```

Java

```
mgRoot.setDefault("Display");
```

6.3.3 Functions on MegaCurrentEnvironments

For examples, see:

- Getting the person or person group used for current session
- Accessing MEGA API Public Objects
- Accessing public objects from another MegaObject
- Getting the current snapshot date
- Flushing the ERQL cache

To compile the environment use the **Compilation** macro:

oRoot.CallFunction("~91tmJFSUQL6I[Compilation]", MetaModel, TechnicalData, Permission, DeleteOldCompilationResults, SaveCurrentCompilationResults)

```
Dim oRoot

Dim MetaModel,TechnicalData,Permission,DeleteOldCompilationResults,SaveCurrentCompilationResults

MetaModel=1

TechnicalData=1

Permission=1

DeleteOldCompilationResults = 1

SaveCurrentCompilationResults = 1

sResult = oRoot.CallFunction("~91tmJFSUQL6I[Compilation]", MetaModel, TechnicalData, Permission, DeleteOldCompilationResults, SaveCurrentCompilationResults)
```

`sResult` is a json that gives a status after compilation:

```
{
  "compilationrequired":1,
  "metamodelcompiled":1,
  "technicaldatacompiled":1,
  "permissionscompiled":1,
  "messages":[]
}
```

- Getting the environment compilation states

6.3.4 Methods on a Reporting Datamart

From HOPEX Administration application (Administration.exe) you can schedule a synchronization on all the repositories for all their Reporting Datamarts.

Using APIs, you can launch and particularly schedule actions on a specific repository, for a specific Reporting Datamart, and without connecting to the Administration application.

See Initializing and synchronizing a Reporting Datamart.

6.4 Summary of Functions

6.4.1 Functions on MegaItems

Functions available on MegaItems, ie. on all objects:

VB Script	Java
`CallFunction`	`public Object callFunction(Object name, Object... arg);`
`CallMethod`	`public void callMethod(Object name, Object... arg);`
`GetMethodID`	`public Object getMethodID(Object name);`
`GetNature`	`public String getNature(String format);`
`GetRoot`	`public MegaRoot getRoot();`
`GetSource`	`public MegaObject getSource();`
`GetTypeID`	`public Object getTypeID();`
`SetDefault`	`public void setDefault(String defaultValue);`

6.4.2 Functions on MatrixContext

Functions available on MatrixContext:

VB Script	Java
`GetRoot As MegaRoot`	`public MegaRoot getRoot();`
`GetMatrixTypeId As Variant`	`public Variant GetMatrixTypeId();`
`GetMatrixId As Variant`	`public Variant GetMatrixId();`
`GetClassId As Variant` ID of the Row MetaClass or Column MetaClass	`public Variant GetClassId();`
`InsertItem(ItemID) As Boolean`	`public boolean InsertItem(ItemID)`

Row or column item insertion. Returns true if the insertion succeeded, else false (object already in or not compatible).	
`GetContainer As Object`	`public Object getContainer (Object name);`
Returns the container of the matrix. Can be Nothing. If the matrix is used in a Property Page context, it returns the AttributeControl corresponding to the matrix.	

See also Using macros to add rows/columns in a matrix.

6.4.3 Functions on MegaCollections

Functions available on MegaCollections:

VB Script	Java
`AdviseRegister`	`public MegaAdviseToken adviseRegister(MegaAdviseTarget target);`
`Add`	`public MegaObject add(Object objectID);`
`Count`	`public int size();`
`Create`	`public MegaObject create();`
`Insert`	`public void insert(Object toInsert);`
`Item`	`public MegaObject get(Object index);`
`Remove`	`public void remove(Object toRemove);`

6.4.4 Functions on MegaObjects

Functions available on MegaObjects:

VB Script	Java
`Delete`	`public void delete(String options);`
`GetClassID`	`public Object getClassID();`
`GetCollection`	`public MegaCollection getCollection(Object vcolID, Object... sortCriteria);`
`GetCollectionID`	`public Object getCollectionID(Object name);`
`GetID`	`public Object getID();`

VB Script	Java
GetProp	public String getProp(Object propID);
GetPropID	public Object getPropID(Object name);
GetRelationship	public MegaObject getRelationship();
GetType	public MegaObject getType(Object name);
GetTarget	public MegaObject getTarget();
Item	public Object item();
Remove	public void remove();
SetProp	public void setProp(Object propID, String value);

6.4.5 Functions on MegaAttributes

Functions available on MegaAttributes:

VB Script	Java
Value	Value public String getValue();
DefaultFormat	
TextType	public Object getTextType();
DescriptionObject	public MegaObject getDescriptionObject();
GetAsPrivateProfile	public MegaPrivateProfile getAsPrivateProfile();
Translate	public MegaAttribute translate(Object language);
SourceObject	public MegaObject getSourceObject();
TestValue	public String testValue(Object value, String format);
Status	public int getStatus();
GetAttributeValue	public Object getValue(String format);

6.4.6 Functions on MegaRoot objects

Functions available on MegaRoot objects:

VB Script	Java
BaseCanClose	public boolean isClosed();
BaseClose	public void close();

VB Script	Java
ContextObject	public MegaCOMObject contextObject(String Name);
CurrentEnvironment	public MegaCurrentEnvironment currentEnvironment();
GetBaseInfo	
EnterPrivilege	public Object enterPrivilege(String description);
LeavePrivilege	public void leavePrivilege();
TryPrivilege	public Object tryPrivilege(String[] description);
GetSelection	public MegaCollection getSelection(String request, Object... sortCriteria);
GetExcecutionStatus	public int getExecutionStatus();
GetObjectFromID	public MegaObject getObjectFromID(Object ident);
GetSystemRoot	
GetClassDescription	public MegaObject getClassDescription(Object classID);
GetCollectionDescription	
MegaCommit	public void megaCommit();
MegaRollback	public void megaRollback();
print Print	public void print(Object value);
SetOpenToken	public void setOpenToken(String token);
SetUpdater	

6.4.7 Functions on MegaCurrentEnvironment

Functions available on MegaCurrentEnvironment:

VB Script	Java
GetCurrentSnapshotDate	public Date getCurrentSnapshotDate();
GetUserOption	myroot.currentEnvironment.getUserOption
GetCompilationStates	
GetCurrentLoginHolder	
GetCurentUserId	

6.5 MEGA Operators

6.5.1 Operator types

The operator types are:

- "Compound": enables specification of behavior on MetaAssociations.

  Examples of "Compound" type operators :

  - Owner
  - Ownership
  - NameSpace
  - Propose
  - Reference

- "Method": implements methods and functions available for APIs. These methods and functions can relate to a MegaObject, a MegaCollection, a MetaClass or to all MetaClasses.

  Examples of "Method" type operators :

  - Edit
  - Explore

    invokeMethod("Explore")
  - IsAvailable
  - Open
  - SaveAs
  - Excel Import
  - Excel Export

- "Atomic": used for basic commands of HOPEX.

  Examples of "Atomic" type operators :

  - Abbreviate
  - Connect
  - Create
  - Disconnect
  - Description
  - Implicit
  - Import
  - Read
  - Delete
  - Translate

To access all operators, search from the "_Operator" MetaClass.

6.5.2 Creating an operator

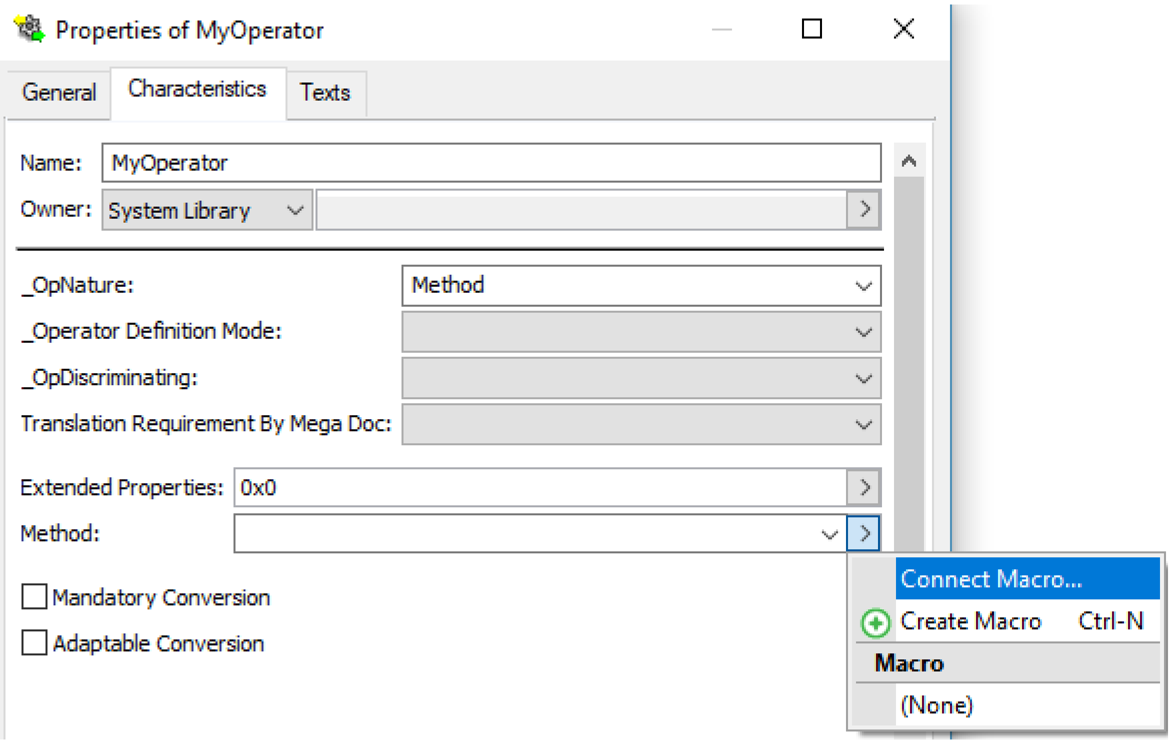
An operator is a standard function that may be called either from the root or from a specific MetaClass.

To create the operator:

1. Open the VB Script editor and enter the following code:

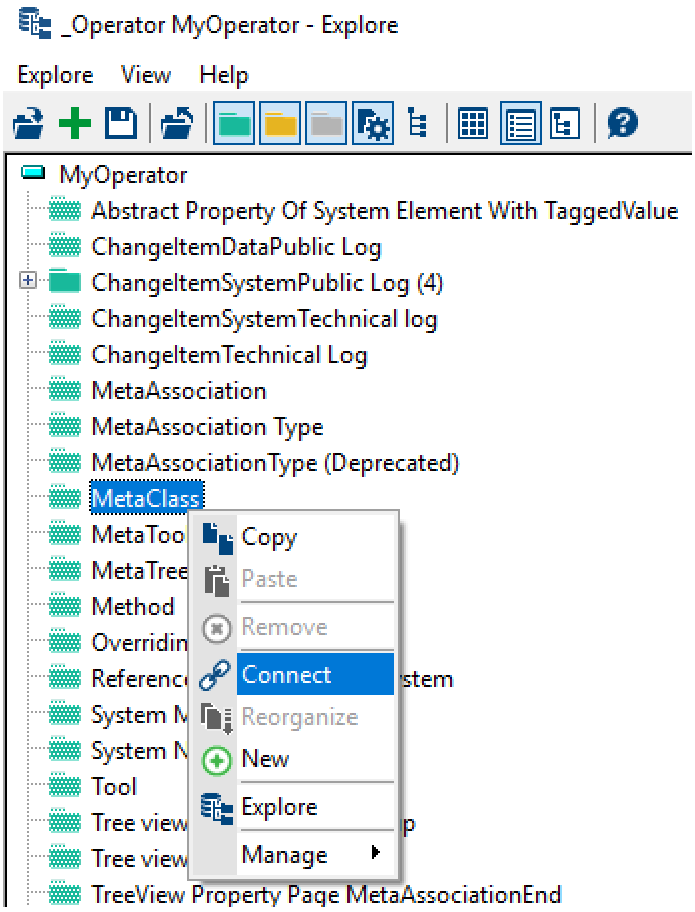
```
[_Operator].Create("MyOperator").explore
[Macro].Create("MyOperator.Macro").explore
```

2. Access to the operator properties and connect the macro you have just created to your operator:



Name: MyOperator
Owner: System Library
_OpNature: Method
_Operator Definition Mode:
_OpDiscriminating:
Translation Requirement By Mega Doc:
Extended Properties: 0x0
Method:
☐ Mandatory Conversion
☐ Adaptable Conversion

If your operator is specific to a MetaClass, connect the MetaClass to the operator through the **MetaClass** MetaAssociationEnd.



3. Navigate to the macro implementing the operator.

   For example: `MyOperator.Macro`.

4. If the operator needs to be written in:

   - Java: specify on the macro properties, the java class factory in the "_ObjectFactory" and the "dispatch" mode in the "SystemComponent" attribute.
   - VB Script: edit the macro.

- If the operator applies on a **MegaObject**, the following methods must be implemented

VB Script The macro must implement the following functions:

```
`specify the number of arguments that will be passed to the operator: here 3.

Function InvokeArgCount() As Integer

  InvokeArgCount = 3

End Function

`Method called when the operator is called with 3 arguments. The first
argument is the object on which the call is made (megaobject).

Sub InvokeOnObject(Obj as MegaObject, sFileName As String, strReportFilename
as String , controlComponent As MegaObject)

  End Sub
```

Java The factory must define a class containing the following functions:

```
public class MyOperator {
  /**
   * Specifies how many parameters are expected
   * if not defined, the method called has only one argument : the megaobject
on which the call has been made
   */
  public int InvokeArgCount() {
    return 3;
  }
  /**
   * function called with the 3 arguments. The first argument is the object
on which the call is made
   */

  public void InvokeOnObject(final MegaObject mgObjectToExport, final String
strFilename, String strReportFilename, final MegaCOMObject controlComponent)
  {
    //procedure called when the operator is called
  }

}
```

Note that the InvokeOnObject method may return a value both in Java or VB Script. In that case, the java function must be defined with a return value and the VB Script method must be defined as a Function.

VB Script

```
'if you want your method to return a value, use
Function InvokeOnObject(Obj As MegaObject)
End Function
```

Java

```
//return a number
public int InvokeOnObject(final MegaObject mgObject)
{
}
```

- If the operator applies on **MegaRoot**, the following methods must be implemented. If you need to add additional arguments, you need to implement **InvokeArgCount** to specify the number of arguments.

VB Script

```
Sub InvokeOnRoot(oroot As MegaRoot)
End Function
'if you want your method to return a value, use
Function InvokeOnRoot(oroot As MegaRoot)
```

```
End Function
```

Java

```
public void InvokeOnRoot(final MegaRoot mgRoot)
{
}
Or if you want your method to return a value, use
//return an integer
public int InvokeOnRoot(final MegaRoot mgRoot)
{
}
```

- If the operator applies on a collection. In that case, the method to be implemented is **InvokeOnCollection**. If you need to add additional arguments, you need to implement **InvokeArgCount** to specify the number of arguments.

VB Script

```
Sub InvokeOnCollection(Coll As MegaCollection)
End Sub
'if you want your method to return a value, use
Function InvokeOnCollection(Coll As MegaCollection)
End Function
```

Java

```
public void InvokeOnCollection(final MegaCollection lst )
{
}
Or if you want your method to return a value, use
//return an integer
public int InvokeOnCollection(final MegaCollection lst )
{
}
```

7 MACROS USED IN HOPEX

Here are examples of where macros (Java or VB Script) are used in HOPEX.

For information on macro about…		**see…**
Command management	MetaCommand Manager	HOPEX Power Studio > Customizing the User Interface > **Versatile Desktop > Configuring the desktop > Examples of macros**
	MetaCommand Item	
	MetaCommand Group	
Wizard implementation kinematic		HOPEX Power Studio > Customizing the User Interface > **Forms - Wizard Implementation – Tutorial** HOPEX Power Studio > Customizing the User Interface > **Forms**
Property Pages	Macro call	HOPEX Power Studio > Customizing the User Interface > **Forms - Property Pages – Tutorial** HOPEX Power Studio > Customizing the User Interface > **Forms**
	viewport management	HOPEX Power Studio > Customizing the User Interface > **Forms**
Metamodel	updateTool implementation – on attributes – on links	HOPEX Power Studio > Customizing the User Interface > **HOPEX APIs > All about starting with APIs** (see Implementing an Update Tool in script) HOPEX Power Studio > Customizing the User Interface > **Forms**
	calculated attributes	HOPEX Power Studio > Customizing the Metamodel > **Managing the MetaModel > MetaAttributes > Using VB Scripts to Calculate Characteristics**
	abstract property implementation	HOPEX Power Studio > Customizing the User Interface > **Forms**
	collection implementation	HOPEX Power Studio > Customizing the User Interface > **Forms**
Operators	Operator implementation	HOPEX Power Studio > Customizing the User Interface > **HOPEX APIs > All about starting with APIs** (see

For information on macro about…		**see…**
		MEGA Operators)
	Writing of operator behavior on links	HOPEX Power Studio > Customizing the Metamodel > **Perimeters > Configuring a perimeter > Configuring perimeter links**
Requests	Writing a dynamic query	HOPEX Power Studio > Customizing the User Interface > **HOPEX APIs > All about starting with APIs** Technical Article (see Writing a dynamic query)
In diagrams	Post-processing	**HOPEX Power Studio > Customizing the User Interface > HOPEX APIs > JavaDoc** documentation
	Interactive Plug-in (diagram plug-in, drag and drop plug-in)	**HOPEX Power Studio > Customizing the User Interface > HOPEX APIs > All about starting with APIs** Technical Article (see Setting up interactive plug-ins in a diagram)
Macro call from RTF and HTML descriptors and code		HOPEX Power Studio > Customizing the User Interface > **HOPEX APIs > All about starting with APIs** (see Calling a macro from HTML, code and RTF descriptors)
Report content		HOPEX Power Studio > Report Studio > **Writing Java report chapters** HOPEX Power Studio > Report Studio > **Writing Java report renderers**
Definition rule	Modifying password	HOPEX Administration (Web) > Managing Users > **Managing the password of a Web User > Modifying password management configuration >** Modifying password definition rules
Desktop	– Click Manager – Event management load, close, save and deactivation – Tool	HOPEX Power Studio > Customizing the User Interface > **Versatile desktop**
Object and link display management	In diagrams via DiagramTypeXXX Condition	

For information on macro about...		see...
Post-processing on business documents		HOPEX Power Studio > Customizing Documentation > **Customizing Business Documents >** Modifying Document Behavior
Assessment	Plug-in	**Assessment** (internal document)
Perimeter	Behavior	HOPEX Power Studio > Customizing the Metamodel > **Perimeters**
Workflow		**HOPEX Power Studio - Workflows** user guide
Specific processing to certain concept	User group management	HOPEX Administration (Web) > Managing Users > Creating and Managing a Person group > **Defining a Person Group > Defining a dynamic person group with a macro**
	Steering Calendar	HOPEX Power Studio > Customizing Steering Calendars > **Steering calendar** **HOPEX Customization - Assessment** (internal document)
	Scheduler	HOPEX Power Studio > **Using the Scheduler**

8 ADMINISTRATION OF HOPEX FROM APIS

The objective of administration applications is to manage workspaces, environments, users, etc. HOPEX administration can be carried out from the application itself or from the outside.

8.1 Introduction

8.1.1 Starting administration

To start an administration application in VBScript, you must use a **MegaApplication** type variable. You can access such a variable by using the standard ***CreateObject*** function:

```
Dim oMegaApplication

Set oMegaApplication = CreateObject ("Mega.Application")
```

When the variable has been created, you can access all the properties accessible from the **MegaApplication** class and its associated objects (environment, repository, etc.).

Before each execution of components using Administration API (**CreateObject** (“**Mega.Application**”)), you must reference **Mega.Application**. To do so, use the **HOPEX-regserver.ps1** powershell script available at the root folder of the HAS instance:

<Path to HAS>\HOPEX Application Server\<Instance port>

```
C;\MEGA\HOPEX Application Server\5000

>.\HOPEX-regserver.ps1
```

8.1.2 Connecting to an open session

To connect to an open session to analyze or modify the current repository:

- use a **MegaCurrentEnv** type variable, or
- use the standard function **CreateObject ("MegaRepository.CurrentEnv")**.

From such a variable you can access the repository root (**MegaRoot**) and therefore all the contained objects.

Example: access to the current repository. This example displays the number of repository procedures:

```
Dim oMegaCurrentEnv

Set oMegaCurrentEnv = CreateObject (“MegaRepository.CurrentEnv”)

Dim oMegaRoot as MegaRoot

Set oMegaRoot = oMegaCurrentEnv.GetRoot

MsgBox oMegaRoot.Getcollection ("Procedure").Count
```

Connection to an already open HOPEX session or manual opening

The **MegaCurrentEnv** object enables you to determine whether a HOPEX session is open or not by means of the **IsSessionOpen** function.

If the HOPEX session is not open, the **GetRoot** function now proposes an interactive connection dialog box enabling the user to open a session.

Intrusion protection

To protect the repository against malicious intrusion, HOPEX displays an acceptance message when calling the ***GetRoot*** function. This behavior can be cancelled in the case of a call originated by HOPEX. To do this, see the reference guide, **SetOpenToken** function.

8.2 Repository Administration Tasks

You can carry out certain HOPEX administration tasks using APIs.

You can execute these tasks by means of VB scripts written and stored in external .vbs files.

HOPEX must be closed during execution of VB scripts carrying out administration tasks. The data regarding environments, workspaces, users and repositories cannot be read when a HOPEX session is open.

8.2.1 Getting the environment IdAbs

To get the environment IdAbs, enter the following code in a .vbs file:

```
Set oMegaApp = CreateObject ("Mega.Application")

Set oEnvironment = oMegaApp.Environments.Item("EnvironmentFolder")

MsgBox "Idabs de l'environnement: " & oEnvironment.GetProp("EnvHexaIdAbs")
```

Replace "EnvironmentFolder" in bold by the environment path.

Code description:

```
Set oMegaApp = CreateObject ("Mega.Application")
```

Creates an instance of the class MegaApplication and assigns it to the "oMegaApp" Variable. This class defines the data corresponding to a HOPEX site.

```
Set oEnvironment = oMegaApp.Environments.Item("EnvironmentFolder")
```

From the environments defined for the application oMegaApp, the environment located in the "EnvironmentFolder" folder is retrieved and assigned to the "oEnvironment" Variable.

You must specify the path in the form in which the environment was referenced (if a UNC path was used, you must use the UNC path).

```
MsgBox "IdAbs of the environment: " & oEnvironment.GetProp("EnvHexaIdAbs")
```

The environment IdAbs is retrieved.

8.2.2 Connecting to an open repository

To connect to an open repository, enter the following code in a .vbs file:

```
Set oMegaApp = CreateObject ("Mega.Application")

Set oEnvironment = oMegaApp.Environments.Item("EnvironmentFolder")

oEnvironment.CurrentAdministrator = "Administrator"
oEnvironment.CurrentPassword = "AdministratorPassword"
```

```
Set oDataBase = oEnvironment.Databases.Item("RepositoryName")
```

Replace the words in bold by the environment path, administrator name, administrator password, repository name and repository logical backup file name respectively.

Code description:

```
Set oMegaApp = CreateObject ("Mega.Application")
```

Creates an instance of the class MegaApplication and assigns it to the "oMegaApp" Variable. This class defines the data corresponding to a HOPEX site.

```
Set oEnvironment = oMegaApp.Environments.Item("EnvironmentFolder")
```

From the environments defined for the application oMegaApp, the environment located in the "EnvironmentFolder" folder is retrieved and assigned to the "oEnvironment" Variable.

You must specify the path in the form in which the environment was referenced (if a UNC path was used, you must use the UNC path).

```
oEnvironment.CurrentAdministrator = "Administrator"
```

The administrator name is entered.

```
oEnvironment.CurrentPassword = "AdministratorPassword"
```

The administrator password is entered.

```
Set oDataBase = oEnvironment.Databases.Item("RepositoryName")
```

From the repositories of the selected environment, the "RepositoryName" repository is retrieved and assigned to the "oDataBase" variable.

8.2.3 Repository logical backup

Having written code for connection to the repository oDataBase, enter the following code in a .vbs file:

```
oDataBase.LogicalSave "LogicalBackupFileName")

WScript.Echo "Processing completed"
```

Code description:

```
oDataBase.LogicalSave "LogicalBackupFileName")
```

Applies the "LogicalSave" function (which carries out repository logical backup) to the selected repository, specifying the complete name of the backup file.

```
WScript.Echo "Processing completed"
```

When logical backup is completed, a window appears and displays "Processing completed".

8.2.4 Compiling the environment

To compile the environment use the **Compilation** macro:

oRoot.CallFunction("~91tmJFSUQL6I[Compilation]", MetaModel, TechnicalData, Permission, DeleteOldCompilationResults, SaveCurrentCompilationResults)

```
Dim oRoot

Dim MetaModel,TechnicalData,Permission,DeleteOldCompilationResults,SaveCurrentCompilationResults

MetaModel=1

TechnicalData=1

Permission=1

DeleteOldCompilationResults = 1

SaveCurrentCompilationResults = 1

sResult = oRoot.CallFunction("~91tmJFSUQL6I[Compilation]", MetaModel,
TechnicalData, Permission, DeleteOldCompilationResults,
SaveCurrentCompilationResults)
```

`sResult` is a json that gives a status after compilation:

```
{
  "compilationrequired":1,
  "metamodelcompiled":1,
  "technicaldatacompiled":1,
  "permissionscompiled":1,
  "messages":[]
}
```

8.2.5 Getting the environment compilation states

The `IsCompiled` function can be used to get information regarding the Metamodel and Technical Data compilations, but this function does not take into account more compilation states like permissions.

To get more information regarding the current environment compilation state, use

```
CurrentEnvironment.GetCompilationStates
```

This function enables to get the compilation states of MetaData, TechnicalData, MetaPicture, MetaTreeNode, DiagramTypeZoom, DiagramType, Query, and Permission.

It returns a bit string (0 to 8) corresponding to each of the states

&H0001 for "MetaData"
&H0002 for "TechnicalData”
&H0004 for "Workflow"
&H0008 for "MetaPicture"
&H0010 for "MetaTreeNode"
&H0020 for "DiagramTypeZoom"
&H0040 for "DiagramType"
&H0080 for "Query"
&H0100 for "Permission"

Example: to get the permission compilation state of the current environment, enter the following code in a .vbs file:

```
Dim uStates
uStates = oRoot.CurrentEnvironment.GetCompilationStates
if uStates and &H0100 then
  print "Permissions are compiled"
else
  print "Permissions are not compiled"
end if
```

8.2.6 Reinitializing a repository backup logfile

Having written code for connection to the repository oDataBase, enter the following code in a .vbs file:

```
Set oDataBaseLog = oDataBase.Log
oDataBaseLog.Reset
WScript.Echo "Processing completed"
```

Code description:

```
Set oDataBaseLog = oDataBase.Log
```

The selected repository logfile is retrieved and assigned to the variable "oDataBaseLog".

```
oDataBaseLog.Reset
```

The "Reset" function, which reinitializes this logfile, is applied to the repository logfile.

```
WScript.Echo "Processing completed"
```

When backup is completed, a window appears and displays "Processing completed".

8.2.7 Deleting a repository

Having written code for connection to the repository oDataBase, enter the following code in a .vbs file:

```
oDataBase.Destroy
WScript.Echo "Processing completed"
```

Code description:

```
oDataBase.Destroy
```

The "Destroy" function that deletes the repository is applied to the selected repository.

```
WScript.Echo "Processing completed"
```

When deletion of the repository is completed, a window appears and displays "Processing completed".

8.2.8 Deleting a workspace

Having written code for connection to the repository oDataBase, enter the following code in a .vbs file:

```
oTransaction.Abort

WScript.Echo "Processing completed"
```

Code description:

```
oTransaction.Abort
```

The "Abort" function that deletes the workspace is applied to the workspace.

```
WScript.Echo "Processing completed"
```

When deletion of the workspace is completed, a window appears and displays "Processing completed".

8.2.9 Disabling repository log

A session parameter enables to disable repository log when running HOPEX API processing.

To prevent repository log deactivation for updates independent from the current thread, you can only deactivate the repository log using the update privilege.

The repository log deactivation is managed by an option, a new script parameter of EnterPrivilege:

```
MegaRoot.EnterPrivilege "DESCRIPTION","DisableLog=1"
```

8.2.10 Deactivating/Reactivating repository log

For performance optimization, you can deactivate/reactivate the repository log (also undo, locks, CRUD test) in a macro.

See Deactivating/Reactivating repository log, undo, locks, CRUD test.

8.2.11 Flushing the ERQL cache

Query results are cached. This avoid repeating calculation of the same query results, so that frequently used queries are executed faster.

For test purpose, if you need to flush this ERQL cache, use the following function:

```
CurrentEnvironment.RefreshQueryResult
```

8.2.12 Initializing and synchronizing a Reporting Datamart

From HOPEX Administration application (Administration.exe) you can schedule a synchronization on all the repositories for all their Reporting Datamarts.

Using APIs, you can launch and particularly schedule actions on a specific repository, for a specific Reporting Datamart, and without connecting to the Administration application.

Use:

`DataInitialize` to initialize the Reporting Datamart data with the HOPEX repository content according to the user/profile permissions and the object type selected at Reporting Datamart creation.

`StructureInitialize` to initialize the structure with the HOPEX repository metamodel. All the tables and columns are created even if there is no data to feed them. The table remains empty, so that the Reporting Datamart structure is always consistent. This ensures the user that any of his queries run properly on the Reporting Datamart.

`IncrementalUpdate` to launch an incremental synchronization to update the Reporting Datamart with all the dispatches performed after the last synchronization.

`ComputedAttributesSync` to launch a calculated MetaAttribute synchronization to scan all the objects and links of the HOPEX repository that can have calculated MetaAttribute values and put their values in the Reporting Datamart.

Do not launch or schedule a calculated MetaAttribute synchronization if you do not use the values of calculated MetaAttributes from the Reporting Datamart.

`DiagramsSync` to launch a diagram synchronization to scan all the HOPEX repository diagrams and update the Reporting Datamart with their drawing representation in the SVG format.

Do not launch or schedule a diagram synchronization if yo u do not use diagrams.

VB Script For example, to initialize the “datamart1” Reporting Datamart data with the “EA” HOPEX repository content:

```
set oEnv = CurrentEnvironment.Site.Environments.Item(CurrentEnvironment.Path)

set oDbEA = oEnv.Databases.Item("EA")

set oDbEADatamarts = oDbEA.Datamarts

set oDm1 = oDbEADatamarts.Item("datamart1")

oDm1.DataInitialize()
```

Code description:

```
set oEnv = CurrentEnvironment.Site.Environments.Item(CurrentEnvironment.Path)

set oDbEA = oEnv.Databases.Item("EA")
```

Specifies the HOPEX repository from which the Reporting Datamart is created: EA.

```
set oDm1 = oDbEADatamarts.Item("datamart1")
```

“datamart1” is the name of the Reporting Datamart to be updated.

```
oDm1.DataInitialize()
```

“datamart1” data is initialized with the “EA” HOPEX repository.

8.2.13 Getting repository information

To get information regarding the repository use the MegaRoot.GetBaseInfo("<information>") function.

If the MegaRoot is on the System Repository, to get the information on SystemDB, add:

```
Set MegaSystemRoot = MegaRoot.GetSystemRoot
MegaSystemRoot.GetBaseInfo("<information>")
```

Use:

`MegaRoot.GetBaseInfo("<information>")` to get the information in string format.

`MegaRoot.GetBaseInfo("<information>","Internal")` to get the information in internal format. This can be particularly useful for dates.

`MegaRoot.GetBaseInfo("#n")` to get all the information

where <information> can be:

TRANSTATE

STARTSTATE

STARTDATE

TRANSTATE_ORIGIN

IDABS

LANGUAGE_IDABS

LANGUAGE_NAME

PATHNAME

PATHNAME_WORK

PATHNAME_USER

DB_ALLOCATEDSIZE

DB_ALLOCATEDUSEDSIZE

MGE_REJECT_FILELONGNAME

MGR_REJECT_FILELONGNAME

MGL_REJECT_FILELONGNAME

BDATABASESYSTEM

BSYSTEM

DMP_FILELONGNAME

TRANSACTION_IDABS

CONNECTION_IDABS

TRAN_NAME

TRAN_LOGACTIVITY

TRAN_CRD_FILELONGNAME

TRAN_PATHNAME

TRAN_TYPE

TRAN_OPENMODE

TRAN_MGL_FILELONGNAME

TRAN_MGE_FILELONGNAME

TRAN_MGR_FILELONGNAME

TRAN_EXT_FILELONGNAME

TRAN_CMP_FILELONGNAME

TRAN_DEL_FILELONGNAME

XMG_REJECT_FILELONGNAME

TRAN_DATE_CREATION

TRAN_DATE_VALIDATION

TRAN_DATE_CONNEXION

TRAN_SIZE

PROFILE_SZ

TRAN_VERSION

DEFAULT_AUTHORIZATION

DEFAULT_VISIBILITY

BASELINE

ADVANCEDOBJECTTRACKING

DB_AREATYPE

DB_CONTENT_UPDATE_COUNT

For example, to get the absolute identifier (IdAbs) of a workspace (transaction):

```
MegaRoot.GetBaseInfo("TRANSACTION_IDABS")
```

8.3 Executing tasks offline

Certain time-consuming processing operations can be batch-executed using administration commands and MEGA APIs.

8.3.1 Reorganizing repositories

The script given as an example below enables reorganization of all repositories of the first environment of a site. It saves active workspaces in logfiles and executes logical backup of repositories. This script is an administration script: HOPEX must therefore be closed before running the script. Workspaces are recreated and their logfiles reinjected.

```
Explicit Option


Dim TabTrans(256,2), sDbSave, sDbRej, STransSave, sTransRej, sAdministratorName,
sPassword

Dim oMegaApp, oDataBase, oEnvironment, oTransaction, i, j, oShell, sSystem

i=0
```

```
' TODO: replace the two values with an available HOPEX user and its optional
password.

sAdministratorName = "a user name"

sPassword = "your password"
Set oMegaApp = CreateObject ("Mega.Application") Set oEnvironment =
oMegaApp.Environments.Item(1)
oEnvironment.CurrentAdministrator = sAdministratorName if sPassword <> "" then
oEnvironment.CurrentPassword = sPassword
' Stores each workspace in a logfile.
’There must be no active workspace
For each oTransaction in oEnvironment.Transactions
 If Right(oTransaction.Name,8)= "(System)" Then
 Else
  i=i+1
  TabTrans(i,1) = oTransaction.User name
  TabTrans(i,2) = oTransaction.Database.name
  STransSave = oEnvironment.path & "\Mega_usr\Trans" & oTransaction.Name & ".mgl"
  oTransaction.Database.Log.Export STransSave
  oTransaction.abort
 End If
Next

' Performs the logical save of each repository except the system repository.
For each oDatabase in oEnvironment.Databases
 If oDatabase.Name="SystemDb" Then
 Else
   sDbSave = oEnvironment.path & "\Mega_usr\DB" & oDataBase.Name & ".mgr"
   sDbRej  = oEnvironment.path & "\Mega_usr\DB" & oDataBase.Name & "Rej.mgr"
   oDataBase.LogicalSave
sDbSave,"meta=off,technical=off,data=On,FileOpen=Rewrite"
   oDataBase.Reset
   odataBase.Import sDbSave, sDbRej
 End If
Next

' Creates new workspace replacing the previous ones.
For j= 1 to i
  oEnvironment.Transactions.create TabTrans(j,2), TabTrans(j,1)
Next
' Imports the saved logs in the corresponding workspaces.
For each oTransaction in oEnvironment.Transactions
```

```
 sTransSave = oEnvironment.path & "\Mega_usr\Trans" & oTransaction.Name & ".mgl"
 sTransRej  = oEnvironment.path & "\Mega_usr\Trans" & oTransaction.Name &
"Rej.mgl"
 If Right(oTransaction.Name,8)= "(System)" Then
 Else
  oTransaction.Database.import sTransSave,sTransRej
 End If
Next

Set oTransaction = Nothing
Set oDataBase = Nothing
Set oEnvironment = Nothing
Set oMegaApp = Nothing

MsgBox "Environment closed"
```

8.3.2 Generating documents

To create a document using a VB script:

➢ Apply the **NewDocument** method to a MegaObject.

This method takes as a parameter the document template from which you create the document.

The **NewDocument** method returns the created document.

Example:

```
Set oRoot = object.GetRoot
Set objDoc = oRoot.GetCollection("~pj)grmQ9pG90[Business Process]").Item("~2F6SN4tSpyh0[Purchasing]").NewDocument("Business Process documentation")
```

Creates a document on the "Purchasing" business process from the "Business Process Description" document template and assigns it to the variable "objDoc".

To update document links:

➢ Apply to the document the **RefreshDocument** method without a parameter.

Example:

```
Set RefreshStatus = objDoc.RefreshDocument()
```

This method returns an object with value:

- 1 when document update is in progress
- 0 if document update is completed.

⚠ **Two document update operations cannot be started simultaneously.**

To detach a document from HOPEX:

- Apply to the document the **DetachDocument** method. This takes as parameter the complete file name (example: C:\My documents\Document.doc).

Example:

```
objDoc.DetachDocument("C:\My documents\Document.doc")
```

This method detaches the current document, cuts document links and creates a file independent of HOPEX.

8.3.3 Generating Web sites

To generate a Web site:

- Apply to the Web site the **GenerateWebSite()** function. Example:

```
Set oRoot = object.GetRoot
Set oWebSiteList = oRoot.GetCollection("Web Site")
For each oWebSite in oWebSiteList
 oWebSite.GenerateWebSite()
Next
```

In this example, Web sites existing in HOPEX are retrieved and we apply to each one the **GenerateWebSite()** function.

VB scripts also enable generation of the CHM file corresponding to a Web site. Apply to the Web site the **GenerateCHM()** function.

Example:

```
Set oRoot = object.GetRoot
Set oWebSiteList = oRoot.GetCollection("Web Site")
For each oWebSite in oWebSiteList
  oWebSite.GenerateCHM()
Next
```

In this example, Web sites existing in HOPEX are retrieved and you apply to each one the **GenerateCHM()** function.

9 COMMUNICATION BETWEEN HOPEX AND THE OUTSIDE

The MEGA application offers various possibilities of communication with the outside, for administration tasks for example.

9.1 API Scripts and .NET

⚠ **Use .NET applications only with Administration APIs. For Dispatch call, see "Late binding" information on Microsoft Help and Support website:**
http://support.microsoft.com/kb/302902/en-us

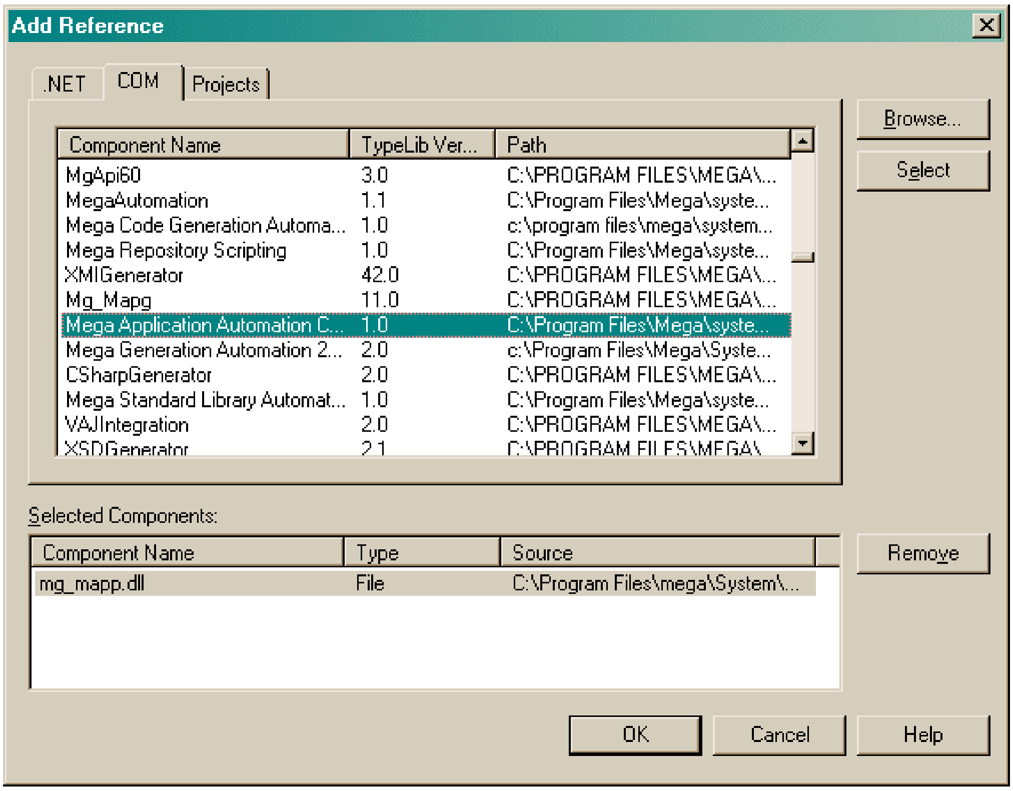
Access and update APIs of the HOPEX repository are presented in the form of COM components. They can be integrated in a .NET application. The user can develop .NET compatible components accessing HOPEX repository data. This section explains the principle used to implement a .NET application accessing data in a HOPEX repository. It also raises points relating to constraints linked to choice of .NET language.

9.1.1 Implementation principle

Whatever the language chosen for your .NET application (Visual Basic.NET, C++. NET, C#, etc.), you can integrate a COM component and use objects and functions exported.

In Visual.NET:

1. Select **Project > Add Reference**.
2. In the **Add Reference** dialog box, select the **COM** tab.
3. Select the **MEGA Application Automation component** component or query the DLL in **<HOPEX folder> \system\mg_mapp.dll**.
4. Add this dll to the list of selected components.



Component Name	TypeLib Ver...	Path
MgApi60	3.0	C:\PROGRAM FILES\MEGA\...
MegaAutomation	1.1	C:\Program Files\Mega\syste...
Mega Code Generation Automa...	1.0	c:\program files\mega\system...
Mega Repository Scripting	1.0	C:\Program Files\Mega\syste...
XMIGenerator	42.0	C:\PROGRAM FILES\MEGA\...
Mg_Mapg	11.0	C:\PROGRAM FILES\MEGA\...
Mega Application Automation C...	1.0	C:\Program Files\Mega\syste...
Mega Generation Automation 2...	2.0	c:\Program Files\Mega\Syste...
CSharpGenerator	2.0	C:\PROGRAM FILES\MEGA\...
Mega Standard Library Automat...	1.0	C:\Program Files\Mega\syste...
VAJIntegration	2.0	C:\PROGRAM FILES\MEGA\...
XSDGenerator	2.1	C:\PROGRAM FILES\MEGA\

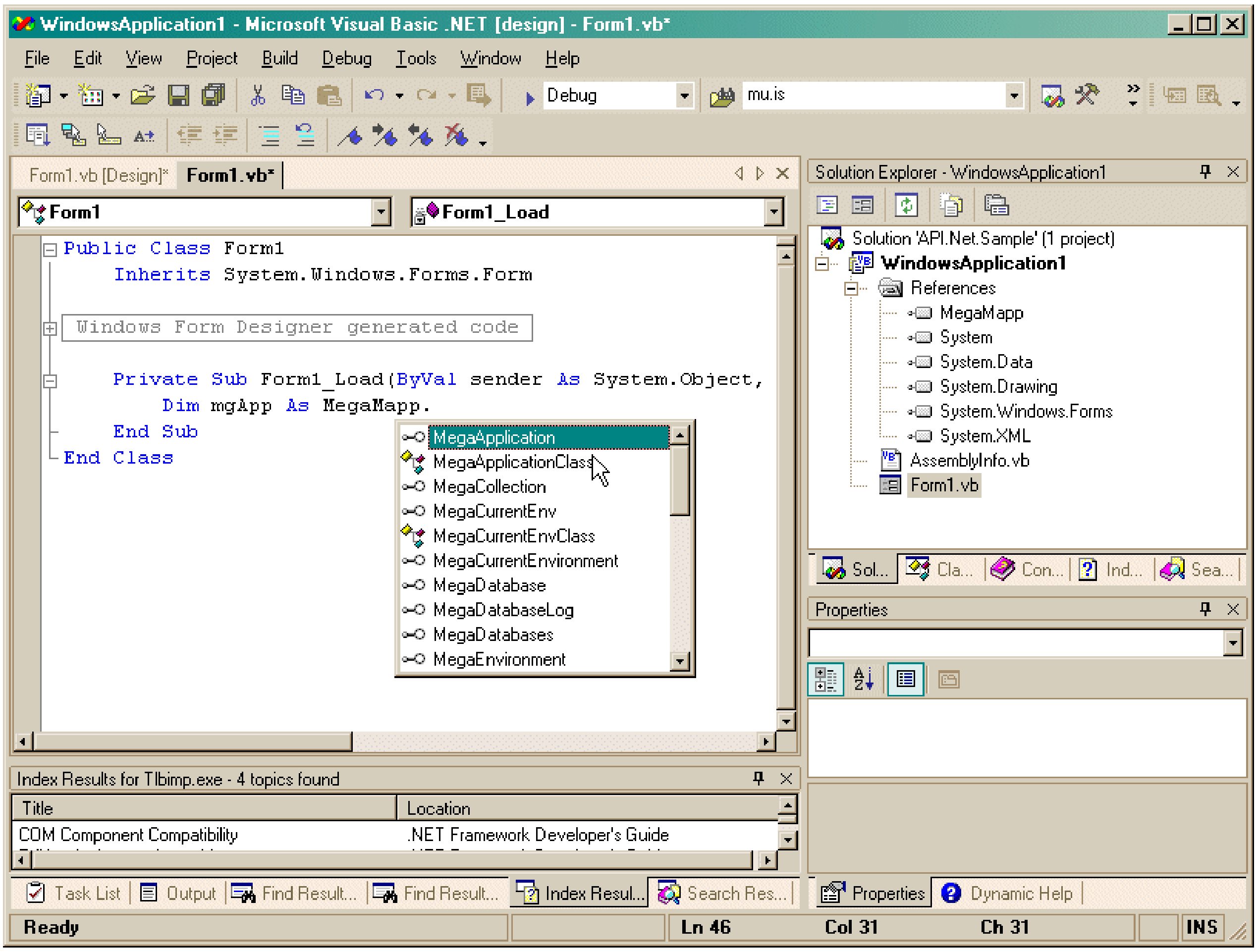
Selected Components:

Component Name	Type	Source
mg_mapp.dll	File	C:\Program Files\mega\System\...

When a reference has been created for the component, API objects are accessible. A namespace has been created automatically and is called MegaMapp.

To access API classes such as MegaApplication, MegaCollection, etc., prefix the class name with this namespace: **MegaMapp**.

Intellisense now handles display of available classes as shown in the following figure.



