HOPEX Studio

User Guide

MEGA
a Bizzdesign company

HOPEX Aquila 6.1

1

HOPEX STUDIO INTRODUCTION

HOPEX Studio desktop is available with **HOPEX Power Studio** technical module.

Report Studio is part of **HOPEX Studio**.

Report DataSet Definition, **GraphSet Definition**, **TreeSet Definition,** and **Report Template** creation are performed in both **HOPEX (Windows Front-End)** and **HOPEX (Web Front-End)** for users with **HOPEX Customizer** or **HOPEX Customizer Publisher** profile.

Creation and use of Report DataSets, GraphSets, TreeSets, and reports are available for all users in **HOPEX Solutions**.

PREREQUISITE FOR ANY CUSTOMIZATION

Customizations must be performed in a development environment.

Before starting any customization, you must first install the **HOPEX Application Server customization** module.

☛ *For detailed information, see* ***MODULES > Managing the Customization Lifecycle*** *documentation.*

HOPEX STUDIO VS JAVA REPORT

HOPEX Studio enables to quickly generate simple reports but does not provide as many possibilities as the reports written in Java do. You first need to identify which of the two techniques best suits your needs.

Java Reports provide more extensive possibilities but require development skills, as it is required to implement a macro in Java.

☛ *See Writing Java Report Chapters Technical Article.*

ACCESSING HOPEX STUDIO DESKTOP

Customization Dedicated Profiles

You can access **HOPEX Studio** desktop with the following profiles:

- **HOPEX Customizer**
  HOPEX Customizer profile gives access to all HOPEX Studio features.
  It enables to customize:
  - Metamodel
  - User Interfaces
  - Documents
- **HOPEX Customizer Publisher**
  HOPEX Customizer Publisher profile is dedicated to document customization, so that you should use this profile when you work in **Report Studio** only.

For example, in **Report Studio**, when creating Definitions (Report DataSet Definition, TreeSet Definition, GraphSet Definition) you might need to create your own queries:

- **HOPEX Customizer - Publisher** profile prevents you from getting, in the IntelliSense feature, all administration attributes and links, which are not of interest in that case. Only business related attributes are proposed and thus you do not get a crowded list.
- with **HOPEX Customizer** profile, you can still hide administration attributes and links: in **Options > Repository > Metamodel**, clear **Display repository administration properties and links** option.

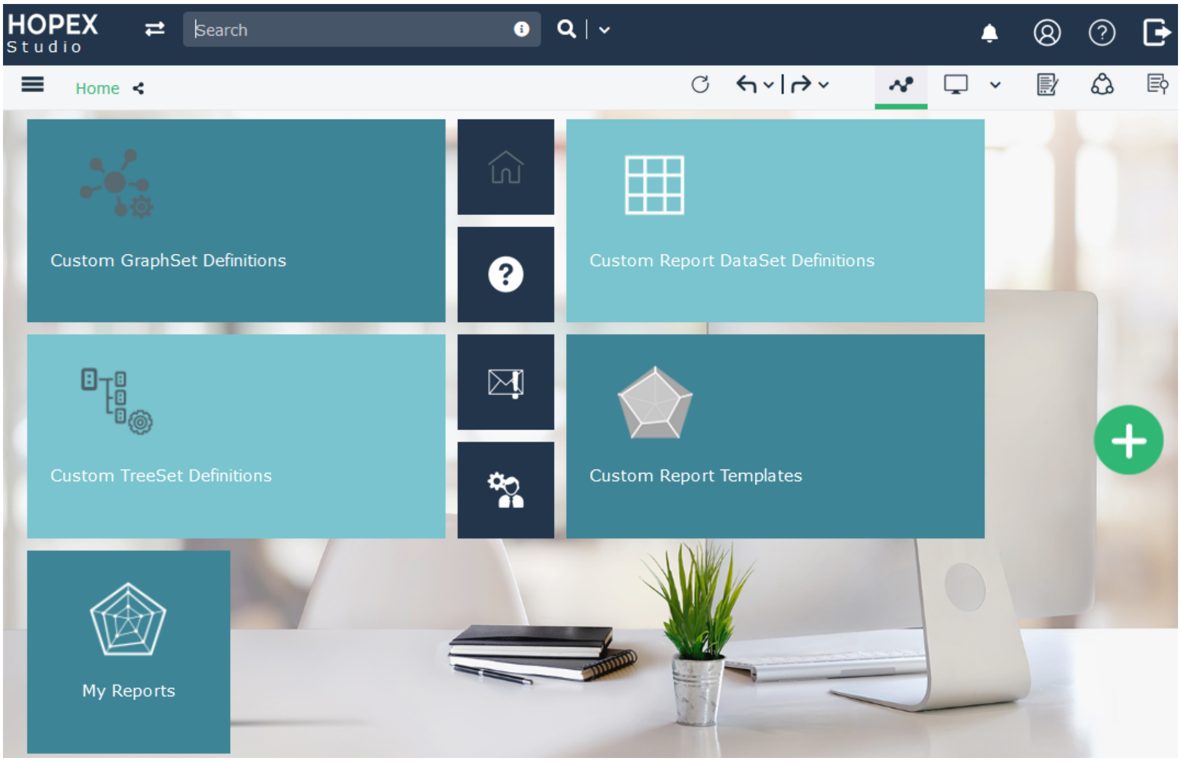
☛ *See Advanced Search Options.*

Connecting to HOPEX Studio Desktop

You can access **HOPEX Studio** desktop with **HOPEX Customizer** or **HOPEX Customizer Publisher** profile.

To connect to **HOPEX Studio** desktop:

- See Accessing HOPEX and connect with either **HOPEX Customizer** or **HOPEX Customizer Publisher** profile.
  Once connected the **HOPEX Studio** desktop appears and a session is opened.

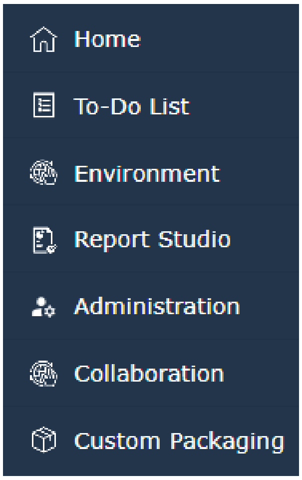


HOPEX STUDIO DESKTOP

The **HOPEX Studio** desktop includes the standard menus and tools of HOPEX desktops.

☛ *For information regarding HOPEX desktops, see HOPEX Desktop.*

HOPEX Studio Navigation Menus



The **HOPEX Studio** desktop includes the following navigation menus:

- ***Standard navigation menus:***
  - **To-Do-List**
  - **Environment**
  - **Collaboration**

  ☛ *For information regarding use of standard navigation menus, see Common features > The HOPEX Web Front-End desktop.*
- ***Reporting navigation menus:***
  - **Home**, for a quick access to standard tiles as well as to **My reports**, **Custom Report Templates**, **Custom Report DataSet Definitions**, **Custom GraphSet Definitions**, and **Custom TreeSet Definitions** tiles, which give access to reports, Report Templates, Report DataSet

Definitions, GraphSet Definitions, TreeSet Definition not provided by **MEGA**.

☛ *It can be useful to add the **History** generic tile to your Home page.*

- **Report Studio**: to create **Report DataSet Definitions**, **Report Template Definitions**, **GraphSet Definitions**, **TreeSet Definitions** and to also create **Reports** from these definitions.

  ☛ *For information regarding report creation, management and customization, see Generating Documentation.*

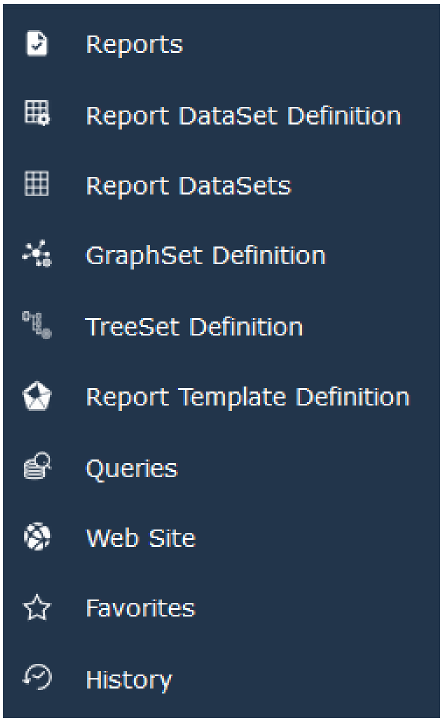
- ***Administration and customization navigation menus:***
  - **Administration**: to manage users (persons, person groups), their profiles and business roles. It also gives access to the scheduling information page.

    ☛ *For information regarding Administration management see HOPEX Administration - Supervisor Web > Managing users.*

  - **Custom Packaging**: to manage customizations

    ☛ *For information regarding customizations, see Modules > HOPEX Application Server Customization.*

Report Studio Navigation Menus



From the **Report Studio** navigation menu you can especially access, create and customize the Report Studio related definitions:

- **Report DataSet Definitions**

  ☛ *See Report DataSet Definition.*

- **GraphSet Definitions**

  ☛ *See GraphSet Definition.*

- **TreeSet Definition Definitions**

  ☛ *See TreeSet Definition.*

- **Report Template Definitions**

  ☛ *See Report Template Definition.*

From these definitions you can create and customize:

- Reports
- Report DataSets
- GraphSets
- TreeSets

☛ *For information regarding report, Report DataSet, GraphSet, and TreeSet creation, management and customization, see Common features > Generating Documentation.*

The **Report Studio** navigation menu includes also access to:

Reports
To access and create reports from Report Templates

Report DataSets
To access and create Report DataSets from Report DataSet Definitions

Queries
To generate gauge reports and create gauge Report templates

Web Site
To generate Web sites

Favorites
For a quick access to your favorites and shared favorites

History
For a quick access to all of the objects you have accessed

2

REPORT DATASET DEFINITION

The following points are covered here:

- ✓ Introduction to Report DataSet Definition
- ✓ Report DataSet Definition Creation
- ✓ Customizing the Report DataSet
- ✓ How To
- ✓ Use Cases
- ✓ Report DataSet Creation

INTRODUCTION TO REPORT DATASET DEFINITION

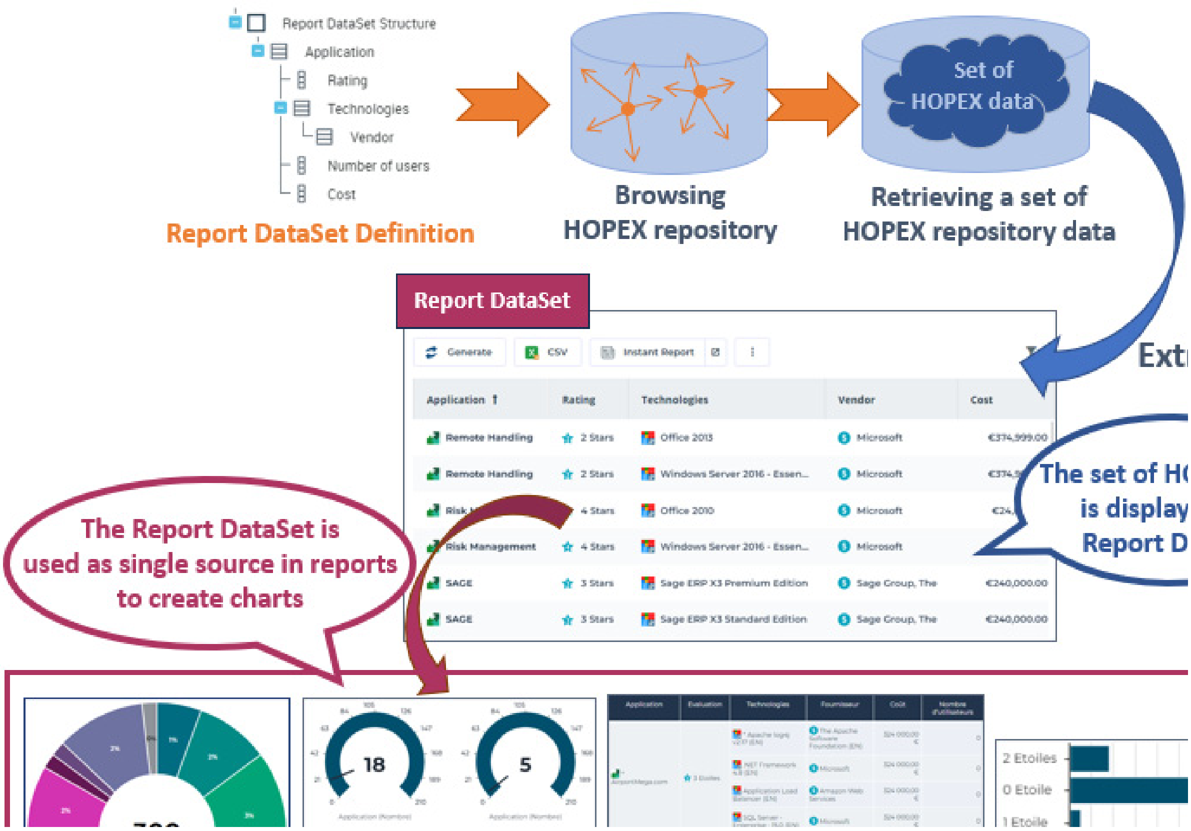
Report DataSet Definition and Report DataSet Principles

☛ ***Report DataSet Definition** is only available with **HOPEX Power Studio** technical module.*

A **Report DataSet** enables to extract HOPEX raw data and show it in tabular form. With this data the end-user can create reports that use tabular data (e.g.: pie-chart, set of gauges, table, bar-chart, matrix) without needed to enter any code.

Report DataSet Definition creation is performed in both **HOPEX** Windows Front-End and Web Front-End for users with **HOPEX Customizer** or **HOPEX Customizer Publisher** profile.

Once a **Report DataSet Definition** is created, any user of a Solution can use it to query **HOPEX** repository and create a **Report DataSet**. Data first shown in tabular form can then be handled and shown in graphical format using **Instant Report** feature.



Report DataSet Definition

The **Report DataSet Definition** defines how to build a tabular set of data (**Report DataSet**). The **Report DataSet Definition** can be used as a Report DataSource in a Report Template.

A **Report DataSet Definition** is MetaClass-specific.

The **Report DataSet Definition** is based either on:

- a collection parameter (an occurrence or a set of occurrences of the root MetaClass)
  At creation you need to define the root MetaClass (concrete or abstract)
- a query (a set of occurrences)
  At creation you need to define the root MetaClass (concrete or abstract) and the query that collects the entry points.

  If the query includes parameters, these parameters will result in Property Parameters in the definition.

Report DataSet

The **Report DataSet** is a set of data defined by a **Report DataSet Definition**. The **Report DataSet** is used as a tabular source of data in a report.

Report DataSet Structure

The Report DataSet **Structure** includes a root Report DataSet **Collection** to which you can add as many items as needed:

- Report DataSet **Properties**
- Report DataSet **Collections**

Example:

The **structure of the** "Application Technologies" **Report DataSet Definition** includes the "Application" **root Report DataSet Collection** with the following added items:

- **Report DataSet Properties**:

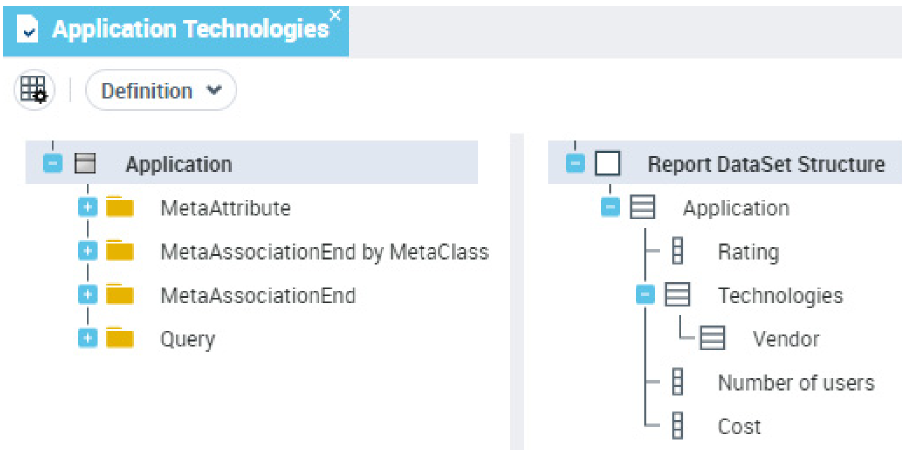
"Rating", "Number of users", and "Cost" MetaAttributes belonging to the Application MetaClass.

- **Report DataSet Collections**:

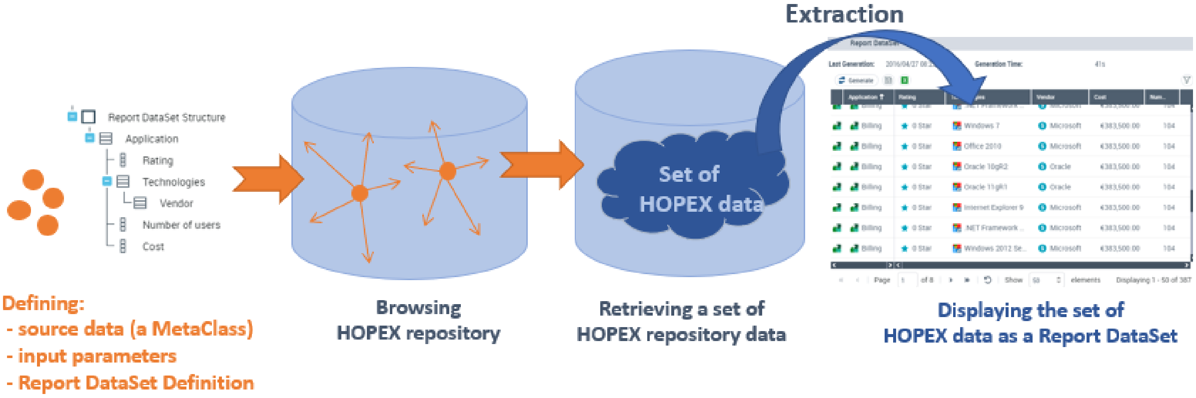
"Technology used", which is a MetaAssociationEnd of the Application MetaClass

"Vendor", which is a MetaAssociationEnd of the "Technology used" MetaClass.

☛ *See Defining the Data that Feeds the Report DataSet.*



Creating and Defining a Report DataSet Definition: the Big Picture



To create and define a **Report DataSet Definition**:

1. Define the **Report DataSet** source data: a MetaClass.
   This MetaClass is the root of the tree that represents the set of data collected from the repository.
   ☛ *See Creating a Report DataSet Definition.*
2. Define which data type feeds the data collection:
   Define the data you want to be displayed in the **Report DataSet**.
   ☛ *See Defining the Data that Feeds the Report DataSet.*
3. (if needed) Define input parameters:
   - **Property parameters**, which can be used in queries to populate a Report DataSet collection.
     ☛ *See Adding Property parameters.*
   - **Collection parameters**, which can be used directly to populate a Report DataSet collection.
     ☛ *See Adding Collection parameters.*

   The user is asked to enter these parameters at **Report DataSet** creation.
4. Customize the **Report DataSet**.
   ☛ *See Customizing the Report DataSet.*
5. Preview the **Report DataSet**.
   ☛ *See Previewing the Report DataSet.*

The **Report DataSet Definition** is available for any user to create a **Report DataSet**.

☛ *See* Creating a Report DataSet.

For examples of **Report DataSet Definitions** see:

- Collecting a Set of Objects, Linked Objects and some of their Properties
- Collecting a Set of Objects Using a Query and Adding a Computed Property

Report DataSet Definition Best Practices

When you create a **Report DataSet Definition** you should follow the best practices.

Defining the Report DataSet Definition

Define a **Report DataSet Definition** to create focused **Report DataSets**, i.e.:

- The Report DataSet should answer a single issue.

  `For examples: a report, an export of specific data.`

- Do not build Report DataSets with huge amount of data that you would use for different purposes.

Optimizing the Report DataSet Definition for better performance

Retrieving the data

When you create a **Report DataSet Definition**, use the most efficient way to retrieve data.

☛ *See Defining the Data that Feeds the Report DataSet.*

Time monitoring

To identify the column that may introduce performance issues, use the debug log to find out how long it takes to get each column.

To get the time monitoring tool:

1. Access the megasite.ini file (in the **HAS console**: **Modules > Module Settings > MegasiteSettings**).
2. Add:

```
[Debug]

ReportDataSetGenerationTimeMonitor=1
```

Keep the last generated Report DataSet

The **Keep last generated result** option is selected by default, so that at next session, the first **Report DataSet** display time is shorten.

☛ *See Speeding up the Report DataSet Display.*

Checking the Report DataSet row count

The **Report DataSet** size is limited to:

- 100 000 rows
- 50 columns

☛ *These limits are defined in the following **Options** (**Options > Tools > Documentation > Reports**):*

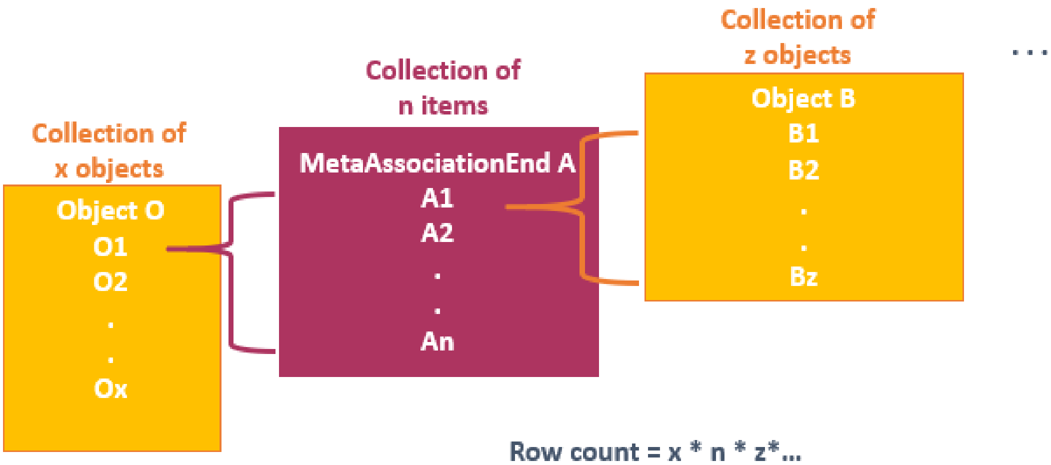
Fix the limit for Report DataSet rows

Fix the limit for Report DataSet columns

As this maximum size can be reached quickly, check that the **Report DataSet** includes all the expected rows, so that it does not miss any data.

☛ *For information regarding Report DataSet related supervision events (e.g.: "Report DataSet Generate" informative event), see the HOPEX Administration > Managing Events > Events to be monitored (Production server).*

Rows are produced by multiplication of the object count of each MetaAssociationEnd.



Advice:

If your **Report DataSet Definition** does not include more than four levels of MetaAssociationEnds or queries the **Report DataSet** should include all the data.

☛ *See Defining the Data that Feeds the Report DataSet.*

Supervising Report DataSet events

Alerts are generated when a Report DataSet exceeds the limited size.

In the HOPEX Supervision console, you can check the Report DataSet events:

☛ *In HOPEX Administration guide, see Events: Report DataSet/ GraphSet/TreeSet Generation section.*

- **Lines count** indicates the number of rows (hidden or not) of the Report DataSet
- **Columns count** indicates the number of Report DataSet items on the Report DataSet collector.

For detailed information regarding Report DataSet related supervision events, see **HOPEX Administration > Technical Articles > Supervision Event Description > ReportDataSet, TreeSet, and GraphSet**.

REPORT DATASET DEFINITION CREATION

The **Report DataSet Definition** creation includes:

- Accessing the Report DataSet Definitions
- Creating a Report DataSet Definition
- Adding Parameters to Filter Data Extraction
- Defining the Data that Feeds the Report DataSet
- Previewing the Report DataSet

☛ *To customize the **Report DataSet Definition**, see Customizing the Report DataSet.*

Accessing the Report DataSet Definitions

You can access :

- all the Report DataSet Definitions
- the custom Report DataSet Definitions only
  Custom Report DataSet Definitions are the Report DataSet Definitions not provided by **MEGA**.

Accessing the Report DataSet Definitions in HOPEX (Web Front-End)

To access the Report DataSet Definitions:

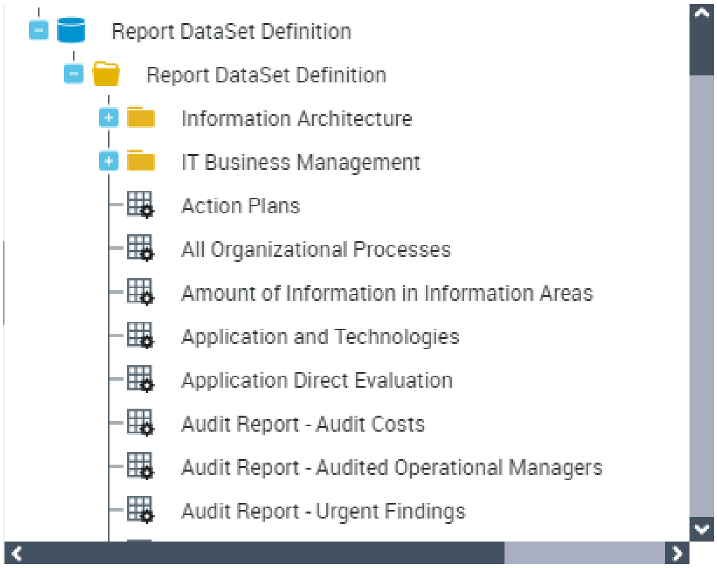
1. Connect to **HOPEX Studio** desktop.

   ☛ *See Connecting to HOPEX Studio Desktop.*

2. Click the navigation menu and select **Report Studio > Report DataSet Definition** .

   In the Edit area, the **Browse** view displays the **Report DataSet Definition** tree.

3. Expand the **Report DataSet Definition** folder.
All the Report DataSet Definitions available are accessible.



☛ *Click the Report DataSet Definition, to automatically display its properties in the **Multi-view** window.*

Accessing the custom Report DataSet Definitions (Web Front-End)

Custom Report DataSet Definitions are the Report DataSet Definitions not provided by **MEGA**.

☛ *Note that when you customize a Report DataSet Definition provided by **MEGA**, it becomes a Custom Report DataSet Definition, see* *How to Modify a Report DataSet Definition Provided by MEGA*

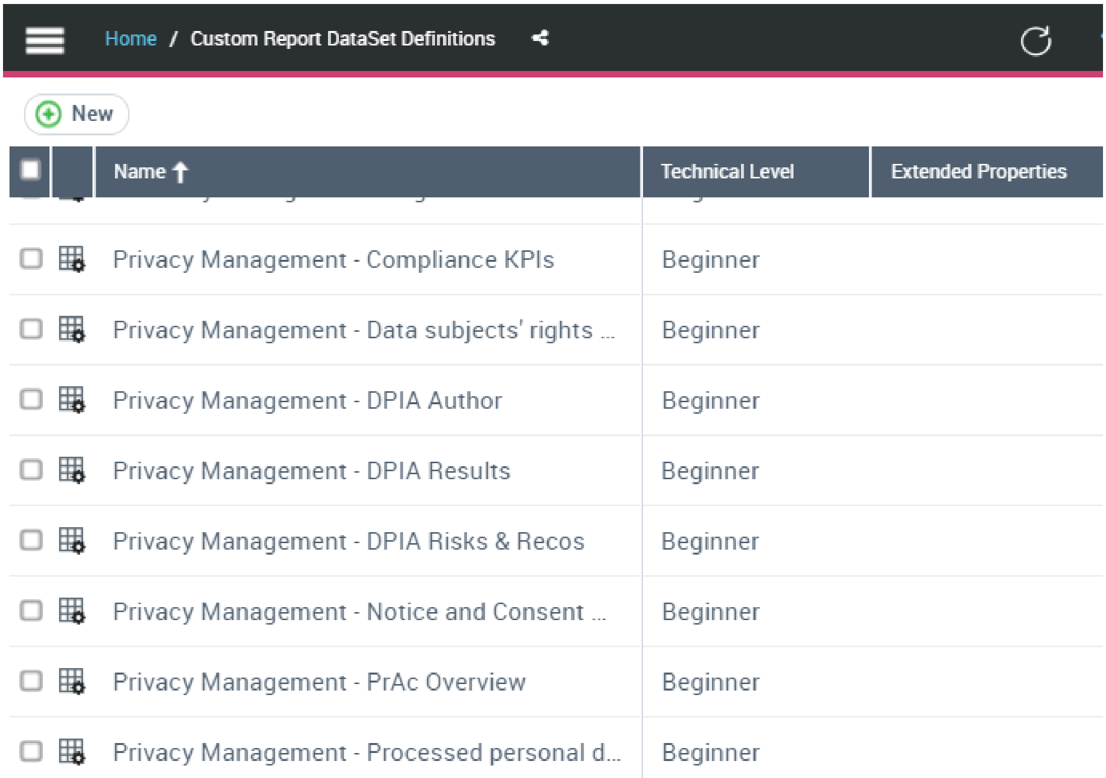
To access the custom Report DataSet Definitions:

1. Connect to **HOPEX Studio** desktop.

☛ *See Connecting to HOPEX Studio Desktop.*

2. In the **Home** page, click the **Custom Report DataSet Definitions** tile.

   In the Edit area, the **Browse** view lists the custom Report DataSet Definitions only.



Accessing the Report DataSet Definitions in HOPEX (Windows Front-End)

To access the Report DataSet Definitions:

1. Connect to **HOPEX** (Windows Front-End) with **HOPEX Customizer** or **HOPEX Customizer Publisher** profile.
2. In **HOPEX** menu bar, select **View > Navigation Windows > Utilities**.
3. In the repository tree, expand **Report DataSet Definition** folder.
   All the Report DataSet Definitions available are accessible.

   ☛ *Right-click Report DataSet Definition and select properties to access the Report DataSet Definition properties.*

Creating a Report DataSet Definition

You can create a Report DataSet Definition populated by:

- occurrences
- a query

☛ *See Report DataSet Definition.*

Defining a Report DataSet populated by occurrences

☛ *For a detailed use case, see Collecting a Set of Objects, Linked Objects and some of their Properties.*

To define a Report DataSet populated by occurrences:

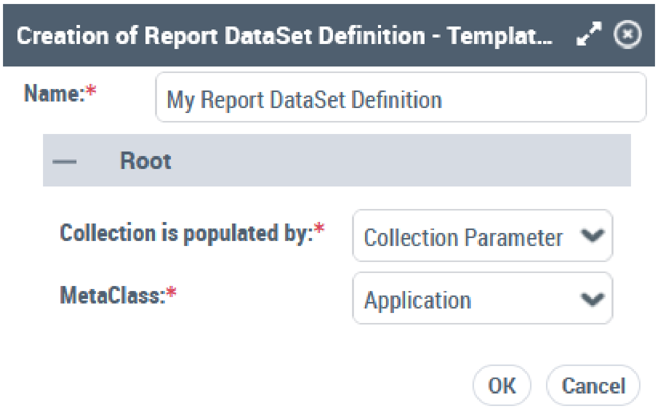
1. Access the **Report DataSet Definition** folder

   ☛ *See Accessing the Report DataSet Definitions.*
2. Right-click the **Report DataSet Definition** folder and select **New > Report DataSet Definition**.
   The **Creation of Report DataSet Definition** window appears
3. In the **Name** field, enter your Report DataSet Definition name.

   ☛ *By default, the Report DataSet Definition name is: Report DataSet Definition-x (x is a number).*
4. Define the **Root** section:
   - in the **Collection is populated by** field keep "Collection Parameter".
   - In the **MetaClass** field, click the arrow and in the drop-down menu select the MetaClass you want to define as the root MetaClass of your Report DataSet Definition.

     ☺ *In the field, enter the first letters of the MetaClass for a quick access.*

   `Example: Application.`



5. Click **OK**.
   Your Report DataSet Definition is created and added to the **Report DataSet Definition** folder.

6. In your Report DataSet Definition properties, display its **Characteristics** page.
It shows:
   - the **Root MetaClass**

     `Example: Application`
   - the **Collection Parameter** characteristics.

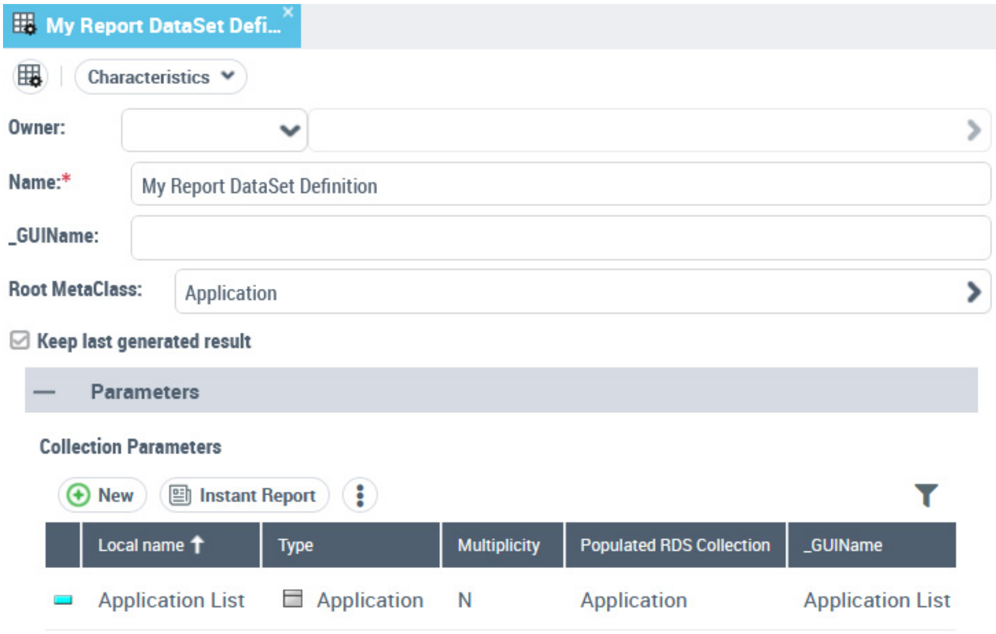
     `For example, with "Application" as root MetaClass, the Collection parameter characteristics are:`

     `- **Type**: "Application" MetaClass`

     `- **Multiplicity**: "N", as several applications need to be collected.`

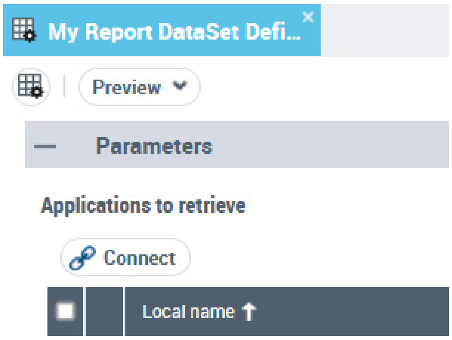
     `- **Populated RDS Collection**: "Application", i.e. this Collection parameter populates the root node of "My Report DataSet Definition" Report DataSet Definition.`

     `- **_GUIName**: "Application List", this parameter is displayed in the Report DataSet Definition Preview page.`
   - the **Keep last generated result** option: selected by default so that at next session, the first `Report DataSet` display time is shorten.



7. (If needed) In the **Collection Parameters** table, **_GUIName** field, modify the GUI name for the Collection Parameter.
   This parameter is displayed in the Report DataSet Definition **Preview** page.

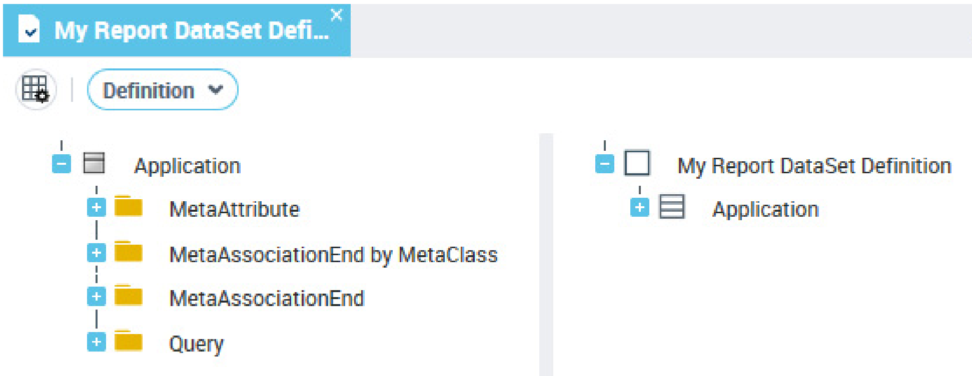
   `E.g.: "Applications to retrieve”`



   ☛ *See Previewing the Report DataSet.*

8. In your Report DataSet Definition properties, display its **Definition** page.
   The right pane displays the root Collection of the Report DataSet Definition structure.

   `Example: "My Report DataSet Definition" Report DataSet Structure includes the "Application" Report DataSet root Collection.`



9. Define your Report DataSet Definition structure.
   ☛ *See Defining the Data that Feeds the Report DataSet*

10. (Optional) You can add parameters to filter the data extraction.
    ☛ *See Adding Parameters to Filter Data Extraction.*

11. Display the Report DataSet Definition **Preview** page.
    ☛ *See Previewing the Report DataSet.*

Defining a Report DataSet populated by a query

☛ *For a detailed use case, see Collecting a Set of Objects Using a Query and Adding a Computed Property.*

To define a Report DataSet populated by a query:

1. Access the **Report DataSet Definition** folder

   ☛ *See Accessing the Report DataSet Definitions.*

2. Right-click the **Report DataSet Definition** folder and select **New > Report DataSet Definition**.
   The **Creation of Report DataSet Definition** window appears
3. In the **Name** field, enter your Report DataSet Definition name.

   ☛ *By default the Report DataSet Definition name is: Report DataSet Definition-x (x is a number).*

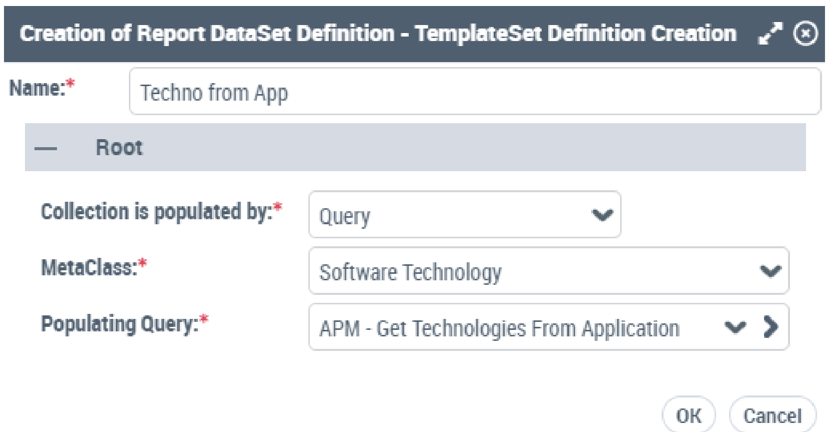
4. Define the **Root** section:
   - in the **Collection is populated by** field select "Query".
   - In the **MetaClass** field, click the arrow and in the drop-down menu select the MetaClass you want to define as the root MetaClass of your Report DataSet Definition.

     ☺ *In the field, enter the first letters of the MetaClass for a quick access.*

     `E.g: Software Technology.`
   - In the **Populating Query** field, click the arrow and in the drop-down menu select the query used to retrieve the input data.