```
Public Class Form1
    Inherits System.Windows.Forms.Form

 Windows Form Designer generated code

    Private Sub Form1_Load(ByVal sender As System.Object,
        Dim mgApp As MegaMapp.
    End Sub
End Class
```

```
MegaApplication
MegaApplicationClass
MegaCollection
MegaCurrentEnv
MegaCurrentEnvClass
MegaCurrentEnvironment
MegaDatabase
MegaDatabaseLog
MegaDatabases
MegaEnvironment
```

Index Results for Tlbimp.exe - 4 topics found

Title	Location
COM Component Compatibility	.NET Framework Developer's Guide

COM component integration mechanism

Life cycle management of COM and .NET objects is not handled in the same way. For COM objects, the application handles reference creation (by QueryInterface::AddRef()) and reference deletion (by QueryInterface::Release()). In the case of a .NET object, it is the framework that handles freeing of objects.

To ensure total integration of the MEGA API component, Visual.NET automatically creates an intermediate.NET DLL. The role ofthis intermediate dll (interop.MegaMapp.dll) is to encapsulate each COM object exported and to handle its life cycle. From the user point of view, the operation is transparent and it can use objects of the API like any other .NET object.

Strong naming

To avoid implementation or compatibility problems caused by updating of certain DLLs, developments can be carried out using strong names. This system allows you to assign to each application object an assembly comprising this name and a unique key.

For more details, refer to **Strong-Named Assemblies** of MSDN.

For this type of development, all components inserted in the project must be signed with an identification key. Therefore, the encapsulation DLL generated by Visual.NET (interop.MegaMapp.dll) is not assigned to a key. It can not therefore be inserted in this type of project.

To alleviate this problem, the encapsulation DLL should be generated manually using the tlbimp.exe utility provided with the .NET framework.

It is this utility that is used to create the encapsulation dll automatically, but all options are not used.

On a command line, enter the sequence:

```
TLBIMP COMdll.dll /out:Netdll.dll /keyfile:keyfile.snk /
namespace:MEGA /asmversion:x.0.0.0
```

Where parameters are as follows:

- `COMDll.dll`: name of the COM DLL to be converted. It can be complemented by the complete name of the folder (use quotes if the name includes spaces).
- `Netdll.dll`: name of the encapsulation DLL
- `keyfile.snk`: name of the file containing the unique key assigned to the application. This file can be generated using the sn.exe utility provided with the .NET framework.
- `namespace: MEGA`: "MEGA" is the name of the encapsulation DLL namespace. In this case, the API objects will be accessible by prefixing class names with "MEGA"
- `asmversion`: defines the version number of the generated DLL.

9.1.2 Language characteristics

In theory, all .NET languages are interchangeable. Classes can be created in VB.NET and overload definitions in C# or C++.NET.

In practice, there are differences between each language.

Optional parameters

Languages like C# do not allow the use of optional parameters. However, this is possible in VB.NET.

The MEGA API has many functions of which parameters are optional. These are either options (MegaDatabase.HistoryReset()), or sort or assignment parameters

(MegaRoot.getSelection(), MegaObject.getCollection(), MegaCollection.Create()). Finally, to enable use of APIs in languages like C#, optional parameters can be indicated as empty character chains. In this case, it can be become complicated to provide the six optional parameters of functions Create() or getCollection(). In addition, it is recommended that an intermediate class be defined exporting underlying functions using different overloads (1 to 6 parameters if necessary).

Unexplained functions

MEGA APIs accessed from VBA or VBScript enable use of a certain number of functions undeclared in the type library. This is the case for functions derived from operators like MegaObject.Extract().

This mechanism is possible in VB6 or VBScript using the tardive (late binding) link and implementation by each COM object of the Dispatch() interface (see MSDN on this subject).

In VB.NET this mechanism is also possible using the Option Strict Off directive. However, highly typed languages (such as C# or Java) do not allow access to undeclared functions from an untyped variable. We must pass to COM objects by a reference.

9.2 VBA Application Example (Visual Basic for Applications)

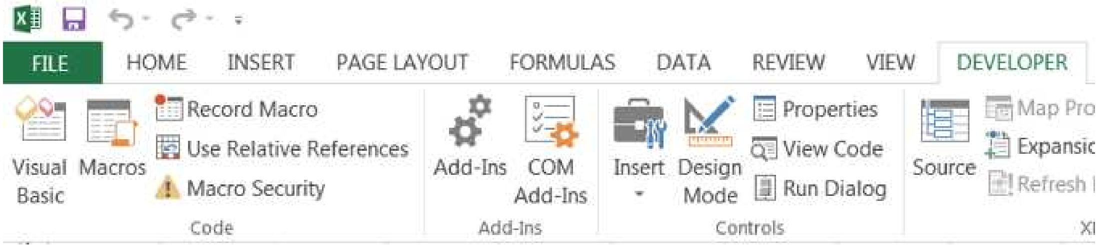
Example of integration between Microsoft Excel® and HOPEX

This example shows how to access HOPEX from Excel and retrieve data contained in a HOPEX repository that can be used by Excel.

In this case, we shall create a macro from Excel. This macro will access HOPEX, retrieve the names of procedures contained in the current repository and display them in an Excel spreadsheet.

To create a new macro in Excel:

1. From Excel menu bar, display the **Developer** menu (in Excel options > Customize Ribbon).
2. In the **Developer** menu select **Visual Basic.**

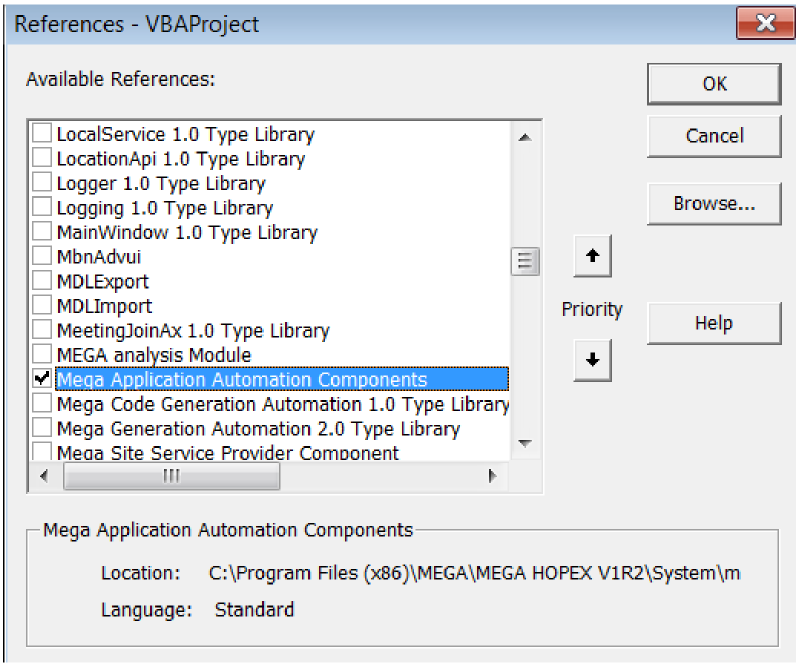


The Visual Basic editor opens.

To use MEGA APIs and benefit from Intellisense (display of names of classes, functions, parameters, etc.) create a reference for a **MEGA** component (DLL) in the VBA project.

To create a reference for this component:

1. In the Visual Basic editor menu bar, select **Tools > References**.
2. In the **VBAProject References** dialog box, select the **Mega Application Automation Components** component.


References - VBAProject
Available References:
☐ LocalService 1.0 Type Library
☐ LocationApi 1.0 Type Library
☐ Logger 1.0 Type Library
☐ Logging 1.0 Type Library
☐ MainWindow 1.0 Type Library
☐ MbnAdvui
☐ MDLExport
☐ MDLImport
☐ MeetingJoinAx 1.0 Type Library
☐ MEGA analysis Module
☑ Mega Application Automation Components
☐ Mega Code Generation Automation 1.0 Type Library
☐ Mega Generation Automation 2.0 Type Library
☐ Mega Site Service Provider Component
OK
Cancel
Browse...
Priority
Help

Mega Application Automation Components

Location: C:\Program Files (x86)\MEGA\MEGA HOPEX V1R2\System\m

Language: Standard

If the component does not appear in the list, click ***Browse*** *and add it from the "System" directory of the HOPEX installation folder (the DLL file to be added is called* ***mg_mapp.dll****).*

3. Enter the macro code in the editor:

```
Sub ImportProcedure()
Dim oMegaCurrentEnvironment As New MegaCurrentEnv
Dim oMegaRoot As MegaRoot
Set oMegaRoot = oMegaCurrentEnvironment.GetRoot
Dim oProcedure As MegaObject
Dim nRow As Integer nRow = 1
For Each oProcedure In oMegaRoot.GetCollection("Procedure")
Range("B" & nRow).FormulaR1C1 = oProcedure.GetProp("Name")
nRow = nRow + 1
Next
End Sub
```

Code description:

```
Sub ImportProcedure()
```

Contains the name of the code used.

```
Dim oMegaCurrentEnvironment As New MegaCurrentEnv
```

The oMegaCurrentEnvironment variable is declared and assigned an instance of the class MegaCurrentEnv corresponding to the current HOPEX environment.

```
Dim oMegaRoot As MegaRoot
```

The oMegaRoot variable of type MegaRoot is declared (corresponding to the root of a HOPEX repository).

```
Set oMegaRoot = oMegaCurrentEnvironment.GetRoot
```

The root of the repository open on the current environment is retrieved (oMegaCurrentEnvironment.GetRoot) and is assigned to the variable oMegaRoot.

```
Dim oProcedure As MegaObject
```

The oProcedure variable of type MegaObject is declared (corresponding to a MEGA object).

The remainder of the code retrieves the collection of procedures and displays the name of each procedure in a different cell of the same Excel spreadsheet column.

```
Dim nRow As Integer nRow = 1
```

variable nRow of integer type is declared.Integer nRow = 1

```
For Each oProcedure In oMegaRoot.GetCollection("Procedure")

Range("B" & nRow).FormulaR1C1 = oProcedure.GetProp("Name")

nRow = nRow + 1

Next

End Sub
```

For each procedure of the collection of repository procedures, the procedure name is inserted in a cell of column B, beginning with cell B1.

Counter nRow enables passage from one cell to another.

To execute this macro:

1. In the Visual Basic editor toolbar, click ▶.

   A message asks if you accept the process that will try to access the HOPEX repository.

2. Click **Yes**.

   The names of the repository procedures are displayed in the Excel spreadsheet.

The same principle is used to access HOPEX from other VBA compatible applications (Word or PowerPoint for example).

10 TOOLKIT

10.1 Metamodel

10.1.1 Accessing an attribute translation using APIs

To access an attribute translation using APIs, you need to use the MegaAttribute component.

To get the "Att" MegaAttribute component, enter:

VB Script

Either:

```
set myMAtt = myObject.GetProp("Att", "Object")
```

or:

```
set myMAtt = myObject.GetAttribute("Att")

MegaAttribute mgattMA = (MegaAttribute) myObject.getProp("Att",
"Object");
```

Java

```
MegaAttribute mgattMA = myObject.getAttribute("Att");
```

After, you can refer to the MegaAttribute interface.

Example:

VB Script

```
'MegaContext(Fields)

set myConcept = GetObjectfromID("~ezHXsMuE3fG0[Analisi]")

set myAtt = GetObjectfromID("~ezHXsMuE3fG0[Analisi]").GetAttribute("Name")

print "Current language: " & myAtt.DescriptionObject.GuiName & " : '" &
myAtt & "' or: " & myAtt.Value

print "In Spanish: " &
myAtt.~n97OO26RrO00[Espanol].DescriptionObject.GuiName & " : '" &
myAtt.~n97OO26RrO00[Espanol] & "' or: " &
myAtt.Translate("~n97OO26RrO00[Espanol]").Value

print "In GUI Language: " & myAtt.GuiLanguage.DescriptionObject.GuiName & "
: '" & myAtt.GuiLanguage & "' or: " & myAtt.Translate("GuiLanguage").Value

' in 2007 SP1:

for each transl in myAtt.DescriptionObject.Translations

set myTranslation = myAtt.Translate(transl.LanguageID)

print "Traduction in " & myTranslation.DescriptionObject.GuiName & " : " &
myTranslation.Value

next
```

Java

```
MegaObject mgobjConcept = mgRoot.getObjectFromID("~ezHXsMuE3fG0[Report]");

               MegaAttribute mgattAtt = mgobjConcept.getAttribute("Name");

               mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]", " Current
language: " + mgattAtt.getDescriptionObject().invokePropertyGet("GuiName") +
" : " + mgattAtt.getValue());
```

```
              mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]", "In Spanish: 
" + ((MegaAttribute)
mgattAtt.invokePropertyGet("~n97OO26RrO00[Espanol]")).getDescriptionObject()
.invokePropertyGet("GuiName") + " : " +
mgattAtt.translate("~n97OO26RrO00[Espanol]").getValue());

              mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]", "In GUI 
Language: " + ((MegaAttribute)
mgattAtt.invokePropertyGet("GuiLanguage")).getDescriptionObject().invokeProp
ertyGet("GuiName") + " : " + mgattAtt.translate("GuiLanguage").getValue());


              //in 2007 SP1:

              for (MegaObject mgobjTransl : (MegaCollection)
mgattAtt.getDescriptionObject().invokeFunction("Translations")) {

                MegaAttribute mgAttMyTranslation =
mgattAtt.translate(mgobjTransl.invokeFunction("LanguageID"));

                mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]",
"Translation in " +
mgAttMyTranslation.getDescriptionObject().invokePropertyGet("GuiName") + " :
" + mgAttMyTranslation.getValue());

              }
```

10.1.2 Accessing the metamodel description using APIs

Do not consult metamodel elements the same way you would do for any MEGA concept, i.e. by directly working with "MetaClass", "MetaAssociation", "MetaAttribute", "MetaAssociationEnd" MetaClasses and related MetaAssociations.

The code written that way is hardly ever efficient, and too dependent on the MEGA metamodel modeling style.

The modeling style is changing over time and takes on some of the implicit data – particularly the fact that conventional MetaAttributes are not necessarily associated with MetaClasses.

Note that the arrival of abstract MetaModel (with MEGA 2007 and even more with its evolutions in MEGA 2009) makes the use of MetaModel native data more and more complex.

To do this, a virtual abstraction layer has been developed, which uses in particular the compiled Metamodel, in this way enabling optimized and precise access to Metamodel concepts.

This layer does not exactly correspond to Metamodel concept implementations, and has been designed to simplify access to data – possibly calculated – which a developer may require to dynamically discover the Metamodel.

Technically, this abstraction layer can be used by APIs by means of MegaObjects and MegaCollections, therefore in exactly the same way as the native objects they represent.

So as to avoid confusion, we try to use a vocabulary for concepts handled in this abstraction layer different from that used in the native Metamodel. In addition, this vocabulary can reference that of APIs as required.

We shall first explain the model object of this abstraction layer. This model has two main entry points, which we shall name ClassDescription and CollectionDescription.

Objects of ClassDescription type enable object MetaClass discovery. Each MegaObject used in APIs makes available its class description by means of the GetClassObject function. A quick method of making contact with such a description and exploring it:

VB Script

```
myMegaObject.GetClassObject.Explore
```

Java

```
mgobjMegaObject.getClassObject().invokeMethod("Explore");
```

Objects of CollectionDescription type enable description of the interface of a MegaCollection. Each MegaCollection used in APIs makes available this description by means of the GetTypeObject function; we can explore such a description with the following script line:

VB Script

```
myMegaCollection.GetTypeObject.Explore
```

Java

```
mgcolMegaCollection.getTypeObject().invokeMethod("Explore");
```

Or:

VB Script

```
Set oRoot = object.GetRoot
oRoot.GetCollection("~KekPBSs3iS10[Diagram]").GetTypeObject.Explore
```

Java

```
mgRoot.getCollection("~KekPBSs3iS10[Diagram]").getTypeObject().invokeMethod("Explo
re");
```

Note that *GetTypeObject* function is also available on a *MegaObject*. In that case, the returned *CollectionDescription* usually corresponds to the description of the collection from which we got the MegaObject (it is not necessary true for collections built from heterogeneous objects).

From version 2007 SP1, we can directly access these descriptions without passing via an already existing MegaObject or MegaCollection, by means of the following functions available on the MegaRoot object:

VB Script

```
myObject.getRoot.GetClassDescription(ClassID)
myObject.getRoot.GetCollectionDescription(CollectionID)
```

Java

```
mgobjMegaObject.getRoot().getClassDescription(ClassID)
mgobjMegaObject.getRoot().getCollectionDescription(CollectionID)
```

ClassID represents here the absolute identifier of a MetaClass or MetaAssociation – this can be supplied in the form of a field. Regarding CollectionID, this can be the identifier of a MetaClass, MetaAssociation, MetaAssociationEnd or Selection.

In earlier versions, or in more generic code, you access the collection of ClassDescriptions of MetaClasses via collection:

VB Script

```
myObject.getRoot.GetCollection("~Ffs9P58kg1fC[ClassDescriptions]")
```

Java

```
mgobjMegaObject.getRoot().getCollection("~Ffs9P58kg1fC[ClassDescriptions]")
```

The following introduces the object Model corresponding to ClassDescriptions and CollectionsDescriptions.

ClassDescription:

Main properties:

- **`Name`**: name in current language
- **`GUIName`**: name in the user interface language

- `Rpbid`: absolute identifier in base64 (you can use GetID to get the identifier in internal format)
- `Level`: concept technical level, corresponding to MetaAttribute Technical Level
- `Visibility`: bit field defining the object visibility (`0x20000:Extension`)
- `Abstraction`: abstraction level (1 for abstract class, 0 for concrete class)
- `Location`: MetaClass location (`S:System D:Data L:External`)

Main collections:

- `Description`:

  List that includes only one item: the CollectionDescription corresponding to the ClassDescription

- `Pages`:

  Property page list defined on the object, including implicit pages (see PageDescription)

- `Groups`:

  Property group list defined on the object, including implicit groups (see GroupDescription)

- `UpperClasses`:

  UpperClass list, seen as *ClassDescription*

- `LowerClasses`:

  LowerClass list, seen as *ClassDescription*

- `CommandAccessor`:

  MetaCommand list explicitly defined on the class (see *CommandDescription*)

CollectionDescription:

Main properties:

- `Name`: name in current language
- `GUIName`: name in the user interface language
- `Rpbid`: absolute identifier in base64 (you can use GetID to get the identifier in internal format)
- `Level`: concept technical level, corresponding to MetaAttribute Technical Level
- `Visibility`: bit field defining the object visibility (`0x20000:Extension`)
- `Order`: order number
- `Major`: indicates that the collection corresponds to a major MetaAssociationEnd

- `Cardinal`: indicates the collection maximum multiplicity (1, U, or N)
- `LType`: absolute identifier of the link type of the collection
- `Permission`: update permission via the user interface for this collection objects
- `Mandatory`: indicates the collection minimum multiplicity (0 or 1)
- `PhysicalClassID`: collection native MetaClass identifier; this can be a MetaAssociationEnd, a MetaAssociation, a MetaClass or a Request, or can be not defined
- `Opposite`: opposite MetaAssociationEnd identifier
- `Association`: MetaAssociation identifier
- `TargetClassID`: source MetaClass identifier (of the collection)
- `SourceClassID`: target MetaClass identifier
- `Abstraction`: abstraction level (1 for abstract class, 0 for concrete class)
- `Location:` Association location (S:`System` D:`Data` L:`External`)
- `SourceTypeID:` source MetaClass identifier, abstract in the case of a generic association
- `TargetTypeID`: target MetaClassidentifier, abstract in the case of a generic association
- `AliasID`: collection alias identifier, if this exists
- `RootID`: generic association identifier if we are on an alias.

Main collections:

- `Properties`: list of the properties (see *PropertyDescription*)
- `Collections`: list of the collections, seen as CollectionDescription
- `ExternalRefs`: list containing, if it exists, the CollectionDescription corresponding to the standard link to external reference
- `Characters:` list containing, if it exists, the CollectionDescription corresponding to the standard link to keyword
- `ImageFormats:` list of image formats defined for this collection (see *ImageFormatDescription*)
- `Concretes`: list of collections corresponding to concrete classes accessible from a generic association
- `MainProperties`: list of the main properties (i.e. those that are not translations)

From MEGA 2007, functions are available on CollectionDescriptions:

VB Script

```
Function CollectionDescription.GetOperatorBehavior(OperatorId) As String
```

Java

```
Function CollectionDescription.callFunction("GetOperatorBehavior",
OperatorId) As Integer
```

This function enables to know the operator behavior given as a parameter according to the MetaAssociationEnd matching the collection.

The returned value corresponds to the Behavior: (65:Abort 83:Standard 76:Link 68:Deep)

VB Script

```
Function CollectionDescription.IsSuperClassOf(ClassId) As Boolean
```

Java

```
Function CollectionDescription.callFunction("IsSuperClassOf", ClassId) As
Boolean
```

VB Script

```
Function CollectionDescription.IsSubClassOf(ClassId) As Boolean
```

Java

```
Function CollectionDescription.callFunction("IsSubClassOf", ClassId) As
Boolean
```

VB Script

```
Function CollectionDescription.IsClassAvailable(ClassId) As Boolean
```

Java

```
Function CollectionDescription.callFunction("IsSubClassOf", ClassId) As
Boolean
```

These functions enable management of correspondence between concrete classes and abstract classes.

IsClassAvailable enables testing whether a corresponding class object can be inserted in the collection

VB Script

```
Function CollectionDescription.Specializations As MegaCollection
```

Java

```
Function CollectionDescription.callFunction("Specializations") As
MegaCollection
```

In MEGA 2007, enables listing of collections corresponding to concrete classes accessible from a generic association. Replaced in 2009 by the collection:

VB Script

```
CollectionDescription.Concretes
```

Java

```
Function CollectionDescription.invokePropertyGet("Concretes") As
MegaCollection
```

VB Script

```
Function CollectionDescription.SameFamily(CollectionId) As Boolean
```

Java

```
Function CollectionDescription.callFunction("SameFamily") As Boolean
```

True if both collections correspond to the same generic MetaAssociation.

Collections accessible from these two types of description lead us to define the following sub-descriptions:

PropertyDescription: description of a property: a property can be a MetaAttribute, TaggedValue or 'LegAttribute (MetaAssociationEnd seen as property)

Main properties:

- **Name**: name in the current language
- **GUIName**: name in the user interface language
- **Rpbid**: absolute identifier in base64 (use GetID to get the identifier in internal format)
- **Level**: concept technical level, which corresponds with the MetaAttribute Technical Level
- **Order**: order number
- **Abbreviation**: property abbreviation (short name)
- **Type:** basic type of the property
- **Format**: property format, corresponding to MetaAttribute Type attribute (X:String 9:Numerical 1:Boolean S:Short L:Long D:Date A:Text B:BinaryText Q:Binary H:64bits F:DoubleFloat)
- **Tabulated**: External format of the property, corresponding to the MetaAttribute Format attribute (S:Standard F:Enumeration T:EnumerationOpened D:Duration P:Percent E:Double O:Object Z:Signed)
- **Length**: property internal length
- **ASCIILength**: Length of property in its ASCII representation
- **ExternalLength**: Length of property in its external representation
- **Occurrence**: indicates that the property (in H Format) represents a MEGA object
- **TextFormat:** identifier of text format managing the property
- **Translatibility:** indicates that the property is translatable (0 or 1)
- **LCID**: identifier of language in which the property is restored
- **Index**: indicates that the property is an index (U:Unique S:UniqueCaseSensitive N:NonUnique)
- **FromLink**: indicates that the property comes from the MetaAssociationEnd (0 or 1)
- **Permission**: bit field characterizing the property UIPermission
- **Mandatory**: indicates that the property is mandatory (0 or 1)
- **Visibility**: bit field characterizing the object visibility (0x20000:Extension 0x1:OutOfList 0x4:Administration 0x40000:Localization 0x10000:NamePart 0x200000:HasDefault 0x400000:DefaultButton)

- **UpdateToolID:** binary attribute containing the identifier of the component managing update of the property. Indicates that the property is not updated by standard method
- **Substitution:** identifier of the attribute substituting the property for this MetaClass
- **OnCreation:** bit field indicating behavior of the property at object creation.

  This field is restored in the form of a character and we should test asc(value)

  (`0x1:DetailedBehavior 0x2:UpdatedOnCreation 0x10:Mandatory 0x20:UpdatableOnlyDuringCration`)
- (2007) **PhysicalClassID**: identifier of the native MetaClass motivating the property; this can be a MetaAttribute, TaggedValue, MetaAssociationEnd… or can be unspecified
- (2007) LanguageID: identifier of the language (when the property is a translation) (2007) **Heritability:** indicates that the property is heritable
- (2007) **DefaultValue:** property default value
- (2007) **RootID:** identifier of the root property, in the case of a translation

Main collections:

Values: lists defined tabulated values (see **AttributeValueDescription**)

AttributeClasses: when the property is an Occurrence, lists the potential MetaClasses of this occurrence (when defined) in the form of a ClassDescription

(2009) **Translations:** property translation list

AttributeValueDescription:

Describes a tabular value defined for a property

Main properties:

- **Name**: name in the current language
- **GUIName**: name in the user interface language
- **Rpbid**: absolute identifier in base64 (use GetID to get the identifier in internal format)
- **Level**: concept technical level, which corresponds with the MetaAttribute Technical Level
- **Order**: order number
- **Value**: ANSI value of the tabular value

Main collections:

- **ImageFormats** lists defined image formats for this tabulated value (see *ImageFormatDescription*)
- *PageDescription*: describes the property page; This page can be implicite, i.e. it does not correspond to a MEGA object (example: Administration page).

Main properties:

- **`Name`**: name in the current language
- **`GUIName`**: name in the user interface language
- **`Rpbid`**: absolute identifier in base64 (use GetID to get the identifier in internal format)
- **`Level`**: concept technical level, which corresponds with the MetaAttribute Technical Level
- **`Order`**: order number
- **`LType`**: identifier of the page type (defines the tab in which it is supposed to appear)
- **`Group:`** identifier of the group that motivates the page
- **`Guid`**: identifier of the macro that implements the page
- (2007) **`Heritability`**: indicates the page heritability (0 or 1)

GroupDescription

Description of a property group. If the group is implicit, it does not correspond to a MEGA object. Main properties:

- **`Name`**: name in the current language
- **`GUIName`**: name in the user interface language
- **`Rpbid`**: absolute identifier in base64 (use GetID to get the identifier in internal format)
- **`Level`**: concept technical level, which corresponds with the MetaAttribute Technical Level
- **`Order`**: order number
- **`LType`**: identifier of the group type (defines the tab in which it is supposed to appear)
- (2007) **`Heritability`**: indicates the group heritability (0 or 1)

Main collections:

- **`Properties`**: group property list (see PropertyDescription)
- **`Pages`**: group associated page (see PageDescription)

ImageFormatDescription

Main properties:

- **`Name`**: name in the current language
- **`GUIName`**: name in the user interface language

- `Rpbid`: absolute identifier in base64 (use GetID to get the identifier in internal format)
- `Level`: concept technical level, which corresponds with the MetaAttribute Technical Level
- `Abbreviation:` standard file extension
- `DefaultUsed:` indicates that ImageFormat is not explicitly defined and that the file used is the default file.

(2007) CommandDescription

Main properties:

- `Name`: name in the current language
- `GUIName`: name in the user interface language
- `Rpbid`: absolute identifier in base64 (use GetID to get the identifier in internal format)
- `Level`: concept technical level, which corresponds with the MetaAttribute Technical Level
- `Order`: order number
- `Heritability`: indicates the commandaccessor heritability (0 or 1)

Comments regarding ClassDescription, XXXDescription, "Name" and other properties

⚠ **XXXDescriptions are MegaObjects and not MetaClasses.**

You cannot use usual MetaAttribute identifiers with these objects.

For example, a search on a MetaClass name returns an error:

VB Script

```
Print
mgobjMyObject.getRoot.getclassdescription("~gsUiU9B5iiR0[Organizational
Process]").getprop("~210000000900[Name]")
```

Java

```
mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]",
mgobjMegaObject.getRoot().getClassDescription("~gsUiU9B5iiR0[Organizational
Process]").getProp("~210000000900[Name]"));
```

How to do this?

Find the identifier of the "Name" property (different from the "Name" MetaAttribute) of MegaObject ClassDescription.

To do this, examine the ClassDescription object type:

VB Script

```
mgobjMyObject.getRoot.GetClassDescription("~gsUiU9B5iiR0[Organizational
Process]").GetTypeObject().Explore
```

Java

```
mgobjMegaObject.getRoot().getClassDescription("~gsUiU9B5iiR0[Organizational
Process]").getTypeObject().invokeMethod("Explore");
```

You can therefore find the "Name" property and copy/paste in the form of a MegaField.

In this way you can use the ClassDescription:

VB Script

```
print
mgobjMyObject.getRoot.getclassdescription("~gsUiU9B5iiR0[Organizational
Process]").getprop("~oKdcP5epmcfC[Name]")
```

Java

```
mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]",
mgobjMegaObject.getRoot().getClassDescription("~gsUiU9B5iiR0[Organizational
Process]").getProp("~oKdcP5epmcfC[Name]"));
```

Note above that the « Name » property identifier is different from the « Name » MetaAttribute one.

10.2 Property Pages

10.2.1 Accessing the description of an object Property Pages

This component obtains a description by code of properties pages of an object.

Retrieving the list of pages and tabs of an object

The propertiesdialog method of a MegaObject can now be used as a function and returns a properties page description component.

```
Set PageDescription = mgObject.PropertiesDialog("Description")
```

This component is a page enumerator, and for this purpose it includes Count and Item standard methods. It returns MegaPropertyPage objects.

These objects are identified by the CLSID of the page, enabling their query in the enumerator.

All visible object pages are accessible via this enumerator.

The script below enables simple test of component on the first MetaClass:

VB Script

```
set ppcol =
MetaClass.item(1).propertiesdialog("Description")

print ppcol.count

for each ppi in ppcol

print ppi.getID & ppi.parentID & ppi.level & " : " &
ppi.name

if ppi.level > 1 then print " Parent Is " &
ppcol.item(ppi.parentID).name

set cnt = ppi.Component

if not cnt is nothing then

cnt.Content.Explore

end if

next
```

Java

```
ComObjectProxy mgcomPpcol = (ComObjectProxy)
mgRoot.getCollection("~P20000000c10[MetaClass]").get(1).invokeFu
nction("propertiesdialog", "Description");

int iSize = (Integer) mgcomPpcol.invokeFunction("count");

mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]", iSize);

for (int j = 1; j <= iSize; j++) {

                MegaPropertyPage mgobjPpi = new
MegaPropertyPage((MegaCOMObject)
mgcomPpcol.invokeFunction("item", j));

                mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]",
mgobjPpi.getID() + " " + mgobjPpi.parentID() + " " +
mgobjPpi.level() + " : " + mgobjPpi.name());

                if (mgobjPpi.level() > 1) {
                  ComObjectProxy mgC = (ComObjectProxy)
mgcomPpcol.invokeFunction("item", mgobjPpi.parentID());
                  String strA = (String)
mgC.invokeFunction("name");

mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]", "Parent Is " +
strA);
                }
                MegaPropertyPageComponent cnt =
mgobjPpi.component();
                if (cnt != null) {
                  cnt.content().invokeMethod("Explore");
                }
              }
```

Object Interface

`MegaPropertyPage.GetID as String`

CLSID of the page, in form {00000000-0000-0000-0000-000000000000}

`MegaPropertyPage.Name as String`

Page name

`MegaPropertyPage.Order as Long`

Page order number

`MegaPropertyPage.Default as Boolean`

Indicates if the page is the default page (visible at first activation)

`MegaPropertyPage.ParentID as String`

CLSID of the parent page if current page is a subtab

`MegaPropertyPage.Level as Long`

Page subtab level: 1 indicates main page, 2 page presented in subtab, …

`MegaPropertyPage.IsTab as Boolean`

Indicate if page is itself a tabbed box. In this case it is the parent of all pages it contains

`MegaPropertyPage.Component as MegaObject`

Page description, if supported by the component implementing the page. If not, is Nothing

`nMegaPropertyPage.TypeID`

Identifier of the page type

`MegaPropertyPage.SourceID`

Identifier of the page source that can be:

- Identifier of underlying MetaPropertyPage when this exists
- Identifier of MetaAttributeGroup when this is not associated with a page – if the page comes from a MetaAttributeGroup
- Identifier of page type (for Tabs containing pages, and for generic pages)
- Another identifier in other cases

Page component

Pages implementing it (currently only standard pages) make available an auto-description accessible by the method. Component of the MetaPropertyPage object.

This object is an explorable standard MegaObject (as the script example shows).

It has the following attributes:

`Name as String ~oKdcP5epmcfC[Name]:` element title

`Nature as String ~VxJRO58xtMuC[Nature]:` Control nature, conforming to configuration syntax:

```
"ListView"

"TreeView"

"Static"

"Edit"

"Text"

"ActiveX"

"HelpComment"

"Schedule"

"Button"

“RadioButtons”

“CheckButtons”

“EditButton”

“Password”

“Viewer”

“SubPage”

"RemotingDropDownListMenu"

“DropDownSelection”

“MatrixView”

"ComboLinks"

“Image”

“Label”

“MetaTree”

“Matrix”

“ExtJsComponent”

“ImageList”

“MultiComboBox”
```

“Toolbar”

“Report”

“Diagram”

“Card”

"MegaEditCheck"

"ComboBox"

"DropDownList"

"CheckBox"

"3StateCheckBox”

"ComboBoxMenu"

"DropDownListMenu"

"StaticMenu"

"EditMenu"

“ComboBitmaps”

“DynamicCombo”

“DatePicker”

“ComboLinks”

Order as Long ~5gs9P58ye1fC[Order]

SourceID as Variant ~Dgs9P58(e1fC[SourceID]: characterizes the property displayed by the control. Usually, the absolute identifier of a MetaAttribute.

Style as Long ~)0nfa1WYhchD[Style]: bit field characterizing the control

```
#define ACWISTYLE_NOTITLE 0x0010000

#define ACWISTYLE_TITLEUP 0x0020000

#define ACWISTYLE_DEFAULTED 0x0040000

#define ACWISTYLE_BOTTOMALL 0x0080000

#define ACWISTYLE_BORDERED 0x0100000

#define ACWISTYLE_EXCLUDED 0x0200000

#define ACWISTYLE_DISABLED 0x0400000

#define ACWISTYLE_HIDDEN 0x0800000

#define ACWISTYLE_DYNAMIC 0x1000000

#define ACWISTYLE_REMOTING 0x2000000

#define ACWISTYLE_CLIPLEFT 0x002

#define ACWISTYLE_CLIPRIGHT 0x004
```

```
#define ACWISTYLE_CLIPTOP 0x008

#define ACWISTYLE_CLIPBOTTOM 0x010

#define ACWISTYLE_CLIPMLEFT 0x020

#define ACWISTYLE_CLIPMRIGHT 0x040

#define ACWISTYLE_CLIPMBOTTOM 0x080

#define ACWISTYLE_CLIPMTOP 0x100

#define ACWISTYLE_NOVCLIP 0x1000

#define ACWISTYLE_NOHCLIP 0x2000
```

Group as String ~0es9P5eQf1fC[Group]: name of the Group in which the control has been placed

Options as String ~T2zVa10))bZD[Options]: options defined for the control in configuration

Read Only as Boolean ~3es9P5ORf1fC[Read Only]

Width as Long ~ths9P5OOf1fC[Width]: control width (in dialog Units)

Height as Long ~uhs9P5eOf1fC[Height]: control height (in dialog units)

Left as Long ~whs9P58Pf1fC[Left]: if specified, control left margin (in dialog units)

Top as Long ~)hs9P5OQf1fC[Top]: if specified, position of top of control relative to group (in dialog units)

Kind as Short ~a2zVa1W00cZD[Kind]: precise nature of control

ControlID as Binary ~v0nfa10XhchD[ControlID]: CLSID of component implementing the control

MapID as Variant ~PDfkighQx400[MapID]: absolute identifier of MAP of control (see Parameterization)

LinkID as 64Bits ~V20000000z50[LinkID]: if the control is obtained from a link, absolute identifier of the MetaAssociationEnd.

ObjectID as 64Bits ~qM44Q5OUd4xC[ObjectID]: absolute identifier of the object of which the control displays a property

ID as Variant ~jKdcP5OomcfC[ID]: Control internal identifier

Child controls of a control are accessible by the Child Collection (« `AttributeControl` ») **~7fs9P58ig1fC[AttributeControl]**

10.3 Accessing MegaObject menus using APIs

The menu of a MegaObject or object collection is accessible via the **MegaCommandManager** component.

The **CommandManager** operator enables access to this component, from a MegaObject or MegaCollection (therefore from a MegaItem).

This component describes the standard menu of the object or collection, as it appears for example in the explorer (the collection menu appears when you click a folder)

Methods of this component are:

```
MegaCommandManager.Object As MegaItem
```

Returns the MegaObject or MegaCollection on which the CommandManager has been invoked

```
Sub MegaCommandManger.TrackPopup(Optional Line As Integer,Optional Column As Integer,Optional Options As String)
```

```
Function MegaCommandManger.TrackPopup(Optional Line As Integer,Optional Column As Integer,Optional Options As String) as Long
```

Displays the object pop-up menu. This method can therefore be called only in interactive mode.

Line and Column indicate the absolute position on the screen of the top left-hand corner of the menu. If they are not specified, the position of the mouse cursor at the time of call is used.

Two options can appear in Option (any separator):

`"NoProperties":` the “Properties” command is not added to the menu

`"ShowOnly":` the menu is displayed but the possibly selected command is not called

When the function is called, the index of selected command is returned, or 0 if no command was selected, or -1 if the "Properties" command was selected.

```
Sub MegaCommandManager.Invoke(CommandID As Variant)
```

Execute the command.

CommandID can be the name of the command or its index. If several commands have the same name, the first one is invoked.

```
Sub MegaCommandManager.InvokeStandard(CommandID As Variant)
```

From 2007 release

This method allows invocation of a standard menu command. Managed commands are:

- `"Copy"`: "Copy" command
- `"Destroy"`: "Destroy" command
- `"Unlink"`: "Unlink" command
- `"Explore"`: "Explore" command

You can invoke the following generic commands on objects:

- `"Open":` default command (open a menu if there are several commands)
- `"AddToFavorites":` "add to favorites" command

You can invoke the following commands on collections:

- `"Paste":` "Paste" command
- `"Create":` "Create" command in in-place mode
- `"QueryCreate":` "Create" command in interactive mode
- `"Link":` "Link" command
- `"ReOrder":` "Reorder" command

This method can be used with or without a return parameter.

If there is a return parameter (here myResult), it is 'True' if the command exists for the CommandManager and has therefore been launched. Otherwise, the command does not exist in this context.

If there is no return parameter and the command does not exist, the method triggers an error.

You can explicitly access to the list of commands that are defined in the Menu:

`MegaCommandManager.Commands As MegaCollection`

Returns the exhaustive list of object commands, including deactivated and invisible commands, and explicit sub-menus.

MegaObjects contained in the resulting collection include the commands – These are not MEGA occurrences.

Their properties are as follows:

`CommandItem.Name As String:`

Command name (in the language of the current environment or site)

`CommandItem.Index As Long:`

`Command internal number`. This number is not stable and should not be used with an instance of CommandManager other than that with which the CommandItem was obtained.

`CommandItem.Style As Long:`

Bit field characterizing the command style and visibility. Significant bits are:

- for the styles:

`POPUP 0x001 (1)` Indicates that the command is a pop-up menu
`CHECKBOX 0x002 (2)` Indicates that the command is a check box: the corresponding menu element can appear with a check mark
`RADIOBUTTON 0x08 (8)` Indicates that the command is a radio button: the corresponding menu element can appear with an exclusive check mark: a single element can be marked among the RadioButtons of the same group (see CommandItem.Group below)

- for the status:

`CHECKED 0x100 (256)` Indicates that the element (CheckBox or RadioButton style) is marked in the context of the occurrence.

`DISABLED 0x400 (1024)` Indicates that the element is grayed and cannot be active in the context of the occurrence.

`CommandItem.Category As Long:`

Bit field specifying category of the element as defined in the corresponding CommandAccessor. Significant bits are:

`DESCRIPTION 0x0010` Object description data entry commands (Open, Zoom, Flowchart,...)
`ENTRY 0x0020` Object characteristics data entry commands (Attributes, Operations, Navigability->, Cardinality->)
`ACTION 0x0004` Object action/activation commands (Generate, Derive, Prototype, Quantify...)
`DOCUMENTATION 0x0040` Object Publication/Documentation commands (Document, Reference,...)
`DISPLAY 0x0002` Object presentation commands (View, Format, Drawing)
CONTAINER 0x0080 Object commands in container (Cut, Copy, Disconnect)
ADMIN 0x0008 Object administration commands (Explore, Delete, Compare,...)
ADMINEX 0x0400 Commands Manage >

`LINK 0x0200` Commands Connect >
`NEW 0x0100` Commands New >

`EXPORT 0x0800` Tool/Export menus in desktop

`INPUT 0x0001` Enumerator commands

`REVERSE 0x1000` Tool/Reverse menu commands in desktop
`LOCALISATION 0x2000` Menu commands relating to languages – managed in desktop and diagram

`STANDARDCMD 0x10000` Standard commands

`STANDARDOPEN 0x20000` Opening command

`NOENUMLISTVIEW 0x100000 Filtre:` Commands not available outside ListViews
`ENUMLISTVIEW 0x200000 Filtre:` Commands specific to ListViews

`CommandItem.Order As Long:` Command order number. This number is used to sort commands of the same category when displaying the menu.
`CommandItem.Group As Long:` Indicates command group. The value returned is 0, except for elements of RadioButton type and elements included in an explicit sub-menu (that is not derived from a category). In this case the Group value corresponds to the index of the pop-up element under which the commands appear.

10.4 Getting the person or person group used for current session

To retrieve the person name and/or person group name connected to the current session, you can use:

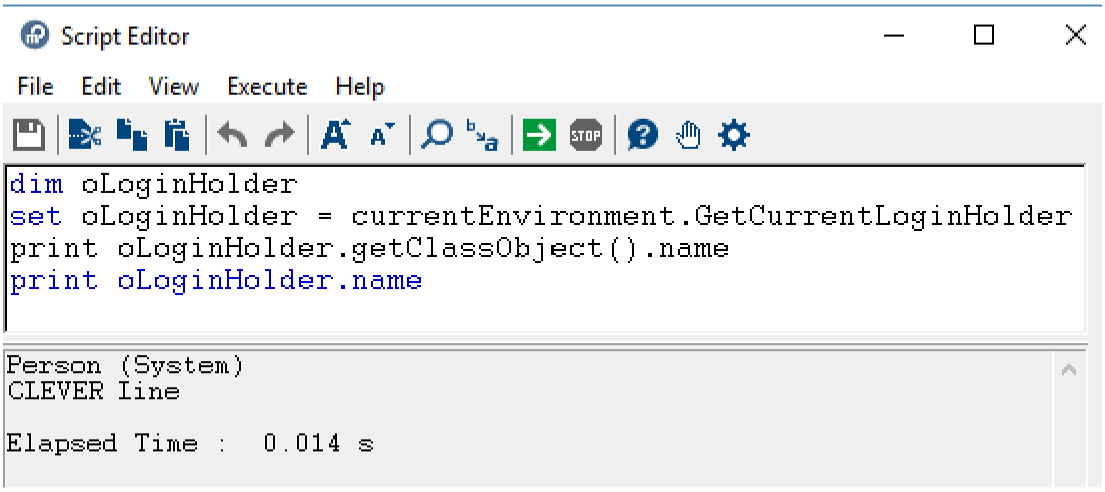
- **GetCurrentLoginHolder**
- **GetCurrentUserId**

A login holder can be a person or a person group. You can use the **GetCurrentLoginHolder** API on currentEnvironment object to retrieve the person or person group used for the current session.

Example:

VB Script

```
dim oLoginHolder
set oLoginHolder = currentEnvironment.GetCurrentLoginHolder
print oLoginHolder.getClassObject().name
print oLoginHolder.name
```



```
dim oLoginHolder
set oLoginHolder = currentEnvironment.GetCurrentLoginHolder
print oLoginHolder.getClassObject().name
print oLoginHolder.name
```

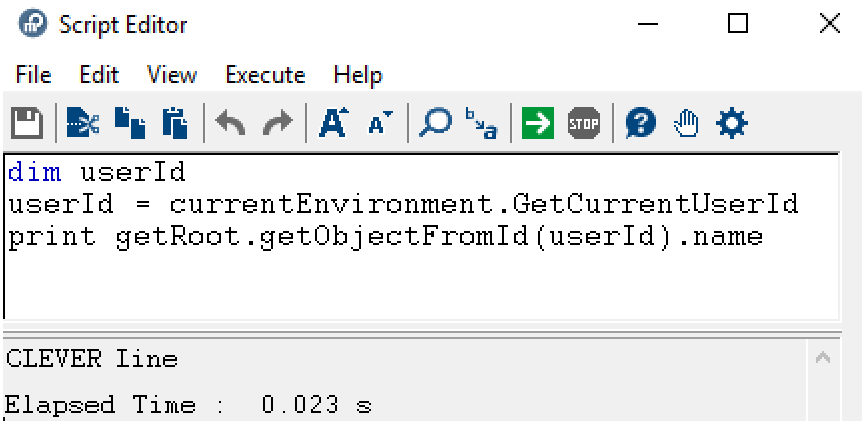
```
Person (System)
CLEVER line

Elapsed Time :  0.014 s
```

When the person is connected to HOPEX via a person group, the **GetCurrentLoginHolder** API returns the person group name. To retrieve the person connected, use the **GetCurrentUserId** API, which returns the IdAbs of the person connected via the person group.

Example:

VB Script

```
dim userId
userId = currentEnvironment.GetCurrentUserId
print getRoot.getObjectFromId(userId).name
```



```
dim userId
userId = currentEnvironment.GetCurrentUserId
print getRoot.getObjectFromId(userId).name
```

```
CLEVER line
Elapsed Time :  0.023 s
```

10.5 Getting the current snapshot date

Use **getCurrentSnapshotDate** to retrieve the date corresponding to the current repository snapshot.

When a repository snapshot has been opened, the function returns:

- a date with internal value format
- 0 otherwise (the session is not open on a snapshot).

  You can customize the code so that it returns a text, for example “you are not in a snapshot”, instead of 0

VB Script

```
Dim oRoot
Set oRoot=GetRoot
Dim date
date = oRoot.currentEnvironment.getCurrentSnapshotDate
if DateDiff("yyyy",0,date) then
     print date
else
     print "you are not in a snapshot"
end if
```

Java

```
Date currentSnapshotDate=
MegaRoot().currentEnvironment().getCurrentSnapshotDate()
public Date getCurrentSnapshotDate();
```

10.6 Getting the default e-mail address

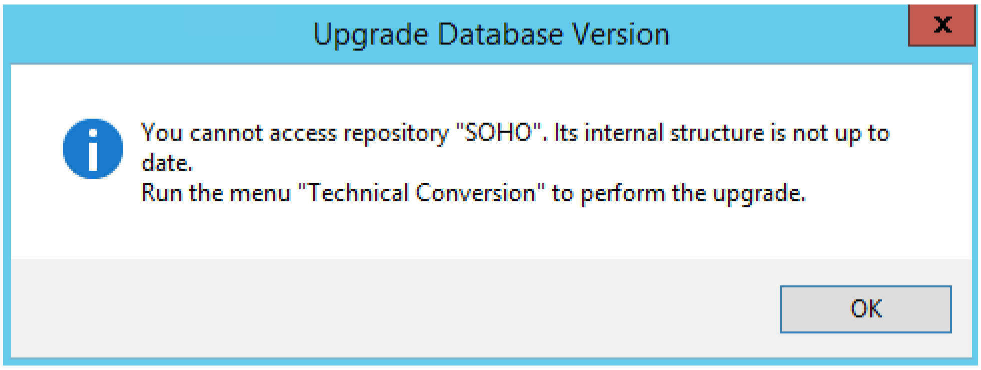
Once the installation options regarding Web applications and SMTP configuration are defined (**Installation > Electronic mail**) to retrieve the email address:

Java

```
 myroot.currentEnvironment.getUserOption("Mail","SMTPDefaultSendingAddres
s");
```

10.7 Triggering technical conversions

To access a repository its technical data must be up-to-date.



You cannot access repository "SOHO". Its internal structure is not up to date.
Run the menu "Technical Conversion" to perform the upgrade.

HOPEX Administration enables you to perform Technical conversions needed for example at migration or upgrade. With HOPEX Administration you can launch the technical conversions on a single environment at a time.

If you have many environments, you can use the **TechnicalConversionProcess** method on the MegaEnvironment object to trigger the technical conversions on all the data repositories and systemdb repositories of your environments.

Example:

VB Script

```
Set oMegaApp = CreateObject("Mega.Application")

Set oEnvironment=oMegaApp.Environments.item(1)

MsgBox "Installation: " & oMegaApp.Path & " - Environment: " & vbcrlf &
oEnvironment.Path

oEnvironment.TechnicalConversionProcess()

Set oMegaApp = Nothing
```

10.8 Managing a semaphore

To create an interprocess semaphore on the current Environment, use:

```
CreateSemaphore(semId, semName, mode)
```

Parameter description:

semId: idabs of the semaphore (hexaidabs format, ~cn64 format, or megafield).

semName: name of the semaphore.
Enter an appropriate name, it can be usefull for debugging.

mode: "try" (default value) or "wait" in order to stop processing until the semaphore is created.

It returns true if the semaphore has been successfully created.