     ☛ *The queries listed are related to the MetaClass chosen as root MetaClass.*

     `E.g: "APM-Get Technologies from Applications".`



5. Click **OK**.
   Your Report DataSet Definition is created and added to the **Report DataSet Definition** folder.

6. In your Report DataSet Definition properties, display its **Characteristics** page.
It shows:

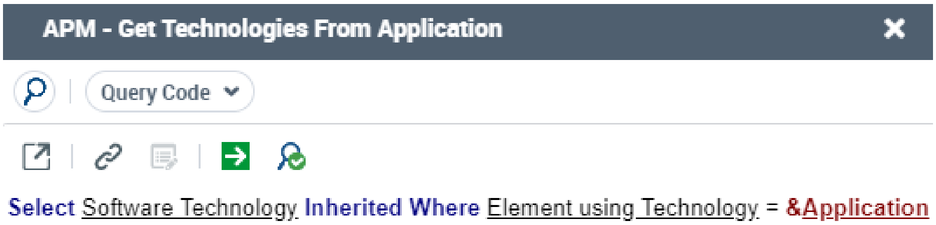
- the **Root MetaClass**

  Example: Software Technology

- the **Root Populating Query** selected to retrieve the objects:

  - **Name**: "APM-Get Technologies From Application" query retrieves the Software Technologies used in an application

  - **Query Parameter Name**: "Application"



- the **Property Parameter** (automatically created).

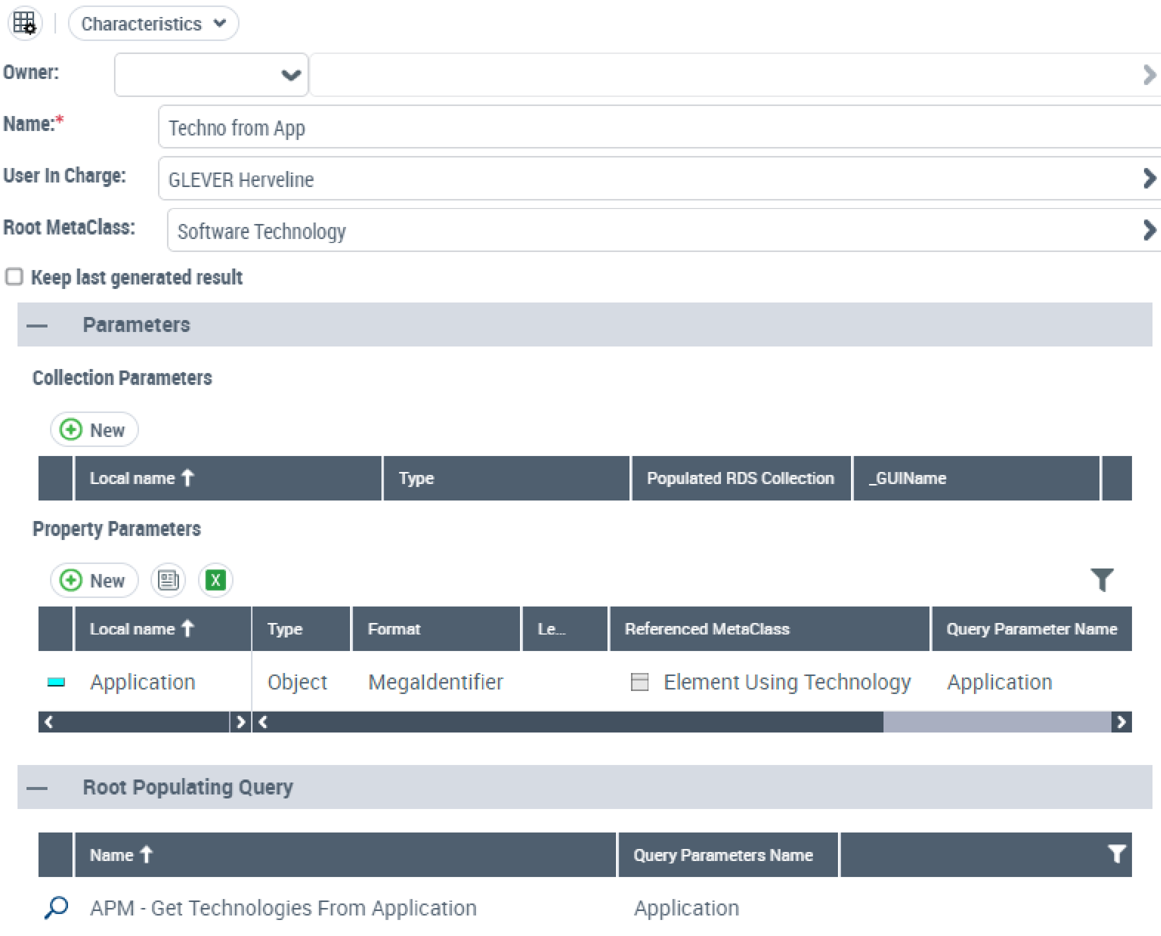
  For example, the "APM-Get Technologies From Application" query uses the "Application" **Query Parameter** to retrieve the Software Technologies:

  - **Name**: "Application"

  - **Type**: "Object" MetaClass

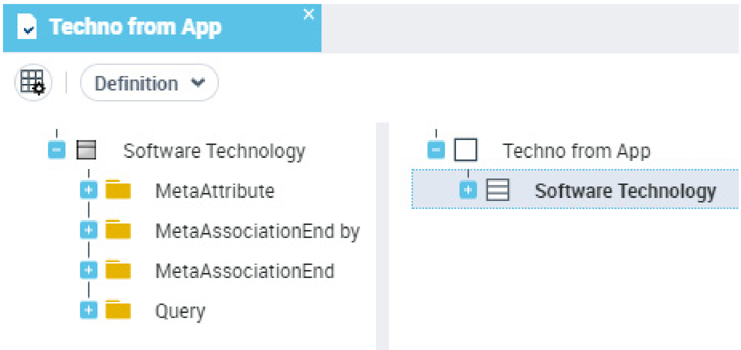
  - **Referenced MetaClass**: "Element Using Technology"

  - **Query Parameter Name**: "Application", this parameter is the exact name used in the query.


Characteristics
Owner:
Name:*
Techno from App
User In Charge:
GLEVER Herveline
Root MetaClass:
Software Technology
Keep last generated result
Parameters
Collection Parameters
New
Local name
Type
Populated RDS Collection
_GUIName
Property Parameters
New
Local name
Type
Format
Le...
Referenced MetaClass
Query Parameter Name
Application
Object
MegaIdentifier
Element Using Technology
Application
Root Populating Query
Name
Query Parameters Name
APM - Get Technologies From Application
Application

7. In your Report DataSet Definition properties, display its **Definition** page.
   The right pane displays the root collection of the Report DataSet Definition tree structure.

   ```
   Example: "Techno from App" Report DataSet Structure ☐ is
   created with its "Software technology" Report DataSet root
   Collection ☰.
   ```



8. Define your Report DataSet Definition structure.
   ☛ See *Defining the Data that Feeds the Report DataSet*
9. (Optional) You can add parameters to filter the data extraction.
   ☛ See *Adding Parameters to Filter Data Extraction.*
10. Display the Report DataSet Definition **Preview** page.
   ☛ See *Previewing the Report DataSet.*

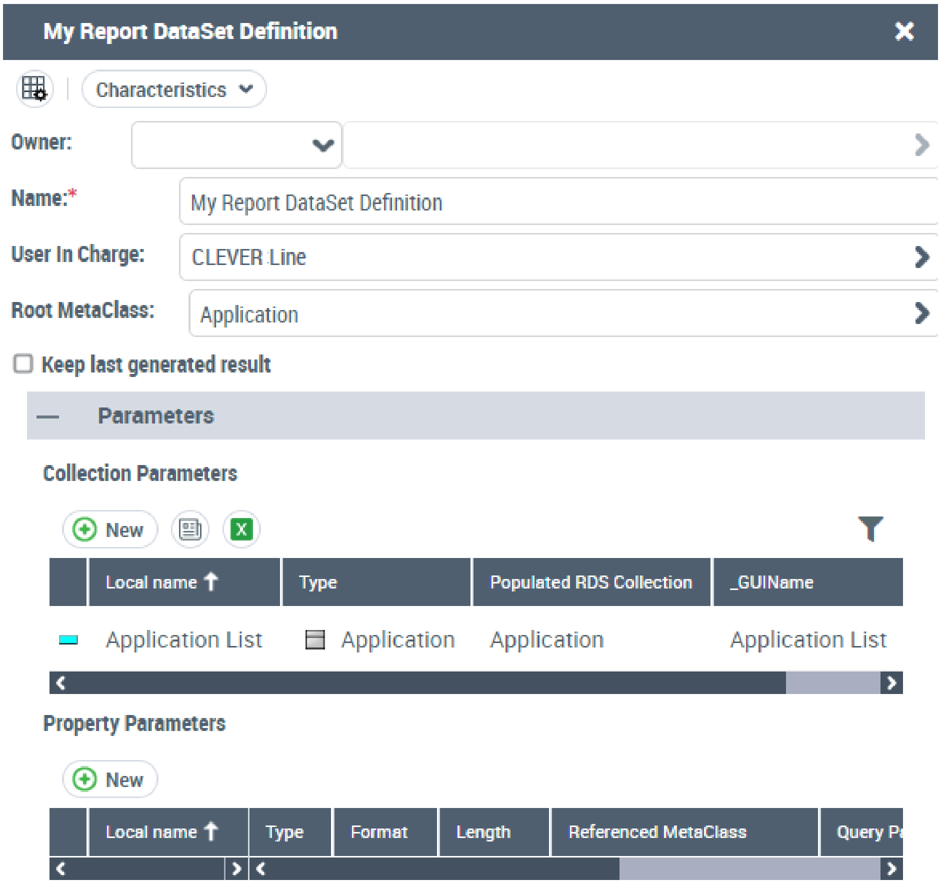
Adding Parameters to Filter Data Extraction

In the Report DataSet Definition you can define the parameters that are used to filter data at Report DataSet data collection.

If needed, you can add either **Property parameters** or **Collection parameters** or both, see:

- Adding Property parameters
- Adding Collection parameters

Adding parameters in the Report DataSet Definition properties (**Characteristics**), is useful for Report DataSet Definition re-usability.



Adding Property parameters

You can limit the collection of data set rows extracted from the Root MetaClass collection. For this purpose, you can use **Property parameters** in your Report DataSet Definition.

At Report DataSet creation the user is asked to enter the Property parameter values, which are taken into account (Property Parameters can be used in queries) to build the Report DataSet.

If no property parameters is specified the Report DataSet operates on the entire repository occurrences.

☛ *See also Adding a computed-type Property to the Report DataSet structure.*

To add Property parameter in the Report DataSet Definition:

1. Access the Report DataSet Definition properties.

   ☛ *See Accessing the Report DataSet Definitions.*
2. Select **Characteristics**.
3. In the **Parameters** section, **Property Parameter** pane, click **New** + .

   ☛ *You can add as many parameters as needed.*
4. In the **Property Parameter** pane:
   - enter your Property Parameter **Local Name** (value shown in the Report DataSet)

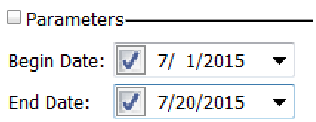
     `Example: Begin date`
   - (if needed) modify the default values for the MetaAttribute **type**, **Format**, and **Length**.

     `Example: MetaAttribute` **`Format`**`: "DateTime".`
   - (if needed) in the **Referenced MetaClass**, connect a MetaClass
   - enter the **Query Parameter Name** (value used in the query you add in the **Definition** tab, see Defining the Data that Feeds the Report DataSet)

     `Example: BeginDate`

Example:

```
When you add "Begin Date" and "End Date" Property
Parameters, at Report DataSet creation the user is asked to
enter both dates, which filter the Report DataSet results.
```



Adding Collection parameters

You can limit the collection of data set rows extracted from the Root MetaClass collection. For this purpose, you can use **Collection Parameters** in your Report DataSet Definition.

At Report DataSet creation the user is asked to enter the Collection parameter values, which are taken into account to build the Report DataSet.

By default once you selected the Root MetaClass, a Collection parameter is added

in the **Collection Parameters** list. When defined, this Collection parameter is used to fill the root collection for data extraction.

☛ *See Collecting a Set of Objects, Linked Objects and some of their Properties.*

To feed the Report DataSet Definition input collection according to a collection:

1. Access the Report DataSet Definition properties.

   ☛ *See Accessing the Report DataSet Definitions.*
2. Select **Characteristics**.

3. In the **Parameters** section, **Collection Parameters** pane, click **New** .

   ☛ *You can add as many parameters as needed.*

4. From the **Collection Parameter** pane, in the **Local Name** field, enter the name of the collection parameter.
5. In the collection **Type** field, click the arrow and select **Connect MetaClass**.
6. Click **Find** .
   The list of MetaClasses is displayed.
7. Select the MetaClass and click **Connect**.
8. In the **_GUIName** field, enter a name for your Collection parameter.
   This name is displayed in the Report DataSet Definition **Preview** page, **Parameters** section.
   The Collection parameters are defined as your Report DataSet Definition input collections.

Example:

You can add the "Applications to extract" Collection Parameter of "Application" collection **Type**, so that at Report DataSet creation the user is asked to select the applications to feed the Report DataSet with this collection.

Defining the Data that Feeds the Report DataSet

The Report DataSet **Structure** includes a root Report DataSet **Collection** to which you can add as many items as needed:

- Report DataSet **Properties**
- Report DataSet **Collections**

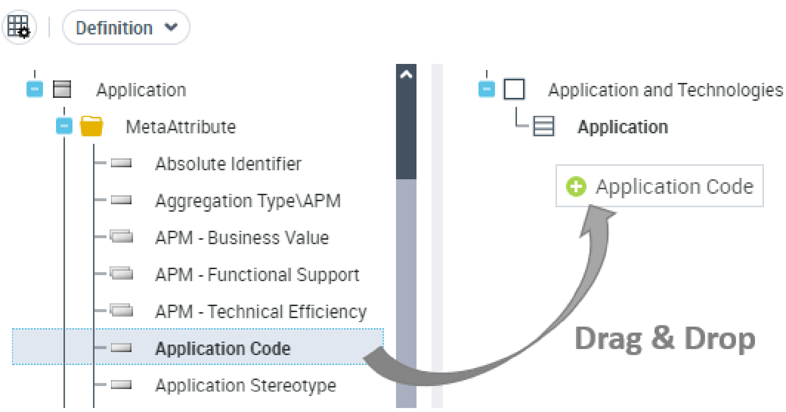
  ☛ *See Report DataSet Structure.*

Two ways to add items to the Report DataSet structure

To add an item to the Report DataSet structure, you can:

- drag and drop it from the left pane to the right pane

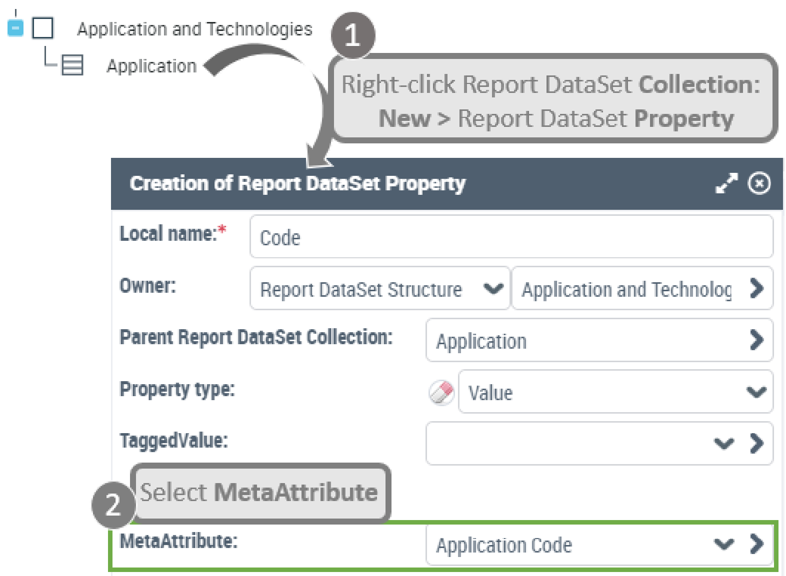
  `E.g: from the` **`MetaAttribute`** `folder of the` **`Application`** `root MetaClass, drag and drop "Application Code" to the right pane`



- add it from a Report DataSet Collection of the structure

  ☛ *Might be an easier way when you do not know in details the metamodel used.*

  `E.g: from the` **`Application`** `Report DataSet Collection, add the "Application Code" MetaAttribute`



Whatever the chosen way, the item is added to the structure under its parent Report DataSet Collection.

> E.g.: the "Application Code" Report DataSet **Property** is added to the Report DataSet **Structure** under the "Application" Report DataSet **Collection**.

How to proceed: the big picture

To define the data that feeds the Report DataSet you can add:

- Report DataSet **Collections**:
  - by drag and drop
    - ☛ *See Adding a Report DataSet Collection/Property by drag and drop.*
  - directly from a Report DataSet Collection of the structure
    - ☛ *See Adding a Report DataSet Collection directly from a Report DataSet Collection.*
- Report DataSet **Properties**:
  - value-type Properties
    - ☛ *by drag and drop, see Adding a Report DataSet Collection/Property by drag and drop.*
    - ☛ *directly from a Report DataSet Collection of the structure, see Adding a value-type Property to the Report DataSet structure.*
  - parameter-type Properties
    - ☛ *See Adding a parameter-type Property to the Report DataSet structure.*
  - computed-type Properties
    - ☛ *See Adding a computed-type Property to the Report DataSet structure.*
  - collection count-type Properties
    - ☛ *See Adding a collection count-type Property to the Report DataSet structure.*

Adding a Report DataSet Collection/Property by drag and drop

You can feed the Report DataSet with:

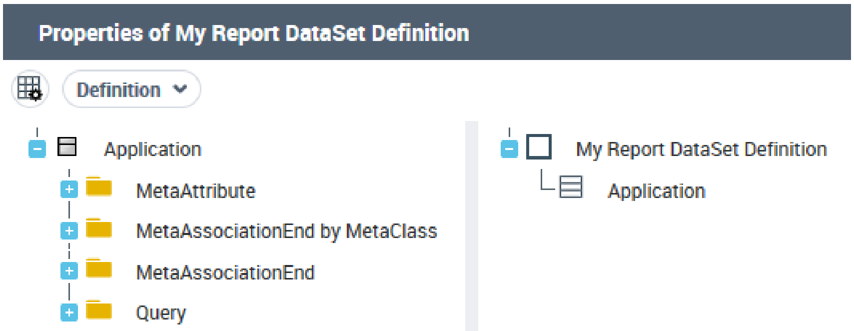
- MetaAttributes
- MetaAssociationEnds
- queries

You can add these items by drag and drop from the left pane to the right pane, by browsing the **MetaAttribute**, **MetaAssociationEnd**, **MetaAssociationEnd by MetaClass**, **Query** folders linked to the root MetaClass.

To add a Report DataSet item to the structure by drag and drop:

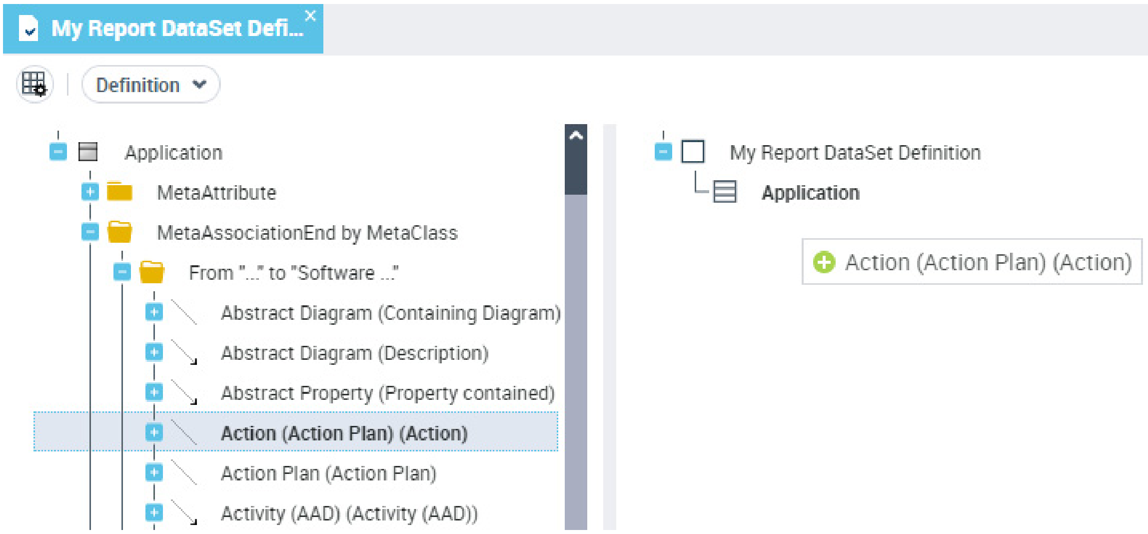
1. Access the Report DataSet Definition properties.

   ☛ *See Accessing the Report DataSet Definitions.*
2. Select **Definition**.
   The left pane displays the Root MetaClass with its associated MetaAttributes, MetaAssociationEnds, and queries.



3. In the left pane, expand the **MetaAttribute**/**MetaAssociationEnd by MetaClass**/**MetaAssociationEnd**/**Query** folder.
4. Select:
   - a MetaAttribute of the MetaClass
   - a MetaAssociationEnd of the MetaClass
   - a query that retrieves data associated with the MetaClass
   - or any item under the MetaAttribute/MetaAssociationEnd of the MetaClass

5. Drag & drop the MetaAttribute/MetaAssociationEnd/MetaClass/query to the right pane.



☛ *You can drag and drop as many MetaAttributes/ MetaAssociationEnds/MetaClasses/queries as needed.*

The MetaAttribute/MetaAssociationEnd/MetaClass or query is added to:

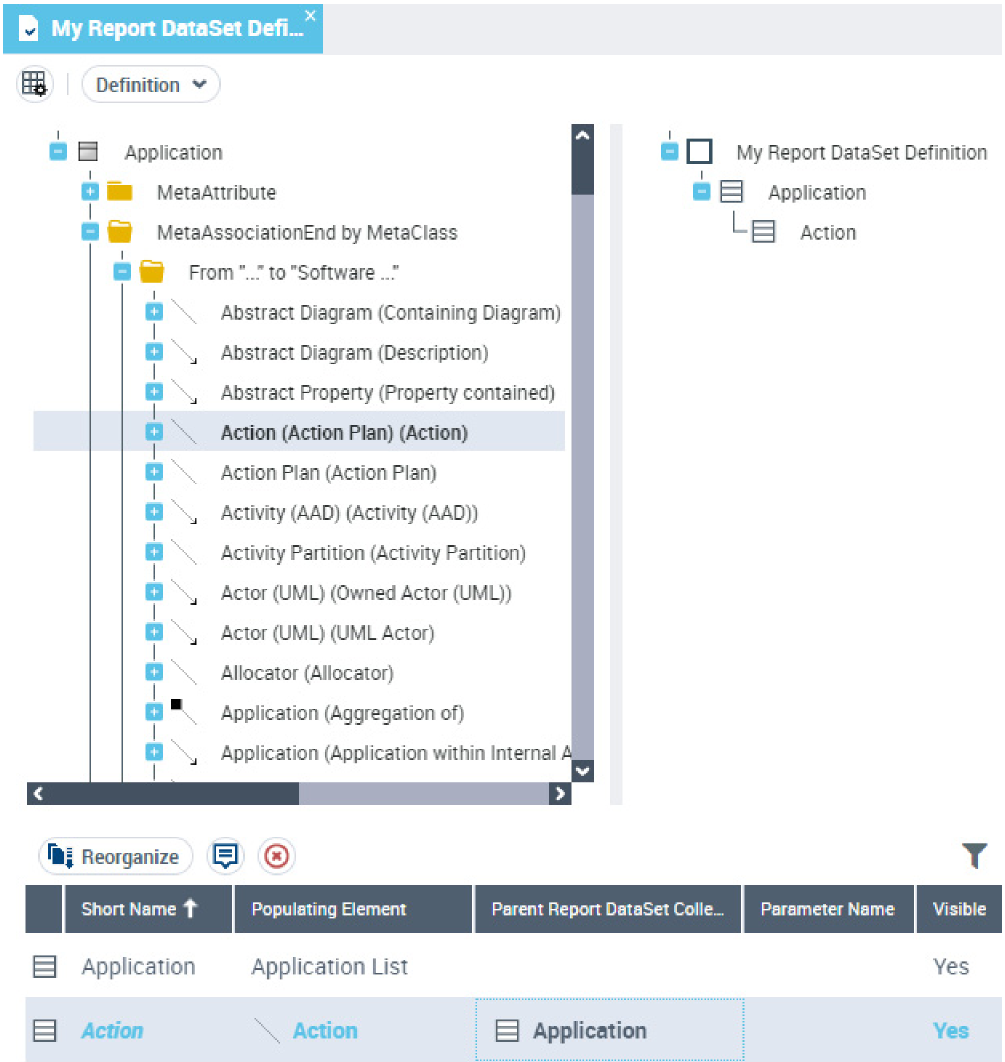
- (right pane) the Report DataSet structure
- (bottom pane) the list of items, where each item represents a column header of the Report DataSet.

☛ *To rename a column header, see Modifying the Name of a Report DataSet Column Header.*

☛ *To sort the columns in the Report DataSet display, see Modifying the Display Order of the Report DataSet Columns.*

☛ *To hide a column in the Report DataSet display, see Hiding a Column of the Report DataSet.*

☛ *To remove a column from the Report DataSet, see Removing a Column from a Report DataSet.*



Adding a Report DataSet Collection directly from a Report DataSet Collection

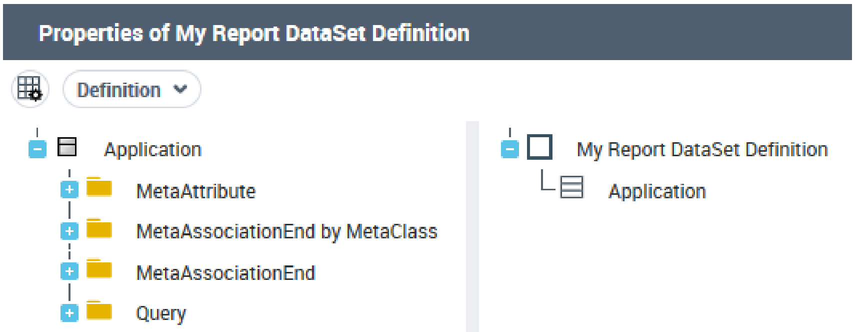
From the Report DataSet structure, you can add a Report DataSet Collection to the Report DataSet Collection.

To add a Report DataSet Collection to the Report DataSet structure:

1. Access the Report DataSet Definition properties.

   ☛ *See Accessing the Report DataSet Definitions.*

2. Select **Definition**.
   The right pane displays the Report DataSet structure.



3. In the right pane, right-click the **Report DataSet Collection** and select **New> Report DataSet Collection**.
4. To define how to populate the Report DataSet Collection:
   - in the **Populating Collection Parameter** field, select the Collection parameter.

     ☛ *A Collection parameter is available if in the Report DataSet Definition **Characteristics** tab, you added a **Collection Parameter**, see Adding Collection parameters.*
   - else, in the **Populating Query** drop-down list, select or create a query.

     ☛ *The drop-down list includes only the queries related to the parent Report DataSet Collection.*

     ☛ *If you created a Property parameter in the Report DataSet Definition **Characteristics** tab (see Adding Property parameters), you can use it in the query.*
   - else, in the **Populating MetaAssociationEnd** drop-down list, select the MetaAssociationEnd.
5. Click **OK**.

Adding a value-type Property to the Report DataSet structure

You can feed the Report DataSet with value-type properties, based on:

- a MetaAttribute
- a TaggedValue

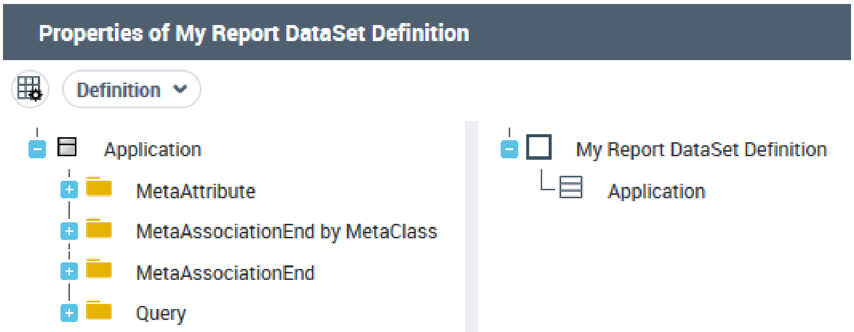
From the Report DataSet **structure**, you can add the **Property** to a Report DataSet **Collection** of the structure.

To add a value-type Property to the Report DataSet structure:

1. Access the Report DataSet Definition properties.

   ☛ *See Accessing the Report DataSet Definitions.*

2. Select **Definition**.
The right pane displays the Report DataSet structure.



3. In the right pane, right-click the **Report DataSet Collection** and select **New> Report DataSet Property**:
   - In the **Property type** field, select "Value".
   - In the **TaggedValue/MetaAtribute** field, select the TaggedValue/ MetaAttribute.
4. Click **OK**.

Adding a parameter-type Property to the Report DataSet structure

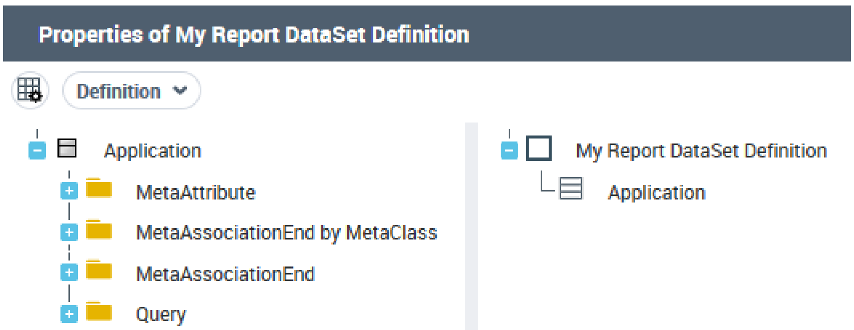
You can feed the Report DataSet with parameter-type Properties.

From the Report DataSet **structure**, you can add the **Property** to a Report DataSet **Collection** of the structure.

To add a parameter-type Property to the Report DataSet structure:

1. Access the Report DataSet Definition properties.

   ☛ *See Accessing the Report DataSet Definitions.*
2. Select **Definition**.
The right pane displays the Report DataSet structure.



3. In the right pane, right-click the **Report DataSet Collection** and select **New> Report DataSet Property**:
   - In the **Property type** field, select "Parameter".
   - In the **MetaParameter** field, select the **MetaParameter**.

     ☛ *A MetaParameter is available if in the Report DataSet Definition **Characteristics** tab, you added a **Property Parameter**, see Adding Property parameters.*
4. Click **OK**.

Adding a computed-type Property to the Report DataSet structure

You can feed the Report DataSet with a computed-type Property. The Property is computed locally for the dataset. As computed, the Property cannot be drag and drop. For this purpose you create a **computed Report DataSet Property** from the **Report DataSet Collection** using a macro.

To add a computed-type Property to the Report DataSet structure:

1. Access the Report DataSet Definition properties.

   ☛ *See Accessing the Report DataSet Definitions.*
2. Select **Definition**.
   The right pane displays the Report DataSet structure.
3. In the right pane, right-click the **Report DataSet Collection** and select **New> Report DataSet Property**:
   - In the **Property type** field, select "Computed".
   - Define the **MetaAttribute** characteristics (**Type**, **Length**, **Format**)
   - In the **Implementation** field, click the right oriented arrow and select **Create Macro**.

     ☛ *For an example, see Collecting a Set of Objects Using a Query and Adding a Computed Property.*
4. Click **OK**.

Adding a collection count-type Property to the Report DataSet structure

You can feed the Report DataSet with a collection count-type Property.

The Property is collected locally for the dataset, with the count of the collection retrieved from a query or a MetaAssociationEnd. As collected, the Property cannot be drag and drop. For this purpose you create a **collection count Report DataSet Property** from the **Report DataSet Collection** using a query or a MetaAssociationEnd.

To add a collection count-type Property to the Report DataSet structure:

1. Access the Report DataSet Definition properties.

   ☛ *See Accessing the Report DataSet Definitions.*
2. Select **Definition**.
   The right pane displays the Report DataSet structure.
3. In the right pane, right-click the **Report DataSet Collection** and select **New> Report DataSet Property**.
4. In the **Property type** field, select "Collection count".

5. From the **Counted Query** (or **Counted MetaAssociationEnd**) field connect the query (or the MetaAssociationEnd).

Previewing the Report DataSet

From the Report DataSet Definition properties you can test the Report DataSet.

To preview the Report DataSet:

1. Access the Report DataSet Definition properties.

   ☛ *See Accessing the Report DataSet Definitions.*

2. Display its **Preview** page.
3. Click **Create Preview**.
4. In the **Parameters** section, define the source data:
   - if needed enter the parameter value

     ☛ *See Adding Parameters to Filter Data Extraction.*

   - if needed, in the source MetaClass, click **Connect** and connect the MetaClass item(s) required.
5. In the **Report DataSet** section, click **Generate** .
   The Report DataSet displays:
   - date and time of computation
   - collected data in tabular form

     ☛ *Objects from a Data Reading access not accessible to the user are not displayed.*

     ☛ *In the **Report DataSet** section, click a column header to sort the result according to this data.*

     ☛ *In the **Report DataSet** section, using the column filters or the column filtering tool perform actions on the Report DataSet display only: all the data (displayed or hidden) is still included in the Report DataSet.*
6. You can:
   - customize the Report DataSet

     ☛ *See Customizing the Report DataSet.*

   - generate an Instant Report from the Report DataSet

     ☛ *The entire Report DataSet data is taken into account, even the hidden data if you used the column filters or the column filtering tool (in the **Preview** page). The customization must be performed in the **Definition** page.*

   - export the Report DataSet in Excel format

     ☛ *If you filtered the Report DataSet display, in that case only displayed data is exported.*

Customizing the Report DataSet

See:

Modifying the Display Order of the Report DataSet Columns

Once the Report DataSet Definition tree is defined you can modify the Report DataSet column display order.

To modify the display order of the Report DataSet:

1. Access the Report DataSet Definition properties.

   ☛ *See Accessing the Report DataSet Definitions.*

2. Select **Definition**.
3. In the bottom pane, click **Reorganize** .
4. Select the items you want to move and drag and drop them where you want.

Hiding a Column of the Report DataSet

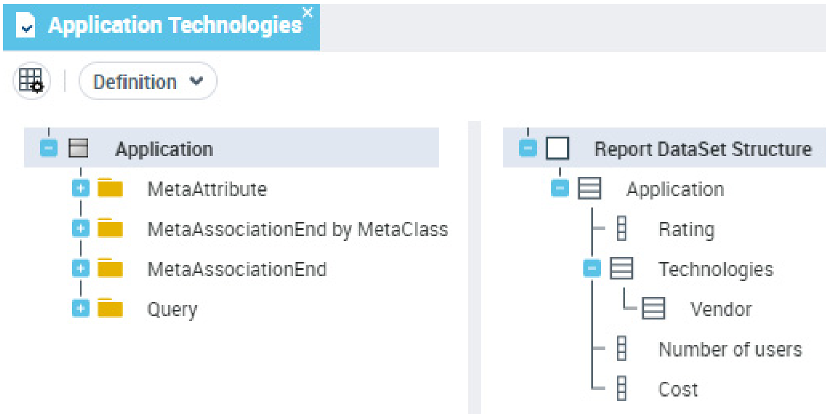
To ease the Report DataSet report readability you can hide columns. At Report DataSet generation, the data is still retrieved and can be displayed if needed.

To hide a column of the Report DataSet:

1. Access the Report DataSet Definition properties.

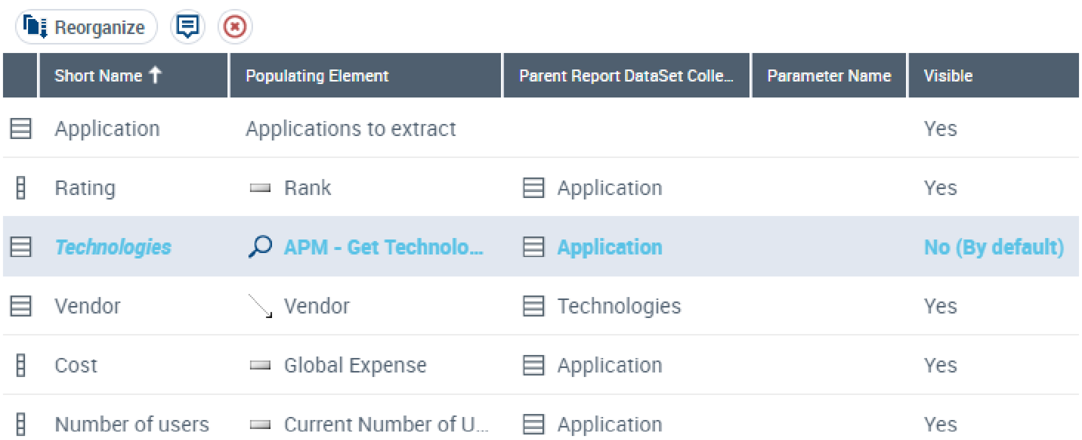
   ☛ *See Accessing the Report DataSet Definitions.*

2. Select **Definition**.



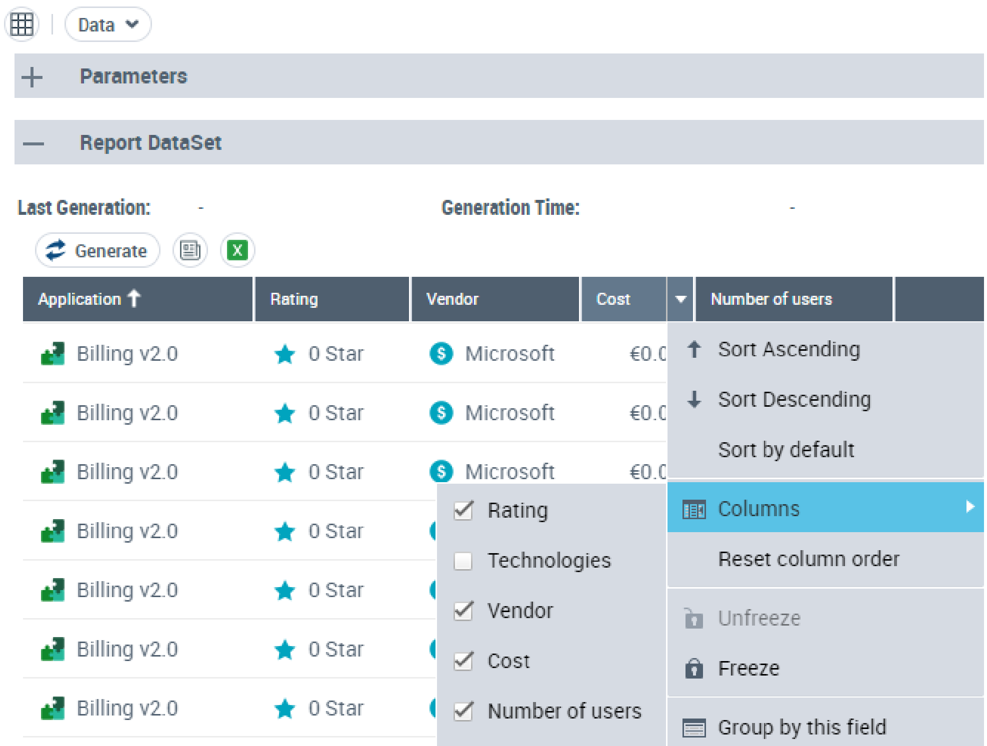
3. In the bottom pane, in the **Visible** field of the column you want to hide select "No (By default)".