```
public boolean createSemaphore(final String semId, final String semName, final String
mode);
```

To destroy the interprocess semaphore on the current Environment, use:

```
DestroySemaphore(semId)
```

Parameter description:

semId: idabs of the semaphore.

```
public void destroySemaphore(final String semId);
```

VB Script

```
if getRoot.CurrentEnvironment().CreateSemaphore("~RSqA9jIt0100", "sem1", "try")
then
`enter your protected code
   getRoot.CurrentEnvironment().DestroySemaphore("~RSqA9jIt0100")
end if
```

Java

```
if (myRoot.currentEnvironment().createSemaphore("~RSqA9jIt0100", "sem1", "try"))
{
       // enter your protected code
      myRoot.currentEnvironment().destroySemaphore("~RSqA9jIt0100");
}
```

10.9 MEGA TextStream an alternative string concatenation

This component is based on the Microsoft ITextStream interface, used in particular by the FileSystemObject component.

- MEGA *TextStreams* can be opened in writing access or reading access; mixed mode however is not supported.
- MEGA *TextStreams* can correspond to character strings: it is possible to instance a *TextStream* in reading access using a character string, and to instance a *TextSteam* in writing access, which will produce a character string output.

- MEGA *TextStreams* offer the possibility of reading or writing files, natively offering UTF8 and ANSI conversion (writing in UNICODE being possible).

Files opened in read-only can use the following functions of the ITextStream interface:

```
TextStream.AtEndOfStream As Boolean
```

indicates that the reader has reached the end of the stream.

```
TextStream.AtEndOfLine As Boolean
```

indicates that the reader is positioned at the end of a line.

```
TextStream.Read(nbChar As Integer) As String
```

reading the number of requested characters in the stream. If line end markers are encountered during reading, they are not transformed and are therefore returned as-is.

```
TextStream.ReadLine As String
```

reading stream to the next line (character CR) OR end of file. Line break characters do not appear in the returned string.

```
TextStream.ReadAll As String
```

reading integrality of stream in a string; not to be used if the file is potentially large…

```
TextStream.Skip(nbChar As Integer)
```

skip the number of requested characters in the stream and continue reading

```
TextStream.SkipLine As String
```

read forward in stream to the next line (character CR) or end of file.

```
TextStream.Line As Integer
```

number of lines read (including line to read: value cannot be less than 1)

```
TextStream.Column As Integer
```

number of columns read in the last line (including column to read: value cannot be less than 1)

Files opened in read-only can use the following functions of the ITextStream interface:

```
TextStream.Write(Text As String)
TextStream.Write(Text As String) As Integer 'Script Only
```

writing string in stream. Use in the form of function does not generate an exception in the case of failure when writing, but returns a not null value in the case of writing failure:

```
TextStream.WriteLine(Text As String)
TextStream.WriteLine(Text As String) As Integer 'Script Only
```

this value corresponds to the OLE (HRESULT) code corresponding to the error. Note that this form is not defined in the ITextStream interface and is therefore reserved for Script or Late-Binded VB use.

writing the character string in the stream, and adding a line end marker. Use in the form of function does not generate an exception in the case of failure when writing, but returns a not null value in the case of writing failure: this value corresponds to the OLE (HRESULT) code corresponding to the error.

```
TextStream.WriteBlankLines(nbLines As Integer)
```

writing requested line end marker number.

```
TextStream.Line As Integer
```

number of lines written (first being number 1).

```
TextStream.Line As Integer
```

number of columns of last line written. Systematically equals 1 in the case of use of WriteLine or WriteBlanlLines.

For all TextStreams:

```
TextStream.Close
TextStream.Close As Integer 'Script Only
```

The Close function can be used only if the file has been explicitly opened by the component (see function Open described below). Use in the form of function does not generate an exception in the case of failure when writing, but returns a not null value in the case of writing failure: this value corresponds to the OLE (HRESULT) code corresponding to the error.

TextStream creation and opening:

In Script you can create an explicit TextStream with CreateMegaObject function:

```
Set myStream = MegaToolkit.CreateMegaObject("MegaTextStream")
```

Then open this TextStream according to its usage.

Opening a Stream corresponding to a file:

```
Sub MegaTextStream.Open(
Source As Object,
Optional Mode As String,
Optional Format As String)

Function MegaTextStream.Open(
Source As String,
Optional Mode As String,
Optional Format As String) As String
```

`Source` corresponds to the file name. It can be ignored in specific opening modes detailed above.

`Mode`: can be **"Write"**, **"Write,Create"** or **"Read"**.

In **"Write,Create"** mode, file creation is possible, else the file must correspond to another existing file.

The default mode is **"Read"** mode.

For files opened in writing access mode, the following keyword can be added: "**NoLineFeed**", indicating that the line end sequence is represented by a CR character alone, not followed by LF when it is generated by functions **WriteLine** or **WriteBlankLines**.

For files open in creation mode, you can add the following key words:

"Temporary": indicates that the file should be created with a "temporary" tag.

"Archive", indicating that the file should be created with the "archive" tag **"WithBOM"**, indicating that format definition characters (UTF8 or UNICODE) should be written in the file header.

Format: by default, format is undefined for reading access files (the reader considers that the format is ANSI, or UNICODE in obvious cases), and ANSI in the current page code for writing access files. It is possible to specify format: In this case, we should also include the mode, the format necessarily being the third parameter of the method.

Format corresponds to the numerical value of LCID in the case of an ANSI file; negative numerical values defined for formats unicode (-1), binary (-3) and utf8 (-4) can also be used, as well as the following keywords: "UNICODE" "BINARY" "UTF8"

When **Open** is used in the form of a function, it does not produce an exception in the event of failure at file opening: The character string returned is empty if opening is successful; in the case of failure, the character string indicates the reason for the error.

Opening a stream corresponding to a character string in reading access:

This operation invokes a method using a *TextStream* as entry system when we have a character string available. In this case, we should not initialize the *TextStream* with the **Open** method, but with the **MapString** method.

```
MegaTextStream.MapString(Input As String)
```

This function also enables reading MEGA text in the form of a TextStream, and therefore to be able to read a text line-to-line.

VB Script

```
myTextStream.MapString myObject.GetProp("Comment", "Display")

Do While Not myTextStream.AtEndOfStream

Print "Line " & myTextStream.Line & " : " & myTextStream.ReadLine

Loop
```

Java

```
MegaCOMObject mgobjMyTextStream = (MegaCOMObject)
mgobjMegaObject.getRoot().currentEnvironment().toolkit().createMegaObject("Me
gaTextStream");

mgobjMyTextStream.invokeMethod("MapString",
mgobjMegaObject.getProp("Comment", "Display"));

while ((Boolean) mgobjMyTextStream.invokePropertyGet("AtEndOfStream") ==
false) {

mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]", "Line " +
mgobjMyTextStream.invokePropertyGet("line") + " : " +
mgobjMyTextStream.invokePropertyGet("ReadLine"));

}
```

Opening a stream corresponding to a character string in writing access:

This function enables creation of a stream managed in live memory and, after writing processing, recovery of its content in the form of a character string.

To do this, we specifically use the **Open** method in the following way:

```
myTextStream.Open 0, "BSTRMODE"
```

The stream is also opened in writing access mode: call on **Write**XXX functions enables optimized concatenation of the value written in an allocated character string.

On completion of the writing operation, we can recover the written string using the **GetString** function.

```
myOutput = myTextStream.GetString
```

This function does not duplicate the string; the string used in the stream is transmitted as-is to the variable and the stream is reinitialized with an empty string, in which we can again write.

This function is particularly optimized and adapted to a generation. For example, consider the two following examples:

VB Script

```
Function TestBSTRNative
TestBSTRNative = ""
For i = 1 To 10000
TestBSTRNative = TestBSTRNative & " Writing of the line number " & i & VbCRLF
Next
End Function

Function TestBSTRStream
Set myTextStream =
CurrentEnvironment.Toolkit.CreateMegaObject("MegaTextStream")
myTextStream.Open 0, "BSTRMODE"
For i = 1 To 10000
myTextStream.WriteLine " Writing of the line number " & i
Next
TestBSTRStream = myTextStream.GetString

End Function
```

Java

```
public String TestBSTRNative() {
    String strTestBSTRNative = "";
    for (int i = 1; i <= 10000; i++) {
    strTestBSTRNative = strTestBSTRNative + " Writing of the line number " + i
+ "\n";
    }
    return strTestBSTRNative;
  }


  public String TestBSTRStream(final MegaRoot mgRoot) {
    MegaCOMObject mgobjMyTextStream = (MegaCOMObject)
mgRoot.currentEnvironment().toolkit().createMegaObject("MegaTextStream");
    mgobjMyTextStream.invokeMethod("Open", 0, "BSTRMODE");
    for (int i = 1; i <= 10000; i++) {
      mgobjMyTextStream.invokeMethod("WriteLine", " Writing of the line number
" + i);
    }
    return (String) mgobjMyTextStream.invokeFunction("GetString");
  }
```

The TestBSTRStream function, while restoring an identical text, is much faster (x50 on a developer machine).

Opening a stream on a volatile temporary file.

It is possible to create a serializedstream on a temporary file, automatically destroyed at freeing of the component.

This system enables script processing to handle generations that risk exceeding machine memory capabilities.

Such a stream is used in two steps:

First, we open it in writing access mode; we can then generate data. To do this, we use the **Open** method in a specific way:

VB Script

```
myTextStream.Open "", "Write,Create,Temporary,DeleteOnClose","UNICODE"
```

Java

```
mgobjMyTextFile.invokeMethod("Open", "",
"Write,Create,Temporary,DeleteOnClose", "UNICODE");
```

On completion of the writing phase, we call the **SetReadMode** method (reserved for this type of stream) which enables switch to reading mode.

We can then use it as a stream in reading mode; however we can no longer write in it.

Finally the **Close** function destroys the temporary file. If it is not called, the file is destroyed when the component is freed.

We can write the example above using a stream on temporary file:

VB Script

```
Function TestBSTRTempo(mgRoot as MegaRoot)
Set myTextFile =
mgRoot.CurrentEnvironment.Toolkit.CreateMegaObject("MegaTextStream")
myTextFile.Open "", "Write,Create,Temporary,DeleteOnClose","UNICODE"
For i = 1 To 10000
myTextFile.WriteLine "Writing of the line number " & i
Next
myTextFile.SetReadMode
TestBSTRTempo = myTextFile.ReadAll
End Function
```

Java

```
public String TestBSTRTempo(final MegaRoot mgRoot) {

MegaCOMObject mgobjMyTextFile = (MegaCOMObject)
mgRoot.currentEnvironment().toolkit().createMegaObject("MegaTextStream");

mgobjMyTextFile.invokeMethod("Open", "",
"Write,Create,Temporary,DeleteOnClose", "UNICODE");

for (int i = 1; i <= 10000; i++) {mgobjMyTextFile.invokeMethod("WriteLine", "
Writing of the line number " + i);

}

mgobjMyTextFile.invokeMethod("SetReadMode");

     return (String) mgobjMyTextFile.invokeFunction("ReadAll");

     }
```

This function executes slower than the **TestBSTRStream** function (around 20%), but in any case much faster than **TestBSTRNative**.

It should be noted that this implementation is of no great interest in this example, since in any case we must provide allocation of the return character string, here in the **ReadAll** function: In this context, it is not possible to manage a stream with insufficient memory…

11 How to

11.1 Supervising HOPEX

"MegaSupervisionManager" component allows to interact and exchange data with HOPEX Server Supervisor.

From "MegaSupervisionManager", you can call:

- `EventCreate(var jsonEventParam)`, which creates a supervision event and returns an object event.

  A supervision event is an object understandable by HOPEX Server Supervisor.

  `jsonEventParam` includes the definition of event code, type and level:

  `{« Name » : « my eventname », « type » : « my eventtype », « level » :mylevel }`

  - `Name` is a character string, default: empty
  - `Type` is a character string, default: « Undefined »

    Its values can be (non-sensitive case): « UserAction », « Error », « ScheduledStateReport », « Alert », « ExitSignal », « GetInfos »
  - `Level` is an integer

- `EventDestroy(object event)`, which deletes the given supervision event.

On the object event, you can call:

- `Set(var jsonEventData)`

  `jsonEventData` is found in the **EventData** column of the HOPEX Server Supervisor tool

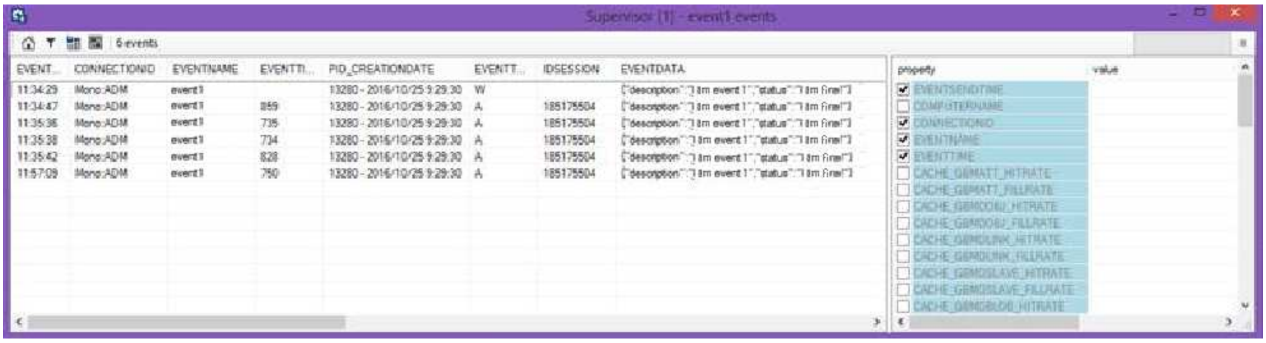
- `Publish()`

  Event publication: the event is sent to HOPEX Server Supervisor. It can be sent only once.

Use case: EventName: "I am event 1" ; Status: "I am fine"

```
MegaSupervisionManager manager=new MegaSupervisionManager(this.megaRoot);

MegaSupervisionEvent event= manager.EventCreate("event java",
MegaSupervisionEventType.ACTION, 3);

event.Set("{\"description\":\"I am event 1\",\”status\”:\”I am fine\”}");

event.Publish();

manager.EventDestroy(event);
```

The result in HOPEX Server Supervisor shows:



Supervisor [1] - event1 events

6 events

EVENT...	CONNECTIONID	EVENTNAME	EVENTTI...	PID_CREATIONDATE	EVENTT...	IDSESSION	EVENTDATA
11:34:29	Mono:ADM	event1		13280 - 2016/10/25 9:29:30	W		{"description":"I am event 1","status":"I am fine!"}
11:34:47	Mono:ADM	event1	859	13280 - 2016/10/25 9:29:30	A	185175504	{"description":"I am event 1","status":"I am fine!"}
11:35:36	Mono:ADM	event1	735	13280 - 2016/10/25 9:29:30	A	185175504	{"description":"I am event 1","status":"I am fine!"}
11:35:38	Mono:ADM	event1	734	13280 - 2016/10/25 9:29:30	A	185175504	{"description":"I am event 1","status":"I am fine!"}
11:35:42	Mono:ADM	event1	828	13280 - 2016/10/25 9:29:30	A	185175504	{"description":"I am event 1","status":"I am fine!"}
11:57:09	Mono:ADM	event1	750	13280 - 2016/10/25 9:29:30	A	185175504	{"description":"I am event 1","status":"I am fine!"}

property	value
☑ EVENTSENDTIME	
□ COMPUTERNAME	
☑ CONNECTIONID	
☑ EVENTNAME	
☑ EVENTTIME	
□ CACHE_GBMATT_HITRATE	
□ CACHE_GBMATT_FILLRATE	
□ CACHE_GBMODU_HITRATE	
□ CACHE_GBMODU_FILLRATE	
□ CACHE_GBMOLINK_HITRATE	
□ CACHE_GBMOLINK_FILLRATE	
□ CACHE_GBMOSLAVE_HITRATE	
□ CACHE_GBMOSLAVE_FILLRATE	
□ CACHE_GBMOBLOB_HITRATE	

11.2 Undo

- undoCollectionStart
- undoCollectionStop
- undoQueueReset

```
public int undoCollectionStart(final String undoCollectionName);
```

From the call to the undoCollectionStart till the call to undoCollectionStop, all the "undo items" of the current session will be registered into a collection.

This "undo items" collection will be visible into the undo queue as an item with the given name. This enables to rollback a whole defined set of actions instead of doing it action by action.

Parameter description:

undoCollectionName: name for the collection of "undo items".

It returns the undoCollectionId that will be used to call the undoCollectionStop

```
public void undoCollectionStop(final int undoCollectionId);
```

Stops the registration of "undo items" into the collection identified by the given id. And makes the collection visible into the undo queue.

Parameter description:

undoCollectionId: id of the collection returned by a call to undoCollectionStart

```
public void undoQueueReset();
```

Resets all the items of the undo queue.

VB Script

```
Dim undoCollectionId

undoCollectionId = getroot.currentenvironment.UndoCollectionStart("group of
applications")

getcollection("application").create

getcollection("application").create

getcollection("application").create

getroot.currentenvironment.UndoCollectionStop undoCollectionId
getroot.currentenvironment.UndoQueueReset
```

Java

```
int undoCollectionId =
oRoot.currentEnvironment().undoCollectionStart("collection 1");
        //list of actions
oRoot.currentEnvironment().undoCollectionStop(undoCollectionId);


oRoot.currentEnvironment().undoQueueReset();
```

11.3 Import/Export

11.3.1 Using MEGA Import/Export command options

import / export shared options

`Meta = On/Off/Full` META occurrences are included - default: On

`Data = On/Off/Full` DATA occurrences are included - default: Full

`Technical = On/Off/Full` Technical occurrences are included - default: Full

`MgxFilter = Meta|Data|Technical|HistoricOff|TextOff|KernelOnly|TranslationOff|DateOff|UserOff|HistoricDisable|LogActivityOff None –` default: None

`CommandFormat = MGL/ XML –` default: MGL

Export specific options

`ExportDateStart = '<date>'`

`ExportDateEnd = '<date>'`

Date format can be:

yyyy/mm/dd,

dd/mm/yy

dd/mm/yyyy

yyyy/mm/dd HH:MM:SS

For example: `ExportDateStart = '19/12/2018'`

XML import specific option

`XmlReport = None|Errors|Warnings|Commited|Skipped`

Import specific options

`ServerMode = On/Off` (with or without GUI) default: Off

`Mega = On/Off MEGA` occurrences are included - default: Off

`Validate = Never/Standard/AtEnd/AtEndOnSuccess –` default: Standard

`ControlIdAbs = On/Off –` default: depends on CommandFormat

`ControlGraph = On/Off –` default: depends on CommandFormat

`Logability = On/Off` Log Activation - default: On

`ReprocessStd = On/Off` Standard Reprocessing - default: Off

`ReprocessUsr = On/Off` User Reprocessing - default: Off

`RepositoryActivity = On/Off` (2005 Release specific) - default: Off

`HistoricDisable` and `LogActivityOff` (2005 Release specific)

HistoricOff option is no longer supported with 2005 Release CheckBase option

`CheckMode = Physical/Logical/Total –` default: Total

`CheckTransaction = On/Off –` default: Off

SaveCommand specific option (backup deducted from ChangeItems)

`DisabledCommandGeneration = On/Off –` default: Off

11.3.2 Exporting Excel data in batch mode

To export Excel data from HOPEX: create an Export Template and use the **Excel Export** operator as follows:

VB Script

```
GetRoot.callMethod "~Ef(7n42ebngG[Excel Export]", "<export template file
name>", "<export file name>", "<WritingMode>", "<name of the Excel sheet to
be processed>", "<MegaCollection of the objects to be exported>", Nothing
```

Code description:
<WritingMode> values:

- Replace: replaces the output file
- Append: opens the output file and adds the Excel sheet to it.

In the following Excel export example, the <WritingMode> = Replace creates a new output file for the first sheet (Applications) to be processed and the <WritingMode> = Append adds a new sheet to the file for the second sheet (Processes) to be processed. In that case the output file includes two Excel sheets.

VB Script

```
GetRoot.callMethod "~Ef(7n42ebngG[Excel Export]", "C:\MyTemplate.xlsx",
"C:\Output.xlsx", "Replace", "SheetApplications",
TheCollectionOfApplications, Nothing

GetRoot.callMethod "~Ef(7n42ebngG[Excel Export]", "C:\MyTemplate.xlsx",
"C:\Output.xlsx", "Append", "SheetProcesses", TheCollectionOfProcesses,
Nothing
```

11.3.3 Importing Excel data in batch mode

To import Excel data into HOPEX use the **Excel Import** operator as follows:

VB Script

```
GetRoot.callMethod "~XDB7q4rPbfo5[Excel Import]", "<name of the Excel file to be imported>", "<report file name>", <boolean>, Nothing
```

Code description:

- < report file name>: this report details (in .xls format) the import result (object, links imported or rejected)
- <Boolean>: indicates where to import the Excel data, in the current library (value: 1) or not (value: 0)

11.3.4 Exporting a Report DataSet in batch mode

To export any Report DataSet in CSV format from HOPEX use the **CSVExport** operator as follows:

VB Script

```
Dim mgRoot
Dim bIsCompressed
bIsCompressed=False
Set mgRoot = mgobjReportDataset.GetRoot
Dim strCSVFilePath
strCSVFilePath = mgobjReportDataset.CallMethod("~hOP7zwnRWbOH[CSVExport]", bIsCompressed)
mgRoot.CurrentEnvironment.Toolkit.ExecCmd strCSVFilePath, "?Open"
```

11.4 Launching MEGA Tools from APIs

11.4.1 Interactive tools

Access to an element menu

`MegaItem.CommandManager As Object:`

`See Name as String ~oKdcP5epmcfC[Name]:` element title

`Nature as String ~VxJRO58xtMuC[Nature]:` Control nature, conforming to configuration syntax:

```
"ListView"
"TreeView"
"Static"
"Edit"
"Text"
```

```
"ActiveX"
"HelpComment"
"Schedule"
"Button"
“RadioButtons”
“CheckButtons”
“EditButton”
“Password”
“Viewer”
“SubPage”
"RemotingDropDownListMenu"
“DropDownSelection”
“MatrixView”
"ComboLinks"
“Image”
“Label”
“MetaTree”
“Matrix”
“ExtJsComponent”
“ImageList”
“MultiComboBox”
“Toolbar”
“Report”
“Diagram”
“Card”
"MegaEditCheck"
"ComboBox"
"DropDownList"
"CheckBox"
"3StateCheckBox”
"ComboBoxMenu"
"DropDownListMenu"
"StaticMenu"
```

```
"EditMenu"

“ComboBitmaps”

“DynamicCombo”

“DatePicker”

“ComboLinks”
```

Order as Long ~5gs9P58ye1fC[Order]

SourceID as Variant ~Dgs9P58(e1fC[SourceID]: characterizes the property displayed by the control. Usually, the absolute identifier of a MetaAttribute.

Style as Long ~)0nfa1WYhchD[Style]: bit field characterizing the control

```
#define ACWISTYLE_NOTITLE 0x0010000

#define ACWISTYLE_TITLEUP 0x0020000

#define ACWISTYLE_DEFAULTED 0x0040000

#define ACWISTYLE_BOTTOMALL 0x0080000

#define ACWISTYLE_BORDERED 0x0100000

#define ACWISTYLE_EXCLUDED 0x0200000

#define ACWISTYLE_DISABLED 0x0400000

#define ACWISTYLE_HIDDEN 0x0800000

#define ACWISTYLE_DYNAMIC 0x1000000

#define ACWISTYLE_REMOTING 0x2000000

#define ACWISTYLE_CLIPLEFT 0x002

#define ACWISTYLE_CLIPRIGHT 0x004

#define ACWISTYLE_CLIPTOP 0x008

#define ACWISTYLE_CLIPBOTTOM 0x010

#define ACWISTYLE_CLIPMLEFT 0x020

#define ACWISTYLE_CLIPMRIGHT 0x040

#define ACWISTYLE_CLIPMBOTTOM 0x080

#define ACWISTYLE_CLIPMTOP 0x100

#define ACWISTYLE_NOVCLIP 0x1000

#define ACWISTYLE_NOHCLIP 0x2000
```

Group as String ~0es9P5eQf1fC[Group]: name of the Group in which the control has been placed

Options as String ~T2zVa10))bZD[Options]: options defined for the control in configuration

Read Only as Boolean ~3es9P5ORf1fC[Read Only]

Width as Long ~ths9P5OOf1fC[Width]: control width (in dialog Units)

Height as Long ~uhs9P5eOf1fC[Height]: control height (in dialog units)

Left as Long ~whs9P58Pf1fC[Left]: if specified, control left margin (in dialog units)

Top as Long ~)hs9P5OQf1fC[Top]: if specified, position of top of control relative to group (in dialog units)

Kind as Short ~a2zVa1W00cZD[Kind]: precise nature of control

ControlID as Binary ~v0nfa10XhchD[ControlID]: CLSID of component implementing the control

MapID as Variant ~PDfkighQx400[MapID]: absolute identifier of MAP of control (see Parameterization)

LinkID as 64Bits ~V20000000z50[LinkID]: if the control is obtained from a link, absolute identifier of the MetaAssociationEnd.

ObjectID as 64Bits ~qM44Q5OUd4xC[ObjectID]: absolute identifier of the object of which the control displays a property

ID as Variant ~jKdcP5OomcfC[ID]: Control internal identifier

Child controls of a control are accessible by the Child Collection `(« AttributeControl »)` **~7fs9P58ig1fC[AttributeControl]**

Accessing MegaObject menus using APIs.

Exploring an element

VB Script `MegaItem.Explore :`

Java `MegaItem.invokeMethod("Explore");`

This command launches HOPEX explorer on any MegaItem (MegaObject or MegaCollection).

When calling a MegaCollection, the root is a field (might be red if the collection does not correspond with a MetaAssociationEnd).

Access to the description of an object property pages

VB Script `MegaObject.PropertiesDialog(« Description ») As Object :`

Java `MegaObject.callFunction("propertiesdialog", "Description");`

See Accessing the description of an object Property Pages.

Opening an object property pages

VB Script `MegaObject.PropertiesDialog(optional option1 As String,optional option2 as String,…) :`

Java `MegaObject.invokeMethod("PropertiesDialog", String optional option1, String optional option2,…);`

The following options are optional and can be entered in any order:

`DefaultSize=w,h`: property window default size

`Position=x,y`: property window position in the page

`CommentSize=h`: help area hight size

`ActivePage={clsid}`: initially active page. You can obtain the page identifier from the object description mentioned above.

Open the macro editor

VB Script

```
MegaObject<Macro>.OpenMacro
```

Java

```
mgRoot.getObjectFromID("~vh1nrtiV95P0[DoDAF Exchange
Utilities]").invokeMethod("OpenMacro");
```

Interactive deletion

VB Script

```
MegaItem.QueryDelete(optional Options As String,optional impactList As String)
:
```

Java

```
mgRoot.getObjectFromID("~vh1nrtiV95P0[DoDAF Exchange
Utilities]").invokeMethod("QueryDelete");
```

This function runs interactive deletion on the object or collection relating to the MegaItem.

If present, the *options* parameter can contain the following keywords:

```
SilentMode: the user interface is not activated; deletion is launched on all
objects, including propagated objects
NoCheckDeletable: deletion rights are not checked; all objects that can be
deleted physically are deleted.

In Mega 2005, a deleted objects backup system can be activated or deactivated

BackupHidden: hide backup view

BackupActive: backup activation

BackupInactive: backup deactivation
```

When deletion of an object can impact an important object (a diagram for example), an indicator is displayed in the interface. It is possible to define a list of objects for which we do not want to display impact; to do this, we must concatenate in the *impactList* parameter the absolute identifiers (in the form of fields) of these objects. In particular, this system allows that, at deletion in a diagram, impacts relating to this specific diagram are not included in the interface.

Interactive query

VB Script

```
MegaRoot.RequestQuery(optional metaClasses As Variant) As MegaCollection
```

Java

```
MegaCollection mgcolQuery = (MegaCollection)
mgRoot.invokeFunction("RequestQuery");
```

This function runs interactive query and returns the collection thus selected.

If the query is cancelled by the user, the result is *Nothing*.

The *MetaClasses* parameter enables definition of the MetaClasses to which the query should relate.

If it is absent, the query relates to all MetaClasses accessible to the user.

If specified with the name, identifier or field representing a MetaClass, query is restricted to these.

If we want to run a query related to several MetaClasses, we should specify as parameter a MegaCollection containing a list of these MetaClasses.

Interactive object selection

```
MegaCollection.SelectQuery(optional title As String,optional monoSel As Boolean)
As MegaCollection
```

Running the selection box enables selection of one or several objects of the start collection, and restores the collection of selected objects.

Title of the box is *title* if the argument is specified.

If *monoSel* is *True*, the box allows selection of only one object. Titles of these boxes should be specified.

The collection restored could be empty.

Running the object creation wizard

```
MegaCollection.InstanceCreator As Object:
```

See *HOPEX Power Studio* > *Customizing the Metamodel* > ***Forms - Wizard implementation***.

Launching a generic wizard

```
MegaObject.CallFunction(«~AfLYxbu47b00[WizardRun]») As Object:
```

See *HOPEX Power Studio* > *Customizing the Metamodel* > ***Forms - Wizard implementation***.

To be documented

MegaObject.Edit: opens object editor

MegaObject<Diagram>.Open: opens diagram

ProgressBarDlg: launches the progress bar

Simulate: launches the simulation tool

11.4.2 Batch Tools

Access to rules and regulations API

```
ApplyRegulation; ApplyRule; ApplyTest,RuleAppliableIs, RegulationActivate,
TestApply :
```

See Accessing rules and regulations using APIs section.

Evaluating a condition on an object

```
MegaObject.ConditionEvaluate(condition) As Boolean
```

The condition conforms to the syntax of conditions used in rules, properties pages or commands.

Creating duplicate of an object

```
MegaObject.CreateDuplicate(copyName As String,subObjectPrefix As String)
```

Creating first variant of an object

```
MetaObject.CreateVariant As MegaObject
```

Testing if object is filtered in the current session of HOPEX

```
MegaObject.IsAvailable As Boolean
```

For a MetaAssociationEnd the test is carried out on the MetaAssociation.

Importing a command file

```
MegaRoot.MegaImport (importfilename as string, rejectfilename as string,
optionalparameters as string) as integer
```

Saving an element

```
MegaItem.SaveAs
```

Importing/Exporting in batch mode

See Import/Export section and see also HOPEX documentation regarding import/export options.

Exporting content of changeItem or changeItem collection

```
MegaItem.SaveAsCommand
```

Function not delivered. To obtain it, import the corresponding **AlignToBase** macro.

To create it directly; internal macro:

```
_objectfactory = "CompareMethodObjectCreate", _server = GBMF
```

Enables comparison of elements in two repositories and generation of alignment file.

```
MegaItem.CallFunction(« AlignToBase » , otherBase, updateFileName As
String,optional optionList As String,optional deleteFileName As String)
```

If MegaItem is a MegaRoot, we compare complete repository.

If MegaItem is a MegaObject, we compare this object (and its content). If a MegaCollection, we compare objects of the collection

otherBase: identifier or name of the repository to be compared

updateFileName: name of the file to be generated

deleteFileName: name of file containing deletion commands, if we want to handle these separately.

optionsList: list of options. Options included in a single character string, separated by commas, and in any order. These options are:

`UpdateExtract`: generates extraction update only, not that of complete repository

Reverse: generates reverse alignment

`Ignore=item1 {;item2…}` does not compare cited attributes. We can include as many Ignore= options as there are attributes; these are grouped, separated by semi-colons. Conventional attributes can be cited using keywords

`Date;Creator;Authorization;Comment;Order`. Others are included either by name, or their absolute identifier in the form of a field.

Used as a function, the returned value indicates if the operation has been suitably executed.

Used as a method, returns an error if the operation has failed

To be documented:

MegaObject.Protection As MegaLock; accesses component managing object protection and lock

DetachDocument: detaches a HOPEX Report (MS Word) object

MegaObject<Diagramm>.Drawing: accesses diagram drawing

ExecuteDescriptor: enables to execute a MEGA descriptor

GenerateCode: generates a Code descriptor

GenerateFmt: generates an HTML Formator

GetMetaClassPicture: retrieves the picture of a specific MetaClass

GetObjectPicture: retrieves the picture of a specific object

GetPropPicture: CDB

InheritedSelect: CDB

InitializeDocument: reinitialize a report(MS Word) content

NewDocument: creates a new Report (MS Word) from an object and a dedicated Report(MS Word)

Template

WebSiteDescription: gives access to the APIs of the web site description

WebSiteTemplateDescription: gives access to the APIs of the web site Template description

11.5 Accessing the Desktop Context using APIs

This component enables an API program to interact with the desktop.

To access to the desktop:

VB Script

```
Set myDesktopContext = CurrentEnvironment.DesktopContext (optional Root
As MegaRoot)
```

Java

```
MegaCOMObject myDesktopContext = (MegaCOMObject)
mgobjMegaObject.getRoot().currentEnvironment().invokePropertyGet("Desktop
Context", mgobjMegaObject.getRoot());
```

Can be Nothing if there is no DesktopContext

If the MegaRoot is not specified, any access to HOPEX repository leads to the message « An external program tries to access HOPEX … »

If the MegaRoot is specified, the DesktopContext can depend on the MegaRoot.

The desktop context enables:

- Actions (activate, deactivate) on desktop client areas
- Creation of specific client areas (currently only browsers can be created)
- Access to current desktop element (the present version does not enable notification of a change of current object)

DesktopContext is a component conforming to the following object model:

DesktopContext:

VB Script

```
DesktopContext.CurrentObject As MegaObject
```

Java

```
MegaObject mgobjCurrentObject = (MegaObject)
myDesktopContext.invokePropertyGet("CurrentObject");
```

Enables access to or modify the current object of the desktop

VB Script

```
DesktopContext.GetRoot As MegaRoot
```

Java

```
MegaRoot mgRootExpl = (MegaRoot)
myDesktopContext.invokePropertyGet("GetRoot");
```

Enables access to the desktop repository.

VB Script

```
DesktopContext.Clients As DesktopClients
```

Java

```
MegaCOMObject myDesktopClients = (MegaCOMObject)
myDesktopContext.invokePropertyGet("Clients");
```

Enables access to the collection of the items of the client area.

DesktopClients:

```
DesktopClients.Count As Integer
```

Number of desktop clients

```
DesktopClients.Item(Index As Variant) As DesktopClient
```

Enables access to a specific client. Index can be:

the index (Integer from 1 to Count)

the tab identifier (can be entered in Field format)

the tool identifier (can be in entered in Field format)

the tab name

This function returns the first client that verifies the index.

```
DesktopClients.Create(Nature As String, Id As Variante, InitialName As String)
As DesktopClient
```

Enables to add a new item in the desktop client area:

`Nature`: nature of the new item (Example: « WebBrowser »)

`Id`: tool identifier (can be in entered in Field format). Via this unique identifier, you can reuse an already existing client area

`InitialName`: tab initial name

DesktopClient

`DesktopClient.Name As String`

Enables client name consultation or modification in clients bar

`DesktopClient.GetID As Variant`

Enables determination of client identifier

`DesktopClient.Flags As Integer`

Enables client configuration consultation or modification (use not currently public)

`DesktopClient.Type As Variant`

Enables determination of client type (diagram, browser, other…)

`DesktopClient.ToolID As Variant`

Enables determination or modification of tool identifier. A specific client can be found using this property. The property can be updated using a field

`DesktopClient.Activate`

Enables client activation (bring to surface)

`DesktopClient.Component As Object`

Enables access to client implementation. In the case of a WebBrowser, this component is the browser itself, supporting the `IWebBrowser2` interface, which among other things enables definition of a URL for the browser…

11.6 Comparing and aligning (CompareTool API)

CompareTool is a MEGA object enabling management of object comparison between the current workspace and a target repository in VBScript. Comparison enables creation of an update file designed to align the target repository. This functionality is available with the HOPEX Supervisor module.

*For detailed information see HOPEX Power Studio > Customizing the Metamodel > **Perimeters**.*

11.7 Exporting (ExportTool API)

ExportTool is a MEGA object enabling management of standard export of an object or collection of objects in VBScript or Java. This functionality is available with the "MEGA Supervisor" module.

*For detailed information see HOPEX Power Studio > Customizing the Metamodel > **Perimeters**.*

11.8 Launching an automatic macro while publishing

The system described below enables launch of automatic processing at HOPEX workspace dispatch.

In particular, this system can be used:

- to execute compliance updates before dispatching
- to run synchronizations with third-party tools when we are sure that the workspace will not be discarded.

In principle: we manage a list of identified 'Jobs' in a workspace

- by the macro to be called
- by a specific parameter.

This list of jobs can be accessed from a MegaRoot by the function:

VB Script

```
Set Jobs = MegaRoot.DispatchJobs
```

Java

```
MegaDispatchJobManager mgdjmJobMgr = new
MegaDispatchJobManager(mgRoot);
```

A job is inserted in this list by:

VB Script

```
Jobs.Insert Macro , Parameter
```

Java

```
mgdjmJobMgr.insert(macro, Parameter);
```

Where:

- **Macro** is the identifier of the macro to be executed for the job, or a character string containing the macro name or field
- **Parameter** is any character string serving as parameter for the Job

*The pair (**Macro,Parameter**) is unique in a workspace: if we insert the same pair twice, the second insertion is ignored.*

We can browse the list already recorded for this repository in this workspace and consult characteristics of each job

VB Script

```
dim Job
For i=1 to Jobs.count
Job=Jobs.item(i)
Print Jobs.item(i).GetID ' unique identifier assigned to the job at its
creation
Print Jobs.item(i).MacroID ' identifier of the macro to be executed for the
job
Print Jobs.item(i).Parameter ' job parameter
Next
```

Java

```
for (int j = 1; j <= (Integer)
mgdjmJobMgr.basedObj.invokeFunction("count"); j++) {
mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]", ((MegaCOMObject)
mgdjmJobMgr.basedObj.invokeFunction("item", j)).invokeFunction("GetID"));
mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]", ((MegaCOMObject)
mgdjmJobMgr.basedObj.invokeFunction("item", j)).invokeFunction("MacroID"));
mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]", ((MegaCOMObject)
mgdjmJobMgr.basedObj.invokeFunction("item",
j)).invokeFunction("Parameter"));
}
```

This list remains empty if you are not in a workspace.

To remove a job from the list:

VB Script

```
Jobs.Remove Macro , Parameter
```

Java

```
mgdjmJobMgr.remove(macro, Parameter);
```

This function does not generate an error when the pair (Macro, Parameter) does not correspond to a job in the list.

Note: In Mega 2009, we can insert jobs concerning only the system repository; this list is accessible by MegaRoot.GetSystemRoot.DispatchJobs

At dispatch:

The macro corresponding to each job is instanced. If the same macro is used for several jobs, as many instances as there are jobs will be created.

It is therefore possible to use globals in these macros. Just before dispatch: we call the following Method in the Macro. This method is necessarily present in the Macro

```
Sub RunJobBeforeDispatch(Root As MegaRoot,Parameter As String)

' work with dispatch data.

' it is not possible at this stage to execute updates that will be taken into
account in the workspace.

End Sub
```

Dispatch then takes place.

If dispatch is successful, call the following method:

```
Sub RunJobAfterDispatch(Root As MegaRoot,Parameter As String)

' good point to execute Commit on third-party DBMS, if required.

' if a global has been defined in RunJobBeforeDispatch, it can be used here.

' MEGA updates however should not be executed in this function.

End Sub
```

If dispatch is not successful, call the following method:

```
Sub RunJobOnFailedDispatch(Root As MegaRoot,Parameter As String)

' good point to execute Commit on third-party DBMS, if required.

' if a global has been defined in RunJobBeforeDispatch, it can be used here.

' MEGA update execution is not recommended in this function, which has not been
tested (to do this, a dispatch must fail...)

End Sub
```

The following call has been added:

```
Sub RunJobEnd(mgobjRoot As MegaRoot,strParameter As String)
End Sub
```

At dispatch, the user interface is locked to prevent any user modifications.

Implementation of this sub is called after user interface unlocking. This allows jobs to propose a user interface, which would otherwise be locked.

Note: RunJobEnd is called whether dispatch is successful or not. To determine if dispatch has been successful, use a global update RunJobAfterDispatch or RunJobOnFailedDispatch, depending on the case.

The following function has been added:

```
Function RunTestJobBeforeDispatch(mgobjRoot As MegaRoot, strParameter As String)
As String

End Function
```

This function is called before dispatch (after click on Dispatch button) in the repository but unlike the RunJobBeforeDispatch procedure, it prevents dispatch if necessary.

If the function returns "" (empty string) then dispatch will be started; if not, the user returns to the workspace after acknowledgement (responsibility of macro to display explanatory message).

11.9 Invoking an object creation wizard using APIs

Objective: Enables creation of an object using its interactive creation wizard.

This function uses the InstanceCreator method, available on a MegaCollection. This method returns (if appropriate) a MegaInstanceCreator component.

```
Set myCreator = myRoot.GetCollection("~QrUiM9B5iCN0[Org-Unit]").InstanceCreator
```

```
[java : MegaInstanceCreator myCreator = new MegaInstanceCreator(myCollection) ]
```

This component enables launch of the object creation wizard offering the possibility of modifying its behavior, in particular:

- By redefining wizard launch mode by means of the mode property – In practice, 3 modes are possible
  - mode = 2:

    Non-interactive launch – no window opens, only specific processing of creation is launched, possibly dependent on supplied parameters. Creation by this mode fails when a required parameter is missing, or when the specific object creation wizard does not implement this function.
  - mode = 1:

    Standard creation mode with entry of name and all parameters.
  - mode = 0: "inplace" mode.

    This mode does not allow name entry and is adapted to creation from a listview, in which the name will be subsequently edited. Depending on the case, no dialog box is opened. If however a required parameter is missing, the interactive wizard may nevertheless be launched.
- By specifying creation context parameters (attributes or cookies).
- (Mega 2009) By inserting additional processing and pages to the wizard.

The interface of this component is an extension of the MegaWizardContext interface (in particular the Mode property mentioned above is a property of this interface). Additions are:

```
create As Variant
```

This function launches the wizard. It returns the identifier of the object created (or possibly reused) and 0 if the wizard has been discarded.

It is not currently possible to continue using the component after call on the Create function.