   `Example: for the "Technologies", in its` **`Visible`** `field select "No (By default)".`



Short Name	Populating Element	Parent Report DataSet Colle...	Parameter Name	Visible
Application	Applications to extract			Yes
Rating	Rank	Application		Yes
Technologies	APM - Get Technolo...	Application		No (By default)
Vendor	Vendor	Technologies		Yes
Cost	Global Expense	Application		Yes
Number of users	Current Number of U...	Application		Yes

4. At Report DataSet generation, the column is hidden by default, but can be displayed (using the filter available on column headers in object lists, see Handling Lists).



Removing a Column from a Report DataSet

At Report DataSet Definition creation, in the Report DataSet structure, you might need to add elements that are necessary to retrieve other elements. These elements are available in the structure for this purpose but you might not want them to be included in the Report DataSet.

You can make them not available in the Report DataSet. In that case, the data is not retrieved.

For example, with a Report DataSet Definition based on the Application MetaClass, you can add:

- the Technologies used for these applications.

- the Vendors of these Technologies: you need to go through the Technologies to retrieve the Vendors.

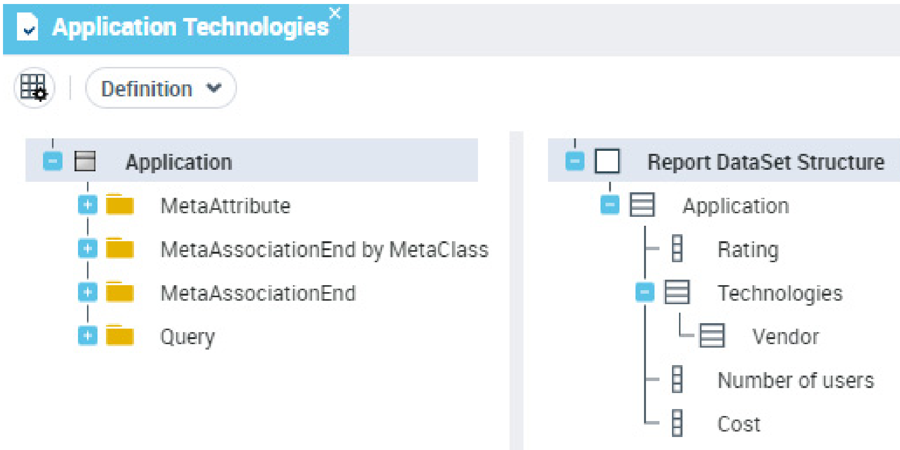
You may be interested in the Vendors of the Technologies but not in the Technologies themselves. You can make the "Technologies" column not available in the Report DataSet.

To remove a column from a Report DataSet:

1. Access the Report DataSet Definition properties.

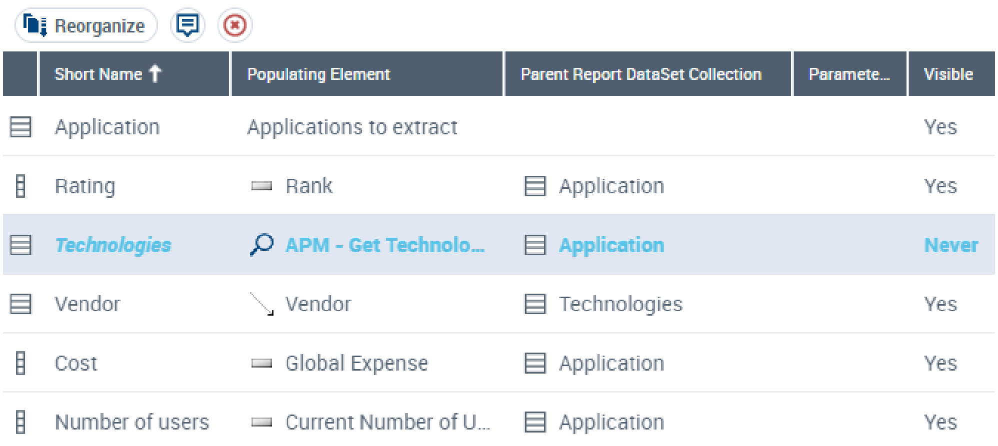
   ☛ *See* *Accessing the Report DataSet Definitions.*

2. Select **Definition**.



3. In the bottom pane, in the **Visible** field of the column you do not want to be available in the Report DataSet, select "Never".

   ```
   For example: for the "Technologies", in its Visible field
   select "Never".
   ```



Short Name	Populating Element	Parent Report DataSet Collection	Paramete...	Visible
Application	Applications to extract			Yes
Rating	Rank	Application		Yes
Technologies	APM - Get Technolo...	Application		Never
Vendor	Vendor	Technologies		Yes
Cost	Global Expense	Application		Yes
Number of users	Current Number of U...	Application		Yes

4. At Report DataSet generation, the column is not displayed, and cannot be added. The data is not retrieved.

Modifying the Name of a Report DataSet Column Header

When performing a drag and drop (see Adding a Report DataSet Collection/Property by drag and drop) Report DataSet item names are automatically defined according to what they stand for. When a name is already used in the same Report DataSet Definition, a suffix is added to the name (-n: -1, -2 etc.).

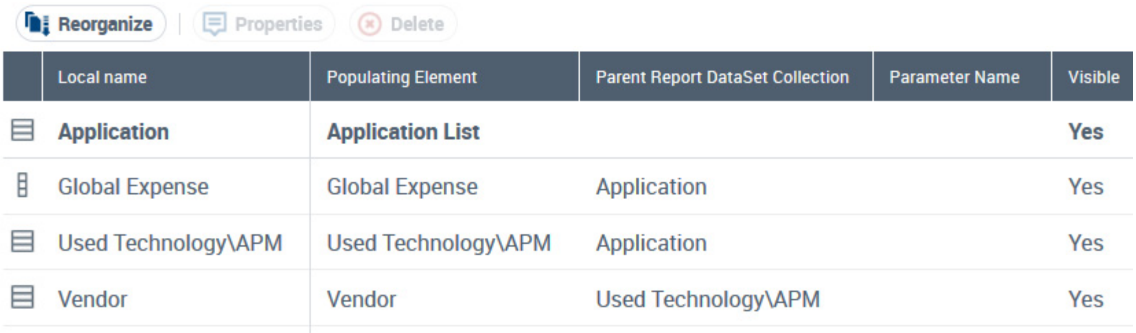
To modify the name of a Report DataSet column header you have to modify the Report DataSet item name.

To modify a Report DataSet item name:

1. Access the Report DataSet Definition properties.

   ☛ *See Accessing the Report DataSet Definitions.*

2. Select **Definition**.

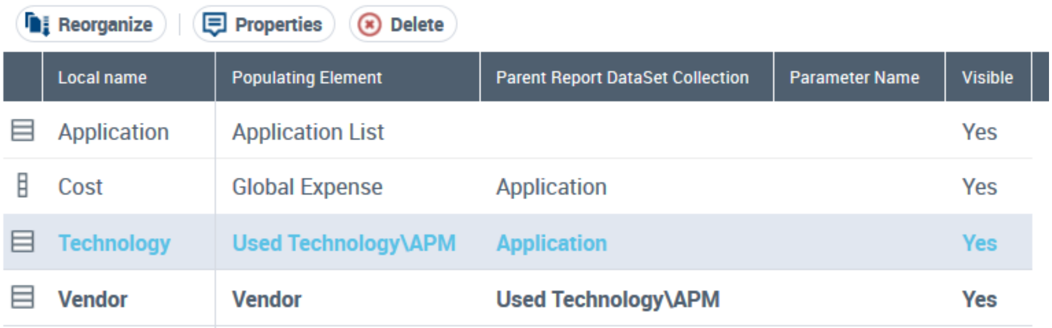


Reorganize | Properties | Delete

	Local name	Populating Element	Parent Report DataSet Collection	Parameter Name	Visible
	Application	**Application List**			**Yes**
	Global Expense	Global Expense	Application		Yes
	Used Technology\APM	Used Technology\APM	Application		Yes
	Vendor	Vendor	Used Technology\APM		Yes

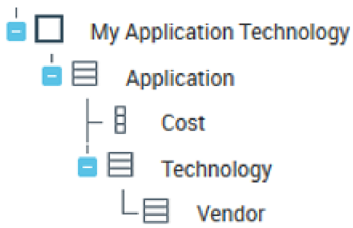
3. In the bottom pane, click the **Local Name** field concerned and modify its name.

   ```
   Example: "Global Expense" in "Cost", "Used Technology\APM"
   in "Technology".
   ```



Local name	Populating Element	Parent Report DataSet Collection	Parameter Name	Visible
Application	Application List			Yes
Cost	Global Expense	Application		Yes
Technology	Used Technology\APM	Application		Yes
Vendor	Vendor	Used Technology\APM		Yes

The Report DataSet Definition tree is updated accordingly.



In the bottom pane, the **Populating Element** column keeps track of the Report DataSet column header names(s) you modified with the corresponding MetaModel element name(s).

Speeding up the Report DataSet Display

By default, the **Keep last generated result** option of the Report DataSet Definition is selected, so that at next session, the first Report DataSet display time is shorten.

To modify the Report DataSet result display:

1. Access the Report DataSet Definition properties.

   ☛ *See Accessing the Report DataSet Definitions.*
2. Select **Characteristics**.
3. Select/Clear **Keep last generated result**.

How To

See:

- How to Add a Column to a Report DataSet
- How to Remove a Column from a Report DataSet
- How to Hide a Column in a Report DataSet
- How to Modify a Report DataSet Definition Provided by MEGA
- How to Duplicate a Report DataSet Definition
- How to Add a Report DataSet in an Object Property Pages

How to Add a Column to a Report DataSet

Each Report DataSet column corresponds to each data you defined as feeding the Report DataSet.

To add a column in a Report DataSet you can either:

- In the Report DataSet Definition definition tab, drag and drop object from the left pane to the Report DataSet tree structure.

  ☛ *See Defining the Data that Feeds the Report DataSet.*

- In the Report DataSet Definition definition tab, from the Report DataSet Collection create a **Report DataSet Property**.

  ☛ *See Adding a computed-type Property to the Report DataSet structure.*

How to Remove a Column from a Report DataSet

Each Report DataSet column corresponds to each data you defined as feeding the Report DataSet.

You might want some of theses columns not to be available in the Report DataSet, i.e. you do not want to retrieve this data from the repository.

To remove a column from the Report DataSet, you can customize the **Report DataSet Definition**.

☛ *See Removing a Column from a Report DataSet.*

How to Hide a Column in a Report DataSet

Any data, displayed or hidden in the Report DataSet is part of the Report DataSet, and taken into account at Instant report creation.

You can hide a column from:

- the **Report DataSet Definition**
  Data is extracted from the repository. By default, it is not displayed in the Report DataSet, but can be displayed. Hidden data remains in the Report DataSet.

  ☛ *See Hiding a Column of the Report DataSet.*

- the **Report DataSet**
  Data is extracted from the repository but not displayed in the Report DataSet.

  ☛ *See Handling the Report DataSet.*

How to Modify a Report DataSet Definition Provided by MEGA

You might want to modify a Report DataSet Definition provided by MEGA. The best way to do so is to duplicate the Report DataSet Definition and modify it according to your needs.

How to Duplicate a Report DataSet Definition

You might need to duplicate a Report DataSet Definition to modify it according to your needs.

To duplicate a Report DataSet Definition:

1. Access the Report DataSet Definition folder.

   ☛ *See Accessing the Report DataSet Definitions.*

2. Right-click the Report DataSet Definition and select **Manage > Duplicate**.
3. Define the name of the duplicated Report DataSet Definition.
4. Click **OK**.

How to Add a Report DataSet in an Object Property Pages

You can define a Report DataSet in an object Property pages as a report.

To do so, you must:

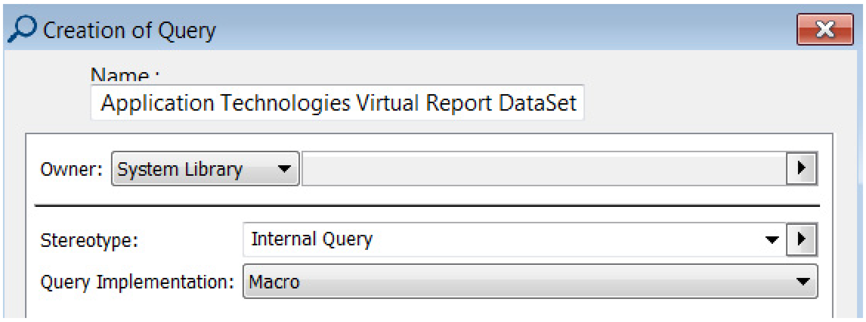
- create a macro-based query, which call the Report DataSet and its collection parameter.
- create a MetaProperty page on the MetaClass concerned, which call the query you created.

  For example, in Application MetaClass properties, you can add a page that displays all the technologies of the application using the Application Technologies Report DataSet.

To define a Report DataSet in an object Property page as a report:

1. Connect to **HOPEX** (Windows Front-End) with **HOPEX Customizer** profile.
2. Use the **Explore** tool to create a macro-based query:
   - in the **Name** field enter a name to your query

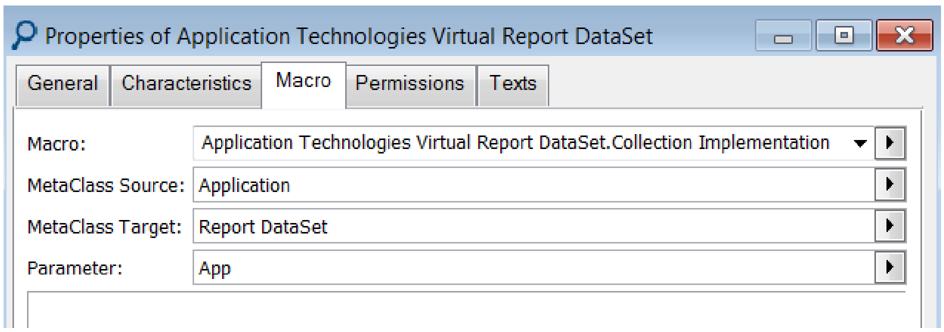
     `Example: "Application Technologies Virtual Report DataSet"`
   - in the **Stereotype** field, select "Internal Query"
   - in the **Query Implementation** field, select "Macro"



3. Click **Finish**.
4. Access your query properties..
5. In the **Macro** tab:
   - In the **Macro** field, click the arrow and select **Create Macro** (select **(VB) script macro** and click **Finish**).
   - In the **MetaClass Source** connect the source MetaClass.

     `Example: "Application" MetaClass`
   - In the **MetaClass Target** connect the "Report DataSet" MetaClass.
   - In the **Parameter** field create a parameter associated with the source MetaClass.

     `Example: "App"`



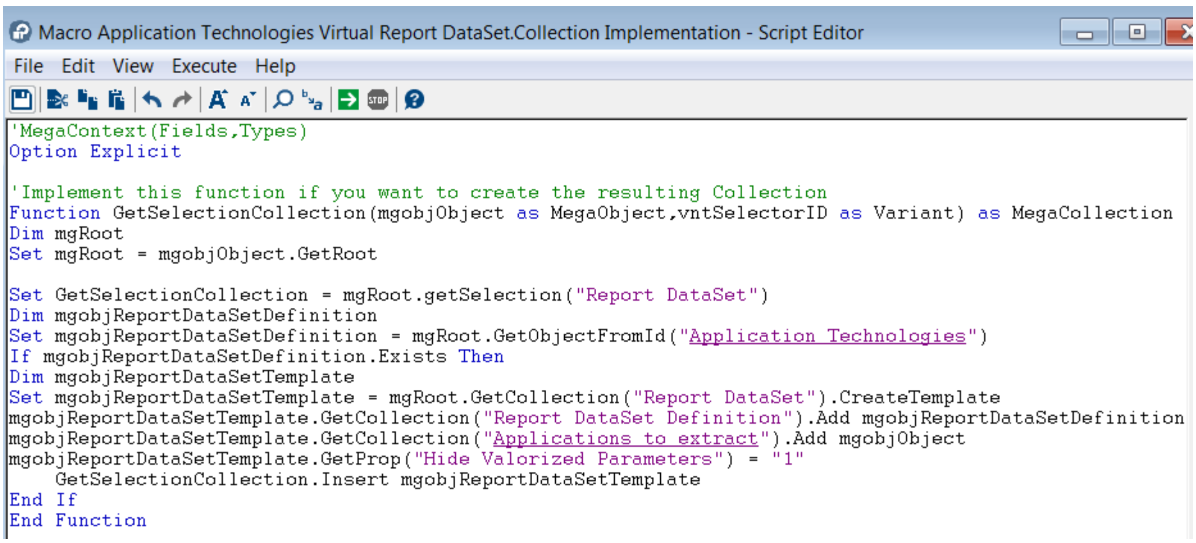
6. Edit the macro, and enter the following code, and replace the following text with your object megafields:
   - <REPORT DATASET DEFINITION MEGAFIELD>

     `Example: ~b)pvF8oxKj34[Application Technologies]`
   - <REPORT DATASET COLLECTION PARAMETER MEGAFIELD>

     `Example: ~8(pvzTrxKPJ5[Applications to extract]`

```
'MegaContext(Fields,Types)
Option Explicit

'Implement this function if you want to create the resulting
Collection Function GetSelectionCollection(mgobjObject as
MegaObject,vntSelectorID as Variant) as MegaCollection
```

```
  Dim mgRoot
  Set mgRoot = mgobjObject.GetRoot

  Set GetSelectionCollection =
mgRoot.getSelection("~)ilXTIGrKPiB[Report DataSet]")
Dim mgobjReportDataSetDefinition
Set mgobjReportDataSetDefinition =
mgRoot.GetObjectFromId("<REPORT DATASET DEFINITION
MEGAFIELD>")
If mgobjReportDataSetDefinition.Exists Then
Dim mgobjReportDataSetTemplate
Set mgobjReportDataSetTemplate =
mgRoot.GetCollection("~)ilXTIGrKPiB[Report
DataSet]").CreateTemplate
mgobjReportDataSetTemplate.GetCollection("~rLU9sCZtKLx6[Rep
ort DataSet Definition]").Add mgobjReportDataSetDefinition
mgobjReportDataSetTemplate.GetCollection("<REPORT DATASET
COLLECTION PARAMETER MEGAFIELD>").Add mgobjObject
mgobjReportDataSetTemplate.GetProp("~CjvjWdhJOXuC[Hide
Valorized Parameters]") = "1"
GetSelectionCollection.Insert mgobjReportDataSetTemplate
End If
End Function
```



7. Create a MetaPropertyPage on the MetaClass you want to add the Report DataSet:

   `Example: on the "Application" MetaClass`

   - In the MetaClass properties, select **UserInterface** tab
   - In the **Property Page** subtab, create a MetaPropertyPage
   - In the **Name** field enter the name.

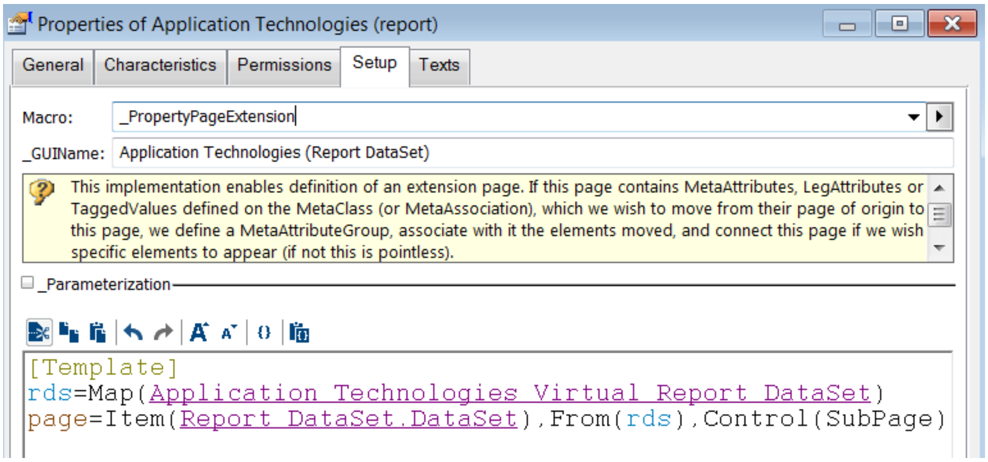
     `Example: "Application Technologies (report)"`

   - Click **Finish**.
8. Define the MetaPropertyPage properties, in the **Setup** tab:
   - In the **_GUIName** field enter the name that you want to be displayed in the object property pages.