```
getRoot As MegaRoot
```

```
classID As Variant
```

Obtains wizard target MetaClass.

The following methods have been added:

VB Script `addTriger(TriggerId As Variant,Optional Options As String)`

Java `public void addTriger(final Object trigger, final String position)`

This method enables insertion of a trigger, of which we specify the identifier here. This trigger enables modification of wizard behavior, and in particular the addition of an initialization or processing code. By default, this trigger is called last (that is after the triggers defined for the wizard). However, with the "OnHead" option, it is possible to arrange that this trigger be called first.

VB Script

```
addPage(PageId As Variant,Optional Options As String)
```

Java

```
public void addPage(final Object page, final String position)
```

this method enables insertion of an additional page in the wizard. PageId is the identifier of the MetaPropertyPage that we want to insert. The option enables specification of the position of the page, and it can take the following values:

`Preliminary:` the page is presented in first position. It does not propose entry of name.

`Preparatory:` the page is presented after the preliminary pages, but before creation of the object.

`Standard:` This is the default; the page is presented after the preliminary and preparatory pages, but before creation of the object. It displays the name of the object if necessary.

`AfterCreation:` the page is presented after creation of the object.

Conclusive: the page is presented in last position.

It is possible to add several pages. If they are of the same type, they will be proposed in the order in which they are added.

If the option is not specified (or equals **Standard**), the page is proposed before object creation.

Example: creating a MetaAssociation

This is a case where InstanceCreator is essential: Creating a Metaassociation in API is impossible, since the two MetaClasses Source and Target must be known before Creation. It is however possible to create a MetaAssociation in a MetaModel diagram, using a link between two MetaClasses.

To create a MetaAssociation in API, we will simulate this creation mode by specifying source and target MetaClasses.

VB Script

```
Set myCreator = myRoot.GetCollection(« MetaAssociation »).InstanceCreator

myCreator.mode = 2

myCreator.SourceID = <source MetaClass identifier>

myCreator.TargetID = <target MetaClass identifier >

myCreator.Name = « <MetaAssociationName> »

myAssocID = myCreator.Create
```

Java

```
MegaInstanceCreator myCreator = new
MegaInstanceCreator(myRoot.getCollection("MetaAssociation");

myCreator.mode(2);

myCreator.sourceID(<source MetaClass identifier >);

myCreator.targetID(<target MetaClass identifier>);

myCreator.name(<association name>);

Object myAssocID = myCreator.create() ;
```

11.10 Checking a script execution

From the execution context of a Macro, we can obtain a system enabling script execution check. This system makes available to the script a status area displayed in real time to the user, a gauge and the possibility of cancelling processing by means of a button.

This system can be implemented by the application launching the script; otherwise a specific window is displayed.

You get the execution context of a macro from the Root

VB Script

```
set mControl = getRoot.ContextObject("#Window")
mControl.Create
mControl.Text = "Hello"
mControl.Status = "This is a test"
mControl.SetRange 1, 20000
for i = 1 to 20000
mControl.Status = "We are in " & i
mControl.SetGauge i
if mControl.IsAborted then i = 20000
next
if mControl.IsAborted then print "The processing as been aborted"
```

Java

```
MegaProgressControl mgmpcControl = new MegaProgressControl(mgRoot);
 mgmpcControl.create();
 mgmpcControl.text("Hello");
 mgmpcControl.status("This is a test");
mgmpcControl.setRange(1, 20000);
for (int j = 1; j <= 20000; j++) {
mgmpcControl.status("On en est à " + j);
 mgmpcControl.setGauge(j);
 if (mgmpcControl.isAborted()) {
 j = 20000;
  }
  }
  if (mgmpcControl.isAborted()) {
mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]", "The processing has been aborted");
 }
mgmpcControl.destroy();
```

Methods available with this component (MegaProgressControl) are the following:

`MegaProgressControl.Text:` title content

`MegaProgressControl.Status:` status content

`MegaProgressControl.Create:` window activation. If the calling context does not manage the StatusWindow, it is created at this occasion

`MegaProgressControl.Destroy:` window closing (if created by the script)

`MegaProgressControl.SetRange min, max:` scroll bar range definition (0 -> 32767)

`MegaProgressControl.SetGauge val:` scroll bar positioning

`MegaProgressControl.IncrementRange inc:` gauje incrementing

`MegaProgressControl.NextIcon:` animate the icon (when needed)

`MegaProgressControl.IsAborted:` indicates that the user wants to abort the processing (he can click Cancel button when present)

`MegaProgressControl.Abort:` activates 'IsAborted' flag

11.11 Setting up a progress bar in macro execution

This section details how to implement a progress bar (called also gauge) to show the execution progression of a script. The progress bar can be put inside an existing window or can create a new window if required. It is useful to give a visual feedback to the user while HOPEX is executing a time consuming script.

When executing a time consuming script you can setup a Progress Bar to let the user know that something is happening, and also show an estimation of the progress if you know how much steps you will need to complete.

HOPEX has the **MegaProgressControl** object to implement progress bars.

If you are:

- inside a wizard then the progress bar will be put in the bottom left corner of the wizard.
- not in a wizard then a pop-up window will appear, with the progress bar and a status line.

To create this use the following code (`root` is a MegaRoot):

VB Script

```
set pBar = root.ContextObject("#Window")
pBar.create
pBar.text = "I'm a progress bar"
pBar.setRange 1, 1000
for i=1 to 1000
pBar.status = "We are at " & i
pBar.setGauge i
next
pBar.destroy
```

Java

```
MegaProgressControl pBar = new MegaProgressControl(root);
pBar.create();
pBar.text("I'm a progress bar");
pBar.setRange(0, 1000);
```

```
for(int i=0; i<1000; i++) {
pBar.status("We are at " + i);
pBar.setGauge(i);
}
pBar.destroy();
```

If you want to know if the user clicked the "Abort" button then check the value pBar.isAborted() (method in Java, value in VB), if the result is true then the user clicked the abort button.

The list of available methods is (Java version, but VB is pretty the same, just use all methods as properties in VB):

`MegaProgressControl.text(String title)` Add a title to the progress bar

`MegaProgressControl.status(String status)` Add a status to the progress bar, useful to show what the script is doing

`MegaProgressControl.create()` The progress bar is created, in this call if a container window exists (like a Wizard) then the progress bar is attached to it, if not a new popup window is created

`MegaProgressControl.destroy()` Destruction of the object. Is mandatory to call it or you will get an error when your script will terminate

`MegaProgressControl.setRange(int min, int max )` Range of values for the progress bar, the minimum min is 0, the maximum max is 32768

`MegaProgressControl.setGauge(int val)` Set the progress bar at the specified value

`MegaProgressControl.incrementRange(int val)` Increment the progress bar of the passed value

`MegaProgressControl.isAborted()` Returns a boolean to understand if the user clicked on the Abort button.

`MegaProgressControl.abort()` Has to be called to activate the Abort button.

Let the user know what's happening!

11.12 Customizing an extraction using APIs

This section supplements documentation on use of compound operators in APIs.

As a reminder, 'compound' operators do not have Macros and are designed to configure behavior of links related to particular concerns; each MetaAssociationEnd has an associated behavior, the most important of these being:

- `'A' (Abort):` the MetaAssociationEnd is ignored.
- `'L' (Link):` the MetaAssociationEnd is taken into account, but we do not continue search on the associated object, which can be extracted from the collection
- `'S' (Standard):` the MetaAssociationEnd is taken into account, but we do not continue search on the associated object, which must nevertheless be included in the collection
- `'D' (Deep):` the MetaAssociationEnd is taken into account and search continues on the associated object

Extraction by operator therefore consists of building a collection containing all objects associated with a start object (or start collection) by scanning associations according to their behavior.

Elements of the resultant collection have a 'Behavior' pseudo attribute containing the 'maximum' value that caused extraction. (L < S < D)

When extracted objects have been listed, this collection can be completed with the exhaustive list of existing associations between each of these extracted objects.

Call to such extraction is by:

```
Set myCollection = MegaItem.OperatorName(Optional options as String,Optional
filter As Object)
```

or

```
Set myCollection = MegaItem.CallFunction(operator As String, Optional options as
String,Optional filter As Object)
```

Examples:

VB Script

```
Set myCol = myObject.Extract

Set myCol = myObject.Extract

Set myCol = myObject.CallFunction("Extract")
```

Java

```
MegaCollection mgcolCollection = (MegaCollection)
mgobjMegaObject.invokeFunction("Extract");

mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]", mgcolCollection.size());
```

Note that extracted objects with an 'L' behavior are included in the collection.

11.12.1 Options

Options enable configuration of this extraction in greater or lesser detail. The character string can contain the following exclusive keywords:

NOMETACLASS: Only the extracted Associations are inserted in the collection

NOMETAASSOCIATION: Only the extracted Occurrences are inserted in the collection

DIRECTONLY

DIRECTASSOCIATIONS

DIRECTOBJECTS: 'Direct' extractions transform 'Deep'behaviors to 'Standard'

Extraction is therefore at only one level.

With DIRECTOBJECTS, associations are not listed;

With DIRECTASSOCIATIONS, objects are not listed

ALLDIRECTASSOCIATIONS: The operator is no longer taken into account, all behaviors are taken

to 'Standard'

Only the associations are listed in the collection

11.12.2 Confidential object filtering

Confidential objects are not usually included in collections and are not scanned.

To change this behavior add the @TAKECONFIDENTIAL keyword to the options.

Example:

VB Script

```
set myCol = myObj.Extract("DIRECTOBJECTS,@TAKECONFIDENTIAL")
```

Java

```
MegaCollection mgcolCollection = (MegaCollection)
mgobjMegaObject.invokeFunction("Extract",
"DIRECTOBJECTS,@TAKECONFIDENTIAL");

mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]", mgcolCollection.size());
```

11.12.3 Advanced filtering using a component

To execute a more precise filtering use an advanced filtering component.

This component enables definition of customized filtering, either at metamodel level, or at the level of occurrences themselves.

The filtering component is passed as parameter; it can be the current macro using keyword MySelf.

Example:

```
Set myCol = myObj.Extract(MySelf)
```

This component can implement the following functions:

```
Sub OnAssociationFilter(Context As MegaExtractContext,AssocEndID,Behavior)
```

This Method is then called once for each browsed MetaAssociationEnd and enables redefinition of its behavior.

```
Sub OnChildCollectionFilter(Context As MegaExtractContext,AssocEndID,Behavior)
```

This Method is then called for each extracted object and allows us to redefine specifically for this object the behavior of a browsed MetaAssociationEnd.

```
Sub OnChildObjectFilter(Context As MegaExtractContext,AssocEndID,Behavior)
```

This Method enables specific filtering of objects browsed by a given MetaAssociationEnd from a given object.

Extraction parameters are obtained from the MegaExtractContext context component.

`MegaContext.RootItem As MegaItem`: extraction start element (object or collection)

`MegaContext.CurrentSource As MegaObject`: element in course of extraction. Valid only for

`OnChildCollectionFilter` and `OnChildObjectFilter` methods

`MegaContext.CurrentTarget As MegaObject`: current browsed element. Valid only for

`OnChildObjectFilter` method

11.13 Using Administration APIs with callback objects

Administration APIs have the specificity to work in a Windows-based process separate from HOPEX.

Any code execution (especially when complex) that can be implemented in a macro (and so which is not dependent on administration API specificities) must be implemented in a macro

The reasons are the following:

- **Performance**
  The macro will be executed in HOPEX and thus does not need the interoperability to execute the functions. This can lead to a dramatic performance boost as an interoperability call cost is 10 to 100 times higher than an in-process call. This is not dramatic for standard administration functions (for example an "import" function interoperability of 1/100 sec instead of 1/10000 sec is not significant comparing to the import itself that lasts 15 min).

  On the other hand, the interoperability cost of a « for each» in GetCollection is proportionally higher and can even be higher than the cost of the function itself: if the function lasts 1/10000 sec, the interoperability will multiply by 100 the execution time.

- **Usual DCOM interoperability limitations**
  You cannot use a component, which has not been specifically created for this purpose, in another process. Especially, HOPEX or Java components created in a specific process can only be used in this process.

11.13.1 Use case example: Customizing an extraction using APIs

See Customizing an extraction using APIs section.

In principal, you can perform an extraction by calling a component that enables to dynamically filter MEGA objects:

Set myCollection = MegaItem.CallFunction(operator As String, Optional options as String,Optional filter As Object)

For example:

```
Class MyFilter
  Sub OnAssociationFilter(Context,AssocEndID,Behavior)
    ' recalculation code of the MetaAssociationEnd behavior …
  End Sub
End Class

Set myComponent = new MyFilter
Set myCollection = MegaItem.CallFunction("Extract","",myComponent)
```

This cannot be applied in a multiprocess context, thus in a .VBS outside HOPEX. This is also true for a Java executable.

The CallFunction will be executed in HOPEX process, but the HOPEX process is not able to call myComponent in callback, as this component has not been created to be performed in the HOPEX process, but to be performed in the VBS process. You would then get a message like « this component has been marshalled for another thread ».

To make all this work, you need to use a MEGA macro:

```
Sub MyExecution(MegaItem)
Set myComponent = MegaItem.CurrentEnvironment.GetMacro("<filtering macro>"
Set myCollection = MegaItem.CallFunction("Extract","",myComponent)
End Sub
```

and call it unitarily in the external script:

```
MegaItem.CurrentEnvironment.GetMacro(<TheMacro>).MyExecution(MegaItem)
```

In that case, calls are one way VBS -> HOPEX and there is no issue.

11.14 Implementing an Update Tool in script

An UpdateTool:

- is a macro linked through the UpdateTool link to a MetaAttribute, a MetaAssociationEnd or a TaggedValue.
- Enables to modify this concept input behavior in the property pages and in-place areas (outside diagrams)

The Script implementation does not allow defining a data entry window; it only allows behavior modification of an existing control.

To be recognized as such, an UpdateTool script must implement the following function:

VB Script

```
Function AttCtl_GetDefaultKind() As String
```

Java

```
public String attCtl_GetKind(final MegaUpdateToolContext
objMegaUpdateToolContext)
```

This function enables to define the window type to use for input and must feed back a control type.

It is either an internal numerical value, or a character string with the following format:

```
<ControlName>{:<option>}{,<option>}
```

<ControlName>: see property pages documentation:

MegaEditCheck

ComboBox

DropDownList

ComboBitmaps

DatePicker

CheckBox

3StateCheckBox

`ComboBoxMenu`

`DropDownListMenu`

`ComboLinks`

`EditButton`

`StaticMenu`

`EditMenu`

`Text`

`Static`

`Edit`

options:

`ReadOnly`: read-only control

`Numerical`: the Edit area displays a number (right-justified)

`WithDefault`: the control displays the default value button

`Mandatory`: mandatory value

`NoEdit`: the Text area (of the combo box, of the Edit menu, of the Edit button) is in read-only

`ResetOnRefresh`: specific usage

`ValidateInput`: specific usage

`External`: display of the attribute external value

An update tool can simply redefine the control to be used. In that case the standard updatetool behavior is called to process the input.

If the control type to be used depends on the occurrence, you can redefine it specifically for this occurrence by implementing:

VB Script

```
Function AttCtl_GetKind(oContext As MegaUpdateToolContext) As String
```

Java

```
public String attCtl_GetKind(final MegaUpdateToolContext
objMegaUpdateToolContext)
```

This function returns the type of control to be used specifically for the current occurrence. The return value is similar to the `AttCtl_GetDefaultKind` one. You can get the current occurrence with the context provided in argument.

This function can also be used to parameterize the standard updateTool behavior.

The `MegaUpdateToolContext` component includes the following functions:

VB Script

```
Function MegaUpdateToolContext.MegaObject As MegaObject
```

Java

```
public MegaObject megaObject()
```

Occurrence to be modified

VB Script `Function MegaUpdateToolContext.AttributeID As Variant`

Java `public Object attributeID()`

Absolute identifier of the attribute (resp. TaggedValue, MetaAssociationEnd) managed by the updateTool.

VB Script `Sub MegaUpdateToolContext.Invalidate As Variant`

Java

Notifies that the element has been modified. This particularly allows to ungrey the Apply button of the property pages.

VB Script `Function MegaUpdateToolContext.GetRoot As MegaRoot`

Java `public MegaRoot getRoot()`

VB Script `Function MegaUpdateToolContext.AttributeControl As MegaObject`

Java

Sends the component that manages the updateTool piloted control. This component includes the interface of previously mentioned attributecontrols, we can however note availability of the property.

VB Script `AttributeControl.Page As MegaPropertyPageStandard`

Java `public MegaObject attributeControl()`

In wizard case, you can get the wizard context by Page.Cookie …

When the update tool implements a generic MetaAssociationEnd or an attribute or a taggedvalue of objet type, use the following property:

VB Script `MegaUpdateToolContext.ValueTypeID`

Java `public Object valueTypeID()`

To consult or update the MetaClass type of the target object. For example:

```
oContext.ValueTypeID = "Broker"
```

enables standard commands of the object (find/list) to do this on the "Broker" MetaClass.

When the control displays an editing area, you can catch the initialization of this area by implementing:

VB Script

```
Function AttCtl_SetText(oContext As MegaUpdateToolContext,sInitialValue as
String) As Boolean
```

Java

```
public void editText(final String setValue)
```

It is then possible to modify the displayed value by modifying the *sInitialValue parameter and sending back True value.*

To catch a command, for example a click on the button when the control has one, implement:

VB Script

```
Function AttCtl_OnCommand(oContext As MegaUpdateToolContext,Item As
Integer,Notification As Integer) As Boolean
```

Java

In the EditMenu case, you can add manual commands by implementing a MetaCommand in the update Tool. For this MetaCommand to be called, implement:

VB Script

```
Function AttCtl_ImplementsMetaCommand(oContext As MegaUpdateToolContext) As
Integer
```

Java

```
public String attCtl_ImplementsMetaCommand(final MegaUpdateToolContext
objMegaUpdateToolContext)
```

The returned value is 0 if you do not want standard menu commands to be displayed.

Otherwise it includes the capability to be used for this standard commands.

To access to the updatetool context in the CommandAccessor functions, you need to pass through the global MegaMacroData

```
Sub CmdInvoke(obj,num)
Dim AttCtlContext As MegaUpdateToolContext
Set AttCtlContext = MegaMacroData.GetBag.AttCtlContext
' …
End Sub
```

In that context, you can particularly exploit the following MegaUpdateToolContext functions:

VB Script

```
MegaUpdateToolContext.ValueTypeID
```

Java

```
public Object valueTypeID()
```

Current object IdAbs in Edit Menu

VB Script

```
MegaUpdateToolContext.EditText ID
```

Java

```
public String editText()
```

Value to be displayed in the area

Simple example (updatetool to link to an object type taggedvalue)

VB Script

```
'MegaContext(Fields,Types)

'Uses(Components)

Option Explicit

Function AttCtl_GetDefaultKind()

AttCtl_GetDefaultKind = "ComboBoxMenu:ValidateInput"

End Function

Function AttCtl_ImplementsMetaCommand(AttCtlContext As MegaUpdateToolContext)

AttCtlContext.ValueTypeID = "~BEy8SnY(yKk0[City Planning Area])"

AttCtl_ImplementsMetaCommand = 7

End Function

Sub CmdCount(obj,count)

count = 3

End Sub

Sub CmdInit(obj,num,name,category)

name = "Command " & num

category = 4

End Sub

Sub CmdInvoke(obj,num)

Dim AttCtlContext as MegaUpdateToolContext

Set AttCtlContext = MegaMacroData.GetBag.AttCtlContext

Dim oResult

Set oResult = AttCtlContext.MegaObject.GetRoot.GetCollection("~QrUiM9B5iCN0[Org-

Unit]").SelectQuery("Invoke Command #" & num & " on Attribute " &

AttCtlContext.AttributeControl.Page.GetID,True)

if oResult.Count = 1 Then

AttCtlContext.ValueID = oResult.Item(1).GetID

AttCtlContext.EditText = oResult.Item(1).ShortName

end if
End Sub
```

Java

```
public class UpdateToolExample extends MegaMacro {
  public String attCtl_GetDefaultKind() {
    return "ComboBoxMenu:ValidateInput";
  }

  public String attCtl_GetKind(final MegaUpdateToolContext
objMegaUpdateToolContext) {
    return "ComboBoxMenu:ValidateInput";
  }

  public String attCtl_ImplementsMetaCommand(final MegaUpdateToolContext
objMegaUpdateToolContext) {
    objMegaUpdateToolContext.valueTypeID("~BEy8SnY(yKk0[City Planning
Area])");
    return "7";
  }

  public void CmdCount(final MegaObject mgobjValidationCandidateObject, final
Integer[] intCmdCount) {
    intCmdCount[0] = 3;
  }

  public void CmdInit(final MegaObject mgobjValidationCandidateObject, final
Integer iCommandNumber, final StringBuffer strNameReturned, final Integer[]
intCategoryReturned)    {
    strNameReturned.append("Command " + iCommandNumber);
    intCategoryReturned[0] = 4;
  }

  public void CmdInvoke(final MegaObject mgobjValidationCandidateObject,
final Integer iCommandNumber) throws MegaException {
    MegaPropertyBag mpbBag = this.getBag();
    ComObjectProxy copAttCtlContext = (ComObjectProxy)
mpbBag.basedObj.invokeFunction("AttCtlContext");
    MegaUpdateToolContext mgutcContext = new
MegaUpdateToolContext(copAttCtlContext);
    MegaCollection mgcolResult;
    String strText = "Invoke Command #" + iCommandNumber;
    MegaObjectProxy mgobjAttCtrl = (MegaObjectProxy)
mgutcContext.attributeControl();
    ComObjectProxy comopPage = (ComObjectProxy)
mgobjAttCtrl.invokeFunction("Page");
    String strID = (String) comopPage.invokeFunction("getID");
    strText = strText + " on Attribute " + strID;
    mgcolResult = (MegaCollection)
mgobjValidationCandidateObject.getRoot().getCollection("~QrUiM9B5iCN0[Org-
Unit]").invokeFunction("SelectQuery", strText, true);
    if (mgcolResult.size() == 1) {
      mgutcContext.valueID(mgcolResult.get(1).getID());
      mgutcContext.editText(mgcolResult.get(1).getProp("Short Name"));
    }
  }
}
```

When the specified control displays a combo box, you can feed it by implementing the following function:

VB Script

```
Function AttCtl_FillCombo(oContext As MegaUpdateToolContext,oFillCollection as
MegaCollection,sInitialValue as String) As Integer
```

Java

```
public int attCtl_FillCombo(final MegaUpdateToolContext
objMegaUpdateToolContext, final MegaCollection mgcolFillCollection, final
StringBuffer strInitialValue)
```

oFillCollection can be used to supply the collection of objects to be displayed in the list. This collection is also available by

```
Function MegaUpdateToolContext.ComboListCollection
```

This collection is a collection of Values (see GetTypeObject.Properties.Item(x).Values)

So that elements will effectively be integrated in the list:

- either these should be effective occurrences of Value (obtained from a description). It is only in this case that we can display bitmaps (NB: There are no bitmaps in ComboEditMenus)
- or these should be MegaObjects created explicitly in the collection. These objects are virtual. To be taken into account, the attributes **`InternalName`** and **`GUIName`** must be specified. InternalName will be used for update (except for an object list) and GUIName for display
- for attributes of object type or legattributes, we must supply the idabs value of the corresponding object. This should be done when creating the value (see example below).

Possible return values are:

- 0: call default processing
- 1: display list from collection - case of simple tabular attribute
- 2: display list from collection, taking account of absolute identifier of the value - applicable particularly to attributes of object type and to legattributes
- -1: as 1, if we want to display (Default) rather than (Empty) to indicate empty value in the combo
- -2: as 2, if we want to display (Default) rather than (Empty) to indicate empty value in the combo

The following example enables management of an object type attribute or a legattribute of which target is compatible with the 'Org-Unit' MetaClass. The list is supplied with all org-units in the repository.

VB Script

```
Function AttCtl_FillCombo(oContext As MegaUpdateToolContext,oFillCollection As MegaCollection) As Integer

Dim oOrg-Unit

for each oOrg-Unit in oContext.MegaObject.GetRoot.GetCollection("~QrUiM9B5iCN0[Org-Unit]")

Dim oAdded

Set oAdded = oFillCollection.Create(oOrg-Unit.GetID) ' the Org-Unit Absolute identifier is set to the created value.

oAdded.GUIName = oOrg-Unit.ShortName

oAdded.InternalName = oOrg-Unit.MegaField() ' Be careful, the internal value must not exceed 20 characters. In that case, you can add a simple counter as in an object case the internal value is not used

Next

AttCtl_FillCombo = 2

End Function
```

Java

// CURRENTLY NOT AVAILABLE - TESTS in progress ...

To catch the control update, implement the following function:

VB Script

```
Function AttCtl_Update(oContext As MegaUpdateToolContext,iStatus As Integer,sErrorMessage As String) As Boolean
```

Java

This function sends back False value, the update defaut code is not called.

11.15 Managing HOPEX undo/redo actions from a Script

The following Macro is available:

```
CreateUndoCollection
```

This Macro Enables the aggregation of several update commands into a single line in the Undo List. It is possible to explicitly undo the registered commands. This can be useful if you want to manage a 'Cancel' button after a complex update operation.

VB Script

```
It provides a MegaUndoRedoManager component
Set UndoCollection = Root.CallFunction("~oeTN2v5z8z10[CreateUndoCollection]")
```

Java

```
MegaUndoCollection class
```

This component implements the followings functions:

VB Script

```
MegaUndoRedoManager.Start "<Command Name>"
```

Java

```
public void start(final String name)
```

Starts the command aggregation. The given name will be shown in the undo list at the end of the collect.

VB Script

```
MegaUndoRedoManager.Active As Boolean
```

Java

```
public boolean isActive()
```

Indicates if the UndoCollection is started or not. This is the default function.

VB Script

```
MegaUndoRedoManager.Stop
```

Java

```
public void stop()
```

Stops the command aggregation. After this command the UndoCollection Name appears in the undo list. This method is automatically called on the component destruction by default.

VB Script

```
MegaUndoRedoManager.Abort
```

Java

```
public void abort()
```

Stops the command aggregation and undo all the commands. After this command the UndoCollection Name appears in the redo list. Stop and Abort methods are exclusive.

11.16 Converting VB Script APIs into Java

You need to know how to convert a VB Script API into java when you are unable to find a correspondence between a Script API and the Java API library. Although most of the usual MEGA specifics API components have been wrapped into Java Classes (see the com.mega.modeling.api.util package) the less used components have not been involved.

However it is possible to use such a component in Java.

The API anonymous components (typed by Object in VB) are accessible in Java through the MegaCOMObject class. In all the cases, you can replace the following VB Code by the following Java Code:

VB Script

```
Dim myObject

Set myObject = XXX
```

Java

```
MegaCOMObject myObject;

myObject = (MegaCOMObject)XXX;
```

Let's now invoke methods and functions on those components.

- The Java language is an early-binded language and all the methods and functions that are called on java classes must be declared explicitly.
- The script components are late-binded. In such a component you can invoke a method that is not declared on an interface.

This is why the MegaCOMObject Java class implements the following functions:

```
invokeFunction(String, Object...)

invokeMethod(String, Object...)

invokePropertyGet(String, Object...)

invokePropertyPut(String, Object, Object...)
```

With those functions you can invoke the component functions in all the cases.

- invoke a propertyGet or a function returning a value (for example a string or an object)

VB Script

```
Dim myString

myString = myObject.PropName

Dim myObj2

Set myObj2 = myObject.FunctionName(Parameter)
```

Java

```
String myString;

myString = (String)myObject.invokePropertyGet("PropName");

MegaCOMObject myObj2;

myObj2 = (MegaCOMObject)myObject.invokeFunction("FunctionName", Parameter);
```

- invoke a method (with no return value)

VB Script

```
myObject.MethodName Param
```

Java

```
myObject.invokeMethod("MethodName", Param);
```

- set a property

VB Script	`myObject.PropName = "Value"`
Java	`myObject.invokePropertyPut("PropName", "Value");`

All the classes of the com.mega.modeling.api.util package are based on the same principle:

- they involve a MegaCOMObject as a member
- they implement a constructor that set this MegaCOMObject
- they implement explicit function that embed the MegaCOMObject invocation

For example a sample on the MegaPropertyBag class.

```
public int count() {

return ((Integer) this.basedObj.invokeFunction("Count")).intValue();

}
```

The MegaCOMObject can handle ALL the vb script calls, unless they have too many parameters (the maximum is 6).

If a Java function seems not to behave as expected, note that all the MEGA API Java classes correspond directly or indirectly to a MegaCOMObject; then you can use the native call in all the cases.

11.17 Duplicating an object or a building block

11.17.1 Duplicating an object

```
Function CreateDuplicate (vToDuplicate As Variant, sPrefix As String,
[vOperator] As Variant, vLibrary As String, vAffixOption As String) As
MegaObject
```

The **CreateDuplicate** function enables to duplicate a given object and its sub-objects defined by a Mega operator. It returns the duplicated root object.

`vToDuplicate` is the name of the duplicated root object. It can be also the internal identifier of an existing object (the identifier is obtained thanks to the GetID function).

`sPrefix` - All duplicated sub-objects short name can be prefixed by the value of this parameter.

`vOperator` – This optional parameter indicates which operator is used to explore the data to duplicate. By default, the Duplicate operator is used.

`vLibrary` – Library owning the duplicated objects

`vAffixOption` – Option that defines if it is a suffix or prefix (by default: suffix)

11.17.2 Duplicating a building block (HOPEX IT Architecture specific)

```
Function SmartDuplicate(ByVal sOptions As String) As MegaObject
```

The **SmartDuplicate** function enables to duplicate or create a new version of a given building block and its components and other description blocks. It returns the created object.

Parameters :

`sOptions`: Options to pass to the duplication tool in the "K1=V1,K2=V2" format.

Possible values:

`Root=[NewCopy|NewVersion]` Default: `NewCopy`

`Components=[Keep|NewCopy|Ignore]` Default: `Keep`

`Boundaries=[Keep|Ignore]` Default: `Keep`

`Scenarios=[Keep|Ignore]` Default: `Keep`

`Environments=[Keep|Ignore]` Default: `Keep`

Example:

```
Set newObject =
myApplication.SmartDuplicate("Root=NewCopy,Components=Keep,Environments=Ignore")
```

11.18 Calling a URL construction function using APIs in HOPEX

You can build a URL that enables to login to HOPEX in a specific context. This URL can be used for example in an email. The URL enables to avoid the user to select the web site and login information (for example: environment, repository, profile). It also enables to launch a tool on a specific object.

Note that the user still needs to authenticate.

You can call the URL construction function (**DesktopURLBuild** or **DesktopURLBuildEx**) on a MegaRoot object. The prototype is as follows:

```
Function DesktopURLBuild(sParameterization as string, sEnvironment as string,
sRepository as string, sApplication as string, sDesktop as string, sProfileId as
string, sGroupId as string)
```

sParameterization parameter is mandatory.

sEnvironment, sApplication, sDesktop, sProfileId are optional, but are all mandatory when one of them is specified.

sGroupId is optional, but when specified all of the other parameters must be specified as well.

If you want to target a web application use:

```
Function DesktopURLBuildEx(sWebApplicationPath as string, sParameterization as
string, sEnvironment as string, sRepository as string, sApplication as string,
sDesktop as string,sProfileId as string, sGroupId as string)
```

where `sWebApplicationPath` enables to define the web application path you want to target.

`sParameterization` defines the tool to be launched as follows:

`Tool:` tool idabs

`Param:` tool parameter idabs

`ParamValue:` parameter value – object idabs if it is a MEGA object

`ParamValueMetaclass:` MetaClass idabs if the parameter value is a MEGA object

`Affinity:` Affinity idabs

`Tool:` tool idabs

`sEnvironment` equals:

- empty if you want the end user to select an environment at connection,
- 0 if you want to connect to the same environment as the one from which the link is created,
- the environment path if you want to access to a specific environment

  `Example: "C:\Users\Public\Documents\HOPEX V2R1\My environment"`

`sRepository` equals:

- empty if you want the end user to select a repository at connection,

- 0 if you want to connect to the same repository as the one from which the link is created,
- the repository name if you want to access to a specific repository

  `Example: “Soho”`

`sApplication` equals:

- empty if you want the end user to select a MEGA application at connection,
- 0 if you want to connect to the same MEGA application as the one from which the link is created,
- the MEGA application name if you want to access to a specific application

  `Example: “MEGA Teamwork”`

`sDesktop` equals:

- 0 if you want to connect to the same desktop as the one from which the link is created,
- the desktop Hexaidabs if you want to access to a specific desktop

  `Example: “FB6CFCE14F6A1D78” (for Teamwork Desktop)`

`sProfileId` equals:

- empty if you want the end user to select a profile at connection,
- 0 if you want the user to use the same profile at connection from which the link is created,
- the profile absolute identifier if you want to connect with a specific profile.

  `Example: “BaeBawooG9GP” (for Teamwork user profile)`

`sGroupId` equals:

- empty if you want the end user to select a group at connection,
- 0 if you want the user to use the same group at connection from which the link is created,
- the Group absolute identifier followed if you want to connect with a specific group.

Is mandatory if the user does not have any assigned profile else than the one through the group.

`Example: 1ehG9JqJGv(Q (for Guest group)`

Example:

```
Dim strWebApplicationPath

strWebApplicationPath = GetRoot.CurrentEnvironment.GetUserParameter("HOPEX",
"WebApplicationPath2")

Dim strParam

strParam =
"Tool=4i(35ZHpEHhN,Param=KXcGkNdEF1bR,ParamValue=63HS0DNRAjD0,ParamValueMetaClas
s=PRjF04qYoSC0,Param=6RQQCjsNFT)M,ParamValue=test"


print GetRoot.DesktopURLBuildEx(strWebApplicationPath, strParam,0,0,"MEGA
TeamWork", "FB6CFCE14F6A1D78", "BaeBawooG9GP", "1ehG9JqJGv(Q")

print GetRoot().DesktopURLBuild(strParam, 0, 0, "MEGA Assessment
Execution","54515BB55085087B", "", "1ehG9JqJGv(Q")
```

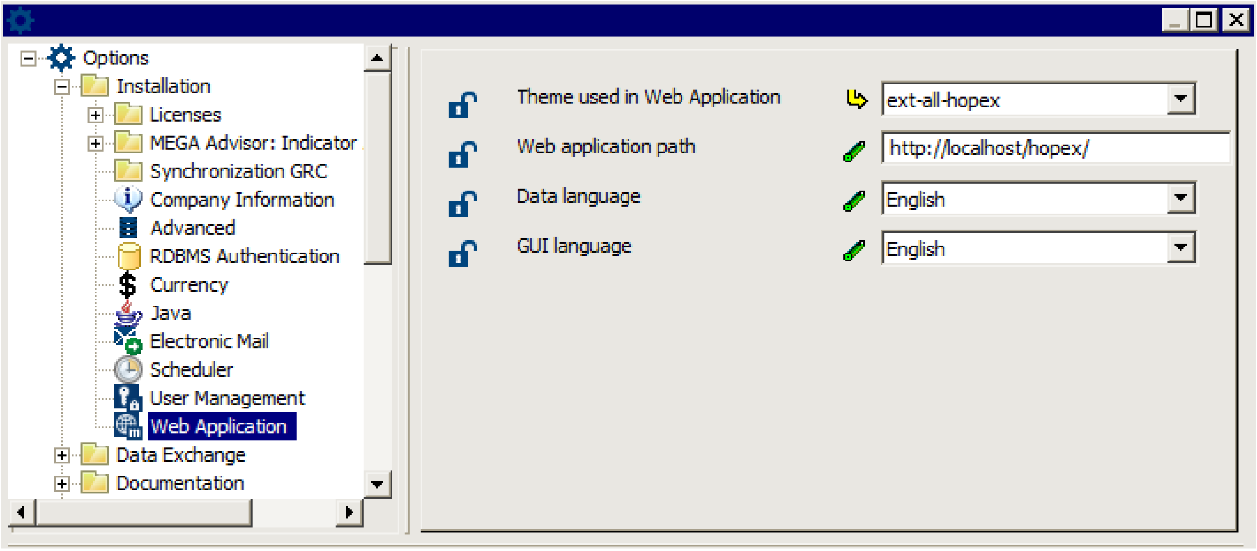
With:

`sEnvironment: 0 = current environment`

```
sRepository: 0 = current repository
sApplication: Mega Teamwork
sDesktop: FB6CFCE14F6A1D78 = Teamwork Desktop HexaIdAbs
sProfileId: BaeBawooG9GP = Teamwork user profile IdAbs
sGroupId: 1ehG9JqJGv(Q = Guest group Idabs

sEnvironment: 0 = current environment
sRepository: 0 = current repository
sApplication: Mega Assessment Execution
sDesktop: 54515BB55085087B = Risk Assessment Desktop HexaIdAbs
sProfileId: "" = no profile defined, the user must select a profile at connection
sGroupId: 1ehG9JqJGv(Q = Guest group Idabs
```

11.18.1 Code examples and results

Note: The **Web application path** must be entered in **Options > Installation > Web Application**.



Theme used in Web Application: ext-all-hopex

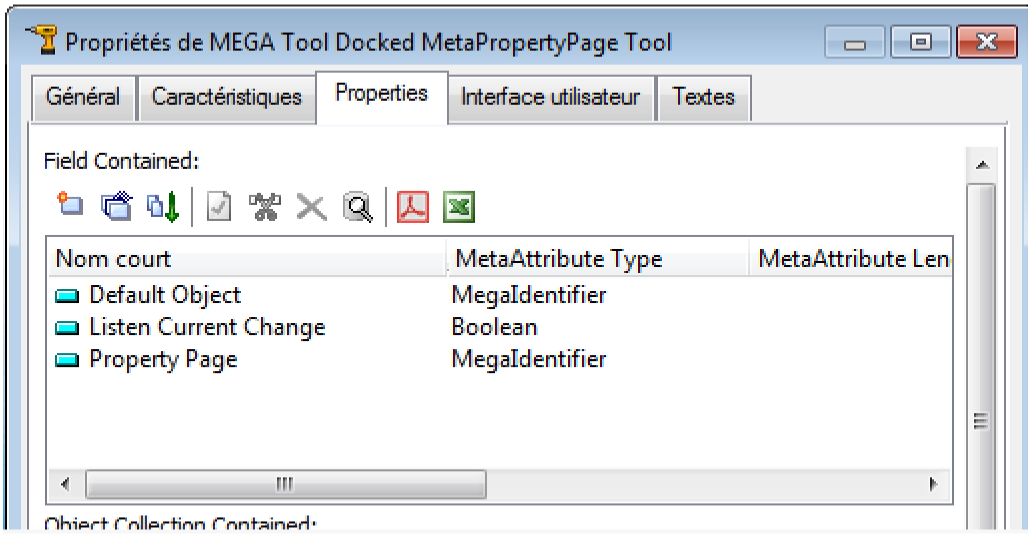
Web application path: http://localhost/hopex/

Data language: English

GUI language: English

Example 1

Display the property page tool and associate « Default object » parameter with the OIZYiqujArn0 object (Advisor Client) of type MrUiM9B5iyM0 (Application).


Propriétés de MEGA Tool Docked MetaPropertyPage Tool
Général
Caractéristiques
Properties
Interface utilisateur
Textes
Field Contained:
Object Collection Contained:

Nom court	MetaAttribute Type	MetaAttribute Len
Default Object	MegaIdentifier	
Listen Current Change	Boolean	
Property Page	MegaIdentifier	

Code:

VB Script

```
print GetRoot.DesktopURLBuildEx("http://localhost/HOPEX/",
"Tool=Sj(3hVHpEXUN,Param=VHRbsTkZFLvH,ParamValue=OIZYiqujArn0,ParamValueMetaclas
s=MrUiM9B5iyM0 ")
```

Java

```
mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]",
mgobjMegaObject.getRoot().invokePropertyGet("DesktopURLBuild",
"Tool=Sj(3hVHpEXUN,Param=VHRbsTkZFLvH,ParamValue=OIZYiqujArn0,ParamValueMetaclas
s=MrUiM9B5iyM0 "));
```

Result:

http://localhost/HOPEX/default.aspx?userdata=Tool-Sj%283hVHpEXUN|VHRbsTkZFLvH-OIZYiqujArn0-MrUiM9B5i...

Exemple 2

1. Display the property page tool and associate « Default object » parameter with the OIZYiqujArn0 object (Advisor Client) of type MrUiM9B5iyM0 (Application).
2. Launch the diagram editor.
3. Launch macroLauncher tool with the macro that has the NiqwTtU5CPB0 identifier and has the « This is a test » additional parameter.

Code:

VB Script

```
print
GetRoot.DesktopURLBuildEx("http://localhost/HOPEX/","Tool=Sj(3hVHpEXUN,Param=VHR
bsTkZFLvH,ParamValue=OIZYiqujArn0,ParamValueMetaclass=MrUiM9B5iyM0,Tool=Fk(3zVHp
EXWN,Tool-
4i(35ZHpEHhN,Param=KXcGkNdEF1bR,ParamValue=NiqwTtU5CPB0,Param=6RQQCjsNFT)M,Param
Value=This is a test ")
```

Java

```
mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]",
mgobjMegaObject.getRoot().invokePropertyGet("DesktopURLBuild", "
Tool=Sj(3hVHpEXUN,Param=VHRbsTkZFLvH,ParamValue=OIZYiqujArn0,ParamValueMetaclass
=MrUiM9B5iyM0,Tool=Fk(3zVHpEXWN,Tool-
4i(35ZHpEHhN,Param=KXcGkNdEF1bR,ParamValue=NiqwTtU5CPB0,Param=6RQQCjsNFT)M,Param
Value=This is a test "));
```

Result:

http://localhost/HOPEX/default.aspx?userdata=Tool-Sj%283hVHpEXUN|VHRbsTkZFLvH-OIZYiqujArn0-MrUiM9B5i...

Example 3

1. Display the same tools as in the previous example.
2. Reuse the same environment, the same repository, the same application and the same desktop.

Code:

VB Script

```
print GetRoot. DesktopURLBuildEx("http://localhost/HOPEX/",
"Tool=Sj(3hVHpEXUN,Param=VHRbsTkZFLvH,ParamValue=OIZYiqujArn0,ParamValueMetaclas
s=MrUiM9B5iyM0,Tool=Fk(3zVHpEXWN,Tool-
4i(35ZHpEHhN,Param=KXcGkNdEF1bR,ParamValue=NiqwTtU5CPB0,Param=6RQQCjsNFT)M,Param
Value=This is a test ",0,0,0,0,0)
```

Java

```
mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]",
mgobjMegaObject.getRoot().invokePropertyGet("DesktopURLBuild", "
Tool=Sj(3hVHpEXUN,Param=VHRbsTkZFLvH,ParamValue=OIZYiqujArn0,ParamValueMetaclass
=MrUiM9B5iyM0,Tool=Fk(3zVHpEXWN,Tool-
4i(35ZHpEHhN,Param=KXcGkNdEF1bR,ParamValue=NiqwTtU5CPB0,Param=6RQQCjsNFT)M,Param
Value=This is a test ", "0", "0", "0", "0", "0"));
```

Result:

http://localhost/HOPEX/default.aspx?Desktop=CCCCCC
CCCCCCCCCC&Db=Adventure&Env=C%3A%5cProgram%20Files...

Example 4

1. Display the same tools as in the previous examples.
2. Reuse the same environment and the same repository.
3. Change the application for «Enterprise Risk Management » application.
4. Use the “risk” desktop with "CF2FD0F14FC556D4" identifier.
5. Connect with « Other Participant in Audit » profile that has s28FH6qOG9qC identifier and the role « Other Participant in Audit » that has T28F56qOGLpC identifier.

Code:

VB Script

```
print GetRoot. DesktopURLBuildEx("http://localhost/HOPEX/",
"Tool=Sj(3hVHpEXUN,Param=VHRbsTkZFLvH,ParamValue=OIZYiqujArn0,ParamValueMetaclas
s=MrUiM9B5iyM0,Tool=Fk(3zVHpEXWN,Tool-
4i(35ZHpEHhN,Param=KXcGkNdEF1bR,ParamValue=NiqwTtU5CPB0,Param=6RQQCjsNFT)M,Param
Value=This is a test ",0,0,"Enterprise Risk Managment", "CF2FD0F14FC556D4",
"T28F56qOGLpCs28FH6qOG9qC")
```

Java

```
mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]",
mgobjMegaObject.getRoot().invokePropertyGet("DesktopURLBuild", "
Tool=Sj(3hVHpEXUN,Param=VHRbsTkZFLvH,ParamValue=OIZYiqujArn0,ParamValueMetaclass
=MrUiM9B5iyM0,Tool=Fk(3zVHpEXWN,Tool-
4i(35ZHpEHhN,Param=KXcGkNdEF1bR,ParamValue=NiqwTtU5CPB0,Param=6RQQCjsNFT)M,Param
Value=This is a test ", 0, 0, "Enterprise Risk Managment", "CF2FD0F14FC556D4",
"T28F56qOGLpCs28FH6qOG9qC"));
```

Result:

http://localhost/hopex/default.aspx/Enterprise Risk Managment?Desktop=z)opn3TnFHjL&from=RWP&Db=0GXuZ9UmHj4Q&Env=xG8xqoktHfgK&Profile=T28F56qOGLpCs28FH6qOG9qC&userdata=,%20Tool|VHRbsTkZFLvH-OIZYiqujArn0-MrUiM9B5iyM0Tool-Fk%283zVHpEXWN,Tool%2d4i%2835ZHpEHhN|KXcGkNdEF1bR-NiqwTtU5CPB0|6RQQCjsNFT%29M-This%20is%20a%20test%20

The new desktop is taken into account as well as the new application.

Example 5

1. Display the same tools as in the previous examples.
2. Specify "C:\Users\Public\Documents\MEGA 2012\Demonstration" environment and "DemoERM" repository.
3. Change the application for «Enterprise Risk Managment » application.
4. Use the desktop with "CF2FD0F14FC556D4" identifier.
5. Connect with « Other Participant in Audit » profile that has s28FH6qOG9qC identifier and the role « Other Participant in Audit » that has T28F56qOGLpC identifier.

Code:

VB Script

```
print GetRoot. DesktopURLBuildEx("http://localhost/HOPEX/",
"Tool=Sj(3hVHpEXUN,Param=VHRbsTkZFLvH,ParamValue=OIZYiqujArn0,ParamValueMetaclas
s=MrUiM9B5iyM0,Tool=Fk(3zVHpEXWN,Tool-
4i(35ZHpEHhN,Param=KXcGkNdEF1bR,ParamValue=NiqwTtU5CPB0,Param=6RQQCjsNFT)M,Param
Value=This is a test ","C:\Users\Public\Documents\MEGA
2012\Demonstration","DemoERM","Enterprise Risk Managment", "CF2FD0F14FC556D4",
"T28F56qOGLpCs28FH6qOG9qC")
```

Java

```
mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]",
mgobjMegaObject.getRoot().invokePropertyGet("DesktopURLBuild", "
Tool=Sj(3hVHpEXUN,Param=VHRbsTkZFLvH,ParamValue=OIZYiqujArn0,ParamValueMetaclass
=MrUiM9B5iyM0,Tool=Fk(3zVHpEXWN,Tool-
4i(35ZHpEHhN,Param=KXcGkNdEF1bR,ParamValue=NiqwTtU5CPB0,Param=6RQQCjsNFT)M,Param
Value=This is a test ", "C:\\Users\\Public\\Documents\\MEGA
2012\\Demonstration", "DemoERM", "Enterprise Risk Managment",
"CF2FD0F14FC556D4", "T28F56qOGLpCs28FH6qOG9qC"));
```

Result:

http://localhost/hopex/default.aspx/Enterprise Risk Managment?Desktop=z)opn3TnFHjL&from=RWP&Db=DemoERM&Env=C%3A%5cUsers%5cPublic%5cDocuments%5cMEGA%202012%5cDemonstration&Profile=T28F56qOGLpCs28FH6qOG9qC&userdata=,%20Tool|VHRbsTkZFLvH-OIZYiqujArn0-MrUiM9B5iyM0Tool-Fk%283zVHpEXWN,Tool%2d4i%2835ZHpEHhN|KXcGkNdEF1bR-NiqwTtU5CPB0|6RQQCjsNFT%29M-This%20is%20a%20test%20

All of the changes are taken into account in the url.

11.19 Calling a macro from HTML, code and RTF descriptors

11.19.1 HTML and code descriptors

The way to call a macro from an HTML or Code descriptor is to use the [ExternalCall/] Tag.

```
[ExternalCall Macro=MyMacroID]
MyUserData
[/ExternalCall]
```

This syntax automatically calls the macro named “MyMacroID” with the following prototypes:

```
Sub Generate(oObject , oContext, sUserData, sResult)
End Sub
```

`oObject` is the current object in the descriptor.

`oContext` is the generation context

`sUserData` is the string contained in the [ExternaCall/] tag, in this sample, it is “MyUserData”.

`sResult` contains the string to be returned to the descriptor.

Or

```
Sub GenerateStream(oObject , oContext, sUserData, oStream)
End Sub
```

`oObject` is the current object in the descriptor

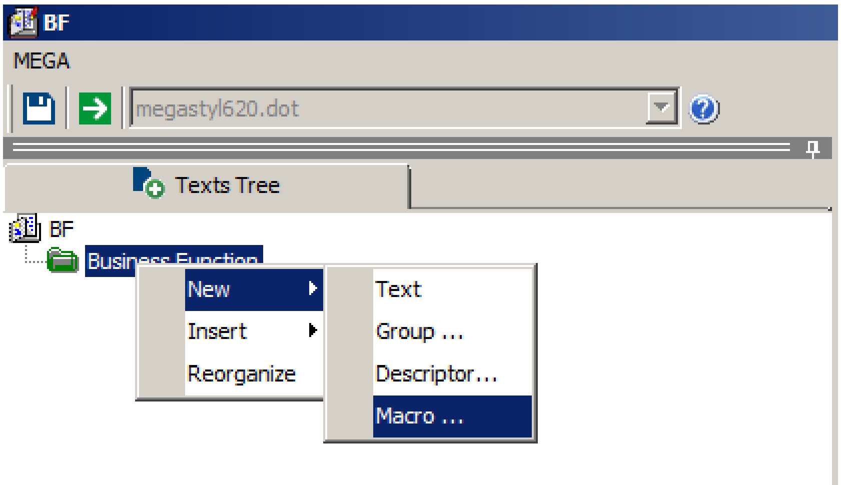
`oContext` is the generation context

`sUserData` is the string contained in the [ExternaCall/] tag, in this sample it is “MyUserData”.

`oStream` is a TextStream containing the string to be returned in the descriptor.

11.19.2 RTF Decriptors

The RTF descriptor may include a new macro:



The macro is created and instanciated with the following prototype:

```
Sub Generate(oObject, oContext, sUserData, sResult)
End Sub
```

`oObject` is the current object in the descriptor.

`oContext` is the generation context

`sUserData` is empty at the moment

`sResult` contains the string to be returned to the descriptor.

11.20 Calling an operator

callMethod and **callFunction** are functions used to call operators (_Operator) and Mega macros. These functions are based on Mega IdAbs. With these functions MEGA recommends that you use the MegaField of the operator, in order to ensure your code upgradability.

invokeMethod and **invokeFunction** are java transcriptions of IDispatch ::GetIdsOfNames and IDispatch ::Invoke. They are accessible to any interop objects to call any method or function. They can be used to call operators but by name (do not use this with name because if the name of the operator changes your code will not work anymore).

11.20.1 Calling a method (message box display)

You do not expect a return from the method.

Example: message box display

VB Script

```
oRoot.MessageBox("Error: Report Parameter is Misconfigured!")
```

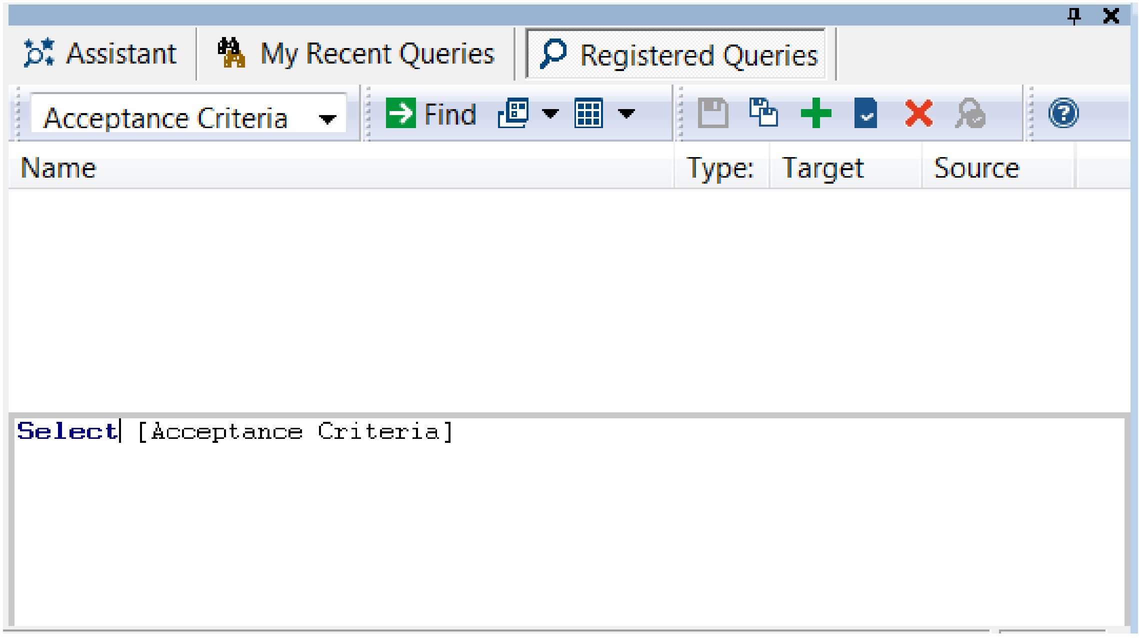
Java

```
MegaRoot.CallMethod("MessageBox” , "Error: Report Parameter is
Misconfigured!")
```

11.20.2 Calling a function (RequestQuery)

The function returns some content.

Example: the function launches the Query window and returns the selected instances.