     `Example: "Application Technologies (Report DataSet)"`

   - In the **Setup** tab enter the following code, and replace `<QUERY MEGAFIELD>` with the query megafield you created:

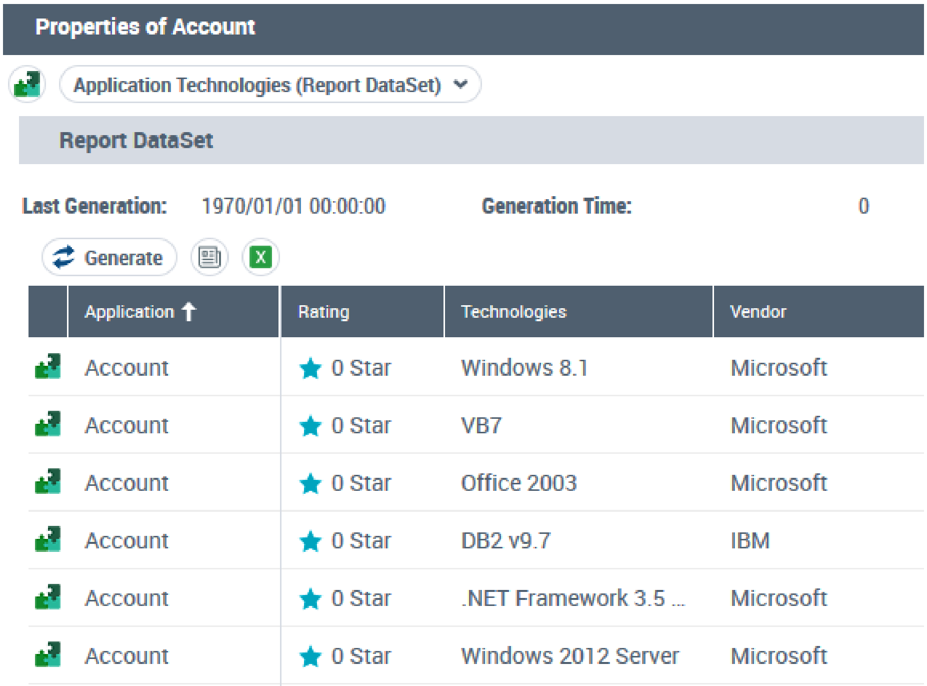
```
[Template]
rds=Map(<QUERY MEGAFIELD>)
page=Item(~Nx(oYLUvK1cF[Report
DataSet.DataSet]),From(rds),Control(SubPage)
```



9. Click **OK**.
10. Exit **HOPEX**.

11. Check that the new page is created.

    For example, check that the "Application Technologies (report DataSet)" page is added to Application property pages.



	Application	Rating	Technologies	Vendor
	Account	0 Star	Windows 8.1	Microsoft
	Account	0 Star	VB7	Microsoft
	Account	0 Star	Office 2003	Microsoft
	Account	0 Star	DB2 v9.7	IBM
	Account	0 Star	.NET Framework 3.5 ...	Microsoft
	Account	0 Star	Windows 2012 Server	Microsoft

Use Cases

See:

- Collecting a Set of Objects, Linked Objects and some of their Properties
- Collecting a Set of Objects Using a Query and Adding a Computed Property

---

Collecting a Set of Objects, Linked Objects and some of their Properties

Description

A use case of Report DataSet Definition consists in defining the root collection as input parameter for the Report DataSet.

In this use case, the Report DataSet Definition is based on the **Application** root MetaClass and on the **Collection Parameter** "Application List" (of Collection Type: **Application**).

The Report DataSet displays for each Application:

- its Software Technologies used and associated Vendors.
- its Global Expenses

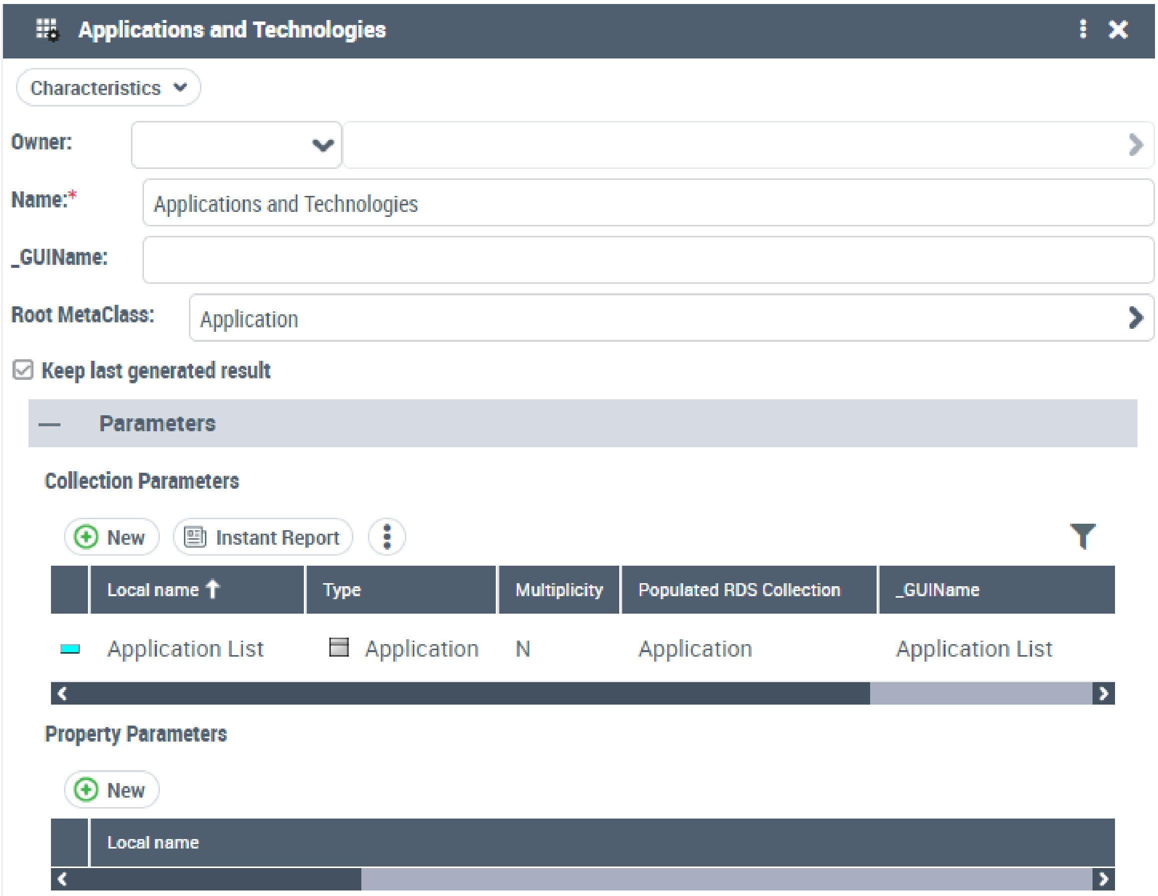
Note that the **Software Technology** MetaClass is not directly connected to the **Application** MetaClass, but to an abstract MetaClass (**Supporting Software Technology Building Block**) of which **Software Technology** and **Software Technology Stack** are sub-types.

So that to add the **Software Technology** to the Definition, you need to go through the **Supporting Software Technology** MetaAssociationEnd and define the **Software Technology** MetaAssociationEnd as target.

Creating the Report DataSet Definition

To create the Report DataSet Definition that displays the applications and their Technologies and associated Vendors:

1. Create a Collection Parameter - based Report DataSet Definition based on the **Application** root MetaClass.
   - ☛ *See Defining a Report DataSet populated by occurrences.*
   - ☛ *See Creating a Report DataSet Definition.*



   The collection parameter is pre-set or already set with the **Application** root MetaClass, the "Application List" Collection Parameter is automatically added with the "Application" Collection Type value.

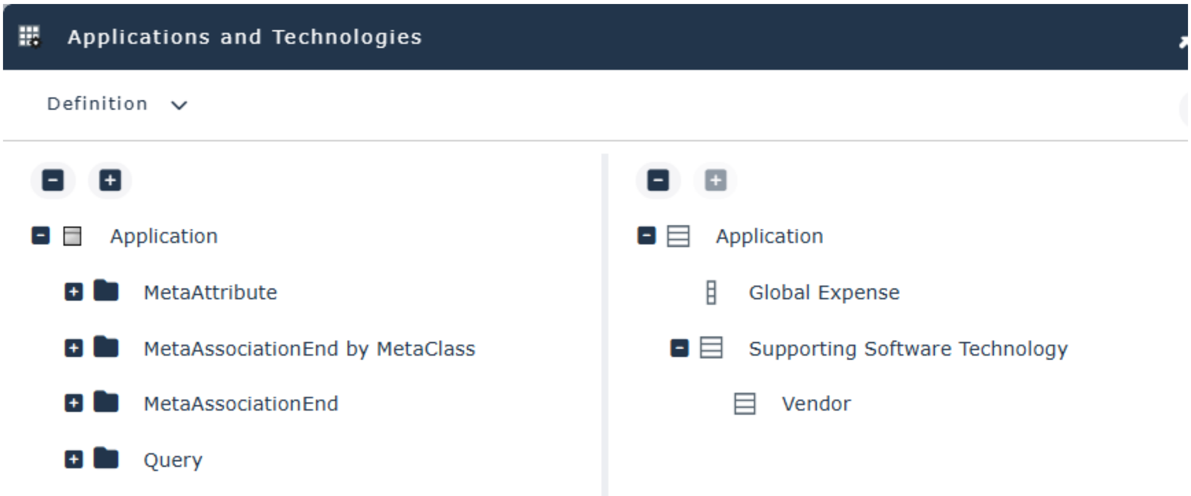
   This Collection parameter is used to list all the applications of the repository.

2. Define the data you want to display in the **Report DataSet**, from the Report DataSet Definition properties: **Definition**:
   - In the left pane, in the **MetaAttribute** folder, select "Global Expense" MetaAttribute and drag and drop to the right pane
   - In the right pane, right-click **Application**, select **New > Report Report DataSet Collection** and in **Populating**

MetaAssociationEnd field select "Supporting Software Technology" and in the **Target MetaClass** pane, click **Connect** and select "Software Technology".

- In the right pane, right-click "Supporting Software Technology", select **New > Report Report DataSet Collection** and in the **Populating MetaAssociationEnd** field select "Vendor".

  ☛ *See Defining the Data that Feeds the Report DataSet.*



3. Modify the name of "Supporting Software Technology" to "Technology".

   ☛ *See Modifying the Name of a Report DataSet Column Header.*

4. Reorganize the Report DataSet column order as follows:

   ☛ *See Modifying the Display Order of the Report DataSet Columns.*

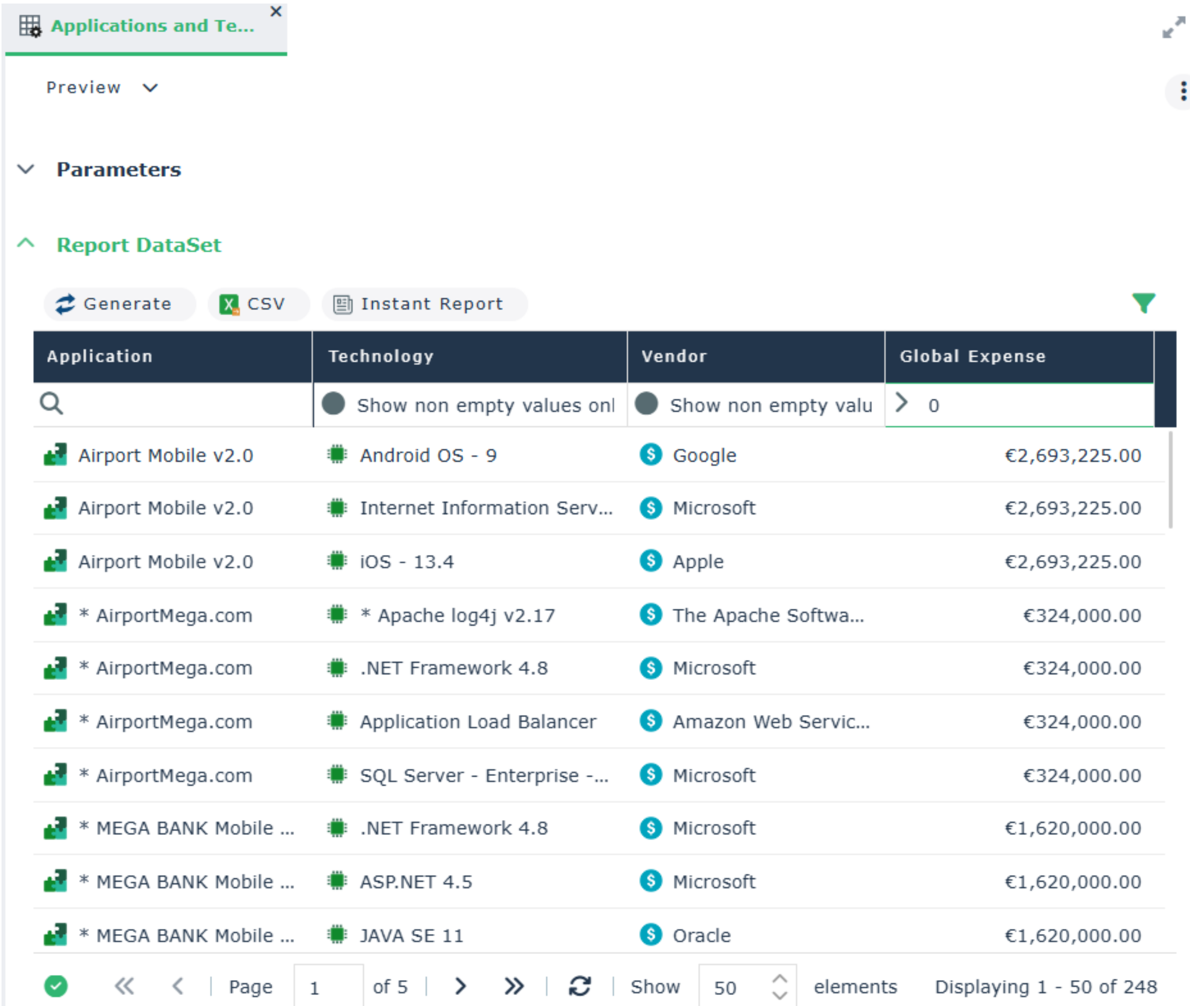
Short Name	Populating Element	Parent Report Data...	Parameter Name	Visible
Application	**Application List**			**Yes**
Technology	Supporting Software Technology	Application		Yes
Vendor	Vendor	Technology		Yes
Global Expense	Global Expense	Application		Yes

5. In the Report DataSet Definition properties, select **Preview** and test your Report DataSet Definition.

☛ *See Previewing the Report DataSet.*

You can use the filters available on the column headers to refine the Report DataSet display. This does not modify the Report DataSet content (only its display).

For example use the "Show non empty values only" for **Used Technology** and **Vendor** columns, and ">0" for **Global Expense**.



You can:

- export the Report DataSet in CSV format
- generate Instant Reports from the Report DataSet

☛ *In both cases, the entire Report DataSet data is taken into account, even data hidden with the filtering used in the **Preview** page.*

Collecting a Set of Objects Using a Query and Adding a Computed Property

Description of the Report DataSet Definition

This use case collects a set of applications of a specific portfolio of a given vendor asking the user to choose the portfolio and the vendor:

The result displays for each application:

- ***properties*** regarding the application (cost, type, rank, used technology) and the technology (end life date, vendor) and the vendor (number of provided technologies)
- a ***computed property***: the number of Business Processes of the application

This Report DataSet Definition is based on the root MetaClass: **Application (1)**.

To define how to collect the input data, you need to create a **Query (2)**, which computes the input data using two Property Parameters ("Vendor" and "Portfolio"):

- ***Query name***
  "APM - Get Applications of a Portfolio using Technologies of a given Vendor"
- ***Query code***

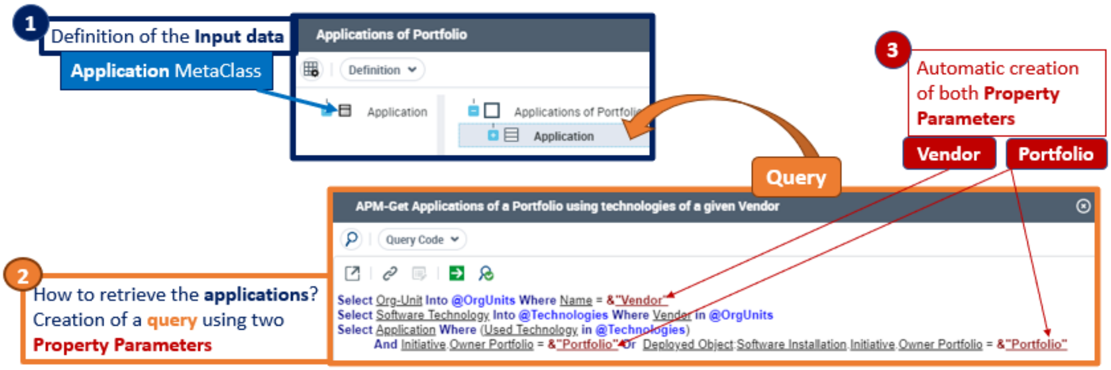
```
Select [Org-Unit] Into @OrgUnits Where [Name] = &"Vendor"

Select [Software Technology] Into @Technologies Where
[Vendor] in @OrgUnits

Select [Application] Where ([Used Technology] in
@Technologies)

          And [Initiative].[Owner Portfolio] = &"Portfolio"
Or   [Deployed Object]:[Software
Installation].[Initiative].[Owner Portfolio] = &"Portfolio"
```

  Where the ***Property Parameters***:

  - "Vendor" defines from who the end user wants to collect the applications
  - "Portfolio" defines from which portfolio the end user wants to collect the applications

Both "Vendor" and "Portfolio" **Property parameters (3)** are automatically created once the query is selected.