```
Select [Acceptance Criteria]
```

VB Script

```
Dim mc As MegaCollection
Set mc = Root. RequestQuery
```

Java

```
MegaCollection    pMegaCollection ;
pMegaCollection  = Root. CallFunction(“RequestQuery”)
```

11.20.3 Calling a function (RegulationApply)

With the **RegulationApply** API the **Check > Regulation with propagation** command is available on an object in HOPEX (Web Front-End):

- The **RegulationApply** function returns an informal object including the result.
- You can customize the standard report displaying the results of this test.

To use the **RegulationApply** API, you need to apply ".RegulationApply” on the object you want to be checked and use the required **Modeling Regulation** parameter.

The processing consists in an extraction from the root object to be checked and in applying the Modeling regulation rules to each candidate item.

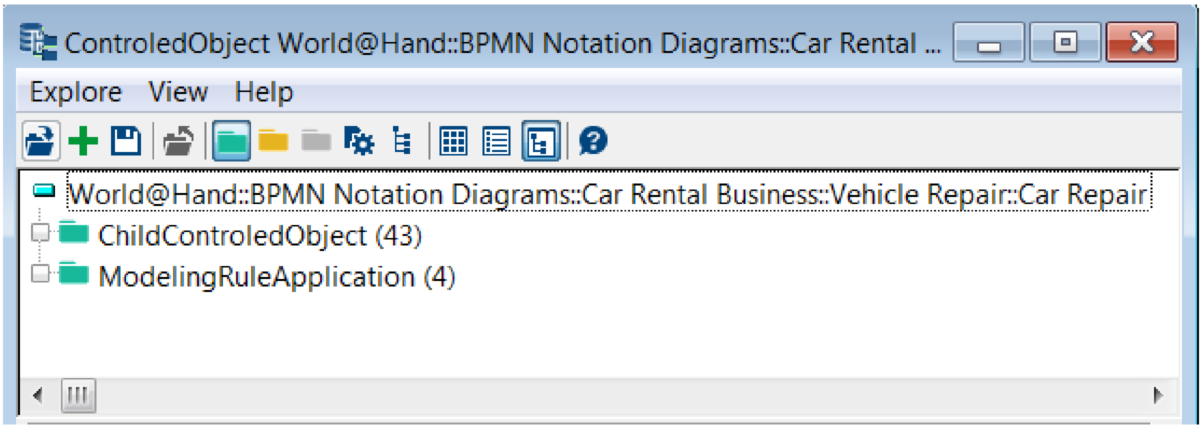
VB Script

```
Const cstrObjectToValidate = "~rpgNUz5T9570[Car Repair]"
Const cstrModelingRegulation = "~IdCWh3eh9T20[BPMN regulation]"

Dim mgobjObjectToValidate
Set mgobjObjectToValidate = GetObjectFromId(cstrObjectToValidate)

Dim mgobjModelingRegulation
```

```
Set mgobjModelingRegulation = GetObjectFromId(cstrModelingRegulation)


Dim mgobjResult

Set mgobjResult =
mgobjObjectToValidate.RegulationApply(mgobjModelingRegulation)


mgobjResult.Explore
```

Java

```
String cstrObjectToValidate = "~rpgNUz5T9570[Car Repair]"

String cstrModelingRegulation = "~IdCWh3eh9T20[BPMN regulation]"

MegaObject mgobjObjectToValidate = megaRoot.getObjectFromID(
cstrObjectToValidate)

MegaObject mgobjModelingRegulation = megaRoot.getObjectFromID
(cstrModelingRegulation)

final MegaObject mgobjResult = (MegaObject)
mgobjObjectToValidate.callFunction("RegulationApply",mgobjModelingRegulation
);
```


ControledObject World@Hand::BPMN Notation Diagrams::Car Rental ...
Explore
View
Help

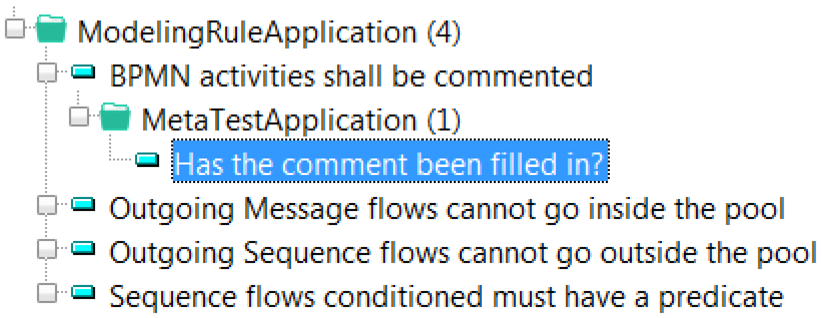
```
World@Hand::BPMN Notation Diagrams::Car Rental Business::Vehicle Repair::Car Repair
    ChildControledObject (43)
    ModelingRuleApplication (4)
```

The **Root object** is the object on which the **RegulationApply** function is invoked.

In the above example:

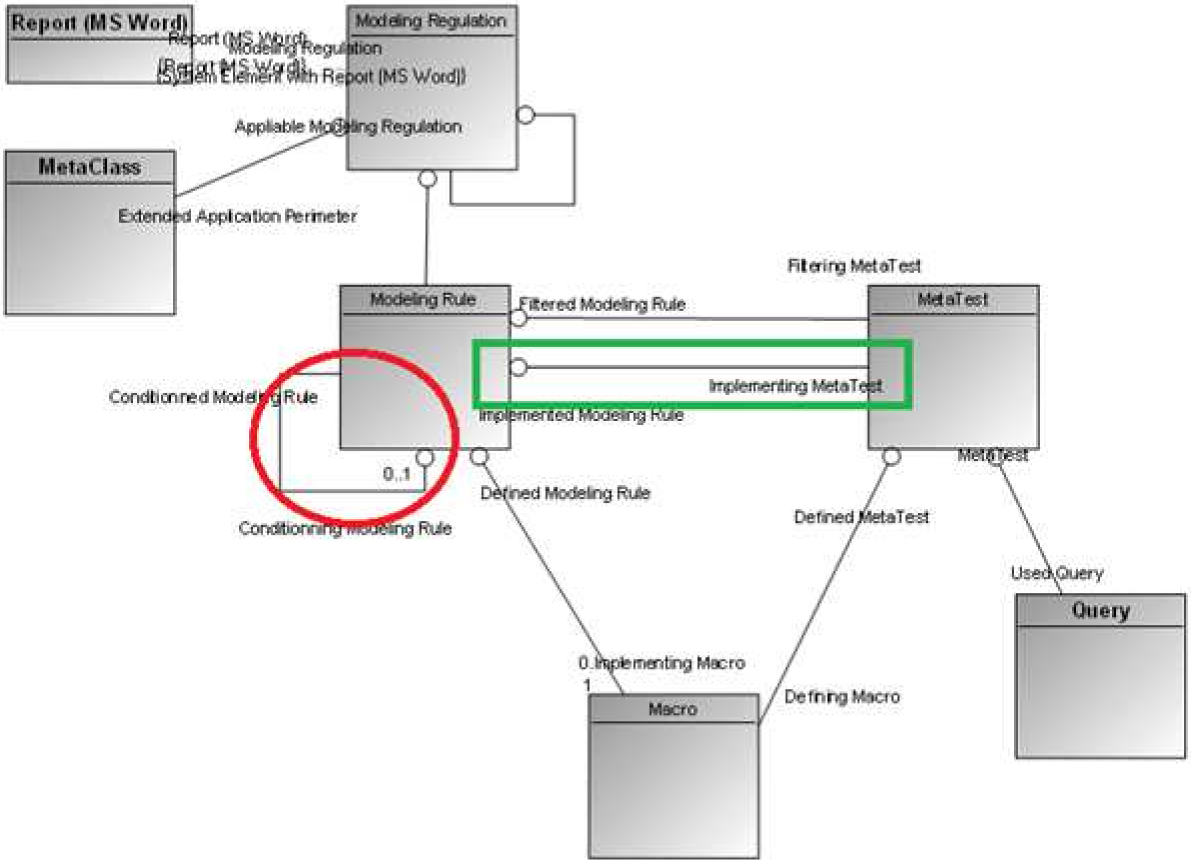
- The root object "ControledObject" holds a set of MEGA objects (43 ControledObjects), which are extracted from the root object to apply the regulations.
- Each "ControledObject" holds a **ModelingRuleApplication** for each ModelingRule applied to this MEGA object.
- Each **ModelingRuleApplication** can include:
    - one or several **MetaTestApplications**

      There are as many MetaTestApplications as there are MetaTests needed to define the modeling rule test.

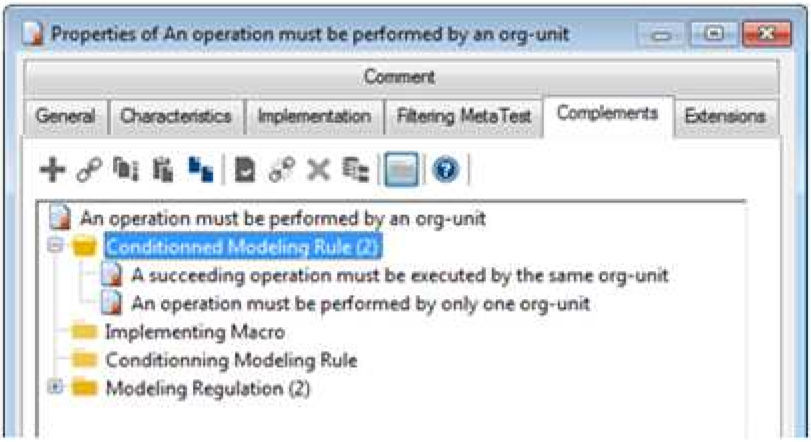


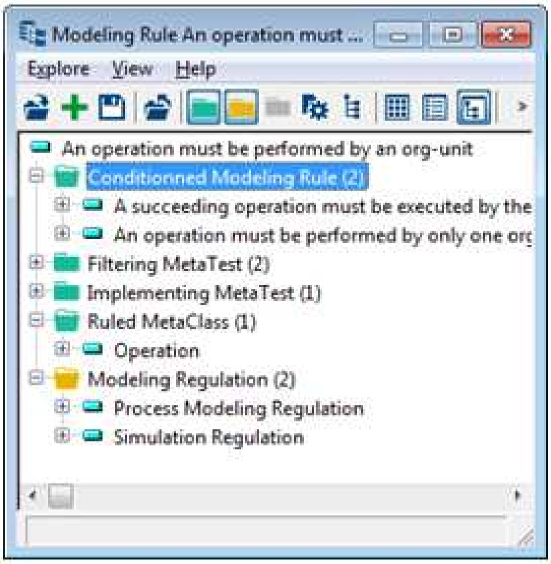
- ModelingRuleApplication (4)
  - BPMN activities shall be commented
    - MetaTestApplication (1)
      - Has the comment been filled in?
  - Outgoing Message flows cannot go inside the pool
  - Outgoing Sequence flows cannot go outside the pool
  - Sequence flows conditioned must have a predicate

- one or several **ChildModelingRuleApplications** (defined as “Conditioning Modeling Rule” in the following Regulation MetaModel diagram)



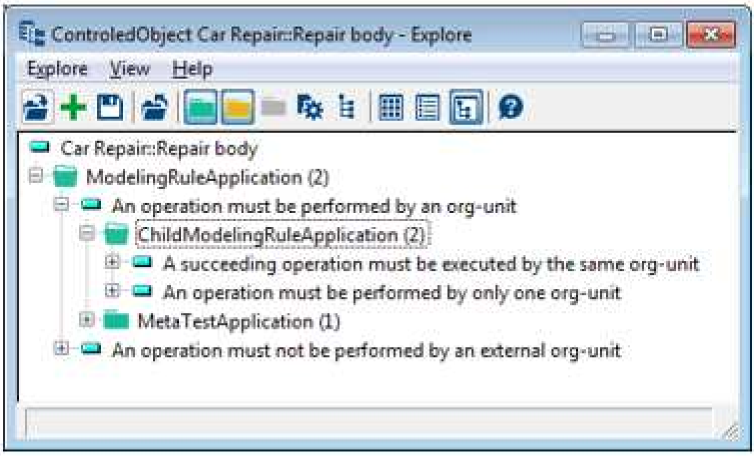
When the **ModelingRule** includes **Conditionned Modeling Rules**, the **ModelingRuleApplication** can hold other **ModelingRuleApplications**.


Properties of An operation must be performed by an org-unit
Comment
General
Characteristics
Implementation
Filtering MetaTest
Complements
Extensions
An operation must be performed by an org-unit
Conditionned Modeling Rule (2)
A succeeding operation must be executed by the same org-unit
An operation must be performed by only one org-unit
Implementing Macro
Conditionning Modeling Rule
Modeling Regulation (2)


Modeling Rule An operation must ...
Explore
View
Help
An operation must be performed by an org-unit
Conditionned Modeling Rule (2)
A succeeding operation must be executed by the
An operation must be performed by only one org
Filtering MetaTest (2)
Implementing MetaTest (1)
Ruled MetaClass (1)
Operation
Modeling Regulation (2)
Process Modeling Regulation
Simulation Regulation

VB Script

```
Const cstrObjectToValidate = "~GhsmIDTJ)qU1[Repair body]"
Const cstrModelingRegulation = "~ga1YNZOP1H80[Process Modeling Regulation]"

Dim mgobjObjectToValidate
Set mgobjObjectToValidate = GetObjectFromId(cstrObjectToValidate)

Dim mgobjModelingRegulation
Set mgobjModelingRegulation = GetObjectFromId(cstrModelingRegulation)

Dim mgobjResult
Set mgobjResult =
mgobjObjectToValidate.RegulationApply(mgobjModelingRegulation)

mgobjResult.Explore
```



- Car Repair::Repair body
  - ModelingRuleApplication (2)
    - An operation must be performed by an org-unit
      - ChildModelingRuleApplication (2)
        - A succeeding operation must be executed by the same org-unit
        - An operation must be performed by only one org-unit
      - MetaTestApplication (1)
    - An operation must not be performed by an external org-unit

11.21 Using MEGA identifiers in the code (Java, VBScript, others)

Here are some useful reminders regarding the use of MEGA identifiers via APIs.

11.21.1 "Physical" type of MEGA identifiers via APIs

You never:

- need to know the data « physical » type of a MEGA identifier.

  e.g.: int, long, float, this physical type that can change from a MEGA release to another.

- have to interpret directly a MEGA identifier that you got from a function of APIs.

In Java:

- all the functions that return an identifier return the Object type

  `e.g.: mgobjMegaObject.getID();`

- any function parameter used to receive a MEGA identifier is of Object type

  `e.g.: mgobjMegaObject.getRoot().currentEnvironment().toolkit().fieldID(Object ID);`

In VB Script &colon;

- all the functions that return an identifier return the Variant type

  `e.g.: MegaObject.getId()`

- any function parameter used to receive a MEGA identifier is of Variant type

  `e.g.: MegaToolKit.fieldID(Vairiant ID)`

Naming convention of a variable that includes a MEGA identifier:

`mgid<etc>`

`e.g.: mgidAttribute, mgidMetaClass, mgidApplication`

11.21.2 Handling identifiers in their « physical » form

Any operation on a MEGA identifier in its “physical” form must be performed through APIs.

Examples:

- Comparing two identifiers:

  `MegaToolKit.sameID(...)`

- Comparing a MegaObject identifier:

  `MegaObject.sameID(...)`

- Identifier transformations:

  ```
  - MegaToolKit.getString64FromID(...)
  - MegaToolKit.getStringFromID(...)
  - MegaToolKit.getIDFromString(...)
  ```

11.21.3 MegaFields

Megafields are character strings that enable object identification while including identified object name.

⚠ **Do not parse megafield (unless absolute need)**

A megafield might not include the identified object name

```
e.g.: a megafield obtained via MegaObject.megaUnnamedField()
```

It is recommended to declare the constants that correspond to identifiers in megafield form.

This enables to carry out impact analysis of objects used in VB Script and Java macros (via the reference link for example).

As a megafield is a string, the variable that includes a megafield starts with "str".

```
e.g.: "strObjectMegaField".
```

11.21.4 MEGA identifier formats

You can handle MEGA identifiers with different format in your code:

Object / Variant (physical form)

This is what you obtained when you enter:

```
> MegaObject.getID()
> MegaObject.getProp("Att","INTERNAL")
```

Where Att is a "Mega Identifier" type attribute, for example Att = "~310000000D00[Absolute Identifier]"

string in numerical format in base 64

This is what you obtained when you enter:

> MegaObject.getProp("Att")

Where Att is a "Mega Identifier" type attribute, for example Att = "~310000000D00[Absolute Identifier]"

Identifiers in MGL files, MGR files, etc.

string in numerical format in base 16 (hexadecimal)

This is what you obtained when you enter:

> MegaObject.getProp("~H20000000550[_HexaIdAbs]")

> MegaToolKit.getStringFromID(...)

Identifiers in XMG files

string MegaField

This is what you obtained when you enter:

> MegaToolKit.fieldID(Object ID)

> MegaObject.megaField()

> MegaObject.megaUnnamedField()

Some function parameters can receive a value in any type amoung the following ones:

Examples:

- MegaRoot.getObjectFromID(Object)
- MegaObject.getProp(Object)
- MegaObject.sameID(Object)
- ...

11.21.5 Bad practice examples

Do not manage on your own:

- identifier comparisons
- conversions of types of variables including identifiers.
- name retrieval in megafields

In only a few particular cases you need to parse megafields.

Do not write:

```
  /**
   * Tells if two megaField having the same Id Abs.
   * @param megaField1
   * @return
   */
  @SuppressWarnings("javadoc")
  public static boolean sameIdAbs(final String megaField1, final String
megaField2) {
    if ((megaField1 == null) || (megaField2 == null)) {
      return false;
    }
    String id1 = Pouet.getIdAbsFromMegaField(megaField1);
    String id2 = Pouet.getIdAbsFromMegaField(megaField2);

    return id1.equals(id2);

  }

  /**
   * Get ID Abs from Mega Field
   * @param megaField megaField
   * @return Id Abs
   */
  public static String getIdAbsFromMegaField(final String megaField) {
    if ((megaField != null) && (megaField.length() > 13)) {
      return megaField.substring(1, 13);
    }
    return "";

  }

  /**
   * Get Name contained into megafield Example: for
   * "~a20000000I20[MetaAttributeValue]" results on "MetaAttributeValue"
   * @param megaField megafield input
   * @return Name extracted
   */
  public static String getNameFromMegaField(final String megaField) {
    int index1 = megaField.indexOf("[");
    int index2 = megaField.lastIndexOf("]");
    String result = "";
    result = megaField.substring(index1 + 1, index2);
    return result;
```

11.22 Getting the parameter value of a query using an API

To get the parameter value of a parameter included in a query, use:

```
CurrentEnvironment.QueryParameters.ParameterValueGet("<parameter name>")
```

`<parameter name>` is the name of the parameter in the query

11.23 Using macros to add calculated attributes

You can:

- define new attributes or parameters for a MEGA object.
- determine value read and save modes for this parameter using MEGA macros.

To calculate an attribute by creating a macro:

1. Open the **Explorer** on the new attribute or new parameter.
2. Right-click the attribute (or parameter) and select **New > Macro**.

   The macro creation wizard appears.
3. Select **Create Macro (VB)Script**.

4. Click **Next**.
5. (Optional) Modify the default Name ("AttributeName".Macro) of your macro.

A macro is an object containing a VB Script code sequence interpreted at execution.

6. Click **Finish**.

Edit the "AttributeName".Macro macro and note that the VB Script contains in particular the following functions:

If they are not present, standard implementation is selected.

- **GetAttributeValue(ByVal Object, ByVal AttributeID, ByRef Value)**

  Defines attribute access mode. The parameters are:

  - Object: corresponds to the object of which the attribute value is requested.
  - AttributeID: absolute identifier of the attribute (or taggedValue).
  - Value: the attribute value returned by the function, concerning this object.

- **SetAttributeValue(ByVal Object, ByVal AttributeID, ByVal Value)**

  Defines attribute save mode. The parameters are:

  - Object: corresponds to the object of which the attribute value must be updated.
  - AttributeID: absolute identifier of the attribute (or taggedValue).
  - Value: the function saves the attribute value for this object.

The attribute nature (_AtNature) should be Virtual.

For both of these functions, attribute change mode is a character string. Conversion must be carried out to change text format to the internal format of the attribute.

Example:

VB Script

```
Sub GetAttributeValue (ByVal object, ByVal AttID, Value)

' internal value reading in integer format. numValue =
CInt(objet.GetProp(AttID, "Physical"))

if numValue < 20 then

Value = "Young"

elseif numValue < 35 then

Value = "Youthful"

elseif numValue < 55 then

Value = "Mature"

else

Value = "Elderly"

end if

End Sub
```

Java

```
public void getAttributeValue(final MegaObject mgobjObject, final Object
AttID, final StringBuffer value) {

// internal value reading in integer format.

String strResult = "";

int iNumValue = (Integer) mgobjObject.getProp(AttID, "Physical");
if (iNumValue < 20) {
strResult = "Young";
} else if (iNumValue < 35) {
strResult = "Youthful";
} else if (iNumValue < 55) {
strResult = "Mature";
} else {
strResult = "Elderly";
}
value.append(strResult);
}
```

You can directly implement read-only and read/write access in the attribute format (without passing via standard conversion).

In this case, you must implement the following two functions, of which prototypes are similar to those above:

VB Script

```
GetExtendedAttributeValue(ByVal Format as LONG, ByVal Object, ByVal
AttributeID, ByRef Value)
```

Java

```
public void getExtendedAttributeValue(final int[] intFormat, final MegaObject
mgobjObject, final Double dAttributeID, final StringBuffer strValue) {}
```

VB Script

```
SetExtendedAttributeValue(ByVal Format as LONG, ByVal Object, ByVal
AttributeID, ByVal Value))
```

Java

```
public void setExtendedAttributeValue(final int[] intFormat, final MegaObject
mgobjObject, final Double dAttributeID, final StringBuffer strValue) {}
```

The difference is in the additional parameter: Format. The possible values are:

- 0 internal: value in internal format (binary, integer,...)
- 1 external: value in external format, but before display processing (certain objects have external form that is textual with the addition of index identifiers, as for class attributes or association roles).
- 3 Display: value in external format used in Web sites or Word documents (expurgated when identifiers in external format occur).

If one of the two extended functions is implemented, call by GetProp with "Physical" format on the same attribute is prohibited since it would lead to an infinite recursion.

11.24 Using macros to add rows/columns in a matrix

To add a row or a column to a matrix you need to implement a macro based on the following functions:

- Function **OnRowAdd(MatrixContext) As Boolean** (to add a row)
- Function **OnColumnAdd(MatrixContext) As Boolean** (to add a column)

If the function is not implemented ou returns the "True" value, the default process is called.

To manage (create/delete) the cells of the matrix, you can use the following functions:

```
Function OnCreateCell(Context,Row As MegaObject,Col As MegaObject) As Boolean
End function
Function OnDeleteCell(Context,Row As MegaObject,Col As MegaObject) As Boolean
End function
```

To call this macro:

1. Access the properties of the Matrix Template.
2. In its Text > _Parameterization enter:

```
[InsertManager]
Macro = <MacroID(field of the macro)>
```

11.25 Adding or launching a tool in HOPEX using APIs

You can use APIs to add or launch a ParameterizedTool using the following commands:

- **CreateParameterizedTool** to add a ParameterizedTool
- **CreateSuspendedParameterizedTool** to launch a ParameterizedTool

In both cases, the same functions are available to customize a ParameterizedTool, see Functions available to customize a ParameterizedTool.

11.25.1 Creating a ParameterizedTool

You can create a ParameterizedTool, using either a MegaField or an IdAbs:

- with a MegaField:

```
Set MyTool = oRoot.CreateParameterizedTool("~Sj(3hVHpEXUN[Docked
MetaPropertyPage Tool]")
```

- with an IdAbs

```
Set MyTool2 = oRoot.CreateParameterizedTool("MglQ)(cpEHE1")
```

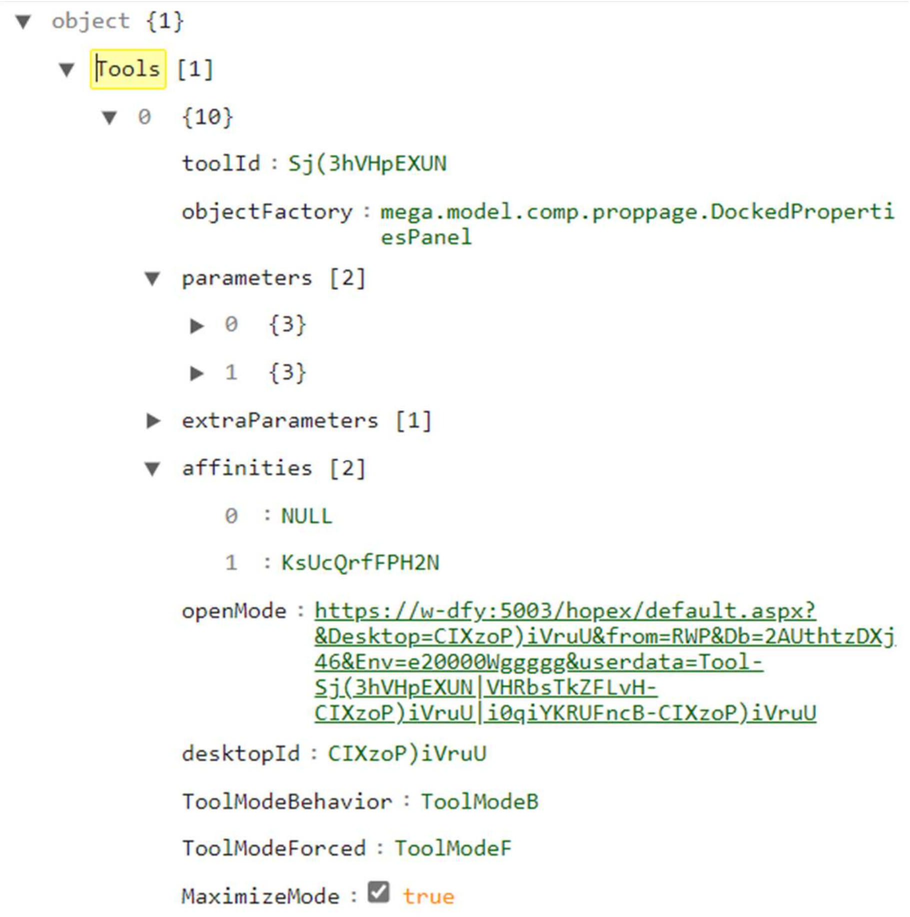
*Note: APIs using **CreateParameterizedTool** replace former APIs using **AddParameterizedTool**.*

Functions are available to customize the ParameterizedTool, see Functions available to customize a ParameterizedTool.

Example of code using functions available to customize the tool:

```
Sub mainRoot(oRoot)
Set MyTool = oRoot.CreateParameterizedTool("~Sj(3hVHpEXUN[Docked MetaPropertyPage Tool]")
MyTool.AddParameter "~CuyCQgJTGv2H[Hide status bar]" ,"1 "
MyTool.AddParameter "~i0qiYKRUFncB[Property Page]"," ~Va1s1gjx51A0[Documentation Report]"
MyTool.AddExtraParameter "~f1KFMsJeFDv2[Display Mode]",  "2"
MyTool.AddAffinity NULL
MyTool.AddAffinity "~7p33r0noIHSM[Search Affinity]"
MyTool.SetMaximizeMode TRUE
MyTool.SetOpenMode "NewTab", "~gV)MTqsnWreM[NewTab Desktop]"
MyTool.SetToolModeForced "A"
MyTool.SetToolModeBehavior "F"
oRoot.Print MyTool.GetJson
end Sub
```

This code enables to retrieve the following Json:



```
▼ object {1}
  ▼ Tools [1]
    ▼ 0 {10}
        toolId : Sj(3hVHpEXUN
        objectFactory : mega.model.comp.proppage.DockedPropertiesPanel
      ▼ parameters [2]
        ▶ 0 {3}
        ▶ 1 {3}
      ▶ extraParameters [1]
      ▼ affinities [2]
          0 : NULL
          1 : KsUcQrfFPH2N
        openMode : https://w-dfy:5003/hopex/default.aspx?&Desktop=CIXzoP)iVruU&from=RWP&Db=2AUthtzDXj46&Env=e20000Wggggg&userdata=Tool-Sj(3hVHpEXUN|VHRbsTkZFLvH-CIXzoP)iVruU|i0qiYKRUFncB-CIXzoP)iVruU
        desktopId : CIXzoP)iVruU
        ToolModeBehavior : ToolModeB
        ToolModeForced : ToolModeF
        MaximizeMode : ☑ true
```

11.25.2 Creating a Suspended ParameterizedTool

In the same way as creating a ParameterizedTool, you can create a ParameterizedTool in suspension mode.

You can create a Suspended ParameterizedTool as follow:

```
Set MyTool = oRoot.CreateSuspendedParameterizedTool("~Sj(3hVHpEXUN[DockedMetaPropertyPage
Tool]")
```

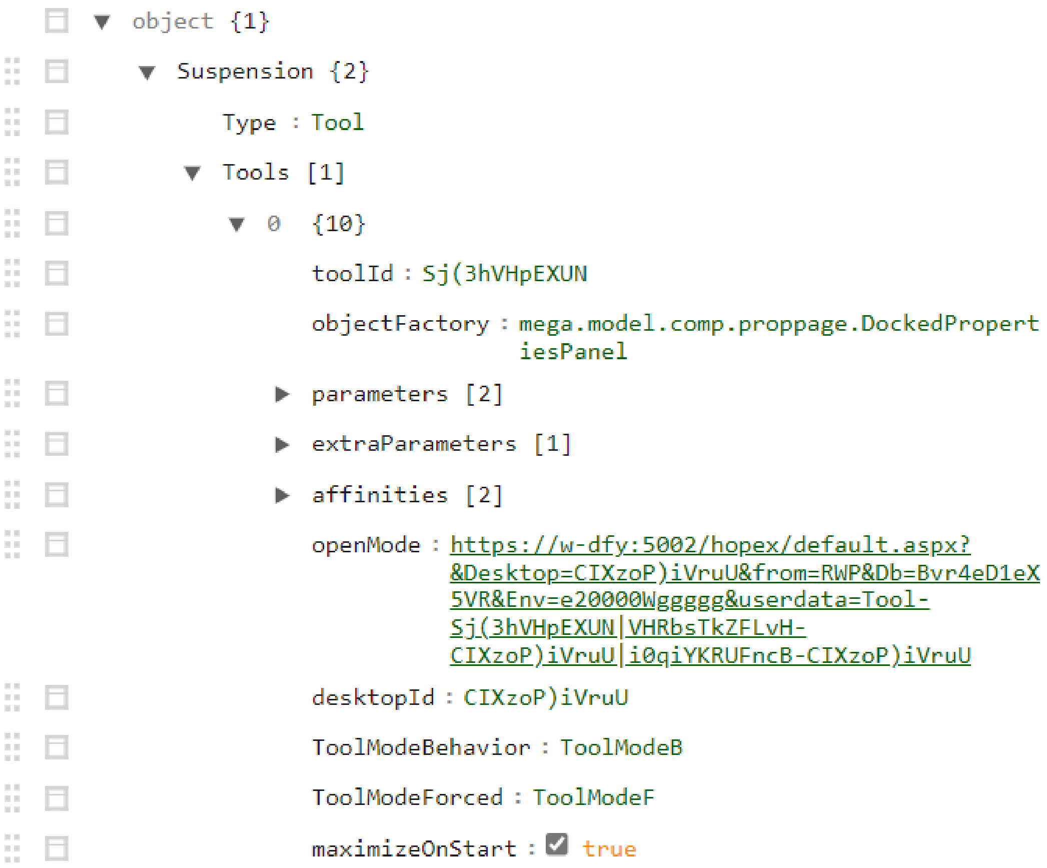
*Note: APIs using **CreateSuspendedParameterizedTool** replace former APIs using **LaunchedParameterizedTool**.*

The same functions as with the ParameterizedTool are available to customize the Suspended ParameterizedTool, see Functions available to customize a ParameterizedTool.

Example of code using functions available to customize the tool:

```
Sub mainRoot(oRoot)

Set MyTool = oRoot.CreateSuspendedParameterizedTool("~Sj(3hVHpEXUN[DockedMetaPropertyPage
Tool]")

MyTool.AddParameter "~VHRbsTkZFLvH[Default Object]", “~CIXzoP)iVruU[1 MetaPropertyPage–
25]"

MyTool.AddParameter "~i0qiYKRUFncB[Property Page]"," ~ANRndAzcKfhB[AF – Person
group.Persons]"

MyTool.AddExtraParameter "~f1KFMsJeFDv2[Display Mode]", "2"

MyTool.AddAffinity NULL

MyTool.AddAffinity "~9zdrlpDlGnSK[GRC – Diagrams Affinity]"

MyTool.SetMaximizeMode TRUE

MyTool.SetOpenMode "NewTab","~gV)MTqsnWreM[NewTab Desktop]"

MyTool.SetToolModeForced "A"

MyTool.SetToolModeBehavior "R"

oRoot.Print MyTool.GetJson

end Sub
```

This code enables to retrieve the following Json:



```
▼ object {1}
    ▼ Suspension {2}
        Type : Tool
        ▼ Tools [1]
            ▼ 0  {10}
                toolId : Sj(3hVHpEXUN
                objectFactory : mega.model.comp.proppage.DockedPropert
                                iesPanel
              ▶ parameters [2]
              ▶ extraParameters [1]
              ▶ affinities [2]
                openMode : https://w-dfy:5002/hopex/default.aspx?
                           &Desktop=CIXzoP)iVruU&from=RWP&Db=Bvr4eD1eX
                           5VR&Env=e20000Wgggggg&userdata=Tool-
                           Sj(3hVHpEXUN|VHRbsTkZFLvH-
                           CIXzoP)iVruU|i0qiYKRUFncB-CIXzoP)iVruU
                desktopId : CIXzoP)iVruU
                ToolModeBehavior : ToolModeB
                ToolModeForced : ToolModeF
                maximizeOnStart : ☑ true
```

With the CreateSuspendedParameterizedTool API you get a **Suspension** array, whose type is indicated (**Type**: Tool).

11.25.3 Functions available to customize a ParameterizedTool

You can use the following functions to customize a ParameterizedTool:

- **AddParameter/AddExtraParameter** function

  The AddParameter/AddExtraParameter function enables to add a parameter / an ExtraParameter to the ParameterizedTool.

  They include the MegaField of the parameter and a value for this parameter.

  Examples:

  ```
  MyTool.AddParameter "~VHRbsTkZFLvH[Default Object]","0"
  MyTool.AddExtraParameter "~MZUVmy73FT1K[Behavior Capabilities]", "2"
  ```

- **AddAffinity** function

  The AddAffinity function enables to add an affinity to the ParameterizedTool.

  It includes a MegaField, but it can also be a NULL affinity.

  Example:

  ```
  MyTool.AddAffinity "~ec3qf4mdU5oL[Addon – Diagram Afffinity]"
  MyTool.AddAffinity NULL
  ```

- **SetOpenMode** function

  The SetOpenMode function enables to specify an OpenMode.

  It includes a character string.

  Example:

  ```
  MyTool.SetOpenMode "M"
  ```

  The open mode values can be:

  - "M" for Modale
  - "F" for Full Screen
  - "P" for Pop-up
  - "NewTab" for New tab

    In that specific case, add the MegaField of the Desktop to be used.

    ```
    MyTool.SetOpenMode "NewTab","~gV)MTqsnWreM[NewTab Desktop]"
    ```

- **SetClickManager** function

  The SetClickManager function enables to specify the ClickManager using an IdAbs.

  ```
  MyTool.SetClickManager "~BuIdYLQ3KfdN[clickManagerForProperties]"
  ```

- **SetToolModeForced** function

  The SetToolModeForced function enables to specify a ToolMode using a character string.

  Examples:

  ```
  MyTool.SetToolModeForced "A"
  MyTool.SetToolModeForced "R"
  ```

  The tool mode values can be:

  - "A" for Add
  - "R" for Replace

- **SetToolModeBehavior** function

  The SetToolModeBehavior function enables to specify a ToolMode using a character string.

  Example:

  ```
  MyTool.SetToolModeBehavior "A"
  MyTool.SetToolModeBehavior "R"
  ```

  The tool mode values can be:

  - "A" for Add
  - "R" for Replace
  - the IdAbs of a desktop.

- **GetJson** function

  The GetJson function enables to return the Json of the ParameterizedTool as a character string format.

  Example:

  ```
  Dim sJson = MyTool.GetJson()
  ```

11.25.4 Using several ParameterizedTools

In some cases, you might need to launch several ParameterizedTools at the same time, in chronological order. In that case, you can create the ParameterizedTool list using the **CreateParameterizedTools** command.

```
Set MyToolsList = oRoot.CreateParameterizedTools
```

This command enables to initialize a ParameterizedTool list. This class uses two functions:

- **AddParameterizedTool**
- **GetJson**

Their use is very similar to the ParameterizedTool class.

Example:

1. Create 3 ParameterizedTools:

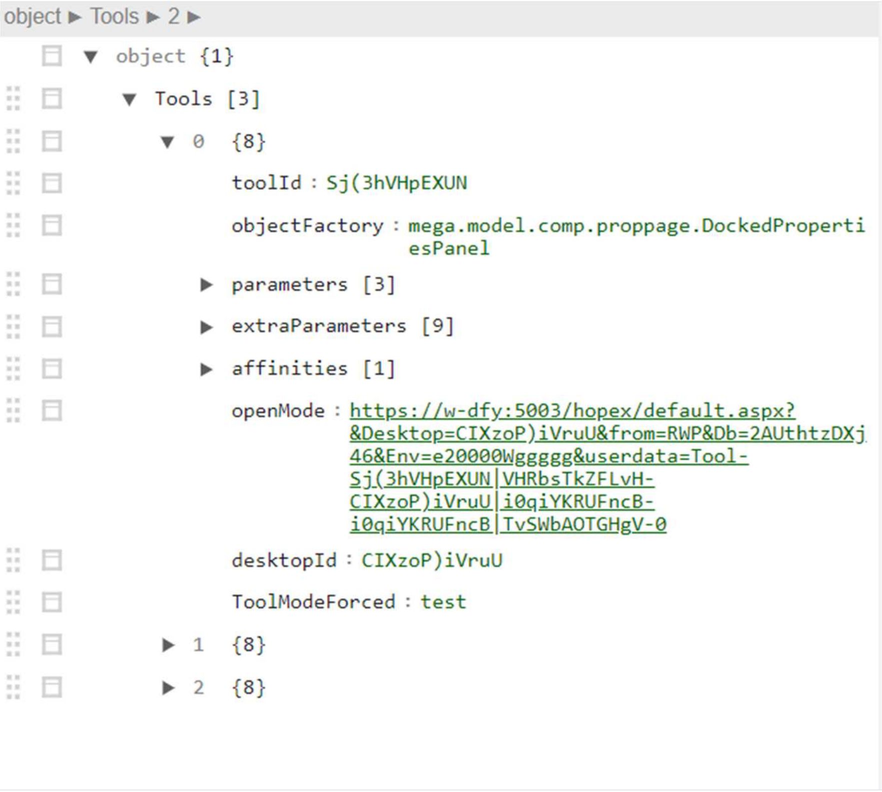
```
Set MyTool1 = oRoot.CreateParameterizedTool("~Sj(3hVHpEXUN[Docked MetaPropertyPage
Tool]")

MyTool1.AddParameter "~VHRbsTkZFLvH[Default Object]", "~CIXzoP)iVruU[1
MetaPropertyPage-25]"

MyTool1.AddParameter "~04VkIJmUFjpU[Listen Current Change]", "1"

MyTool1.AddExtraParameter "~flKFMsJeFDv2[Display Mode]", "2"

MyTool1.AddAffinity NULL

MyTool1.AddAffinity « ~I4bVrX9OGTqF[GRC – Property Page Affinity]"

MyTool1.SetMaximizeMode TRUE

MyTool1.SetOpenMode "NewTab", "~gV)MTqsnWreM[NewTab Desktop]"

MyTool1.SetClickManager "~BuIdYLQ3KfdN[clickManagerForProperties]"


Set MyTool2 = oRoot.CreateParameterizedTool("~)l(3iWHpEXaN[HTML Display Tool]")
```

```
MyTool2.AddParameter "~TvSWbAOTGHgV[Container title strategy]", "0"

MyTool2.AddParameter "~i0qiYKRUFncB[Property Page]"," ~OnfWIEf2L5ML[AF – Objects
and person groups of reading access area] "

MyTool2.AddExtraParameter "~flKFMsJeFDv2[Display Mode]", "2"

MyTool2.AddAffinity NULL

MyTool2.AddAffinity "~3drCv5qtWbGD[Center Tool Affinity]"

MyTool2.SetMaximizeMode FALSE

MyTool2.SetOpenMode "NewTab", "~gV)MTqsnWreM[NewTab Desktop]"

MyTool2.SetClickManager "~BuIdYLQ3KfdN[clickManagerForProperties]"


Set MyTool3 = oRoot.CreateParameterizedTool("~6k(3dXHpEHfN[Docked MetaWizard
Tool]")

MyTool3.AddParameter "~ZzDR68tjGPoS[enableNavigation]", "1"

MyTool3.AddParameter "~qKt3U3nxLHXQ[Property Macro]", "
~xAaCx9LZQnpL[DiveProperties.Implementation]"

MyTool3.AddExtraParameter "~flKFMsJeFDv2[Display Mode]", "2"

MyTool3.AddAffinity NULL

MyTool3.AddAffinity "~46)M(t5sMHv6[Universal Desktop Browse Affinity]"

MyTool3.SetMaximizeMode TRUE

MyTool3.SetClickManager"~BuIdYLQ3KfdN[clickManagerForProperties]"

MyTool3.SetOpenMode "NewTab", "~gV)MTqsnWreM[NewTab Desktop]"
```

2. Use:
   - **AddParameterizedTool** to add a ParameterizedTool to the list, and
   - **Getjson** to concatenate all of the 3 jsons of the ParameterizedTool in a single one.

```
Set MyToolsList = oRoot.CreateParameterizedTools

MyToolsList.AddParameterizedTool MyTool

MyToolsList.AddParameterizedTool MyTool2

MyToolsList.AddParameterizedTool MyTool3

oRoot.Print MyToolsList.GetJson
```

You get the following Json:

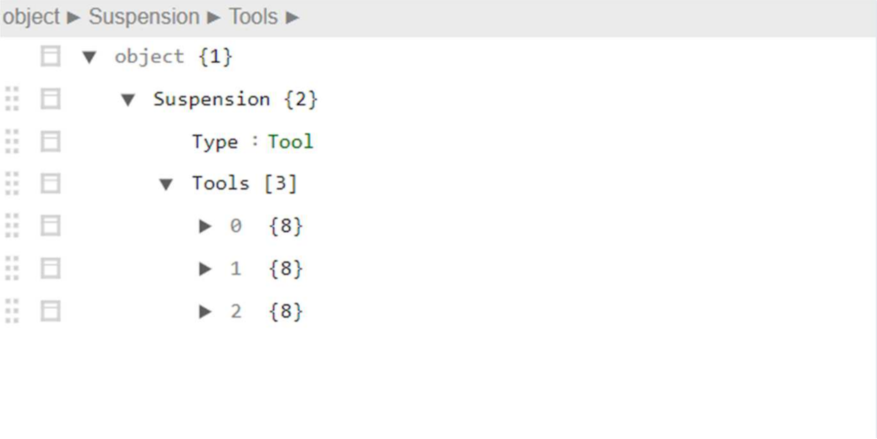


```
object ▶ Tools ▶ 2 ▶
▼ object {1}
    ▼ Tools [3]
        ▼ 0 {8}
            toolId : Sj(3hVHpEXUN
            objectFactory : mega.model.comp.proppage.DockedProperti
                            esPanel
          ▶ parameters [3]
          ▶ extraParameters [9]
          ▶ affinities [1]
            openMode : https://w-dfy:5003/hopex/default.aspx?
                       &Desktop=CIXzoP)iVruU&from=RWP&Db=2AUthtzDXj
                       46&Env=e20000Wggggg&userdata=Tool-
                       Sj(3hVHpEXUN|VHRbsTkZFLvH-
                       CIXzoP)iVruU|i0qiYKRUFncB-
                       i0qiYKRUFncB|TvSWbAOTGHgV-0
            desktopId : CIXzoP)iVruU
            ToolModeForced : test
        ▶ 1 {8}
        ▶ 2 {8}
```

You can also manage suspensions using **CreateSuspendedParameterizedTools**:

```
Set MyToolsList = oRoot.CreateSuspendedParameterizedTools
MyToolsList.AddParameterizedTool MyTool
MyToolsList.AddParameterizedTool MyTool2
MyToolsList.AddParameterizedTool MyTool3
oRoot.Print MyToolsList.GetJson
```

You get the following Json:



```
object ▶ Suspension ▶ Tools ▶
▼ object {1}
    ▼ Suspension {2}
        Type : Tool
        ▼ Tools [3]
            ▶ 0 {8}
            ▶ 1 {8}
            ▶ 2 {8}
```

11.25.5 Creating a ParameterizedTool from a ParameterizedTool

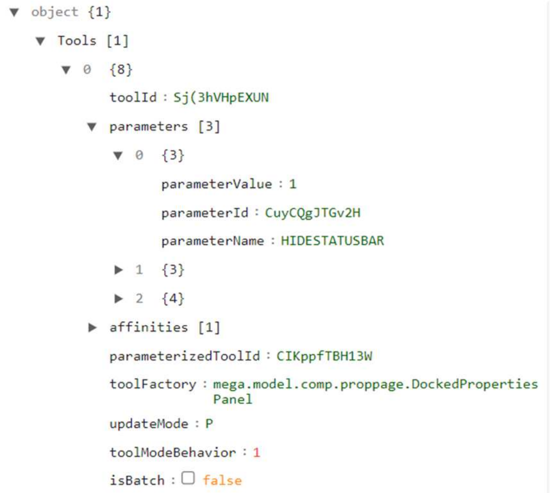
You can generate a ParameterizedTool from an existing ParameterizedTool using its IdAbs. The syntax remains the same:

```
Set MyTool = oRoot.CreateParameterizedTool(~CIKppfTBH13W[ITPM – Docked
MetaPropertyPage – My Application to Update Finance])
```

In that case the (only) available function is GetJson. If you try to add parameters, you will get an error. This is handy to get a Json when the ParameterizedTool already exists:

```
Set MyTool = oRoot.CreateParameterizedTool(~CIKppfTBH13W[ITPM – Docked
MetaPropertyPage – My Application to Update Finance])

oRoot.Print MyTool.GetJson
```

You get the following Json:

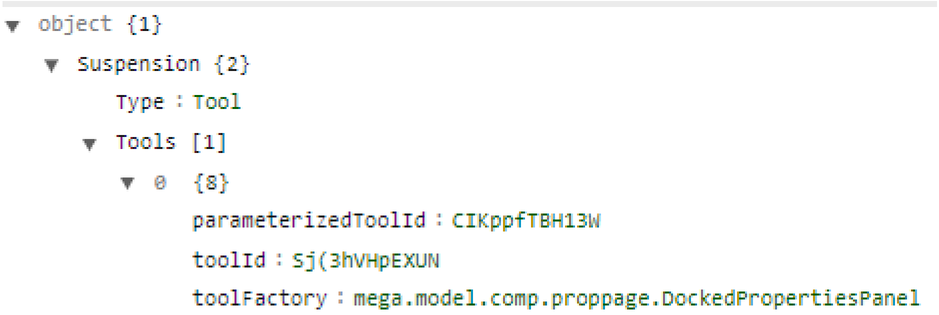


```
▼ object {1}
   ▼ Tools [1]
      ▼ 0  {8}
            toolId : Sj(3hVHpEXUN
         ▼ parameters [3]
            ▼ 0  {3}
                  parameterValue : 1
                  parameterId : CuyCQgJTGv2H
                  parameterName : HIDESTATUSBAR
            ▶ 1  {3}
            ▶ 2  {4}
         ▶ affinities [1]
            parameterizedToolId : CIKppfTBH13W
            toolFactory : mega.model.comp.proppage.DockedProperties
                          Panel
            updateMode : P
            toolModeBehavior : 1
            isBatch : □ false
```

You can also create a SuspendedParameterizedTool

```
Set MyTool = oRoot.CreateSuspendedParameterizedTool(~CIKppfTBH13W[ITPM – Docked
MetaPropertyPage – My Application to Update Finance])

oRoot.Print MyTool.GetJson
```

You get the following Json:



```
▼ object {1}
   ▼ Suspension {2}
        Type : Tool
      ▼ Tools [1]
         ▼ 0  {8}
              parameterizedToolId : CIKppfTBH13W
              toolId : Sj(3hVHpEXUN
              toolFactory : mega.model.comp.proppage.DockedPropertiesPanel
```

11.26 Report DataSets and APIs

11.26.1 Getting a Report DataSet content using an API

To get a Report DataSet content use GetCollection("~Yvazr2mvKf21[DataSet Create]") on the Report DataSet object.

Example: Opening the selected Report DataSets in HOPEX explorer

VB Script

```
Dim mgcolDataSubSet
Set mgcolDataSubSet = GetRoot.RequestQuery("~)ilXTIGrKPiB[Report DataSet]")
If Not mgcolDataSubSet Is Nothing Then
  Dim mgobjDataSubSet
  For Each mgobjDataSubSet In mgcolDataSubSet
    Dim mgcolDataSubSetValues
    Set mgcolDataSubSetValues =
mgobjDataSubSet.GetCollection("~Yvazr2mvKf21[DataSet Create]")
    mgcolDataSubSetValues.explore
  Next
End If
```

Java

```
String datasetId="";
MegaObject mgobjDataSubSet = megaRoot.getCollection("~)ilXTIGrKPiB[Report
DataSet]").get(datasetId);
MegaCollection mgcolDataSubSetValues =
mgobjDataSubSet.getCollection("~Yvazr2mvKf21[DataSet Create]");
```

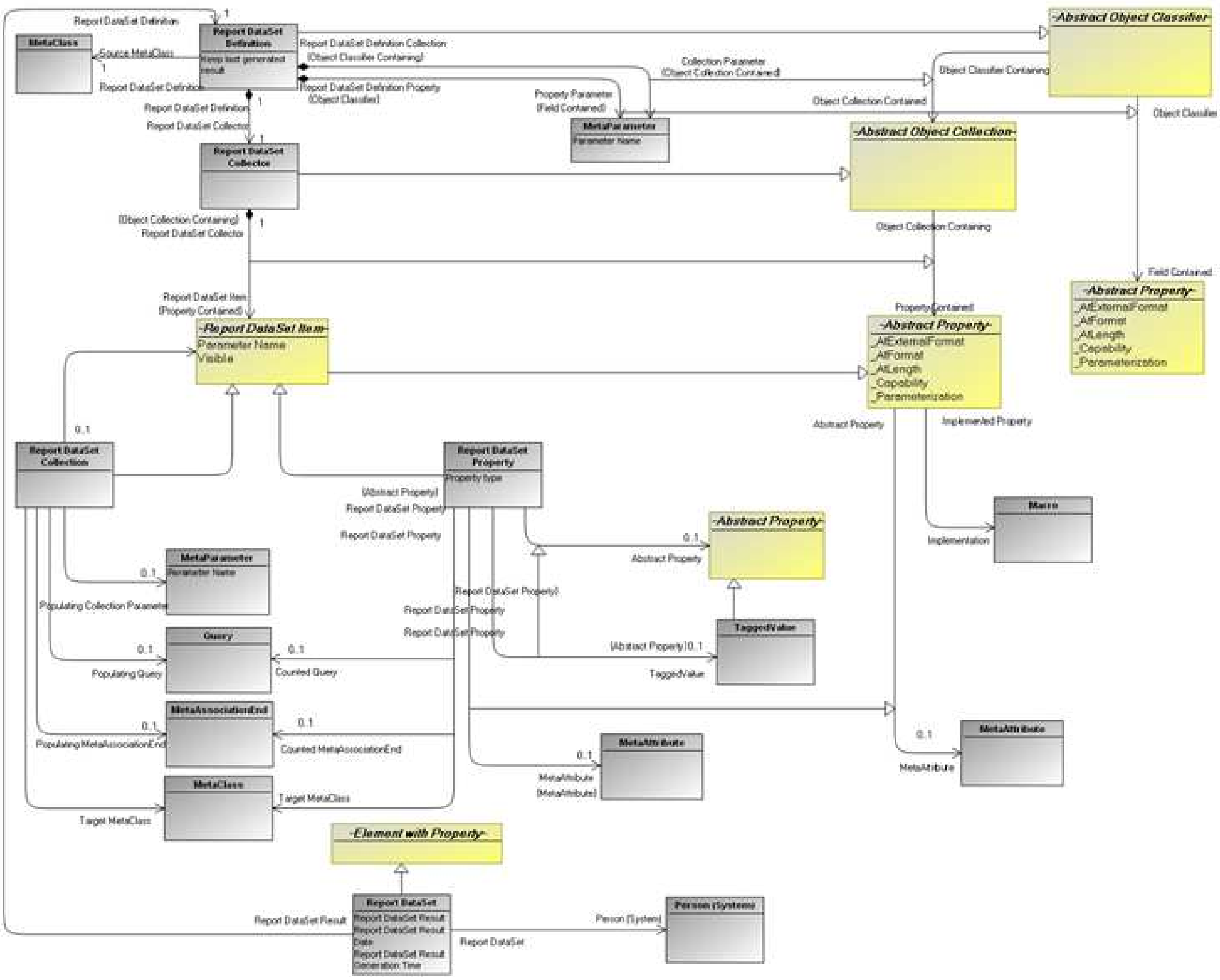
11.26.2 Regenerating a Report DataSet content using an API

Once a Report DataSet has been generated in a HOPEX session, each successive consultation of this Report DataSet in the same session returns the same previous Report DataSet (the Report DataSet is not calculated at each consultation, it is stored in the cache).

To refresh the Report DataSet content, you need to invoke the CollectionCacheReset operator to delete this cache. In this way, the next consultation of the Report DataSet triggers a new Report DataSet calculation.

To do so, call the CollectionCacheReset operator on the MEGA root, with the Report DataSet identifier and the Report DataSet Collector identifier as argument.

The Report DataSet MetaModel is as follows:



Example: Deleting the cache of all selected report Dataset

VB Script

```
Dim mgcolDataSet
Set mgcolDataSet = GetRoot.RequestQuery("~)ilXTIGrKPiB[Report DataSet]")
If Not mgcolDataSet Is Nothing Then
  Dim mgobjDataSet
  For Each mgobjDataSet In mgcolDataSet
    GetRoot.CollectionCacheReset mgobjDataSet.GetId,
mgobjDataSet.GetCollection("~rLU9sCZtKLx6[Report DataSet
Definition]").Item(1).GetCollection("~NMU9ZfYtKHh6[Report DataSet
Collector]").Item(1).GetId()
  Next
End If
```

Java

```
megaRoot.callMethod("CollectionCacheReset",
mgobjDataSubSet.getID(),mgobjDataSubSet.getCollection("~rLU9sCZtKLx6[Report
DataSet Definition]").get(1).getCollection("~NMU9ZfYtKHh6[Report DataSet
Collector]").get(1).getID());
```

11.27 GraphSet and APIs

11.27.1 Getting the current GraphSet IdAbs

You can get the current GraphSet IdAbs in a macro implementing the GraphSet Arc Definition.

With a GraphSet Arc Definition implemented by a calculated query, to get the current GraphSet in the query use the "CurrentGraphSet" parameter:

```
mgobjObject.GetRoot.CurrentEnvironment.QueryParameters.ParameterValueGet("Curren
tGraphSet")
```

Example of a macro getting the current GraphSet

```
Sub FillSelectionCollection(mgobjObject as MegaObject, vntSelectorID as Variant,
mgcolCollection as MegaCollection)

    Dim graphSetIdabs

    graphSetIdabs =
    mgobjObject.GetRoot.CurrentEnvironment.QueryParameters.ParameterValueGet("Cu
    rrentGraphSet")

    Dim mgobjGraphSet : Set mgobjGraphSet =
    mgobjObject.GetRoot.GetObjectFromId(graphSetIdabs)

    If mgobjGraphSet.Exists Then

        mgcolCollection.Add mgobjGraphSet

    End If

End Sub
```

11.28 Diagrams and APIs

Diagrams API can be useful to programmatically extract information from diagrams, modifying existing diagrams or creating new diagrams from scratch.

11.28.1 Opening an existing diagram

The first step to handle a diagram is to obtain a MegaDrawing API object. This is achieved by first obtaining a reference on the diagram repository object using the repository API.

The following code sample retrieves a diagram in the repository using its unique identifier:

```
Set oRepositoryDiagram = GetObjectFromId("hk5HbIaO8b70")
```

The diagram can then be "opened", either for read-only or read/write access. The following function returns a MegaDrawing which will allow further manipulation:

```
Set oDrawing = oRepositoryDiagram.Drawing("RO") 'Read-Only

Set oDrawing = oRepositoryDiagram.Drawing("RW") 'Read-Write
```

Note that invoking the Drawing function is like opening the diagram in the desktop: it is a costly call, and for improved performance scripts should avoid opening multiple times the same diagram.

Also note that if the diagram is already opened in the desktop, this function will open a separate instance of the diagram in the state of its last save. Most API modifications on this diagram will not therefore be reflected live in the graphical user interface: it will be necessary to close it without saving, then reopen it. To handle diagrams live see Setting up interactive plug-ins in a diagram section.

11.28.2 Creating a new diagram

Like any repository object, a diagram must first be created in the repository. However, there are two additional requirements to successfully create a diagram:

- A diagram must have a "nature", which indicates the type of diagram (a flowchart, an organizational chart …). Available natures are listed as Meta MetaAttributeValue of the MetaAttribute "Nature".
- A diagram must describe a repository object (a Business Process, an Application …) and be linked to this object via the adequate MetaAssociationEnd. To find out which MetaAssociationEnd is used as well as the list of allowed natures for a given described object, it is necessary to browse the DiagramTypeZoom linked to the MetaClass of the described object.

```
' Retrieving a Business Process

Set oBusinessProcess = GetObjectFromId("0ynRE)tLxy81")

Set colDescription = oBusinessProcess.GetCollection("Description")

Set oNewDiagram = colDescription.Add("My new diagram")

oNewDiagram.GetProp("Nature") = "BPDD" ' Business Process Component Diagram

Set oDrawing = oNewDiagram.Drawing("RW")
```

Note that initialization macros are not called when a diagram is created programmatically.

11.28.3 Saving the diagram

Unlike repository modification, diagram manipulations are not committed automatically. It is necessary to explicitly save the diagram:

```
oDrawing.Flush
```

11.28.4 Storing/Retrieving an information on/from a diagram

You can store an information on a diagram using:

```
myDrawing.prop("UserData") = "myData"
```

You can retrieve the information from the diagram using:

```
Drawing.prop("UserData")
```

Example:

```
dim oDiag


set oDiag = GetObjectFromId("~j6H4z71sYLlJ[MEGA - Business Process Diagram]")
```

```
dim Drawing
set Drawing = oDiag.Drawing("RW")

Drawing.Prop("UserData") = "toto"
print Drawing.Prop("userData")

Drawing.flush
```

11.29 Accessing graphical objects in a diagram

The all-around way to access graphical objects in a diagram is via the DrawingItems collection available on the MegaDrawing. Some items can also have child items. The structure of the diagram is therefore a tree of DrawingItems. Items are sorted by depth, from the background to the foreground.

An important property of all DrawingItems is the DrawClassName which indicates its object type. The most important types will be detailed in the sections below.

Sample code illustrating how to print the object tree of a diagram

```
Sub PrintDrawingItems( colDrawingItems, identLevel )
  sPrefix = "-"
  For i = 1 to identLevel
    sPrefix = " " & sPrefix
  Next
  For Each oItem in colDrawingItems
    Print sPrefix & oItem.DrawClassName
    PrintDrawingItems oItem.SubDrawingItems, identLevel+1
  Next
End Sub

PrintDrawingItems oDrawing.DrawingItems, 0
```

11.29.1 Accessing repository objects

Repository objects present in the diagram can be accessed in two ways:

- Via the generic DrawingItems tree. Their class name is “ModeOcc”. Objects returned by this collection support the DrawingItem interface.
- Via the legacy DrawingObjects collection which returns only repository objects in a flat list. Objects returned by this collection support the DrawingObject interface and are NOT sorted by depth.

The newer DrawingItem interface has more features than the legacy DrawingObject interface. It is possible to switch between interfaces, as shown in the example below.

```
For Each oDrawingObject in oDrawing.DrawingObjects
  Print oDrawingObject.Name
  Set oDrawingItem = oDrawingObject.DrawingItem
  Print oDrawingItem.ItemProperty("ModeOcc_Name")
  Set oDrawingObjectBis = oDrawingItem.ItemProperty("ModeOcc_DrawingObject")
  Print oDrawingObjectBis.Name
Next
```

To add a repository object in a diagram, the MegaObjectInsert function must be used. It is possible to add several graphical representations of the same repository object. If the object was not already present in the diagram, the repository link between the repository object and the diagram will be created automatically. Note that even if the modifications of the diagram are not saved, this link will still be present in the repository.

```
Set oRepositoryObjectToAdd = GetObjectFromId("C(Rt1VUV9n42")
Set oNewDrawingItem = oDrawing.MegaObjectInsert(oRepositoryObjectToAdd)
oNewDrawingItem.SetPos 200, 200, 4200, 2200
Set oNewDrawingItem2 = oDrawing.MegaObjectInsert(oRepositoryObjectToAdd)
oNewDrawingItem2.SetPos 5200, 200, 9200, 2200
```

Removing a repository object from the diagram is done using the Erase function, as for any DrawingItem. If the DrawingItem removed is the last representation of a repository object, the repository link between the object and the diagram will be automatically deleted.

11.29.2 Accessing repository links

Repository links present in the diagram can be accessed in two ways:

- Via the generic DrawingItems tree. Their class name is "ModeLink". Objects returned by this collection support the DrawingItem interface.
- Via the legacy DrawingLinks collection which returns only repository links in a flat list. Objects returned by this collection support the DrawingLink interface and are NOT sorted by depth.

The newer DrawingItem interface has more features than the legacy DrawingLink interface. It is possible to switch between interfaces, as shown in the example below.

```
For Each oDrawingLink in oDrawing.DrawingLinks
  Print oDrawingLink.DirectLegName
  Set oDrawingItem = oDrawingLink.DrawingItem
  Set oDrawingLinkAgain = oDrawingItem.ItemProperty("ModeLink_DrawingLink")
  Print oDrawingLinkAgain.DirectLegName
Next
```

Repository links are automatically added or removed in a diagram when a link is added or removed in the repository and connected objects are present in the diagram.

```
' Creation of a link in repository
Set oOperation1 = GetObjectFromId("lktopHlNBL51")
Set oOperation2 = GetObjectFromId("JltopHlNBb71")
Set oNextLeg = GetObjectFromId("KsUirDB5iCu2")
oOperation1.GetCollection(oNextLeg.GetId()).Add(oOperation2)
```

```
` Retrieving the graphical representation of the new link
Function FindDrawingLink( oDrawing, idRpMaster, idRpSlave, idLeg )
  ReDim arDrawingLink(0)
  For Each oDrawingLink In oDrawing.DrawingLinks
    If ( ( idLeg   = oDrawingLink.OppositeLegId          _
           And idRpSlave  = oDrawingLink.OppositeDrawingObject.Id _
           And idRpMaster = oDrawingLink.DirectDrawingObject.Id ) _
      Or ( idLeg   = oDrawingLink.DirectLegId            _
           And idRpSlave  = oDrawingLink.DirectDrawingObject.Id   _
           And idRpMaster = oDrawingLink.OppositeDrawingObject.Id ) ) Then
      ReDim Preserve arDrawingLink(UBound(arDrawingLink)+1)
      Set arDrawingLink(UBound(arDrawingLink)-1) = oDrawingLink
    End If
  Next
  FindDrawingLink = arDrawingLink
End Function

arDrawingLink = FindDrawingLink( oDrawing, oOperation1.GetId(), _
                                 oOperation2.GetId(), oNextLeg.GetId() )
Set oDrawingLink = arDrawingLink(0)
```

It is then possible to modify graphical aspects of the link. Many visual properties of texts and lines are available in structures which must be manipulated by value, and not by reference: API functions usually return copies of these structures and not direct handles to the original structures. This allows preparing a structure and then being able to assign it to different objects. The following example illustrates this distinction when changing the color of a line.

```
Set oDrawingItem = oDrawingLink.DrawingItem
Print Hex(oDrawingItem.Pen.Color) ` For example, returns FF0000FF - blue

` The Pen property returns a copy of the MegaDrawingPen structure so
` the following statement will NOT change the color of the link
oDrawingItem.Pen.Color = &hFFFF0000 ` red
Print Hex(oDrawingItem.Pen.Color) ` still blue

Set oLinePen = oDrawingItem.Pen
oLinePen.Color = &hFFFF0000
` Only the structure copy in oLinePen has been modified
Print Hex(oDrawingItem.Pen.Color) ` still blue,

` This statement will commit the change in the original structure
oDrawingItem.Pen = oLinePen
Print Hex(oDrawingItem.Pen.Color) ` red at last
```

```
` Short Version
Set oLinePen = oDrawingItem.Pen
oLinePen.Color = &hFF00FF00
oDrawingItem.Pen = oLinePen

oDrawing.Flush ` Do not forget to save the diagram
```

Modifying the points of a line is done via a special interface. To be successful, the new points collection must keep the original extremities coordinates (the first and the last points). If the line is orthogonal, extra care must be taken to ensure that the new point collection is orthogonal; otherwise the line points will be reset.

```
' Setting orthogonal style
Set oLineStyle = oDrawingItem.LineStyle
oLineStyle.Style = 7 ' orthogonal
oDrawingItem.LineStyle = oLineStyle

` Utility function for debug which prints the points of a collection
Sub DumpPoints( colPoints )
  Print colPoints.Count & " points"
  For i = 0 to colPoints.Count - 1
    Print "[ " & colPoints.Abscissa(i) & ", " & colPoints.Ordinate(i) & " ]"
  Next
End Sub

Set colPoints = oDrawingItem.ItemProperty("DrwLine_Points")

' Removing old points, keeping both extremities
For i = 1 to colPoints.Count - 2
  colPoints.RemovePoint(1)
Next

' Creating a "stair"
nbSteps = 7
dw = (colPoints.Abscissa(1)-colPoints.Abscissa(0)) / nbSteps
dh = (colPoints.Ordinate(1)-colPoints.Ordinate(0)) / nbSteps
For i = 1 to nbSteps
  colPoints.InsertPointBefore colPoints.Count-1, _
                              colPoints.Abscissa(colPoints.Count-2), _
                              colPoints.Ordinate(colPoints.Count-2)+dh
  colPoints.InsertPointBefore colPoints.Count-1, _
                              colPoints.Abscissa(colPoints.Count-2)+dw, _
                              colPoints.Ordinate(colPoints.Count-2)
Next
colPoints.Abscissa(colPoints.Count-2) = _
```

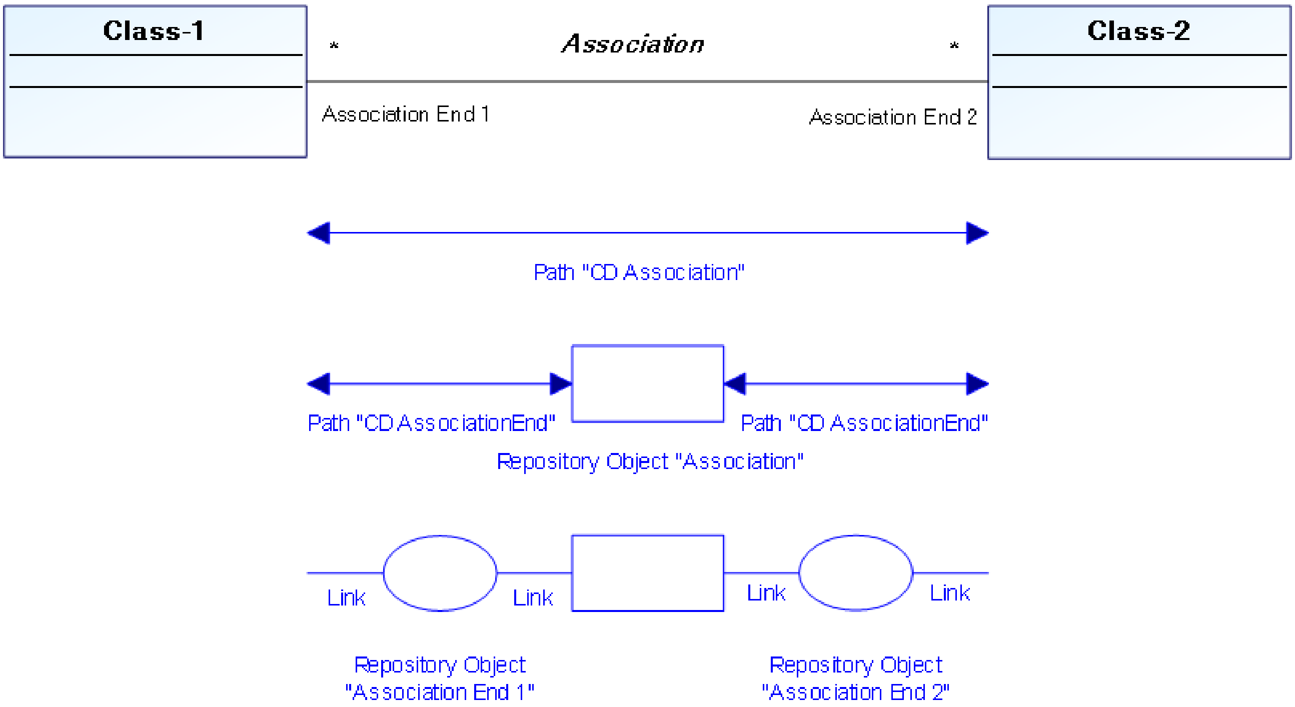
```
                         colPoints.Abscissa(colPoints.Count-1)

        oDrawingItem.ItemProperty("DrwLine_Points") = colPoints
```

11.29.3 Paths

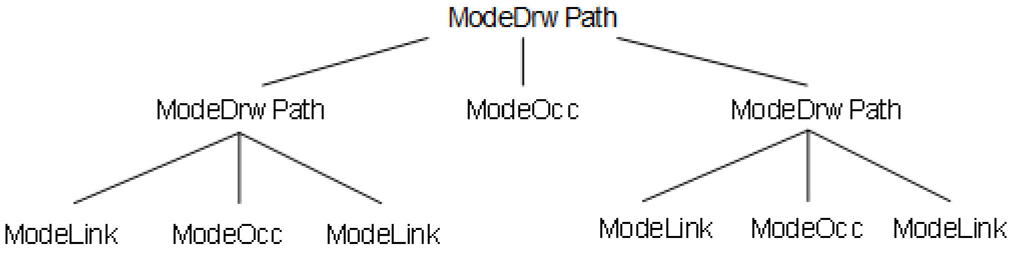
A path is a special kind of graphical object: it is displayed as a link, but actually hides a sequence of repository links and objects. Paths are declared as DiagramTypePath in the diagram parameterization. It is declared as a sequence of:

1. A link or a path
2. An object
3. A link or a path

An example of path is an association in a UML class diagram. It is a multi-level path which chains 3 repository objects.



There is no specific API for paths. They must be accessed via the DrawingItem collection, their DrawClassName is "ModeDrwPath". The DrawingObjects and DrawingLinks hidden by the path can still be accessed via the legacy collections, as they are actually children of the path.



However, as these graphical objects are not displayed, there is no real interest in manipulating them. To modify the aspect of the path (like the color or the points, it is necessary to modify the root ModeDrwPath DrawingItem. However, access to the underlying DrawingObjects and DrawingLinks is necessary to identify the path.

```
Function FindPathItem( oDrawing, idRpObjectCenter )
  ReDim arDrawingItem(0)
  For Each oDrawingObject in oDrawing.DrawingObjects
    If oDrawingObject.Id = idRpObjectCenter Then
      Set oParent = _
          oDrawingObject.DrawingItem.ItemProperty("ParentDrawingItem")
      If oParent.DrawClassName = "ModeDrwPath" Then
        ReDim Preserve arDrawingItem(UBound(arDrawingItem)+1)
        Set arDrawingItem(UBound(arDrawingItem)-1) = oParent
      End If
    End If
  Next
  FindPathItem = arDrawingItem
End Function

Set oAssociation = GetObjectFromId("PhnuDD(NB1W0")
arPathDrawingItem = FindPathItem( oDrawing, oAssociation.GetId() )
Set oPathDrawingItem = arPathDrawingItem(0)
```

To add a path to diagram, all necessary objects and links must first be created in the repository and the extremity objects must appear in the diagram. The center object of the top level path must then be added. The remaining objects and links will be added automatically and a path will be generated. It can then be accessed as previously.

```
oDrawing.MegaObjectInsert oAssociation
arPathDrawingItem = FindPathItem( oDrawing, oAssociation.GetId() )
Set oPathDrawingItem = arPathDrawingItem(0)

Set oPathPen = oPathDrawingItem.Pen
oPathPen.Color = &hFF0000FF
oPathDrawingItem.Pen = oPathPen
```

11.29.4 Getting the Text Field of a ModeOcc of a path

The coordinates of a ModeOcc of a path correspond with the virtual location of this ModeOcc on this path. Thus, the central ModeOcc of a path (invisible in the drawing) is a virtual rectangle located in the middle of the path.

To get the coordinates of the **&Name&** field of the central ModeOcc, you must go through the twin of the ModeOcc attribute corresponding to the **&Name&** field.

11.30 Setting up interactive plug-ins in a diagram

You can set macros as diagram plug-ins for a given DiagramType. These macros are called when:

- diagrams are opened or saved,
- graphical representations of repository objects are moved or resized
- represented objects are modified.

In response to a user input, they can then further modify the diagram or the repository, for example to perform an automatic layout or to create complex repository relationships which cannot be achieved with standard parameterization.

In addition, diagram plug-ins help declare and implement macro commands; each macro command is associated with a button in a specific “macro” toolbar.

11.30.1 Writing a diagram plug-in

A diagram plug-in must be implemented in a Mega “Macro” repository object. It defines a number of functions which are called in response to events occurring in the diagram.

Write the following code in the macro editor and save it as a macro.

```
' ********************************************************************
' Event handlers - each handler implementation is optional
Sub OnLoadDrawing( oDrawing As MegaDrawing )
End Sub

Sub OnMoveDrawingObject( oDrawing       As MegaDrawing, _
                         oDrawingObject As MegaDrawingObject )
End Sub

Sub OnResizeDrawingObject( oDrawing       As MegaDrawing, _
                           oDrawingObject As MegaDrawingObject )
End Sub

Sub OnSaveDrawing( oDrawing As MegaDrawing )
End Sub

Sub OnObjectChanged( oDrawing       As MegaDrawing, _
                     oDrawingObject As MegaDrawingObject )
End Sub

' New events in Mega 2009 SP2
Sub OnInsertDrawingObject( oDrawing       As MegaDrawing, _
                           oDrawingObject As MegaDrawingObject)
End Sub

Sub OnEraseDrawingObject( oDrawing       As MegaDrawing, _
                          oDrawingObject As MegaDrawingObject)
End Sub

Sub OnUndo( oDrawing As MegaDrawing)
  ' WARNING: no modification should be made, this is only for updating
  ' internal structures
end Sub


' ********************************************************************
' Plug-in custom toolbar button declaration
' To add macro commands, the three following methods must implemented.
' Otherwise none is required to be implemented.

' Returns the number of macro commands implemented by the plug-in.
Sub GetCommandCount( oDrawing, nCmd )
  nCmd = 1
End Sub

' For each command from 1 to count (returned by GetCommandCount),
' this method should return the text description of the command in sCmdName
' and optionally the ID of an existing MetaPicture which will be used for the
' button in the toolbar instead of the default picture.
Sub GetCommandDescription( oDrawing, nCmd, idPict, sCmdName)
  If nCmd = 1 Then
    sCmdName = "ZOrderCompare test"
    idPict = oDrawing.GetRoot.CurrentEnvironment.Toolkit _
                     .GetIDFromString("~uxuU1odd5z00[Book]")
  End If
End Sub

' This method will be called when the user clicks the corresponding button
Sub CommandCall( nCmd, oDrawing )
  If nCmd = 1 Then
    Set mySelCol = drawing.SelDrawingItems
    Set myItem   = mySelCol.Item(1)
    Set myObjCol = drawing.DrawingObjects
    For Each obj In myObjCol
      If myItem.ZOrderCompare(obj.DrawingItem) = 1 then
        MsgbBx "Selected object is BEHIND " & obj.Name
      Else
        MsgBox "Selected object is IN FRONT of " & obj.Name
      End If
    Next
  End If
End Sub
```

11.30.2 Writing a drag'n drop plugin

Just like the diagram plug-in, you can handle your own behavior for the drag'n drop in the diagram. The drag'n drop plug-in must be implemented in a MEGA "Macro" repository object. It defines a number of functions which are called in response to events occurring in the drag'n drop.

Even if technically a diagram plug-in macro can be used to implement the custom drag'n drop behavior it is strongly recommended to use distinct macro for diagram plug-in and drag'n drop plug-in.

```
Function OnDragEnter(oDragnDropContext as Object)
End Function
```

The On DragEnter function is called each time a drag'n drop enters the diagram area.

The return value must be one of the following options:

- 0: This is the default behavior. The Drag'n drop plug-in does not implement a specific behavior for this drag'n drop
- 1: The behavior of the drag'n drop is handled by the plug-in. The data of the drag'n drop can be dropped in the current diagram.
- 2: The behavior of the drag'n drop is handled by the plug-in. The data is not accepted for the drop action in the current diagram.

The oDragnDropContext is a context object available for each methods of this plug-in. It is created when a given drag'n drop action enters the given diagram area for the first time. This object is available until the drag'n drop is dropped in the given diagram or when another drag'n drop enters the diagram's area.

This object supports the following methods:

- GetRoot: Returns the MEGARoot
- GetDrawing: Returns the drawing of the current diagram.
- GetSourceObjClassId: Returns the MetaClass Id of the data attached to the current drag n drop.
- GetSourceObjCollection: Returns the collection of the data attached to the drag n drop.
- GetDragPosX: Returns the position in the diagram of the cursor.
- GetDragPosY: Returns the position in the diagram of the cursor.
- GetBag: Returns a bag attached to the context object.

```
Function OnDragOver(oDragnDropContext as Object)
End Function
```

The OnDragOver is called each time the drag'n drop moves over the diagram's area. Only very light computation should be implemented in this function.

To lighten the computation of this function you can for example do your computation in the OnDragEnter function and store the result on the context object thanks to its bag.

```
Sub OnDragLeave(oDragnDropContext as Object)
End Sub
```

This procedure is called each time the drag'n drop leaves the diagram's area. It does not have any return value.

```
Function OnDragDrop(oDragnDropContext as Object)
End Function
```

This function is called when the drag'n drop data is dropped on the diagram's area. If you allowed the drag'n drop action with function OnDragEnter or OnDragOver, you must handle the drag'n drop here.

The available return values are:

- 0: The drag'n drop was not handled by the plug-in.
- 1: The drag'n drop was handled by the plug-in.

11.30.3 Registering the macro on a DiagramType:

In the properties box of the DiagramTypeParam of the DiagramType of a diagram, in the _settings text, in the [Macros] section, just add an entry like <n>=<macroIdAbs> where n is any integer not already used and macroIdAbs is the macro idAbs which can be easily obtained via the "Field insert" button of the Text page toolbar.

Note:

When the macro is modified, the diagram needs to be reloaded to take changes into account.

The drawing given to each method of the plugin is a Read/Write MegaDrawing.

Several plugins can be set up on one MetaDiagramType. Each plugin can provide macro commands, however the overall maximum number of macro commands is 16.

11.31 Writing a dynamic query

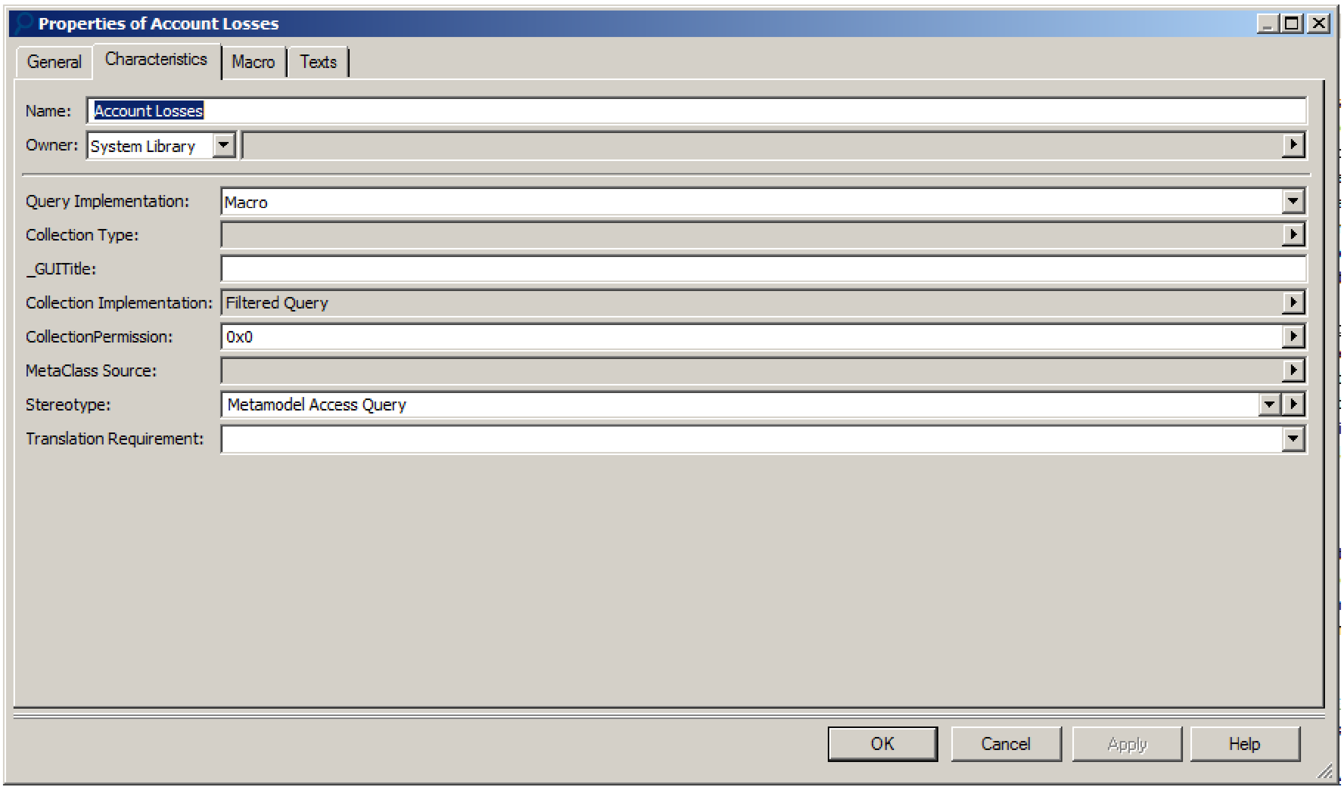
A query retrieves a set of objects.

A dynamic query is a query that cannot be written with an ERQL syntax; Algorithm is necessary to retrieve the collection content.

A dynamic query is based on the Query concept.

To write a dynamic query:

1. Open the query **Properties** window.
2. Set its **Query Implementation** attribute to "Macro".
3. Set its **collection Implementation** attribute to the macro that will compute the collection:



Properties of Account Losses

Name: Account Losses

Owner: System Library

Query Implementation: Macro

Collection Type:

_GUITitle:

Collection Implementation: Filtered Query

CollectionPermission: 0x0

MetaClass Source:

Stereotype: Metamodel Access Query

Translation Requirement:

4. It is recommended to implement the query code in VB Script except if you have very complex code. In that case, you can use Java

   Two prototypes can be defined when computing the collection:

VB Script

```
Function getSelectionCollection (oSourceObject as MegaObject, SelectorID as Variant)

End Function


Sub FillSelectionCollection(oSourceObject as MegaObject, SelectorID as Variant, mgcoll as MegaCollection)

End Sub
```

Java

```
public void InvokeOnCollection(final MegaCollection lst )
{
}
Or if you want your method to return a value, use
//return an integer
public int InvokeOnCollection(final MegaCollection lst )
{
}
```

The **getSelectionCollection** creates and sends back the collection to be retrieved. This function can be used particularly when the collection contains different kind of objects.

The **FillSelectionCollection** has the collection already initialized. The function role is just to fill it in.

The arguments are:

- the source object from which the query may be requested or the MegaRoot
- the query identifier.

11.32 Accessing rules and regulations using APIs

11.32.1 Accessing regulations using APIs

Implementing a regulation

```
myObject.ApplyRegulation(myRegulation as MegaObject) as Boolean
```

Applies the regulation to the object.

Returns `True` or `False` according to the regulation compliancy.

Defining the default active regulation

```
myRegulation.RegulationActivate = True
```

myRegulation becomes the active regulation.

To get the result of implementing this regulation to a specific object, read the calculated text “object control report”.

Example:

VB Script

```
set myRule = GetObjectFromID("~W6Or(qjB2zB0[Architecture Modeling
Regulation]")

myRule.RegulationActivate = True

print Application.item(1).getprop("object control report","Display")

myRule.RegulationActivate = False
```

Java

```
MegaObject mgobjMyRule =
mgobjMegaObject.getRoot().getObjectFromID("~W6Or(qjB2zB0[Architecture
Modeling Regulation]");

                  mgobjMyRule.invokePropertyPut("RegulationActivate", true);

                  mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]",
mgobjMegaObject.getRoot().getCollection("Application").get(1).getProp("object
control report ", "Display"));

                  mgobjMyRule.invokePropertyPut("RegulationActivate", false);
```

By default «object control report» text is in HTML format. « Display » keyword enables to display it in text mode.

```
myRegulation.RegulationActivate = False restores the active regulation defined
in the user options
```

11.32.2 Accessing rules using APIs

To apply "myRule" rule to "myObject" objet, you can use the following symmetrical methods:

```
myRule.RuleApply(myObject as MegaObject) as Boolean

myObject.ApplyRule(myRule as MegaObject) as Boolean
```

Both functions send `False` when the rule is not complied with.

Reminder: the rule description is included in the text "Description of the rule". When the rule is not complied with, the message to be displayed is "When the rule is not complied with"

Example:

VB Script

```
set myRule = GetObjectFromID("~o6OrCgnB2fp2[An application cannot be its own
component]")

print myRule.RuleApply(Application.item(1))

print Application.item(1).ApplyRule(myRule)

print myRule.GetProp("Description of the rule")

print myRule.GetProp("When the rule is not complied with")
```

Java

```
MegaObject mgobjMyRule =
mgobjMegaObject.getRoot().getObjectFromID("~o6OrCgnB2fp2[An application
cannot be its own component]");

                  mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]",
mgobjMyRule.invokeFunction("RuleApply",
mgobjMegaObject.getRoot().getCollection("Application").get(1)));

                  mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]",
mgobjMegaObject.getRoot().getCollection("Application").get(1).invokeFunction
("ApplyRule", mgobjMyRule));

                  mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]",
mgobjMyRule.getProp("Description of the rule"));
```

```
                    mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]",
    mgobjMyRule.getProp("When the rule is not complied with"));
```

11.33 Business Documents and APIs

The following APIs are available in the frame of the Document Management module regarding static documents:

- StaticDocumentFilePathGet
- DesktopUrlBuild with DocumentLauncher
- SaveAsStatic

11.33.1 StaticDocumentFilePathGet

Retrieving the address of a physical file from a static document (static document version, system static document or system static document version) that can be used to send a document as an attachment to a mail for example.

Function prototype:

```
Function StaticDocumentFilePathGet() As String
```

Use example:

mgoDoc a static document (static document version, system static document or system static document version) retrieves the path to the file containing the static document.

```
strFilePath = mgoDoc.CallFunction("~FFopcJjTGnIC[StaticDocumentFilePathGet]")
```

or

```
strFilePath = mgoDoc.StaticDocumentFilePathGet()
```

Result examples:

C:\PROGRAM FILES (X86)\MEGA\ENVS\DEMONSTRATION\SYSDB\USER\User\mg_webtmp\68412896\129955724059390000\Exemple de DOCX v1.docx

Warning: the file life cycle is the responsibility of the StaticDocumentFilePathGet function caller. It should be deleted after use.

However the file will be deleted when the workspace is closed.

11.33.2 DesktopUrlBuild with DocumentLauncher tool

Generation of a link enabling static document opening from the HOPEX Web Front-end.

Use the API DesktopUrlBuild (see Calling a URL construction function using APIs in HOPEX).

The Mega Tool used here is "Document Launcher": ~LA)RAINPGnHP[Document Launcher]

The Mega Tool parameter is "documentId": ~LB)R72OPGDLP[documentID]

Use as parameter the identifier of the document we want to open and the identifier of the MetaClass of the document we want to open (for example MetaClass Document (static)).

Use example:

Let us take a static document with the following absolute identifier: WEsYsj8PGnU4print

VB Script

```
Print mgobjMegaObject.getRoot.DesktopURLBuild
("Tool=LA)RAINPGnHP,Param=LB)R72OPGDLP,ParamValue=WEsYsj8PGnU4,ParamValueMe
taclass=UkPT)TNyFDK5")
```

Java

```
mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]",
mgobjMegaObject.getRoot().invokePropertyGet("DesktopURLBuild",
"Tool=LA)RAINPGnHP, Param=LB)R72OPGDLP, ParamValue=WEsYsj8PGnU4,
ParamValueMetaclass=UkPT)TNyFDK5"));
```

Result example:

```
http://localhost/HOPEX/default.aspx?userdata=Tool-LA%29RAINPGnHP|LB%29R72OPGDLP-WEsYsj8PGnU4-
UkPT%29TNyFDK5
```

Note:

This MegaTool also functions for static document versions, system static documents, system static document versions and also for external references and system external references. To do this, just modify the ParamValueMetaClass.

11.33.3 SaveAsStatic

Saving a dynamic document (Word document, book, report) as a static document.

Function prototype:

```
Function SaveAsStatic(mgoDocPattern As MegaObject, bBatch As Boolean, strFormat As String)
As String
```

- `mgoDocPattern` static document template (can be Nothing)
- `bBatch` a boolean indicating if we want to create the static document in batch or not (display of static document wizard or not)
- `strFormat`, the generation format. Can be empty "".

  For books, available formats are: "PDF" or "RTF" (RTF by default)

  For reports, available formats are: "PDF", "RTF", "XLS" (RTF by default)

  For documents, format is not taken into account. It depends on the option defined in HOPEX (DOC or RTF) and on the generation context (HOPEX Windows Front-End or HOPEX Web Front-End)
- It returns the identifier of the object (static document) created.

Use example:

Let us take mgoDynamicDoc a book, a report or a document (Word dynamic)

```
sDocId = mgoDynamicDoc.CallFunction("~9Zy9faIVGbdG[SaveAsStatic]",
mgoDocPattern, bBatch, strFormat)
ou
sDocId = mgoDynamicDoc.SaveAsStatic(mgoDocPattern, bBatch, strFormat)
```

11.33.4 Macro Script global properties (MegaPropertyBag)

Script macro global data

At creation of a MEGA macro, a global component is made accessible to macro Script code. This component is named **MegaMacroData**. It enables indication of the script instancing context, and also handling of macro global variables.

- Macro context data: this data comprises read-only variables.
    - **MegaMacroData.GetID**: absolute identifier of the MEGA macro instancing the current script.
    - **MegaMacroData.ServiceID**: absolute identifier of the service implemented by the macro. For example, when a macro implements a *MetaCommand*, this variable contains the absolute identifier of the *MetaCommand*. This function offers the possibility of implementing a reusable macro, since it can access a configuration available on the implemented service.
    - **MegaMacroData.InitString**: macro initialization string. In certain use cases, a macro is initialized with a configuration string; this enables macro reuse without creating a specific service. This variable enables access to this initialization string.
- *PropertyBag* associated with a macro: this function enables definition of macro global properties, which is independent of script instances executing macro methods.

  The Microsoft script interpreter used to execute macro code is strictly monothread: it can be accessed (or destroyed) only in the thread in which it was created. A macro can be globally instanced and called in a multithread context: in particular this is the case for implementations of *MetaAttribute* and *_Operator*. To operate in these contexts, the macro execution engine instances as many scripts as there are calling threads. Each script instance is strictly independent of the others: In particular this is true for globals declared in a script – including the global code, which is then executed for each new script instance. If, for optimization reasons, it is desirable to maintain globals valid for all macro script instances, we must have available a mechanism independent of the script. To do this, a global *PropertyBag* is made available to the macro.

  MegaMacroData.GetBag enables access to the global *PropertyBag* of the macro. This *PropertyBag* is a component implementing the **MegaPropertyBag** interface described in the paragraph below.

MegaPropertyBag component

Such a component enables management of a list of named properties, in a similar way as for a javascript class. This data can be managed and maintained independently of the instances of scripts that handle them.

You can access a **MegaPropertyBag**, either:

- by using MegaMacroData.GetBag property, available on each macro script, or
- by instantiating specifically a MegaPropertyBag as follows:

VB Script	`MegaToolkit.CreateMegaObject(« MegaPropBag »)`

Java	`MegaPropertyBag mgpbBag = new`
	`MegaPropertyBag(mgRoot.currentEnvironment().toolkit());`

Default mode of MegaPropertyBag operation does not require prior declaration of stored variables. Its operation is therefore very similar to that of a javascript class, as the following example shows:

```
Set Bag = CurrentEnvironment.Site.Toolkit.CreateMegaObject("MegaPropBag")

Bag.Prop1 = "Value 1"

Bag.Prop2 = 17

print Bag.Prop1

print Bag.Prop2
```

Unlike VBScript assignment, object reference assignment is automatic, and keyword **Set** should not be included in assigning an object to a property:

```
Bag.Prop3 = GetCollection("Package").Item(1)

print Bag.Prop3.GetProp("Name")

Keyword Item enables generic access to a property:

print Bag.Item("Prop1")
```

You can configure a propertyBag to impose declaration of properties by means of the Explicit method; this system limits risk of error when you want to share content of a propertyBag, or simply to avoid programming bugs (assured by the Explicit Option in VBScript).