☛ *Note that the **Query Parameter Name** value in the query code and in the Property Parameters ("Vendor" and "Portfolio") are exactly the same, ans that they are case sensitive.*



To define **(4)** the data you want to display in the Report DataSet, you need to:

- drag and drop the Report DataSet properties **(a)**, i.e here MetaAttributes related to the **Application** root MetaClass.
- from Report DataSet Collections add Report DataSet Collections and Report DataSet Properties **(b)**

  ☛ *Note that the **Software Technology** MetaClass is not directly connected to the **Application** MetaClass, but to an abstract MetaClass (**Supporting Software Technology Building Block**) of which **Software Technology** and **Software Technology Stack** are sub-types.*

  *So that to add the **Software Technology** to the Definition, you need to go through the **Supporting Software Technology** MetaAssociationEnd and define the **Software Technology** MetaAssociationEnd as target.*

- create a computed property **(c)**:
  - a **Report DataSet Property**, which displays the number of Business Processes of each application

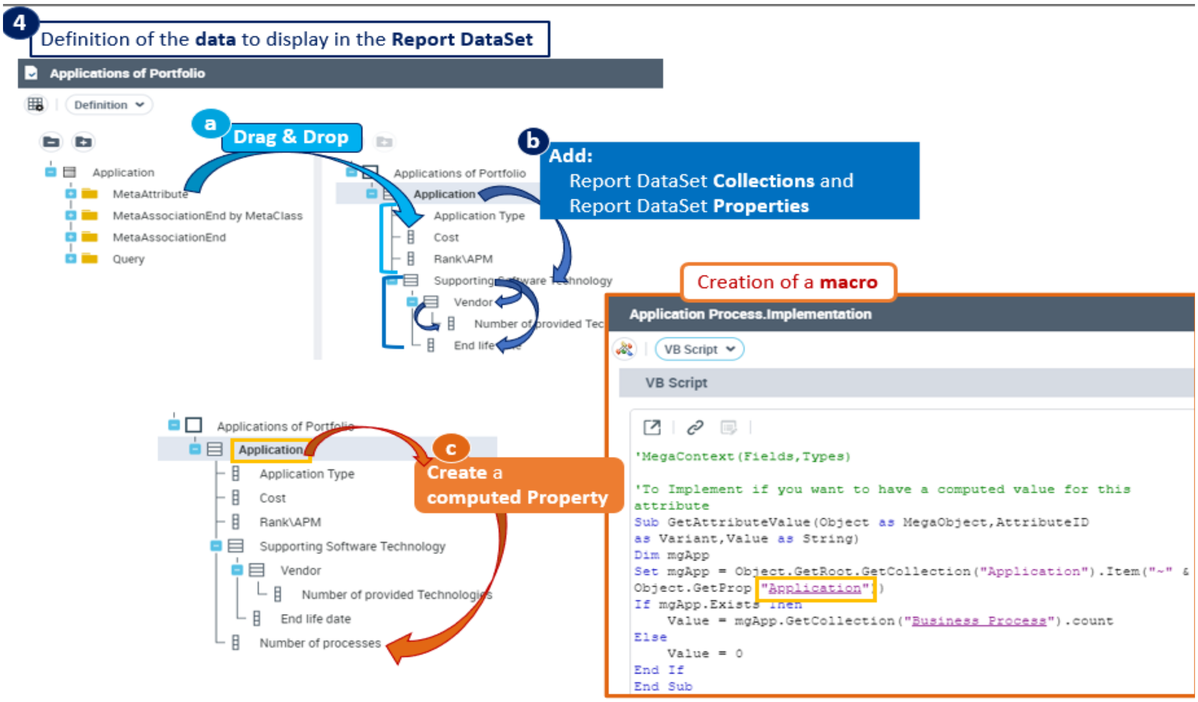
    `Report DataSet Property:` "***Number of processes***".

  - the **Macro**, which computes the "Number of processes"

    `Macro:` "***Application Process.Implementation***".

```
Sub GetAttributeValue(Object as MegaObject,AttributeID as Variant,Value as String)
  Dim mgApp
  Set mgApp =
Object.GetRoot.GetCollection("Application").Item("~" &
```

```
Object.GetProp("<Field of the Report DataSet Collection
"Application">"))
  If mgApp.Exists Then
    Value = mgApp.GetCollection("~h4n)MzlZpK00[Business
Process]").count
  Else
    Value = 0
  End If
End Sub
```



Creating the Report DataSet Definition

To create the Report DataSet Definition that collects applications of a specific portfolio of a given vendor:

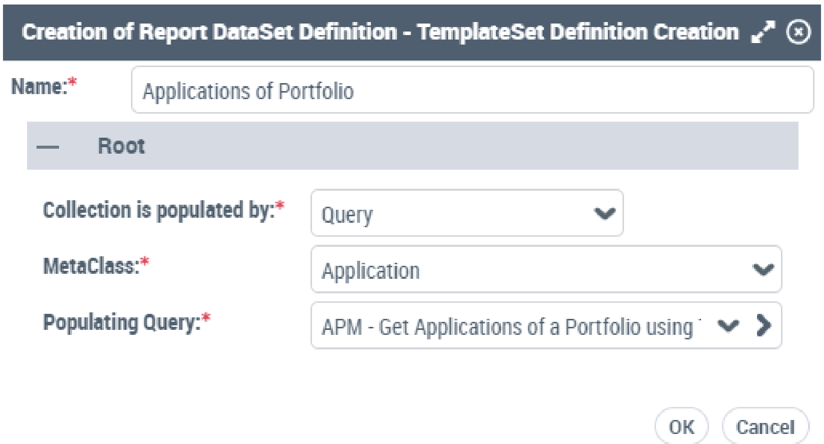
1. Create a query-based Report DataSet Definition based on the **Application** root MetaClass.

   e.g.: Report DataSet Definition **Name**: `Applications of Portfolio`.

   ☛ *See Defining a Report DataSet populated by a query.*

2. In the **Populating Query** field, create the query that defines how to collect the input data, with the following characteristics:
   - **Name**: "APM - Get Applications of a Portfolio using Technologies of a given Vendor"
   - **Query Implementation**: "Select"
   - **Stereotype**: "Internal Query"
   - **Query Code**:

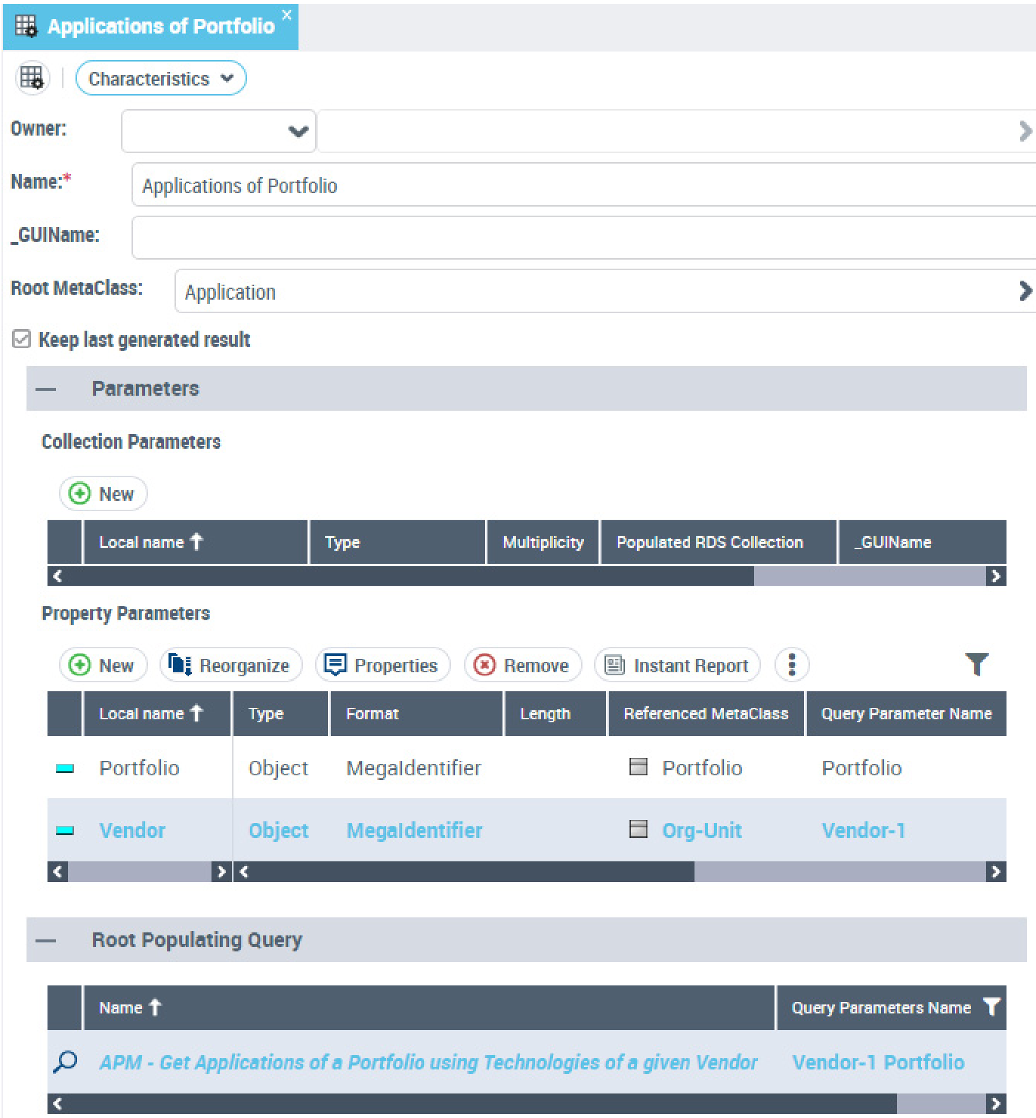
```
Select [Org-Unit] Into @OrgUnits Where [Name] = &"Vendor"
Select [Software Technology] Into @Technologies Where
[Vendor] in @OrgUnits
Select [Application] Where ([Used Technology] in
@Technologies)
         And [Initiative].[Owner Portfolio] = &"Portfolio"
Or  [Deployed Object]:[Software
Installation].[Initiative].[Owner Portfolio] = &"Portfolio"
```



3. Note that both "Portfolio" and "Vendor" **Property parameters** are automatically created:
   - **Type**: "Object"
   - **Format**: "MegaIdentifier"
   - **Referenced MetaClass**: Portfolio/Org-Unit MetaClass
   - **Query Parameter Name**: "Portfolio"/"Vendor-1"

   ☛ *Note that "Vendor"* ***Query Parameter Name*** *has been renamed in "Vendor-1" (in the Query Code as well). You can rename its* ***Local Name*** *in "Vendor".*

   Only the ***Query Parameter Name*** *value must have the exact same name in both places (be careful as it is case sensitive).*



4. Define the data you want to display in the Report DataSet, from the Report DataSet properties: **Definition**:

Drag and drop MetaAttributes:

☛ *See Adding a Report DataSet Collection/Property by drag and drop.*

In the left pane, select each object and drag and drop it to the right pane:

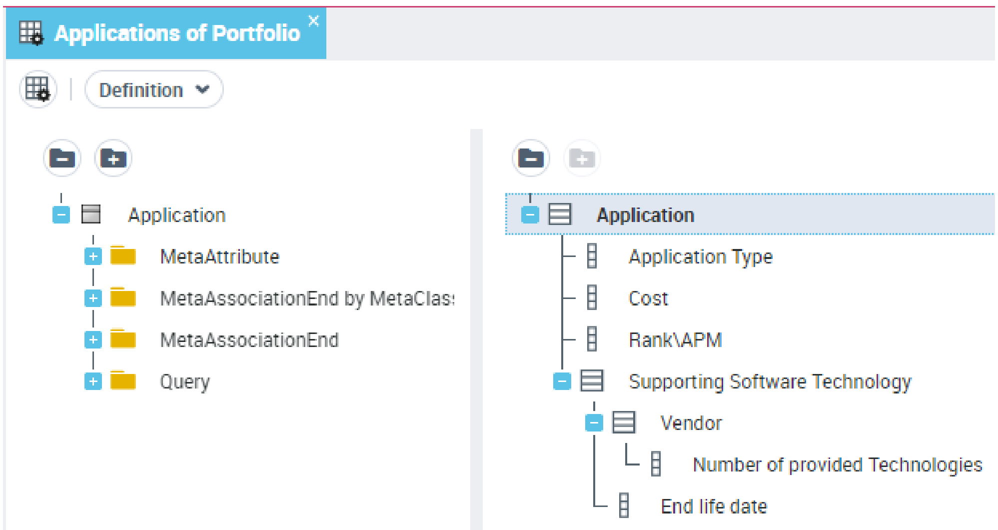
- **Application Type** MetaAttribute
- **Cost** MetaAttribute
- **Rank/APM** MetaAttribute

Add Report DataSet Collections and Report DataSet Properties

☛ *See Adding a Report DataSet Collection directly from a Report DataSet Collection.*

In the right pane:

- right-click **Application**, select **New > Report DataSet Collection** and in **Populating MetaAssociationEnd** field select "Supporting Software Technology" and in the **Target MetaClass** pane, click **Connect**, select "Software Technology" and click **OK**.
- right-click **Supporting Software Technology**, select **New > Report DataSet Collection** and in the **Populating MetaAssociationEnd** field select "Vendor" and click **OK**.
- right-click **Vendor**, select **New > Report DataSet Property**, in the **Local name** field enter "Number of provided Technologies", and in the **MetaAttribute** field select "Number of provided Technologies" and click **OK**.
- right-click **Supporting Software Technology**, select **New > Report DataSet Property** in the **Local name** field enter "End life date", and in the **MetaAttribute** field select "End life date" and click **OK**.



Add a computed Property

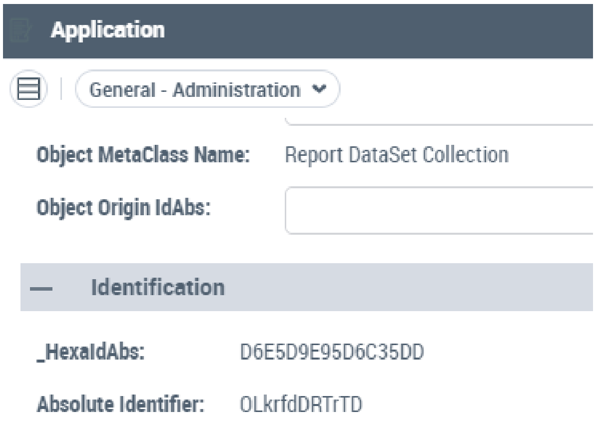
☛ *See Adding a computed-type Property to the Report DataSet structure.*

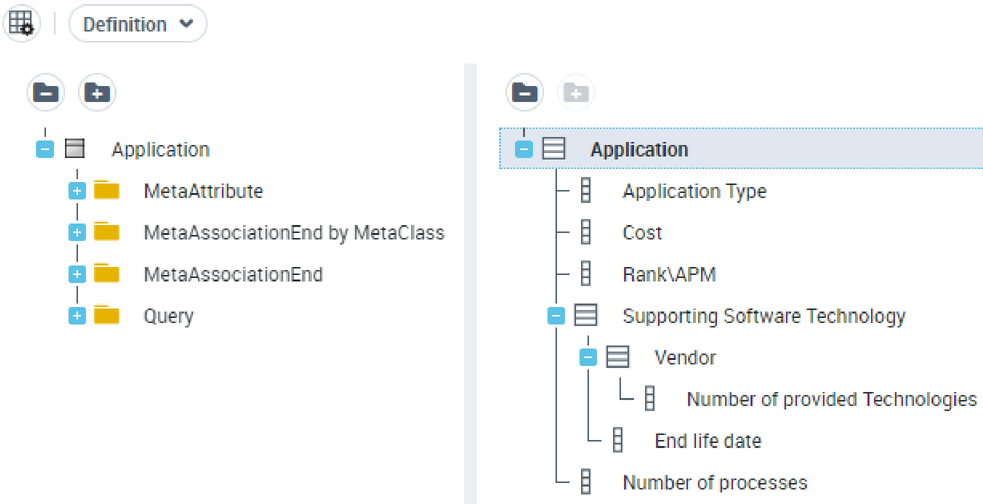
From the **Application** Report DataSet Collection, create a computed parameter "Number of processes":

- **Property type**: "Computed"
- **MetaAttribute Type**: "String"
- **MetaAttribute Length**: 63
- in the **Implementation** field, select **Create macro** and create the "Application Process.Implementation" macro:

```
Sub GetAttributeValue(Object as MegaObject,AttributeID as Variant,Value as String)
  Dim mgApp
  Set mgApp =
Object.GetRoot.GetCollection("Application").Item("~" &
Object.GetProp("~OLkrfdDRTrTD[Application]"))
  If mgApp.Exists Then
    Value = mgApp.GetCollection("~h4n)MzlZpK00[Business
Process]").count
  Else
    Value = 0
  End If
  ' Value = 4
End Sub
```

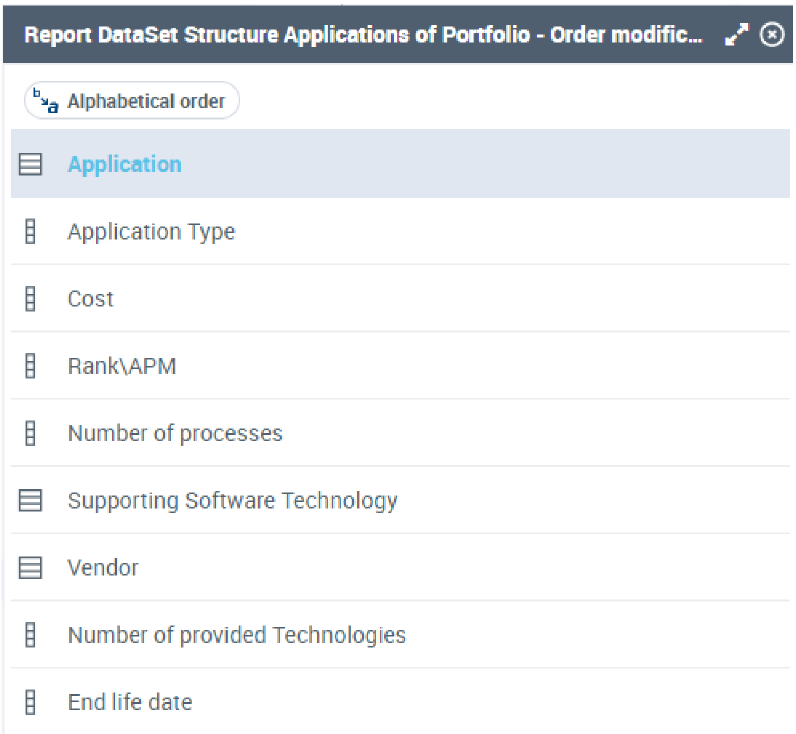
Where here "***~OLkrfdDRTrTD[Application]***" is the field of the Report DataSet Collection "Application".





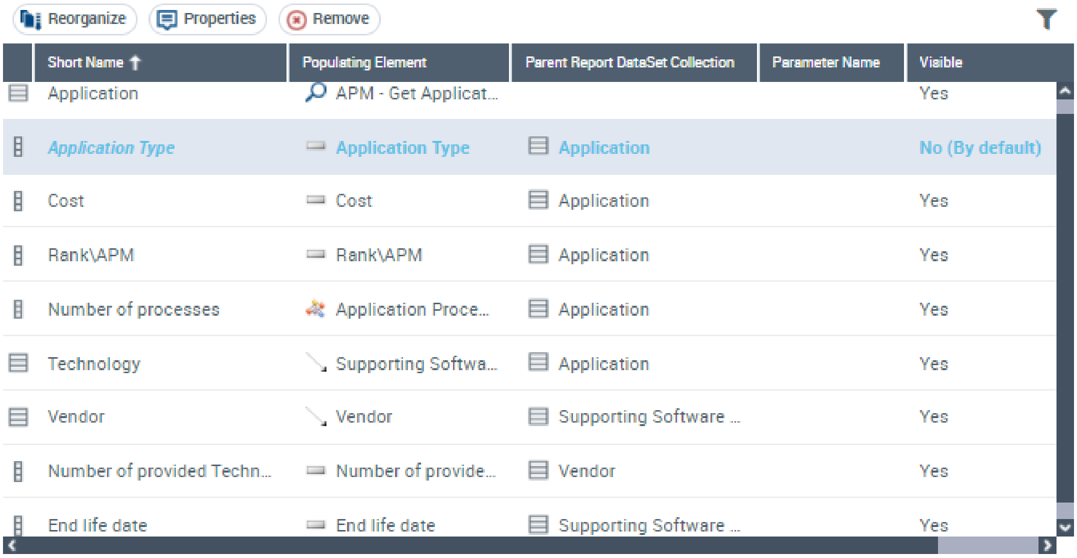
5. Reorder the Report DataSet columns.