```
Bag.Explicit
```

Used as a function, Explicit indicates the PropertyBag operating mode.

In Explicit mode, you need to declare a property – using Dim method – before using it, either for reading or update.

```
If Bag.Explicit Then

Bag.Dim "Prop1"

Bag.Dim "Prop2"

End If

Bag.Prop1 = "Value 1"

Bag.Prop2 = 17
```

in VBScript, the propertyBag is also an iterator enabling listing of properties used (either in update or consultation).`For Each propName In Bag`

```
print "Prop " & propName & " = " & Bag.Item(propName)

Next
```

In languages in which the iterator cannot be used (such as java or javascript), we can access properties using an index and the Count function. In this case, we cannot access the property name (future function ItemName).

VB Script

```
For propIndex = 1 To Bag.Count

print "Prop #" & propIndex & " = " & Bag.Item(propIndex)

Next
```

Java

```
MegaPropertyBag mgpbBag = new
MegaPropertyBag(mgRoot.currentEnvironment().toolkit());

mgpbBag.explicit(true);

mgpbBag.dim("Prop1");

  mgpbBag.dim("Prop2");

  mgpbBag.basedObj.invokePropertyPut("Prop1", "Value1");

  mgpbBag.basedObj.invokePropertyPut("Prop2", 17);

  for (int j = 1; j <= mgpbBag.count(); j++) {

  mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]", "Prop #" + j + " = " +
mgpbBag.item(j));

  }
```

You can test the nature of a *propertyBag* variable, so as to determine if it is an object or a data item, by means of the **IsObject** function.

If the list of properties indicated in the above example includes objects, these should be treated differently:

```
For Each propName In Bag

If Bag.IsObject(propName) Then

Set propItem = Bag.Item(propName)

print "Prop " & propName & " is an Object"

Else

propItem = Bag.Item(propName)

print "Prop " & propName & " = " & propItem

End If

Next
```

propertyBag expression evaluator.

The *propertyBag* has an expression interpretation function; Evaluated expressions are based on a VBScript syntax and allow only elementary actions:

- numerical or alphanumerical expression calculation

- assignment of properties with these expressions.

In particular, this evaluator is used by indicators in HOPEX; it enables call on dynamic code without calling **ExecuteGlobal**. This function offered by Microsoft scripting has the significant drawback of enabling implementation of a trojan horse, since the Script code called has MEGA macro execution rights, and more specifically of the connected user - this therefore representing a potential security risk. However the expressions interpreted by a *propertyBag* do not have all VBScript functions (the **CreateObject** function cannot be called) thus limiting security fault risks.

In these expressions, variables used correspond to *propertyBag* properties. It is nevertheless possible to cite external variables managed by a component given as a parameter to the evaluator; these variables are presented in the expression in the form of fields. The evaluator calls the function whose name is passed as parameter when a field is found in the expression.

A field can comprise a series of fields separated by dots and possibly terminated by an option:

`Field1.Field2.option`

The function implemented allows as parameter the propertyBag itself, which can supply context information required for evaluation of the field using the following contextual functions:

- Number of consecutive fields:

VB Script

```
Bag.FieldCount
```

Java

```
mgpbBag.fieldCount()
```

- Identifier corresponding to nuField field:

VB Script

```
Bag.FieldValue(nuField)
```

Java

```
mgpbBag.fieldValue(nuField)
```

- Option value:

VB Script

```
Bag.FieldOption
```

Java

```
mgpbBag.fieldOption()
```

The evaluator exists in two forms:

- an expression evaluator, which returns a value.

VB Script

```
MegaPropertyBag.Evaluate(
EvaluableString As String,
FieldResolver As Object,
ResolvingFunction As String) As Variant
```

Java

```
public Object evaluate(final String formula, final Object
fieldProcessor, final String callbackName)
```

- a code evaluator, which enables assignment of *propertyBag* properties with expressions. This evaluator can include several VBScript instructions separated by ':' or on different lines.

VB Script

```
MegaPropertyBag.Execute(
EvaluableString As String,
FieldResolver As Object,
ResolvingFunction As String)
```

Java

```
public String execute(final String script, final Object fieldProcessor,
final String callbackName)
```

Example:

VB Script

```
Class FieldResolver
Function Field(Bag)
```

```
' nothing to evaluate

End Function
End Class


Set Bag =
CurrentEnvironment.Site.Toolkit.CreateMegaObject("MegaPropBag")

Bag.Prop1 = "Value 1"

Bag.Prop2 = 17

Bag.Sel = GetCollection("Package")

print Bag.Evaluate("Prop1 & Prop2 & Sel.Count", New FieldResolver,
"Field")

Bag.Execute("Prop1 = Prop2 & Sel.Count", New FieldResolver, "Field")

print Bag.Prop1
```

Java

```
class FieldResolver {

    public FieldResolver() {}

    public String Field(final MegaPropertyBag mgpbBag) {

      //nothing to evaluate

      return "";

    }

  }

MegaPropertyBag mgpbBag = new
MegaPropertyBag(mgRoot.currentEnvironment().toolkit());

mgpbBag.basedObj.invokePropertyPut("Prop1", "Value1");

mgpbBag.basedObj.invokePropertyPut("Prop2", 17);

mgpbBag.basedObj.invokePropertyPut("Sel",
mgRoot.getCollection("Package"));

mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]",
mgpbBag.evaluate("Prop1 & Prop2 & Sel.Count", new FieldResolver(),
"Field"));

mgpbBag.execute("Prop1 = Prop2 & Sel.Count", new FieldResolver(),
"Field");

mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]", (String)
mgpbBag.item(3));
```

12 Coding recommendations & Performances

12.1 Coding recommendations

12.1.1 Handling Identifiers

You must make reference to an object using its absolute identifier rather than its name: in this case the code is most highly optimized and resists renaming of the instance as well as change of language.

An absolute identifier is a unique identifier that can be assigned to any instance, characteristic, or link in HOPEX.

However, an absolute identifier is less readable than a name, which is why the MEGA Script Editor offers the possibility of using fields rather than the name of the MEGA object.

Each MEGA instance is identified uniquely by its absolute identifier. This identifier is a 64-bit key, represented by MEGA in the form of a 12 character string.

Objects that define the MEGA metamodel (MetaClasses, MetaAssociations and MetaAttributes amongst others) are also MEGA objects; and they therefore have absolute identifiers.

Obtaining identifiers

A data item representing a MEGA object is implicitly considered as an identifier. In particular, an identifier enables positioning in a collection. For example, if "OperationID" variable contains an operation identifier, it can be used to execute a search in a collection of operations as follows:

VB Script

```
Set myOperations = ...
' myOperations is a collection of operations. OperationID = ...
' OperationID is the identifier of an operation.
Set anOperation = myOperations(operationID)
If anOperation.Exists Then
OperationID = anOperation.GetID ' the variable keeps the same value.
End If
```

Java

```
MegaCollection mgcolOperations = mgobjProject.getCollection("Operation");
              // mgcolOperations is a collection of operations.
              Object objOperationID =
mgobjProject.getCollection("Operation").get(1).getID();
              // objOperationID is the identifier of an operation.
              MegaObject mgobjOperation =
mgcolOperations.get(objOperationID);
              if (mgobjOperation.exists()) {
                objOperationID = mgobjOperation.getID();
                // the variable keeps the same value.
              }
```

The **GetID** function obtains the identifier of an object in its internal format.

The value returned by the GetID function is only meaningful within the same executable. This value cannot be generalized and must not be used globally.

The **GetPropID** and **GetCollectionID** functions are used to obtain the identifier (not the value) of an object or a MetaAssociationEnd accessible from an instance. These values are persistent and independent of language (in the context of a multilingual repository for example).

VB Script

```
Ope.GetProp("Name")
Ope.GetProp(Ope.GetPropID("Name"))
' These last two lines are synonymous.
```

Java

```
String strName = mgobjOperation.getProp("Name");
                        strName =
mgobjOperation.getProp(mgobjOperation.getPropID("Name"));
                        strName = (String) mgobjOperation.invokePropertyGet("Name");
```

Note that **GetProp** (like **GetCollection**) accepts a name or identifier as parameter. The **Item** function also accepts name or identifier as parameter:

VB Script

```
Set myOperations = oProject.GetCollection ("Operation")
' the code finds a collection of operations.
Set anOperation = myOperations.Item("operation-1")
```

Java

```
MegaCollection mgcolOperations = mgobjProject.getCollection("Operation");
                     //the code finds a collection of operations.
                     MegaObject mgobjAnOperation = mgcolOperations.get("operation-
1");
```

The object type for an instance is obtained with the **GetClassID** function.

Using fields

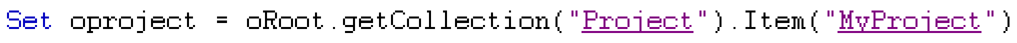
This consists of replacing the character string containing the name by a character string starting with escape character '~', followed by the absolute identifier, then by the object name between square brackets.

Field display with standard editor

```
Set myproject=
oRoot.getCollection("~qekPESs3iG30[Project]").Item("~7qv3W01mCz10[MyProject]")
```

Field display with scriptSet editor

The script editor masks the field code and displays the name only, underlined.



```
Set oproject = oRoot.getCollection("Project").Item("MyProject")
```

In the properties dialog box of a macro, the VB Script tab enables edit of VB Script of the macro. The {} ***Hide/Show Fields*** *button enables display of fields used in the macro, according to"ScriptSet" mode or to "standard" mode.*

Fields can be used in different functions such as: ***GetCollection****,* ***GetProp****,* ***SetProp****,* ***Item****,* ***GetMacro****,* ***GetObjectFromID****.*

Using standard functions available to convert and compare identifiers

Using the MegaField function

To represent a MEGA field identically from one editor to another, the **MegaField** function builds the field corresponding to the identifier of an object.

VB Script

```
myprojectField = myproject.MegaField

Set operations =
Project.Item(myprojectField).getCollection("Operation")
```

Java

```
String strProjectField =
mgRoot.getCollection("~qekPESs3iG30[Project]").item("My
Project").megaField();

               MegaCollection mgcolOperations =
mgRoot.getCollection("~qekPESs3iG30[Project]").item(strProjectField).g
etCollection("Operation");
```

Using directive fields

Fields can also be used to invoke methods of a ***MegaObject***. This type of MEGA script cannot be used directly and must be transformed in order to be executed. This transformation is controlled by the ***Fields*** directive in the **MegaContext** command invoked below.

```
'MegaContext(Fields)    - field interpretation activation

'MegaContext(Fields,Batch) - field interpretation and global DBRoot deactivation
```

When the "Fields" context is activated, the VB script is pre-interpreted so that its fields can be replaced by expressions compatible with VB Script syntax. MetaClasses, MetaAttributes and MetaAssociationEnds can then be pasted in the script in the form of fields rather than the corresponding object names.

<u>Example:</u>

The following code that does not use fields:

```
for each ope in myOperations

  print ope.Nom

next
```

can be replaced by the following which uses them:

```
'MegaContext(Fields)

for each ope in myOperation

  print ope.Name

next
```

When this script is opened with an editor that does not process MEGA fields (for example Windows Notepad, the fields appear in their storage format.

```
'MegaContext(Fields)

for each ope in myOperation

  print ope.~210000000900[Name]

next
```

This file cannot be directly executed by the script interpreter. The **MegaContext(Fields)** option enables transformation of this code to a script acceptable by the script interpreter. For information, this transformation is limited to moving an opening square bracket: the code below can therefore be executed.

```
'MegaContext(Fields)

for each ope in [~gsUiU9B5iiR0myOperation]

  print ope. [~210000000900Name]

next
```

12.1.2 How to speed up queries in API code by using Absolute Identifiers

The way you use to write queries in API code may affect the performance of the query.

The following information shows how to raise performance in queries used inside the code (Java or VB) by simply changing names to absolute identifiers.

Example:

VB Script

```
root=getRoot

s=Timer

for i=1 to 10000

set res = root.getSelection("Select [Application] Where [Defined-
Service].[Operation].[Organizational Process]='World@Hand::BPMN Notation
Diagrams:: Purchasing:: Purchase Goods & Services:: Contract Negotiation'")

next

e=Timer

print "Query without IdAbs: " & (e-s)*1000 & "ms"

s=Timer

for i=1 to 10000

set res = root.getSelection("Select ~MrUiM9B5iyM0[Application] Where
~ltSTdNNHjqj0[Defined-Service] in (~TsUiT9B5iyQ0[IT Service] WHERE
~hqUiTCB5iK72[Operation].~mrUiaCB5iCB2[Organizational
Process]='~W3qoNsjV91e6[Contract Negotiation]')")

next

e=Timer

print "Query with IdAbs: " & (e-s)*1000 & "ms"
```

Java

```
Date dCurrDateS1 = new Date();

for (int j = 1; j <= 10000; j++) {

MegaCollection mgcolTest = mgRoot.getSelection("Select [Application] Where
[Defined-Service].[Operation].[Organizational Process]='World@Hand::BPMN
Notation Diagrams:: Purchasing:: Purchase Goods & Services:: Contract
Negotiation'");

}

Date dCurrDateS2 = new Date();

mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]", "Query without IdAbs: " +
(dCurrDateS2.getTime() - dCurrDateS1.getTime()) + "ms");

dCurrDateS1 = new Date();

for (int j = 1; j <= 10000; j++) {

MegaCollection mgcolTest = mgRoot.getSelection("Select
~MrUiM9B5iyM0[Application] Where ~ltSTdNNHjqj0[Defined-Service] in
(~TsUiT9B5iyQ0[IT Service] WHERE
~hqUiTCB5iK72[Operation].~mrUiaCB5iCB2[Organizational
Process]='~W3qoNsjV91e6[Contract Negotiation]')");

}

dCurrDateS2 = new Date();

mgRoot.callFunction("~U7afnoxbAPw0[MessageBox]", "Query with IdAbs: " +
(dCurrDateS2.getTime() - dCurrDateS1.getTime()) + "ms");
```

The environment is Demonstration with MEGA 2009 SP5 CP6 R6.

We execute (a lot of times, 10000 times) the same query and then we will print the elapsed time.

The first query uses the names for MetaClass, MetaAssociationEnds, and the name for the source object.

The second query replace names with Absolute Identifiers (in the MegaField format).

The first cycle of queries took 20312,51 ms to be executed (about 20 seconds), the second cycles took about 16796,885 ms to be executed (a little bit more than 1 second and half).

By using idabs for the query we gain about the 25% in performances.

Remember (for the entry point) that MEGA accept a megaField when you can put a name.

To get the megaField of an object (its Id) just write `myObject.megaField` (you can use also `.megaUnnamedField`, in this case the name of the object will be replaced by a X, for the system it is the same).

In complex reports where you do a lot of queries the execution time of the report can benefit from queries with Identifiers.

This approach can also be implemented for imbedded queries in HTML descriptions for websites.

Not only it is interesting as far as peformance is concerned, it is also more maintainable (as written text for names of MetaClasses, MetaAssociationEnds and MetaAttributes is not stable in the long term).

12.1.3 Browsing repository (collection use)

There is no other way than browsing through the repository the information you want to handle in your code.

⚠ **However, be aware that browsing the repository is time consuming (e.g.: finding objects, listing related objects).**

You must pay particular attention to write optimized code.

Do not ask several times the same information to the repository.

Do not ask an information you do not need to the repository.

Take advantage of indexes in your browsings.

Here are examples of good vs bad :

Example 1:

- Bad: count is called as many time as there are elements

```
for (int i = 1; i <= mgobj.getCollection(…).count(); i++) {
  ...
}
```

- Good:

```
int iCount = mgobj.getCollection(…).count();
for (int i = 1; i <= iCount; i++) {
  ...
}
```

Example 2:

mgidSearchedId is the object identifier that you find via a MetaAssociationEnd from another object

Bad: browsing all the items of a list when you are interested in only one item and you know how to identify it by an indexed attribute

```
int iCount = mgobj.getCollection(…).count();
for (int i = 1; i <= iCount; i++) {
  MegaObject mgobjChild = mgobj.getCollection(…).get(i);
  if (mgobjChild.sameId(mgidSearchedId)) {
    mgobjSearched = mgobjChild;
    // my code
  }
}
```

Good:

```
MegaObject mgobjSearched = mgobj.getCollection(…).get(mgidSearchedId);
if (mgobjSearched.exists()) {
  // my code
}
```

Example 3:

You have an identifier in Hexadecimal format (Base 16) and you want it in Base 64 (or any other variant enabling to retrieve the identifier in any other format)

Bad: searching for an object to retrieve information you already have in another format.

```
strId = mgRoot.getObjectFromId(strHexaIdentifier).getProp(
"~310000000D00[Absolute Identifier]");
```

- Good:

```
strId = mgToolKit.getString64FromId(strHexaIdentifier);
```

12.1.4 Writing code rules

Writing code rules regarding GUIs

For web compatibility:

- do not use:
  - Java GUIs in MEGA code
  - VB (Visual Basic) GUIs in MEGA code
- use HOPEX forms to execute your own GUIs, with:
  - Property pages
  - Wizards

For detailed information on Property Pages and Wizards, see HOPEX Power Studio > Customizing the user Interface > ***Forms****.*

Web compatibility: using Java or VB (Visual Basic) GUIs in MEGA code would launch the GUIs on HOPEX server instead of the Web client and block HOPEX server.

Writing code rules regarding Performance

Calculated MetaAttributes:

- browsing an excessive number of objects is forbidden
- do not over consume resources

Release objects:

See Avoiding processes to go slower: Tracking down non released instances.

- Java: release objects instead of using the default garbage collector, enter:

```
yObj.release()
```

- VB Script:

`Set myObj = Nothing` only if the process may take a long time.

Do not use GetObjectFromID, instead use:

```
myRoot.GetCollection(«  ~MrUiM9B5iyM0[Application]  »).Item(AppID)
```

Use of field with IdAbs is mandatory:

See MegaFields, MegaFields, and Handling Identifiers.

- the field format is: absolute identifier followed by the object name:

```
~MEsJ0p5rATw0[AllStoppedWorkflows]
```

- to retrieve the field form an object, enter:

VB Script `sID = myObject.megafield`

Java `sID = myObject. megaField();`

When the object name is not required, to improve performance, use megaUnnamedField:

VB Script `sID = myObject. megaUnnamedField`

Java `sID = myObject.megaUnnamedField();`

Use variables to store objects

12.1.5 Confidentiality

Objects in HOPEX may be confidential.

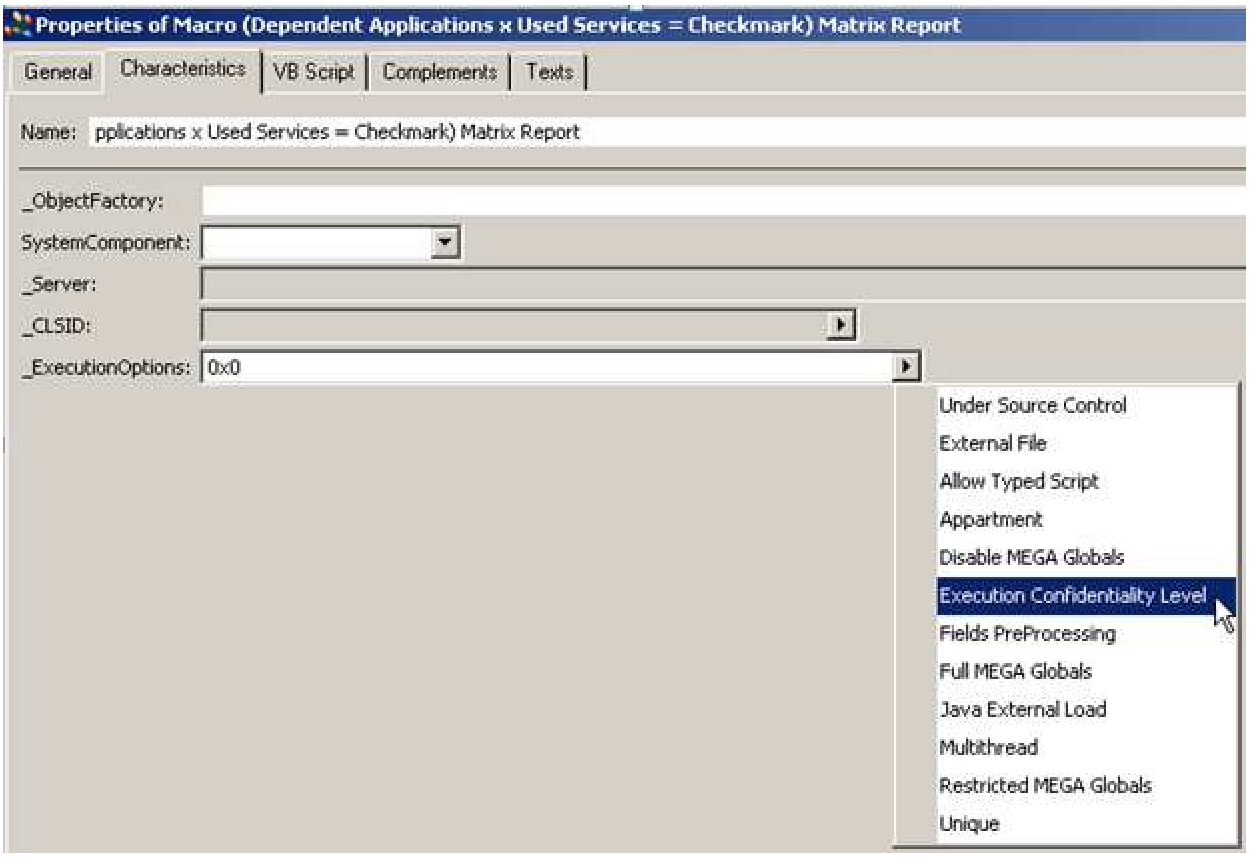
A macro does not give you acces to confidential objetcs:

`GetProp(MyProp , « display »)` sends back ****

`GetCollection` sends back a collection with the confidential objects but you cannot access to any properties

According to the macro parameterization, scripts are executed at:

- the macro confidentiality level : algorithm
- the user confidentiality level: displayed information



Properties of Macro (Dependent Applications x Used Services = Checkmark) Matrix Report

Name: pplications x Used Services = Checkmark) Matrix Report

_ObjectFactory:

SystemComponent:

_Server:

_CLSID:

_ExecutionOptions: 0x0

When creating a macro, always keep in mind the confidentiality issue.

The macro must give a valid result.

Examples:

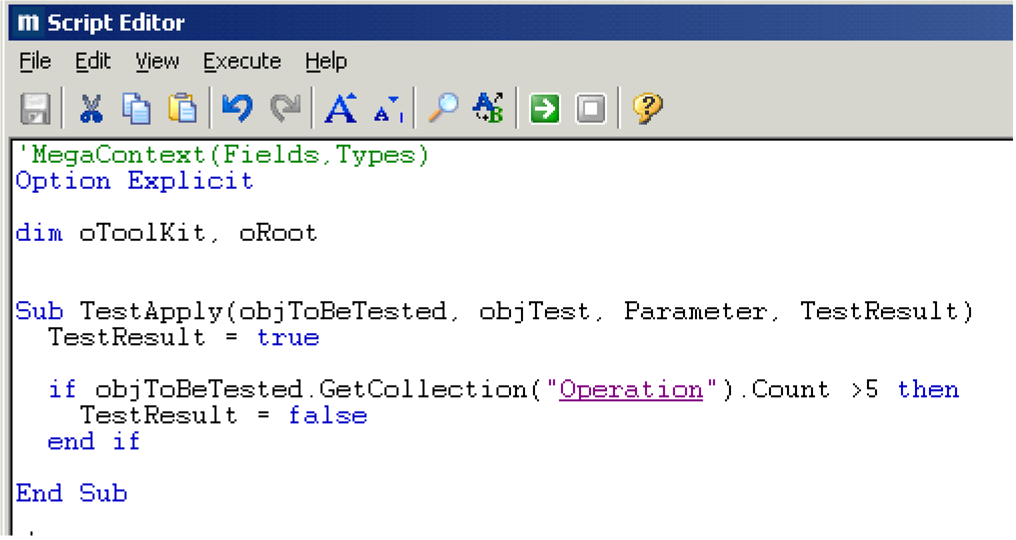
- **Regulation rule**: the organizational process cannot include more than five operations
- **Properties**: display the name and comment of an organizational process operations
- **Matrix**: displays the relations between the organizational processes and IT Services through operations

Example1: Regulation rule

Regulation rule: the organizational process cannot include more than five operations

For example:

- The organizational process P1 includes six operations: Op1, … , Op6.
- User U1 can see P1 but Op3 is confidential
- User U2 can see P1 and all of its operations
- Script:



```
'MegaContext(Fields,Types)
Option Explicit

dim oToolKit, oRoot


Sub TestApply(objToBeTested, objTest, Parameter, TestResult)
  TestResult = true

  if objToBeTested.GetCollection("Operation").Count >5 then
    TestResult = false
  end if

End Sub
```

When the above script is executed with:

- U1 view, the procedure execution result gives:

  5 ≤ 5 -> TestResult = true

  The regulation rule is valid and displays that the organisational process respect the rule.

- U2 view, the procedure execution result gives:

  6 > 5 -> TestResult = false

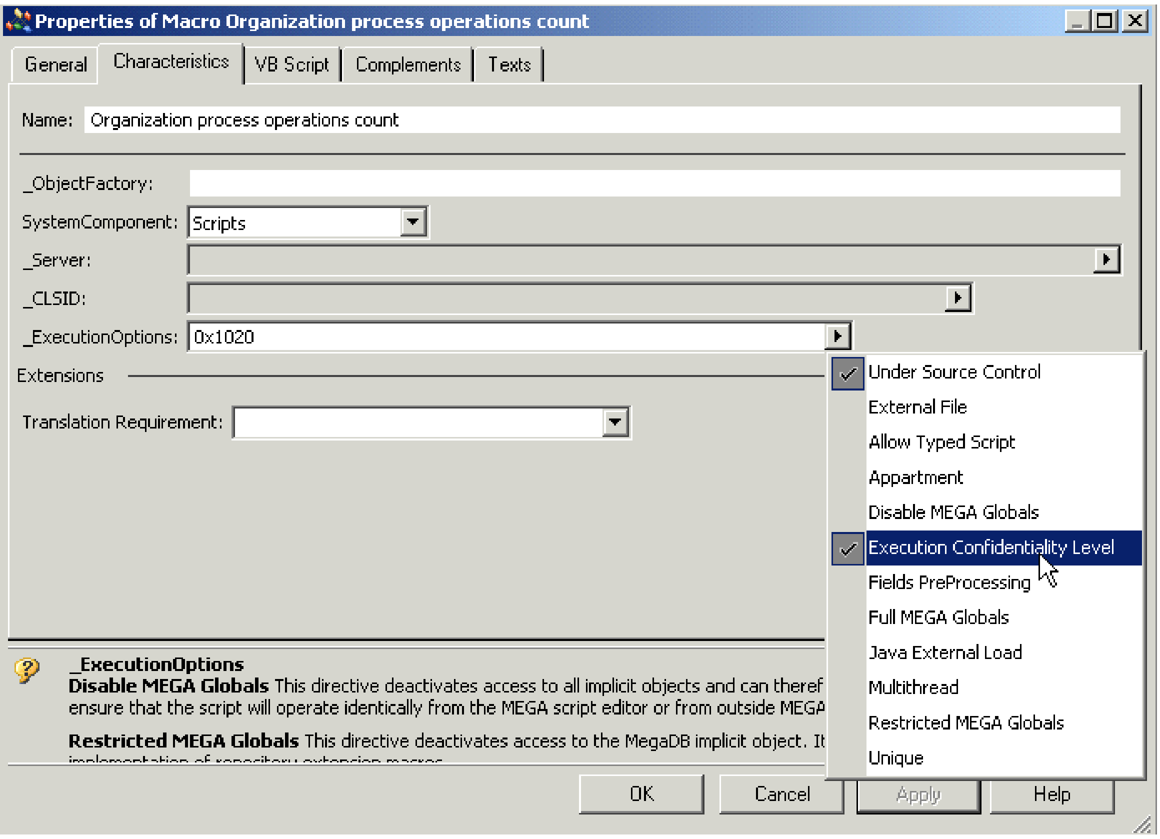
  The regulation rule is not valid and displays that the organisational process does not respect the rule.

Results are different for user U1 and user U2

This script is not correct: the rule must give the same result for all the users.

To configure the macro related to the organizational process

1. In the macro **Properties** window, **Characteristics** tab, set **_ExecutionOptions** to “Execution Confidentiality Level”.



Properties of Macro Organization process operations count

Name: Organization process operations count

_ObjectFactory:

SystemComponent: Scripts

_Server:

_CLSID:

_ExecutionOptions: 0x1020

Extensions

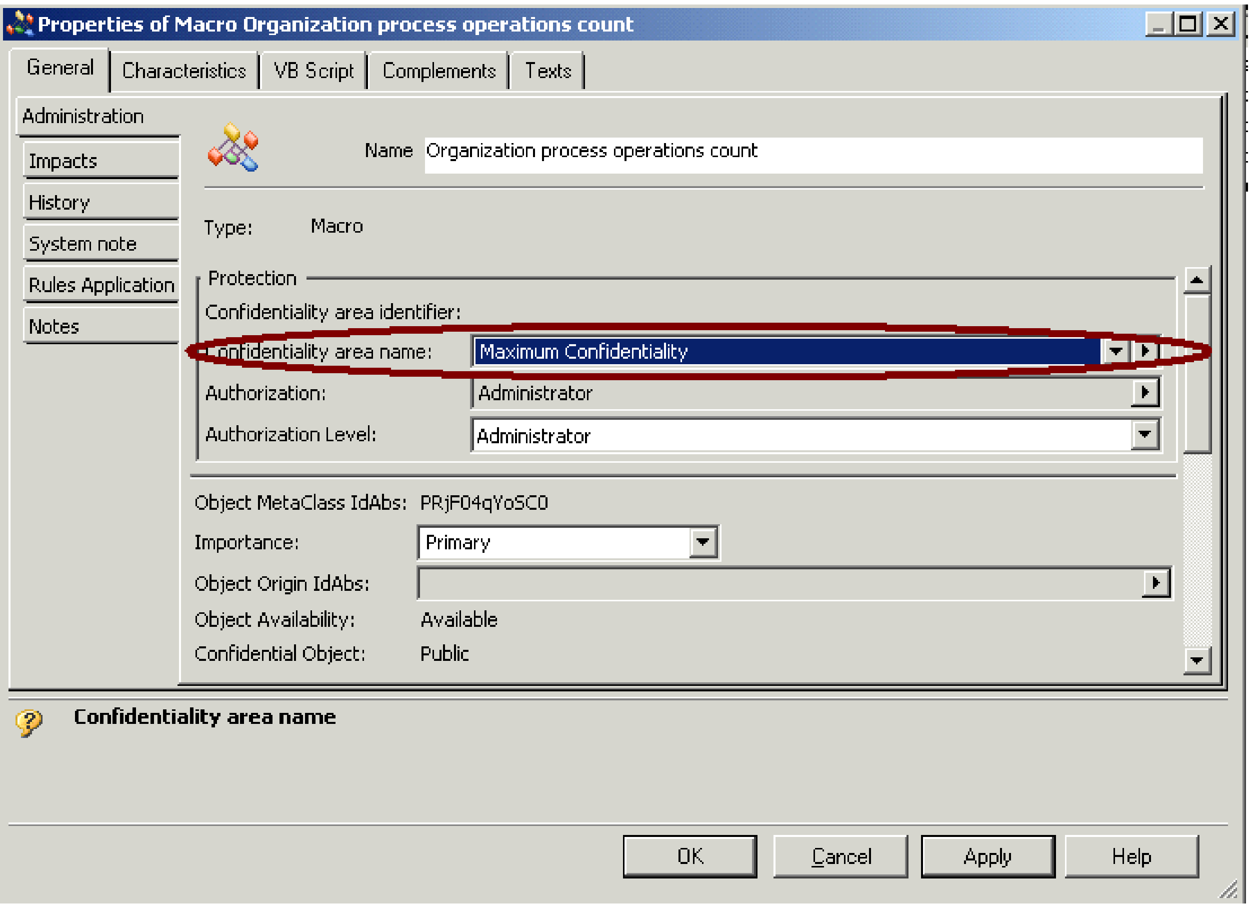
Translation Requirement:

☑ Under Source Control

☑ Execution Confidentiality Level

_ExecutionOptions
Disable MEGA Globals This directive deactivates access to all implicit objects and can theref
ensure that the script will operate identically from the MEGA script editor or from outside MEGA
Restricted MEGA Globals This directive deactivates access to the MegaDB implicit object. It
implementation of repository extension macros

2. In the macro **Properties** window, **General** tab, **Administration** sub-tab, set the macro Confidentiality level to “Maximum confidentiality” level



Name: Organization process operations count

Type: Macro

Confidentiality area identifier:

Confidentiality area name: Maximum Confidentiality

Authorization: Administrator

Authorization Level: Administrator

Object MetaClass IdAbs: PRjF04qYoSC0

Importance: Primary

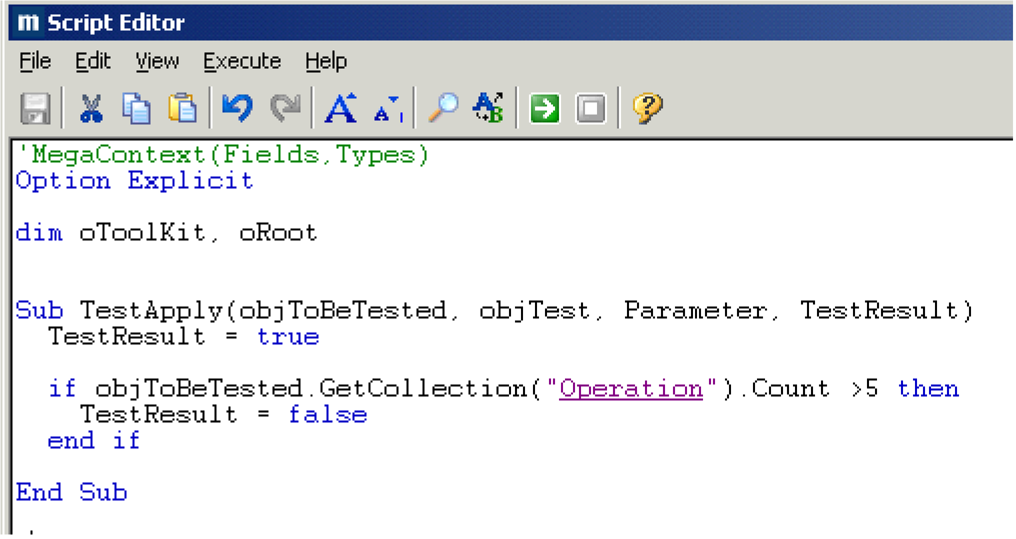
Object Origin IdAbs:

Object Availability: Available

Confidential Object: Public

Confidentiality area name

3. Enter the code:



```
'MegaContext(Fields,Types)
Option Explicit

dim oToolKit, oRoot


Sub TestApply(objToBeTested, objTest, Parameter, TestResult)
  TestResult = true

  if objToBeTested.GetCollection("Operation").Count >5 then
    TestResult = false
  end if

End Sub
```

The code is executed as if the end user had the maximum confidentiality level.

When the above script is executed with U1 view, the procedure execution result gives:

```
6 > 5 -> TestResult = false
```

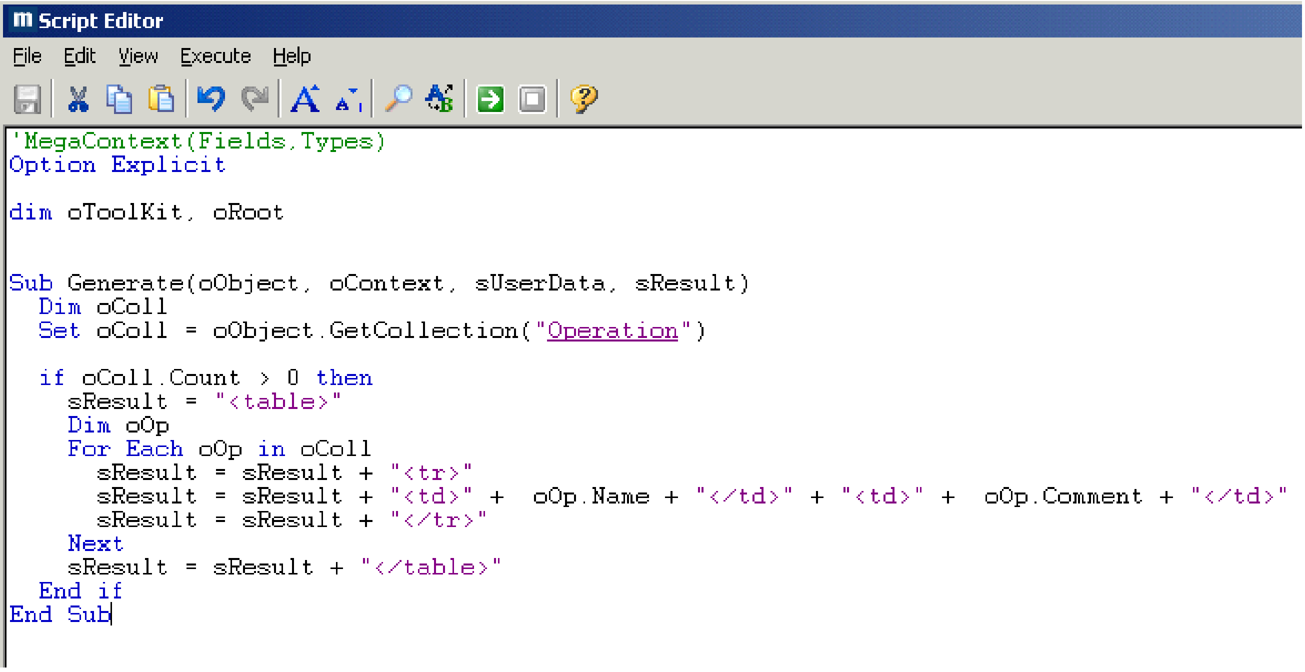
For both users, the regulation rule fails and displays that the organisational process does not respect the rule.

Example2: Properties

When no algoritm is needed, execute the macro at the user confidential level.

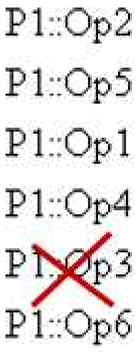
To display the name and comment of an organizational process operations

1. In the Script Editor, enter the following code:



```
'MegaContext(Fields,Types)
Option Explicit

dim oToolKit, oRoot


Sub Generate(oObject, oContext, sUserData, sResult)
  Dim oColl
  Set oColl = oObject.GetCollection("Operation")

  if oColl.Count > 0 then
    sResult = "<table>"
    Dim oOp
    For Each oOp in oColl
      sResult = sResult + "<tr>"
      sResult = sResult + "<td>" +  oOp.Name + "</td>" + "<td>" +  oOp.Comment + "</td>"
      sResult = sResult + "</tr>"
    Next
    sResult = sResult + "</table>"
  End if
End Sub
```

When the code is executed at the maximum confidentiality level, with user U1, the result is wrong as Op3 is confidential for U1.



P1::Op2

P1::Op5

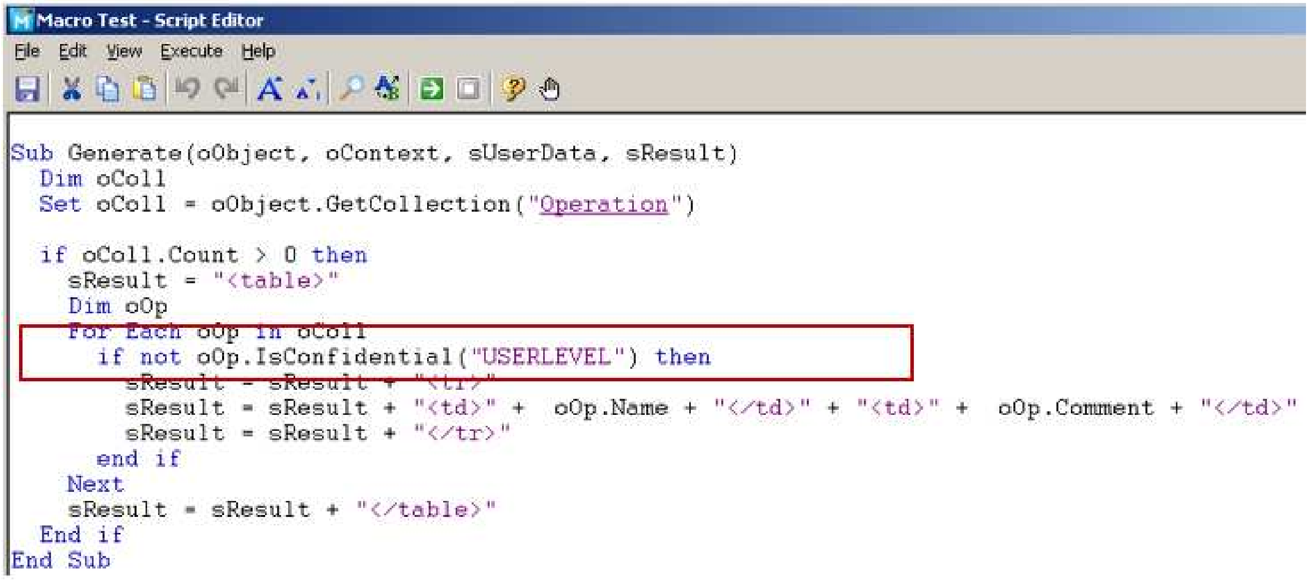
P1::Op1

P1::Op4

~~P1::Op3~~

P1::Op6

2. For the confidential objects not to be displayed use IsConfidential(“UserLevel”).



```
Sub Generate(oObject, oContext, sUserData, sResult)
  Dim oColl
  Set oColl = oObject.GetCollection("Operation")

  if oColl.Count > 0 then
    sResult = "<table>"
    Dim oOp
    For Each oOp in oColl
      if not oOp.IsConfidential("USERLEVEL") then
        sResult = sResult + "<tr>"
        sResult = sResult + "<td>" +  oOp.Name + "</td>" + "<td>" +  oOp.Comment + "</td>"
        sResult = sResult + "</tr>"
      end if
    Next
    sResult = sResult + "</table>"
  End if
End Sub
```

The result is good, Op3 is not displayed:

P1::Op2

P1::Op5

P1::Op1

P1::Op4

P1::Op6

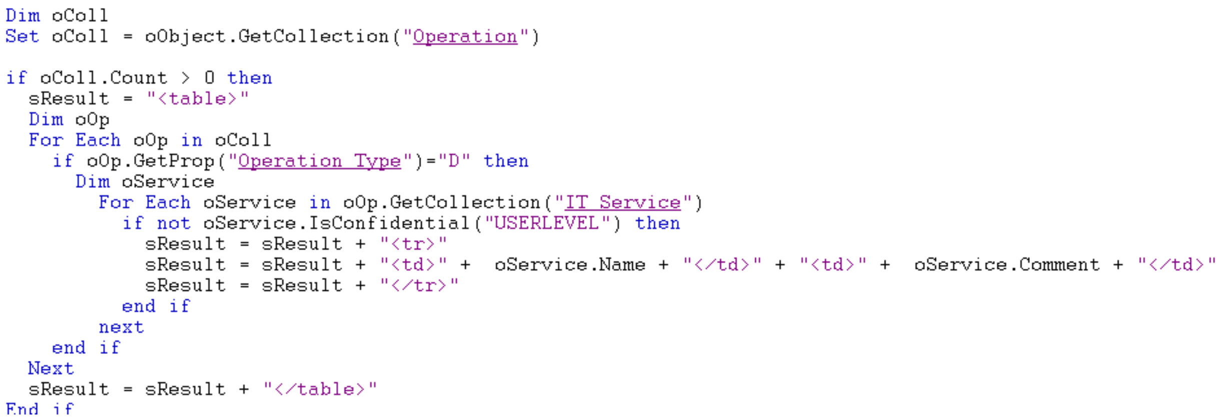
Example2: Matrix

When algorithm is required, execute the macro at the maximum confidentiality level and take into account the confidentiality with the IsConfidential(“USERLEVEL”) function.

All the Operations have the operation type “Decision”.

The purpose is to display the IT Services linked to the organizational process through the operations of Decision type.

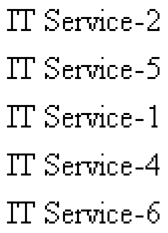
1. In the Script Editor, enter the following code:



```
Dim oColl
Set oColl = oObject.GetCollection("Operation")

if oColl.Count > 0 then
  sResult = "<table>"
  Dim oOp
  For Each oOp in oColl
    if oOp.GetProp("Operation Type")="D" then
      Dim oService
        For Each oService in oOp.GetCollection("IT Service")
          if not oService.IsConfidential("USERLEVEL") then
            sResult = sResult + "<tr>"
            sResult = sResult + "<td>" +  oService.Name + "</td>" + "<td>" +  oService.Comment + "</td>"
            sResult = sResult + "</tr>"
          end if
        next
    end if
  Next
  sResult = sResult + "</table>"
End if
```

2. When the code is executed:

   - at the User level, with the user U1

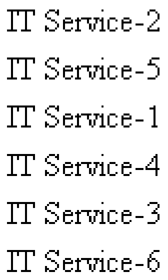


```
IT Service-2
IT Service-5
IT Service-1
IT Service-4
IT Service-6
```

The result is wrong.

Even if Op3 is confidential, the “IT Service-3” IT service linked to P1 should be displayed.

   - at the Maximum confidentiality level, with the user U1,



```
IT Service-2
IT Service-5
IT Service-1
IT Service-4
IT Service-3
IT Service-6
```

The result is correct. Even if Op3 is confidential, the IT service “IT Service – 3” linked to P1 is displayed.

12.2 Performances

This section gives some advices to improve code performance. For these recommendations to be useful the code algorithmic needs to be well written. These recommendations are mainly useful for code handling a large amount of data. They concern:

- access to HOPEX data
- language code

The way the code access data can determine the overall performance of the process. Each query to the repository costs in term of performance so it is better to do the minimal queries or to use cache systems.

12.2.1 Navigating through the metamodel with APIs

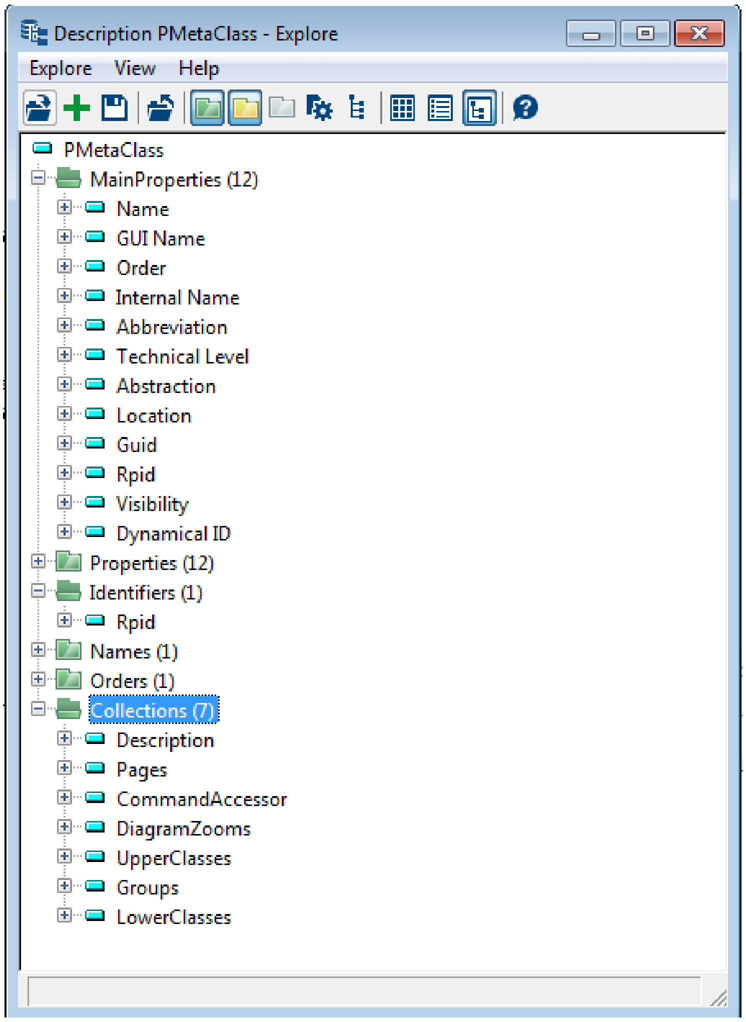
If you need to access to metamodel data (MetaClasses, MetaAttributes, MetaAssociations,…), you must know that this kind of data is rather static and HOPEX offers a metamodel cache to improve navigation performance. Specific APIs are available to access the metamodel, see Accessing the metamodel description using APIs section.

When navigating through the metamodel, use these APIs (instead of standard APIs) to avoid access to repository and improve application performance.

To:

- get a different view on the metamodel, use:

```
getroot.GetClassDescription("~gsUiU9B5iiR0[Organizational
Process]").GetTypeObject().Explore
```

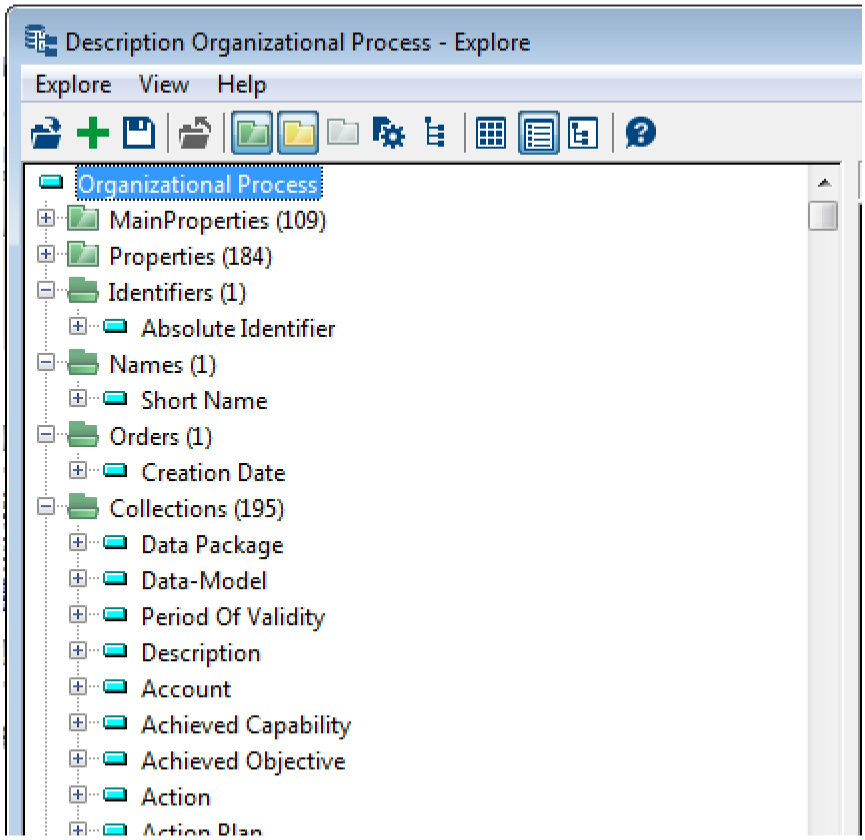

Description PMetaClass - Explore

Explore View Help

```
PMetaClass
  MainProperties (12)
    Name
    GUI Name
    Order
    Internal Name
    Abbreviation
    Technical Level
    Abstraction
    Location
    Guid
    Rpid
    Visibility
    Dynamical ID
  Properties (12)
  Identifiers (1)
    Rpid
  Names (1)
  Orders (1)
  Collections (7)
    Description
    Pages
    CommandAccessor
    DiagramZooms
    UpperClasses
    Groups
    LowerClasses
```

- access the Organizational Process MataClass, enter:

```
getroot.GetCollection(("~gsUiU9B5iiR0[Organizational
Process]").GetTypeObject.Explore
```

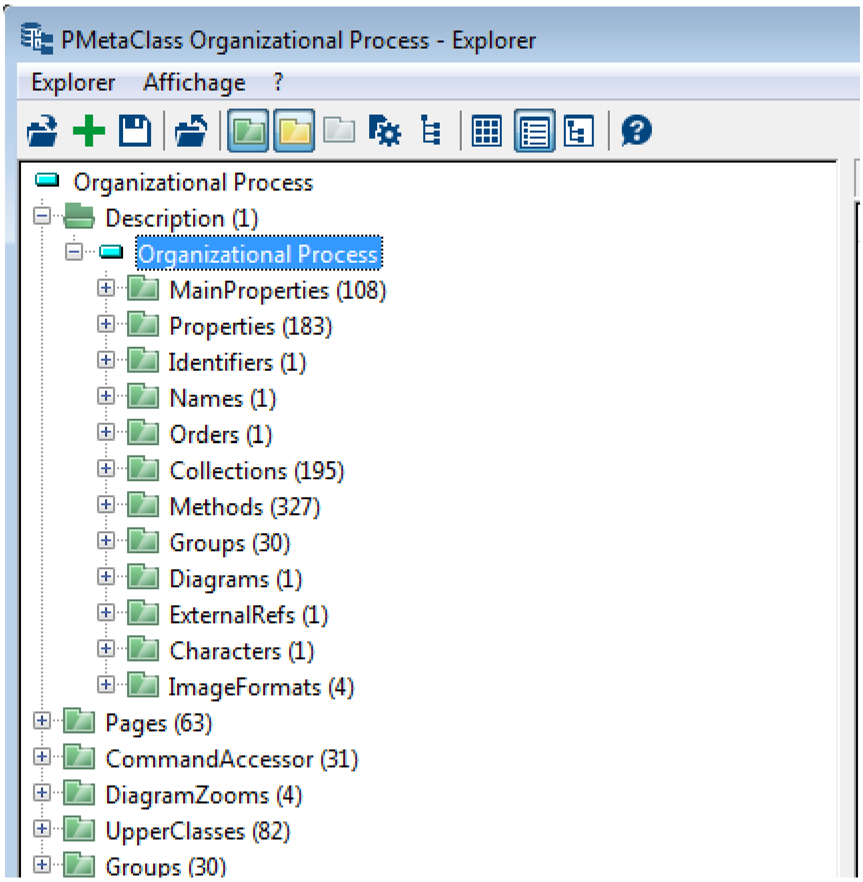


Description Organizational Process - Explore

- Organizational Process
  - MainProperties (109)
  - Properties (184)
  - Identifiers (1)
    - Absolute Identifier
  - Names (1)
    - Short Name
  - Orders (1)
    - Creation Date
  - Collections (195)
    - Data Package
    - Data-Model
    - Period Of Validity
    - Description
    - Account
    - Achieved Capability
    - Achieved Objective
    - Action
    - Action Plan

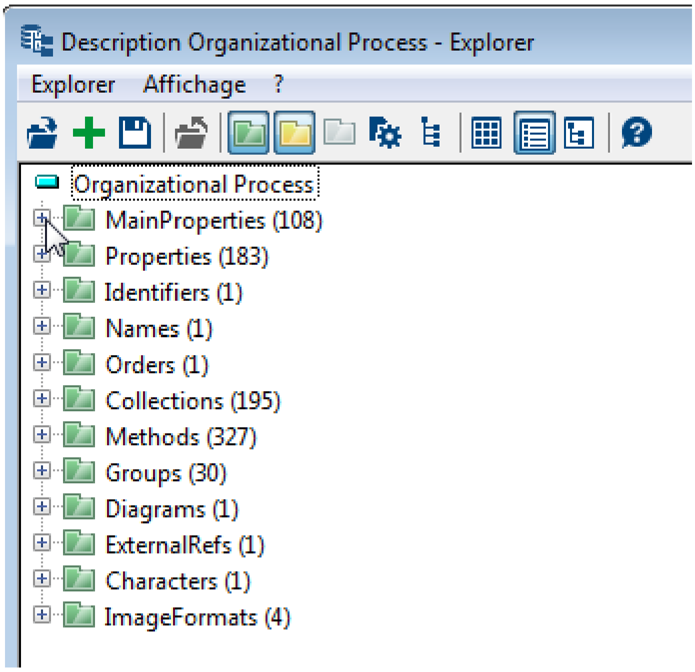
- get a full description of the MetaClass, use **GetClassdescription**:

```
getroot.GetClassDescription("~gsUiU9B5iiR0[Organizational Process]").Explore
```



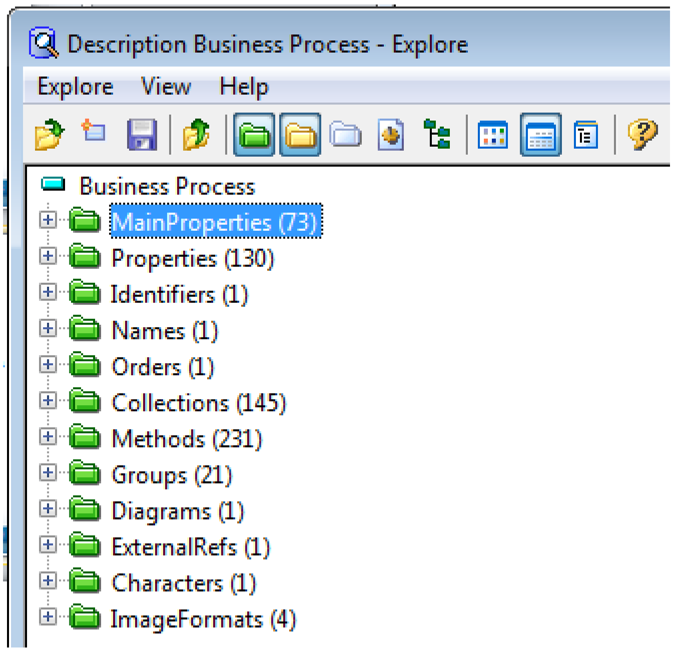
- get a description of the MetaClass only, use GetCollectionDescription

```
getroot.GetCollectionDescription("~gsUiU9B5iiR0[Organizational Process]").
Explore
```



- navigate from an occurrence, use **GetTypeObject** or **GetClassObject**:

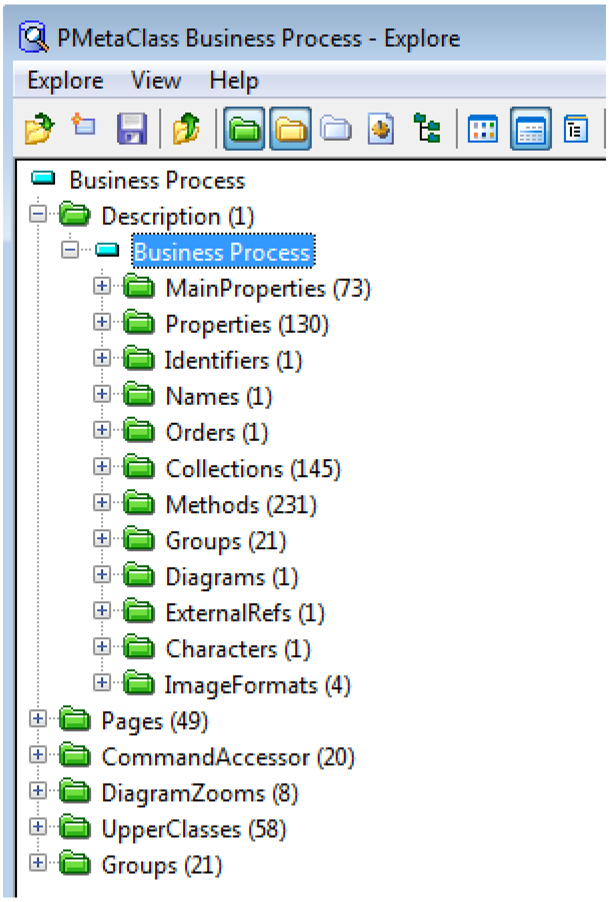
```
GetRoot.GetObjectFromID("~TK0gySBDCjl0[1. Training
Program]").GetTypeObject.explore
```



Description Business Process - Explore

```
Business Process
+ MainProperties (73)
+ Properties (130)
+ Identifiers (1)
+ Names (1)
+ Orders (1)
+ Collections (145)
+ Methods (231)
+ Groups (21)
+ Diagrams (1)
+ ExternalRefs (1)
+ Characters (1)
+ ImageFormats (4)
```

```
GetRoot.GetObjectFromID("~TK0gySBDCjl0[1.  Training
Program]").GetClassObject.explore
```



PMetaClass Business Process - Explore

```
Business Process
- Description (1)
   - Business Process
      + MainProperties (73)
      + Properties (130)
      + Identifiers (1)
      + Names (1)
      + Orders (1)
      + Collections (145)
      + Methods (231)
      + Groups (21)
      + Diagrams (1)
      + ExternalRefs (1)
      + Characters (1)
      + ImageFormats (4)
+ Pages (49)
+ CommandAccessor (20)
+ DiagramZooms (8)
+ UpperClasses (58)
+ Groups (21)
```

VB Script

```
for each oUpperClass in
oRoot.getClassDescription(oMetaClass.GetId).UpperClasses

    set oUpperClass = oRoot.getObjectFromId(oUpperClass.GetId)

    for each omoAnalysisParameter in oUpperClass.Getcollection
("~8yHXTTuE31W0[Report Parameter]")

      set ocAnalysisTemplate.Insert = omoAnalysisParameter.Getcollection
("~kyHXAOuE3jN0[Parameterized Report Template]")

    next

  next
```

Java

```
MegaObject odMetaClass = oRoot.getClassDescription("~" + sClassId);

MegaObject odCollDescrip = odMetaClass.getCollection("Description").get(1);
```

12.2.2 Navigating through data with APIs

This case is the most often encountered.

Megafield usage

A megafield is a string containing the absolute identifier of an object followed by its name. It identifies a unique object. The name between the brackets is optional and can be replaced for example by "[X]".

Example of megafield:

`~MEsJ0p5rATw0[MyApplication]` is the same as `~MEsJ0p5rATw0[X]`

The megafield can be built manually or retrieved using the api megafield available on MegaObjects:

VB Script	Java
id = mgObject.MegaField	id = mgObject.megaField();
Returns ~MEsJ0p5rATw0[MyApplication]	
id = mgObject.MegaField()	id = mgObject.megaUnnamedField();
Returns ~MEsJ0p5rATw0[X]	

If you do not need the name, choose always the second one which improves the performance (it does not compute the name of the object).

The usage of megafields is compulsory for all Mega APIs.

<u>Example</u>:

```
mgObject.getProp("~310000000D00[Absolute Identifier]")
```

is better than

```
mgObject.getProp("Absolute Identifier").
```

This is the same for all Mega APIs (getProp, getCollection, etc.).

Searching an object

It is not recommended to use GetObjectFromId function. If you know exactly the MetaClass which instantiates what you are looking for, prefer writing this:

```
getCollection("~MrUiM9B5iyM0[Application]").item("~MEsJ0p5rATw0[MyApplication]")
;
```

When searching for an instance with its name, do not use a query with getSelection API, but the following code which uses dedicated indexes:

```
getCollection("~MrUiM9B5iyM0[Application]").item("MyApplication");
```

Manipulating identifiers

To compare the identifiers of two objects do not convert them in the same format and then compare them. Prefer the usage of the sameID API. With this one you can use whatever identifier (absolute identifier, hexa identifier or getId). It is available on MegaObject and it can be used as follows:

```
mgObject.sameID("~MEsJ0p5rATw0[MyApplication]");
```

If you have two identifiers, you can use the sameID API available on the MegaToolkit:

```
mgToolkit.sameID("MEsJ0p5rATw0","B1EDB2562C14016F");
```

If you have an identifier in a given format and you need it in anther format, do not go to the MegaObject and ask for the good identifier format. Instead use the APIs available on the MegaToolKit that allow converting identifiers:

```
getString64FromID(myID);

getStringFromID(myID);

getIDFromString(myStringID);

generateID();
```

Loops

In almost all the algorithms we can find loops that allow navigating through objects. If a code like this is needed:

```
for (int i = 1; i <= mgobj.getCollection(…).count(); i++) {

  ...

}
```

or

```
while (i <= mgobj.getCollection(…).count()) {

  i++;

  ...

}
```

Be careful to store the count of the collection before using it in the loop. Otherwise the count function will be evaluated at each loop.

Prefer syntax like this:

```
int iCount = mgobj.getCollection(…).count();

for (int i = 1; i <= iCount; i++) {
```

12.2.3 Deactivating/Reactivating repository log, undo, locks, CRUD test

For performance optimization, you can deactivate/reactivate the following actions in a macro:

- repository log
- undo
- locks
- CRUD test

Use the following method respectively:

```
mgobjRoot.BatchModeEnter "RepositoryLog=Stop",
"RepositoryUndo=Stop","RepositoryLock=Stop","UIPermission=Stop"

// Processing

mgobjRoot.BatchModeLeave "RepositoryLog=Start",
"RepositoryUndo=Start","RepositoryLock=Start","UIPermission=Start"
```

12.2.4 Optimizing the macro of a dynamic data access rule

Dynamic rules for reading or writing data access (permissions) are defined by a macro.

To optimize your macro execution performance, instead of using ERQL queries, use the following methods, which benefit from an efficient session cache:

- **oObject.IsInCurrentAssignment**, to check if a Business Role (e.g. Risk Manager) is assigned to the current person.

  ```
  oObject.IsInCurrentAssignment ("Object=idAbsObject, Location=idAbsLocation,
  Role=idAbsRole, RoleQuery=idAbsQuery, LocationQuery=idAbsQuery,
  ObjectQuery=idAbsQuery")
  ```

  The parameter `idAbsRole` is mandatory, except if the method is called from a Business Role object. The value can be the keyword "Any".

  If i`dAbsObject` is set, the method checks if the Business Role is assigned for this object. The value can be the keyword "Any".

  If `ObjectQuery` is set, the method checks if the Business Role is assigned to an object from which the query result (the query takes the assigned object as parameter) includes the given idAbsObject.

  If `idAbsLocation` is set, the method checks if the Business Role is assigned at this location. The value can be the keyword "Any".

  If `LocationQuery` is set, the method checks if the Business Role is assigned at a location from which the query result (the query takes the assigned location as parameter) includes the given `idAbsLocation`.

- **oObject.IsInCurrentAssignedLocation**, to check if a Business Role (e.g. Risk Manager) is assigned to the current person at a specific location.

  ```
  oObject.IsInCurrentAssignedLocation("Role=idAbsRole, Object=idAbsObject,
  RoleQuery=idAbsQuery, ObjectQuery=idAbsQuery, LocationQuery=idAbsQuery")
  ```

  Same parameters as oObject.IsInCurrentAssignment, except "`idAbsLocation`" parameter, which is set with the oObject on which is called the method.

- **oObject.IsInCurrentAssignedObject**, to check if a Business Role (e.g. Risk Manager) is assigned to the current person for a specific object.

```
oObject.IsInCurrentAssignedObject ("Role=idAbsRole, RoleQuery=idAbsQuery,
Location=idAbsLocation, LocationQuery=idAbsQuery, ObjectQuery=idAbsObject")
```

Same parameters as oObject.IsInCurrentAssignment, except "`idAbsObject`" parameter, which is set with the oObject on which is called the method.

Macro example: macro of the "Incident - Reading.Implementation" Data Access Rule

```
Option Explicit

'--------------------------------------------------------------------------------'

'---------------------------------------------------------------------------------

Sub GetAttributeValue(ByVal mgIncident, ByVal vMetaAttributeId, ByRef Value)

  Dim mgColOrgUnit

  Dim mgOrgUnit


  Value = "0"

  If mgIncident.SameId(mgIncident.GetProp("~(10000000v30[_Creator]"),
mgIncident.GetRoot.CurrentEnvironment.GetCurrentUserId) Then

    Value = "1"

  ElseIf mgIncident.IsInCurrentAssignedObject("Role=~XnhUqvd(HztR[Declarer]")
Then

    Value = "1"

  Else

    Set mgColOrgUnit = mgIncident.GetCollection("~p0b9Go4tH9eC[Declarant's
Entity]")

    If mgColOrgUnit.Count = 0 Then

      Value = "1"

    Else

      For Each mgOrgUnit In mgColOrgUnit

        If
mgOrgUnit.IsInCurrentAssignedLocation("CollectionFromLocation=~(yYIf)afGHsC[Org-
Unit and all its tree]","RoleQuery=~TiS9GRoANzf5[Business Role is LDC]") Then

          Value = "1"

          Exit For

        End If

      Next

    End If

  End If

End Sub
```

12.2.5 Avoiding processes to go slower: Tracking down non released instances

Some behaviors concern codes dealing with many object instances. The process seems to be slower as it progresses. In most cases, it is due to a large amount of living MegaObjects.

As a reminder, the following code:

```
MegaOject mgObj = mgRoot.getObjectFromId("xxxxxxxxxxx");
```

instantiates a MegaObject class and connect it to the HOPEX repository object. As long as the java instance is not destroyed, the reference to the MegaObject still exists. If a lot of these references lives at the same time, they are notified each time a modification is made in the repository. So the more you have this kind of references in memory the more slowly the process goes.

⚠ **To solve the problem, release explicitly the instances as soon as you do not need them anymore.**

This is to be done while coding. To release java instances a posteriori, see Processes going slower: releasing non released instances procedure.

To release an instance, enter:

VB Script	`Set mgObject = Nothing`
Java	`mgObj.release();`

This problem concerns MegaObjects or MegaCollections but also all Classes provided by MEGA (MegaEnvironment, MegaRoot, MegaToolkit…).

Especially in Java, beware of unsuspected instantiations.

For example, the following code:

```
String codeTemplate = mgRoot.currentEnvironment().resources().codeTempla
te("xxxxxx", "");
```

instantiates two objets: one for the currentEnvironment and one for the resources.

These two objects should be released and you should replace the above code by the following one:

```
MegaCurrentEnvironment mgCurrEnv = mgRoot.currentEnvironment();

MegaResources mgRes = mgCurrEnv.resources();

String codeTemplate = mgRes.codeTemplate("xxxxxx", "");

mgRes.release();

mgCurrEnv.release();
```

⚠ **Releasing all objects is really something important in big programs.**

To help you to find the non-released objects in your java code, HOPEX provides a class called “**com.mega.modeling.api.util.MegaDebugLivingInstances**”. It is located in mj_api.jar.

```
com.mega.modeling.api.util.MegaDebugLivingInstances
```

To use this class:

1. Indicate in your code, the moment from which you want to start monitoring the Mega Classes instances. To do that, enter:

```
MegaDebugLivingInstances.activate();
```

Since V5 CP5, all objects where Keep or Release has not been explicitly called will be reported. The behavior has been changed as the purpose of this tool is to find all objects without explicit Release.

If you need the previous behavior where implicit released objects through garbage collector are not reported, start this tool with:

```
MegaDebugLivingInstances.activate(Mode.AllRelease);
```

2. You can use three APIs:

```
MegaDebugLivingInstances.getLivingInstancesCount();
```

This function allows knowing the number of living instances. It retrieves an integer.

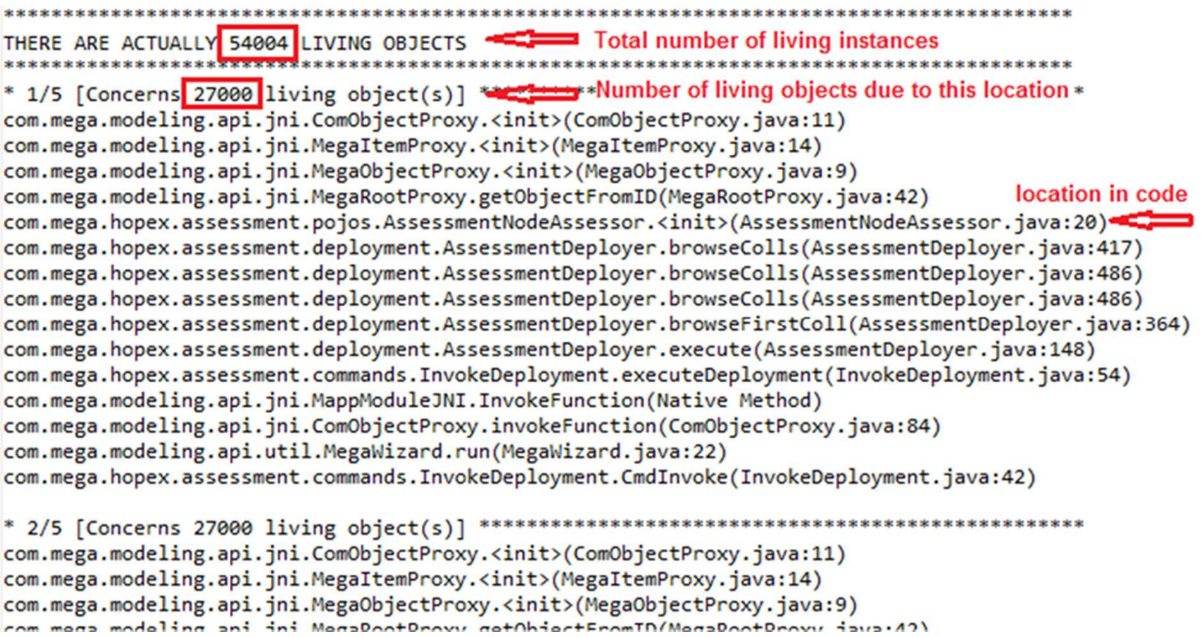
```
MegaDebugLivingInstances.getLivingInstances();
```

This function allows retrieving, as a string, the exact locations in your code and the number of instances still living due to this location.

```
MegaDebugLivingInstances.dumpLivingInstances("c:\\livingObjects.txt");
```

This function allows dumping, in a specified file, the exact locations in your code and the number of instances still living due to this location.

Dump example:



```
************************************************************************************
THERE ARE ACTUALLY 54004 LIVING OBJECTS
************************************************************************************
* 1/5 [Concerns 27000 living object(s)] ********************************************
com.mega.modeling.api.jni.ComObjectProxy.<init>(ComObjectProxy.java:11)
com.mega.modeling.api.jni.MegaItemProxy.<init>(MegaItemProxy.java:14)
com.mega.modeling.api.jni.MegaObjectProxy.<init>(MegaObjectProxy.java:9)
com.mega.modeling.api.jni.MegaRootProxy.getObjectFromID(MegaRootProxy.java:42)
com.mega.hopex.assessment.pojos.AssessmentNodeAssessor.<init>(AssessmentNodeAssessor.java:20)
com.mega.hopex.assessment.deployment.AssessmentDeployer.browseColls(AssessmentDeployer.java:417)
com.mega.hopex.assessment.deployment.AssessmentDeployer.browseColls(AssessmentDeployer.java:486)
com.mega.hopex.assessment.deployment.AssessmentDeployer.browseColls(AssessmentDeployer.java:486)
com.mega.hopex.assessment.deployment.AssessmentDeployer.browseFirstColl(AssessmentDeployer.java:364)
com.mega.hopex.assessment.deployment.AssessmentDeployer.execute(AssessmentDeployer.java:148)
com.mega.hopex.assessment.commands.InvokeDeployment.executeDeployment(InvokeDeployment.java:54)
com.mega.modeling.api.jni.MappModuleJNI.InvokeFunction(Native Method)
com.mega.modeling.api.jni.ComObjectProxy.invokeFunction(ComObjectProxy.java:84)
com.mega.modeling.api.util.MegaWizard.run(MegaWizard.java:22)
com.mega.hopex.assessment.commands.InvokeDeployment.CmdInvoke(InvokeDeployment.java:42)

* 2/5 [Concerns 27000 living object(s)] ********************************************
com.mega.modeling.api.jni.ComObjectProxy.<init>(ComObjectProxy.java:11)
com.mega.modeling.api.jni.MegaItemProxy.<init>(MegaItemProxy.java:14)
com.mega.modeling.api.jni.MegaObjectProxy.<init>(MegaObjectProxy.java:9)
com.mega.modeling.api.jni.MegaRootProxy.getObjectFromID(MegaRootProxy.java:42)
```

Total number of living instances
Number of living objects due to this location
location in code

****

Using this class can slow down the main code so consider using it only during conception phase.

Batch mode

When entering a part of code which will handle a lot of data, it is recommended to disable some notifications for better performances. To do that, use the following API:

```
mgRoot.currentEnvironment().enterBatchUpdate();
```

To enable notifications use the following API:

```
        mgRoot.currentEnvironment().leaveBatchUpdate();
```

Objects Creation

If you know the name of the occurrence you want to create, specify it as a parameter of the create API:

```
getCollection(…).create("My Name");
```

This prevents the system to allocate a temporary computed name to the object. This computation can be a cause of slow performance. This is especially useful when creating a lot of objects.

If the objects you are creating are supposed to be linked to a main object, you should not have a code like this:

```
MegaObject myObject = getCollection(…).create();

myMainObject.getCollection(…).add myObject;
```

Prefer the following code which is optimized to do a create link operation:

```
MegaObject myObject = myMainObject.getCollection(…).create();
```

This is really compulsory when dealing with concepts having namespace (most of the objects). These two technics can be composed to be more efficient.

12.2.6 Processes going slower: releasing non released instances

You can release all at once a set of Java instances of ComObjectProxy (MegaObject, MegaCollection, MegaRoot…) without needing to track all of the objects, and release each of them one by one.

To do so, use the following class, which enables to collect and release the objects concerned:

```
com.mega.modeling.api.util.MegaManageLivingInstances
```

⚠ **This is to be done a posteriori only, when founding out that java instances are not released. To release java instances while coding, see Avoiding processes to go slower: Tracking down non released instances procedure.**

To use this class:

1. In your code, indicate when you want to start collecting Mega instances: call the following method:

```
com.mega.modeling.api.util.MegaManageLivingInstances.startCollectReferences();
```

2. Call the mass release method using the following API:

```
com.mega.modeling.api.util.MegaManageLivingInstances.releaseReferences();
```

3. Indicate when you want to stop collecting Mega instances: use the following API:

```
com.mega.modeling.api.util.MegaManageLivingInstances.stopCollectReferences();
```

⚠

- **Be accurate when defining the different calls, as any object instantiated after the start will be released at the release call without any distinction.**
- **Do not forget to stop the collect.**
- **Do not use this procedure with code that calls successively wizards or callFunction/callMethod, as in Web Front-End the collection will not be performed correctly due to suspensions that implies several different java threads preventing from collecting the right reference.**

Advice:

The best way is to:

- call the start right after a loop start,
- call the stop after the loop
- call the release at the right end of an iteration:

```
startCollectReferences();
for(init;condition;incr/decr){
// code to be executed
releaseReferences();
}
stopCollectReferences();
```

Tips:

You can also use other APIs to put the collection in stand-by/continue:

```
com.mega.modeling.api.util.MegaManageLivingInstances.suspendCollectReferences();
com.mega.modeling.api.util.MegaManageLivingInstances.resumeCollectReferences();
```

These APIs enable to isolate Java objects so as not to release them. These APIs are effective only after a:

```
startCollectReferences();
```

Advantage:

The main advantage is that you do not need to modify any code: you only need to insert the three calls at the right place. So it is especially efficient with legacy code. You can even nest the `startCollectReferences/stopCollectReferences`.

Best practice:

Use this class for existing code: do not use this class while coding, but rather while detecting performance issues on existing code, so as to avoid rewriting working code.

The best way is to breakdown all of your Java objects and apply the release when needed.

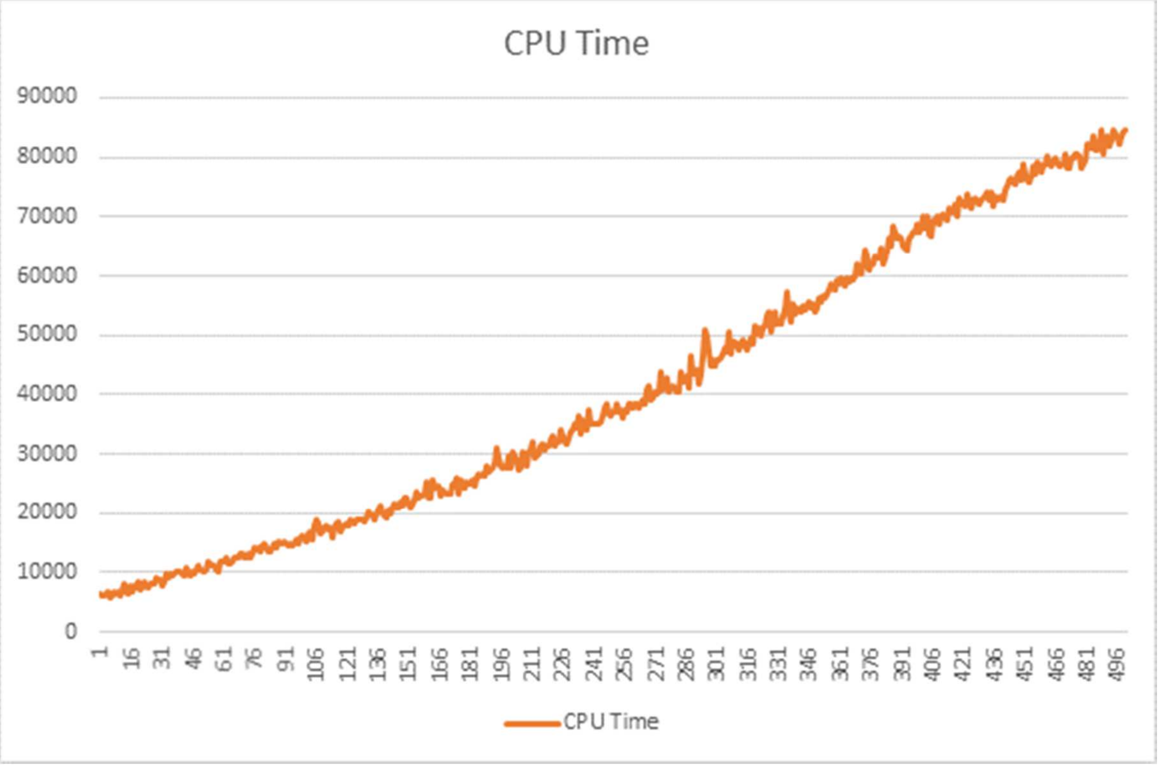
Use case 1: mass generating questionnaires in an assessment session

The time to generate 500 questionnaires:

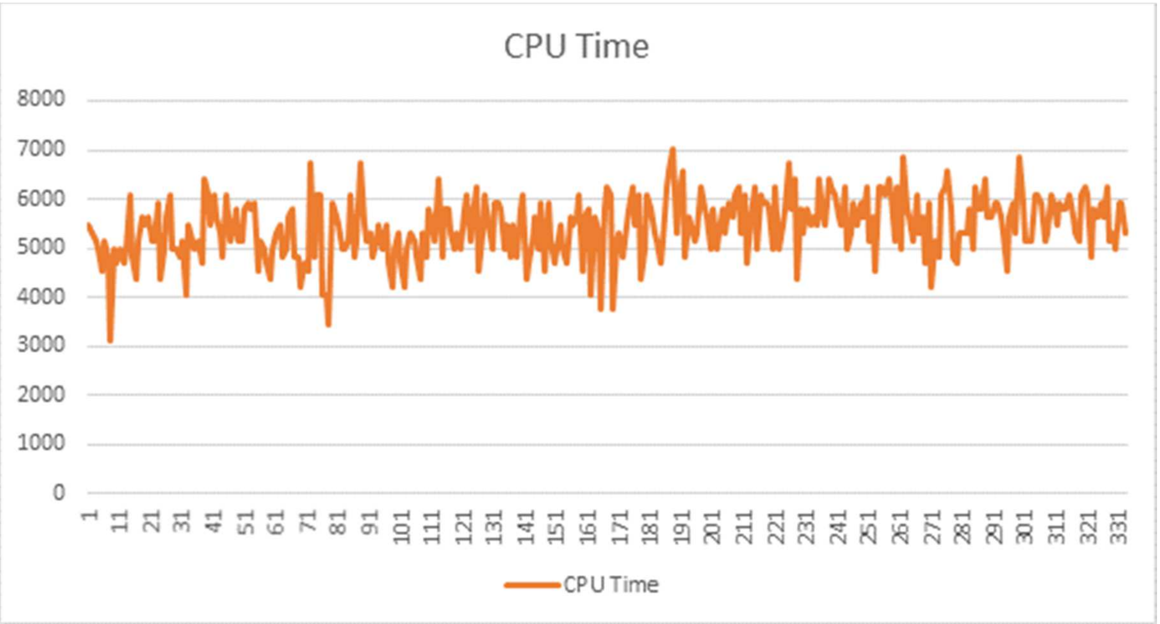
- before using this class: 45 min
- after using this class: 9 min

Moreover the generation time is stable.

Before:



After:



Use case 2: workflow action enabling mass transitions

The time to perform the mass transition of 500 questionnaires:

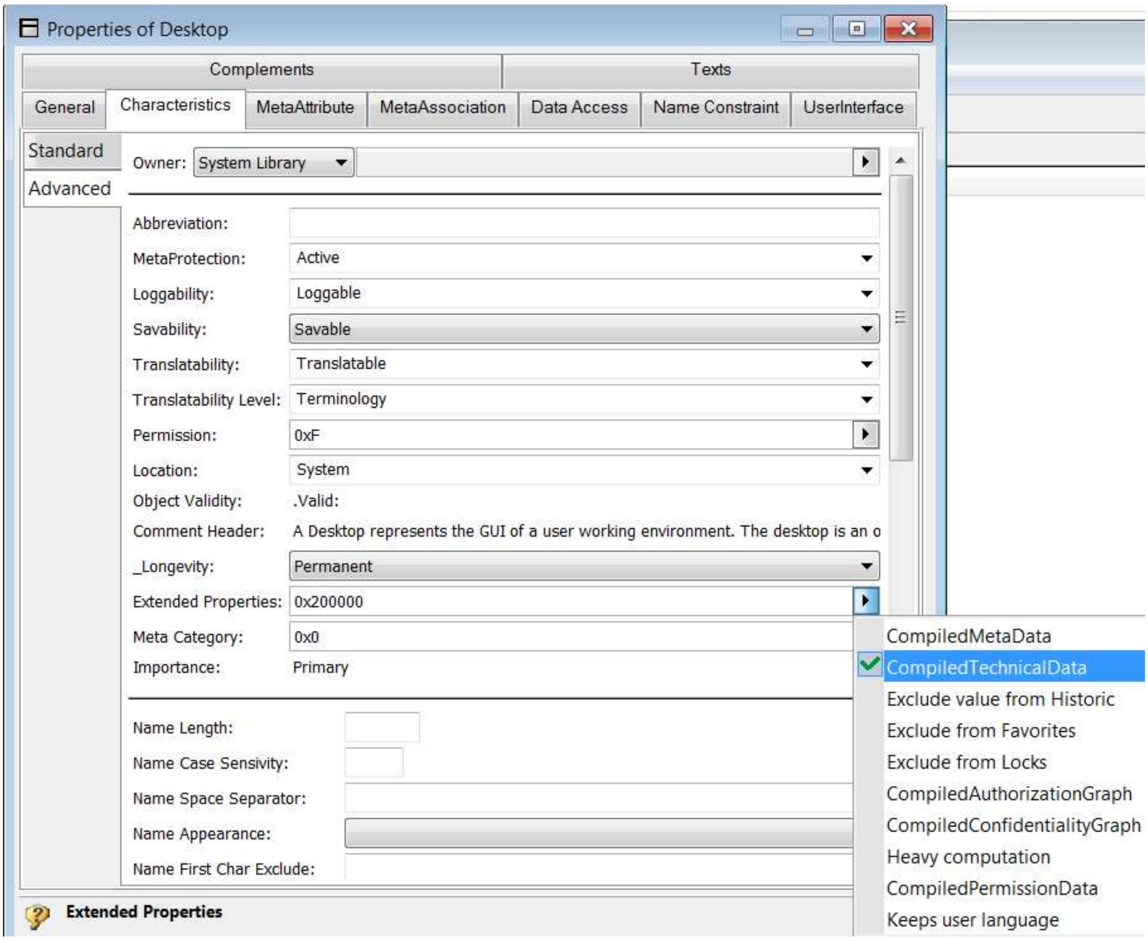
- before using this class: 2 hours
- after using this class: 17 min

Moreover the CPU is stable.

12.2.7 Navigating through the technical data with APIs

Technical data is stored in the system repository.

- Data must be stable, i.e.:
  - serialized in a file
  - loaded on demand from the file
  - when a data often changes, there is no point in tagging it as "technical data"



Owner: System Library

Abbreviation:

MetaProtection: Active

Loggability: Loggable

Savability: Savable

Translatability: Translatable

Translatability Level: Terminology

Permission: 0xF

Location: System

Object Validity: .Valid:

Comment Header: A Desktop represents the GUI of a user working environment. The desktop is an o

_Longevity: Permanent

Extended Properties: 0x200000

Meta Category: 0x0

Importance: Primary

Name Length:

Name Case Sensivity:

Name Space Separator:

Name Appearance:

Name First Char Exclude:

- To navigate through the data, use the scanner as follows:
  - Define a scanner class
  - Define public member that can be reused by the caller
  - Caller uses ScanCollection to retrieve a collection from a specific MetaAssociationEnd
  - Callee defines **onItem** procedure called for each data available in the collection

VB Script

```
Class ScannerReportTemplate
  Public mgResource
  Public mgToolkit
  Public mgData
  Public mgFunction
  Public mgName
  Public mgMetaclassidabs
  Public mgbTrouve
  Public mgIdAttribute
  Public mgFirstIdAbs
  Public  coont

  Public Sub OnItem(Context, Id)
    mgFirstIdAbs=Id
    mgbTrouve =true
    Select Case mgFunction
      Case 1
           mgData=Context.targetProperty(mgIdAttribute & ":T" )
      Case 2
           mgData = mgResource.name(Context, "~Z20000000D60[Nom
court]")
    End Select
    Context.Abort
  End Sub
End Class
Function GetShortNameRT(idDepart,mgRoot As MegaRoot)
  Dim mgScanner
  Set mgScanner = New ScannerReportTemplate
  Set mgScanner.mgResource = mgRoot.CurrentEnvironment.Resources
  Set mgScanner.mgToolkit = mgRoot.currentEnvironment().toolkit()
  mgScanner.mgFunction = 2
  dim idLink
  dim listAttributes
  idLink = "~kyHXAOuE3jN0[Rapport type parametre]"
  ` T means the information is on the target object and not on the
link
  listAttributes ="~Z20000000D60[Nom court]" & ":T"
  mgScanner.mgbTrouve = false
  mgScanner.mgResource.ScanCollection idDepart ,idLink , mgScanner ,1
, listAttributes
  if(mgScanner.mgbTrouve ) then
      GetShortNameRT= mgScanner.mgData
  else
      GetShortNameRT= null
  end if
 end function
```

Java

```
public class MyScannerEvent extends MyScanner implements
CollectionScanner {
   private List<Event> m_lEvents = Collections.synchronizedList(new
ArrayList<Event>());

   public MyScannerEvent(final MegaRoot megaRoot, final MegaResources
megaResources, final MegaToolkit megaToolkit) {

    super(megaRoot, megaResources, megaToolkit);

   }

  public void OnItem(final MegaCollectionScannerContext context,
final Object endId) {

  Event event = new Event();

         event.setMacroId(Util.getIdAbsBase64(this.getMegaToolkit(),
endId));


  event.setEvent(context.property(VocLinkDesktopLinkMacro.MA_EventTyp
e));

         synchronized (this.m_lEvents) {

         this.m_lEvents.add(event);

    }

    public List<Event> getListEvents() {

         return this.m_lEvents;

    }

    public void setListEvents(final List<Event> listEvents) {

         this.m_lEvents = listEvents;

    }

}

`Caller

MyScannerEvent myScannerEvent = new
MyScannerEvent(this.getMegaRoot(), this.getMegaResources(),
this.getMegaToolkit());

ScannerProperties spEvent = new ScannerProperties();

PropertiesForObject.addPropertiesForDesktopEvents(spEvent);

this.getMegaResources().scanCollection(objDesktopID,
VocDesktop.MAE_EventBehaviorMacro, myScannerEvent,
CollectionScanMode.synchrone, spEvent.getAllProperties());

desktop.setEvents(myScannerEvent.getListEvents());
```

12.3 Log error management

In standard, HOPEX logs all the errors in the megaerr file.

To get the stack written on the log file, you must not use try…catch in your code.

However, if you want to send back an unrecoverable error, use the MegaException class.

```
public void myMethod(final String sParam) throws MegaException

{

   if (sParam.equals("bad"))
```

```
{ throw new MegaException("Bad param", Mode.APPLICATIF);
}
```

JAVADOC

See the JavaDoc documentation:

- Reports API
- MEGA API
- Toolkit API
- Workflow Engine API