☛ *See Modifying the Display Order of the Report DataSet Columns.*



6. Hide the "Application Type" column and rename "Supporting Software technology" in "Technology".

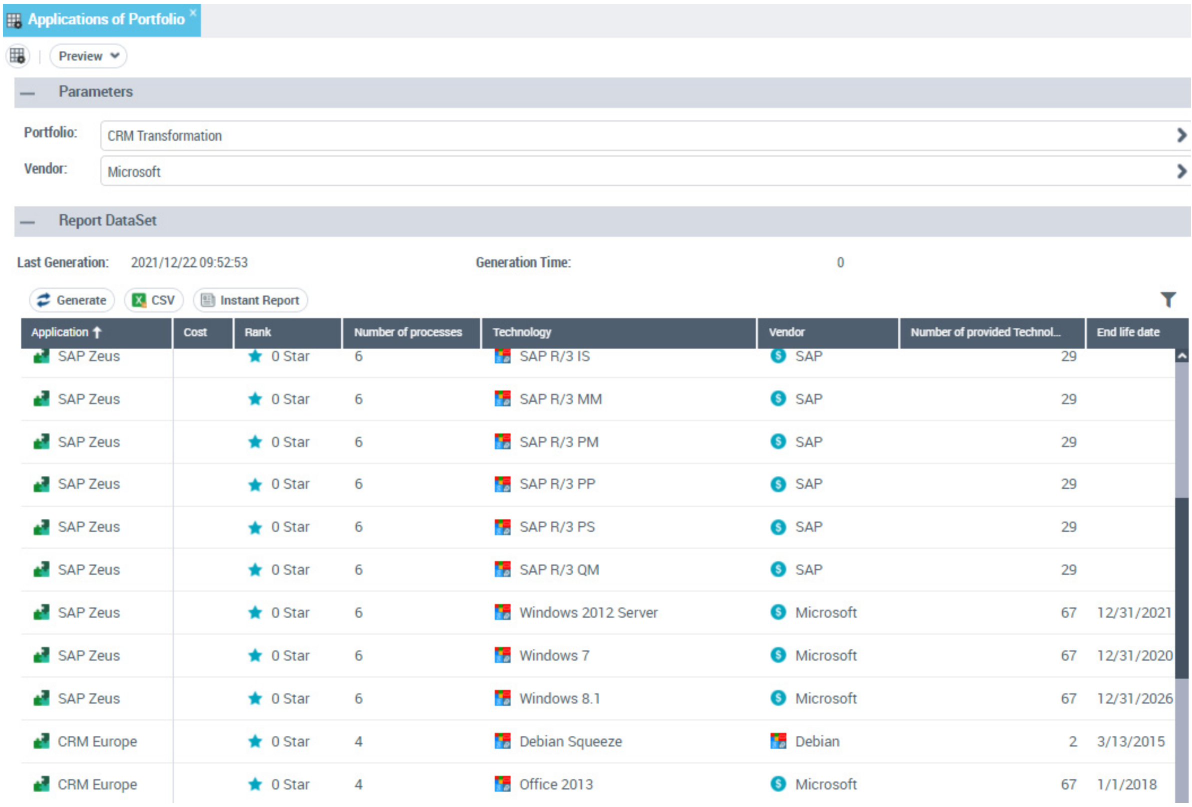
   ☛ *See Hiding a Column of the Report DataSet and Modifying the Name of a Report DataSet Column Header.*



7. In the Report DataSet Definition properties, select **Preview** and test your Report DataSet Definition.

   ```
   For example, use "CRM Transformation" Portfolio and
   "Microsoft" Vendor.
   ```

   ☛ *See Previewing the Report DataSet.*



You can:

- export the Report DataSet in Excel format (only displayed data is exported)
- generate Instant Reports from the Report DataSet (the entire Report DataSet data is taken into account, even data hidden with the filtering used in the **Preview** page).

REPORT DATASET CREATION

A **Report DataSet Definition** is available for any user to create a **Report DataSet**.

See:

- Creating a Report DataSet
- Handling the Report DataSet
- Creating a Report Template from the Report DataSet

Creating a Report DataSet

To create a Report DataSet:

1. Connect to **HOPEX** (Web Front-End), with a **Solution** profile (or with **HOPEX Cutomizer Publisher** profile).
2. Click the **Environment > Common: Report DataSets** navigation menu (or **Report Studio > Report DataSets**).
3. In **My Report DataSets** page, click **New** .
4. In the **Report DataSet Definition** drop-down list, select a **Report DataSet Definition**.

   `E.g. : Application Technologies.`
5. (Optional) Modify the **Name** of your Report DataSet**.**
   By default its name is: <name of its Report DataSet Definition>-x.

   `E.g. : Application Technologies-1.`
6. (If needed) In the **Parameters** section, enter the requested parameters.

   `E.g.: connect the applications to be extracted.`
7. Click **OK**.
   Your **Report DataSet** is created.
8. Access your Report DataSet properties.
9. Display its **Data** page.
10. In the **Report DataSet** section, your **Report DataSet** is fed with data.

    ☛ *If in the **Parameters** section you modify the parameters, then in the **Report DataSet** section, click **Generate** to update your Report DataSet.*

Handling the Report DataSet

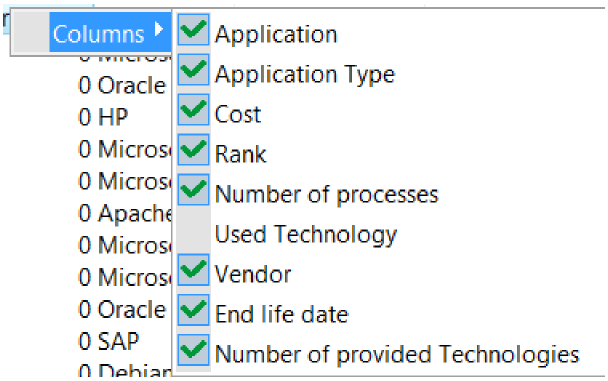
Once the **Report DataSet** is created, you can:

- modify the **Report DataSet** status.
  By default the **Report DataSet** is public. It is stored in **HOPEX** and shared with other users.

  ☛ *To facilitate the access to a **Report DataSet** you can make it private (from the **Report DataSet** properties > **Characteristics** page, in the **Sharing** field, select "Private") and define the persons concerned.*

*For these persons, the **Report DataSet** is then available in **My Report DataSets**. The **Report DataSet** is no longer available in **Public Report DataSets** for any user.*

- create an Instant Report from your **Report DataSet**: click **Instant Report** .
- refresh the **Report DataSet** with updated data: click **Refresh** .
- export the **Report DataSet**: click **CSV** .
- hide **Report DataSet** columns: right-click the table header, select **Columns** and clear the items you want to hide.
  This can be useful to facilitate readability of the Report DataSet.
  Hidden column data remains included in the Report DataSet. When you launch an instant report, the instant report is performed on all the Report DataSet data.



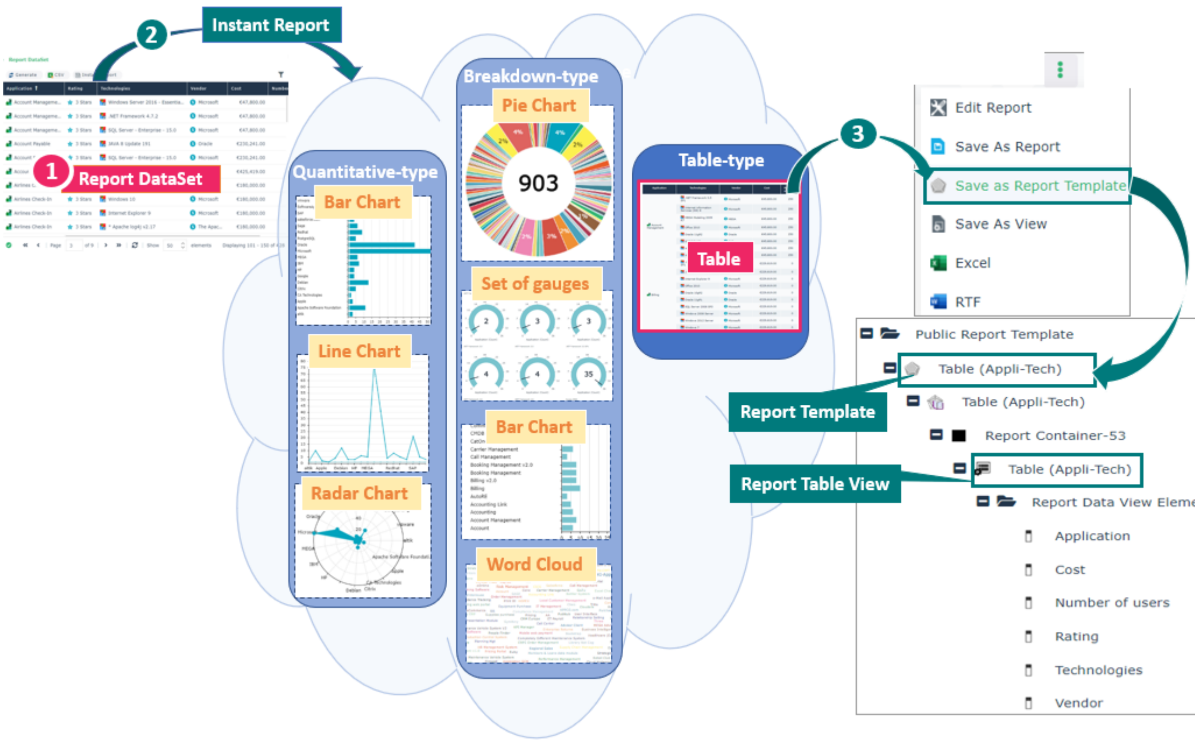
☛ *For detailed information on **Report DataSet** handling see Common Features documentation.*

Creating a Report Template from the Report DataSet

From the **Data** page of the **Report DataSet**, you can launch an Instant Report, then save the report as:

- a Report Template
- a Report Data View, to use it in a Report Template

> ☛ *Note that* ***Save as Report****, generates a public report and also a private* ***Report Template****, which you can access in* ***Private Report Template > My Report Templates*** *folder.*



To create a Report Template from a **Report DataSet**:

1. Access the **Report DataSet** properties (from Report **Studio > Report DataSets**).
2. Display its **Data** page.
3. In the **Report DataSet** section, click **Instant Report** .
4. Select the required instant report type.

   `E.g.:` ***`Breakdown`*** `(pie chart, bar chart, set of gauges, word cloud),` ***`Matrix`*** `(bar chart, radar chart,` ***`Quantitative`*** `(bar chart, line chart, radar chart),` ***`Table`***`.`
5. Click **OK**.
   The instant report is displayed in full page, **Report** section.

6. In the **Report** section, click **More** **> Save As Report Template** .

   ☛ *Note: if you customize the report display, this customization is not taken into account in the Report Template.*

7. In the **Name** field, enter your Report Template name.
8. Click **OK**.
   A public Report Template is created, with its public **Report Table View**. They both have the same name.

   You can access the Report Template:

   - directly from your **Home** page, from **Custom Report Templates** tile, or
   - from the **Public Report Template** folder (**Report Studio > Report Template Definition > Report Templates**)

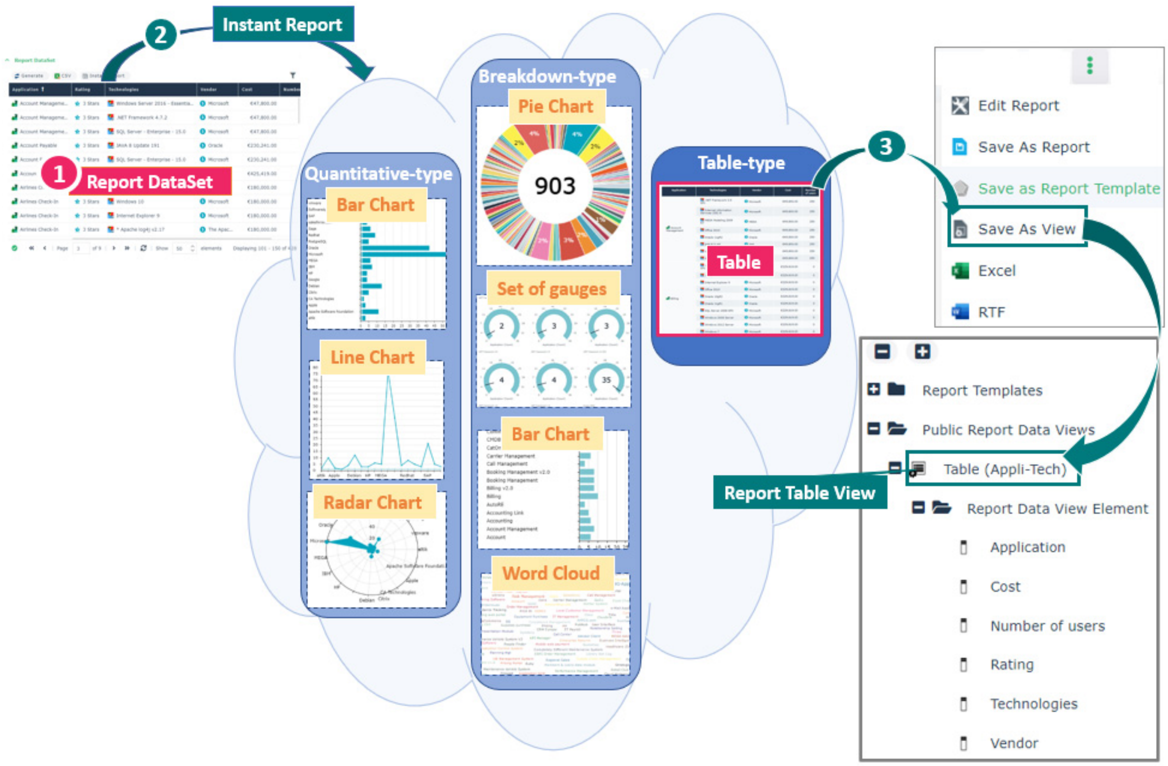
Creating a Report Data View from the Report DataSet

From the **Data** page of the **Report DataSet**, you can launch an Instant Report, then save the report as:

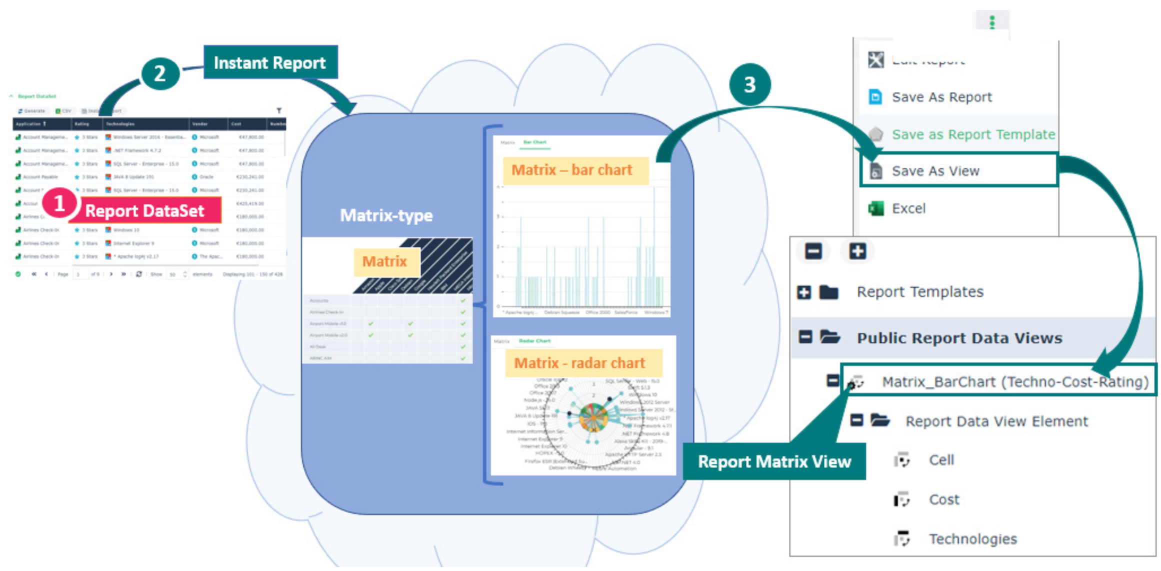
- a Report Template, to get its Report Template
- a Report Data View, to use it in a Report Template

From a **Report DataSet**, you can create a Report Data View:

- a **Report Table View**



- a **Report Matrix View**



To create a Report Data View:

1. Access the **Report DataSet** properties (from **Report Studio > Report DataSets**).
2. Display its **Data** page.
3. In the **Report DataSet** section, click **Instant Report** .
4. Select the required instant report type.

   E.g.: ***Breakdown*** (pie chart, bar chart, set of gauges, word cloud), ***Matrix*** (bar chart, radar chart, ***Quantitative*** (bar chart, line chart, radar chart), ***Table***.
5. Click **OK**.
   The instant report is displayed in full page, **Report** section.
6. (Optional) You can customize the report display.

   ☛ *See Managing Instant Reports.*
7. In the **Report** section, click **More** > **Save as View** .
8. Enter a **Name** to the view.
9. Click **OK**.
   The Report Data View is created and accessible in the **Report Studio > Report Template Definition > Public Report Data Views** folder.

   Any HOPEX user can add the View in a report:
   - use the View to create a Report Template

     ☛ *See Creating a Report Template using a Report Data View.*
   - add the View as a report Chapter in a Report Template

     ☛ *See Adding a Report Chapter to a Report Template.*

GRAPHSET DEFINITION

The following points are covered here:

Introduction to GraphSet Definition

☛ ***GraphSet Definition*** *is only available with* ***HOPEX Power Studio*** *technical module.*

A **GraphSet Definition** creation is performed in **HOPEX** (Web Front-End) for users with **HOPEX Customizer** or **HOPEX Customizer Publisher** profile.

Creation and use of **GraphSet** are available in **HOPEX** (Web Front-End) for all users.

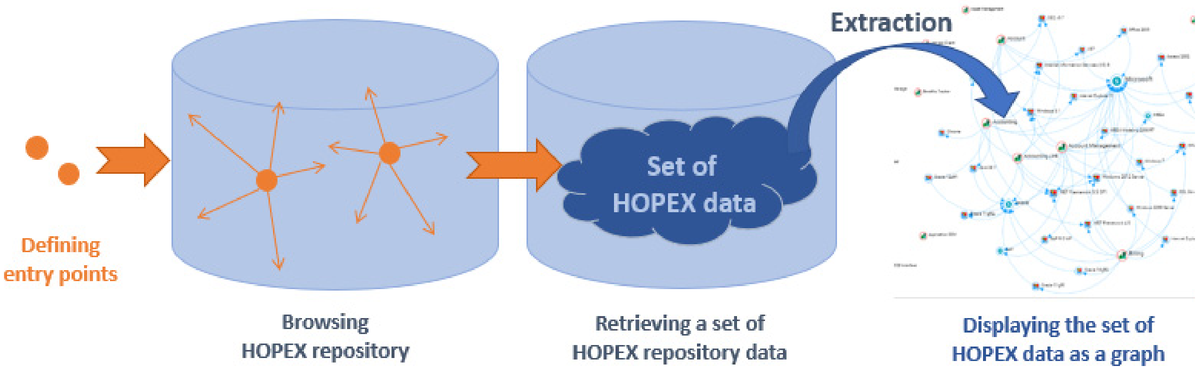
Once a **Report Template** is created from a **GraphSet Definition**, any user of a Solution can use it to query **HOPEX** repository and create **graph** reports.

GraphSet Definition and GraphSet Principle

GraphSet Definition

A **GraphSet Definition** enables to create **GraphSet** reports as follows:

- ***Phase 1***: Extraction of a set of **HOPEX** data.
  This extraction consists in defining entry points, from which the repository is browsed to retrieve a set of **HOPEX** data that will constitute the **GraphSet**.
- ***Phase 2***: Display of this set of **HOPEX** data as a graph.
- ***Phase 3***: Creation of **graph** reports from this display.



This data extraction constitutes the **GraphSet** data, with no style specification.

The data extraction is oriented and comply with the rules defined on node and arc definitions.

GraphSet entry point

A **GraphSet entry point** is an object or an object collection, which are the entry points of the **GraphSet** data extraction.

The **GraphSet** is computed and built from these entry points.

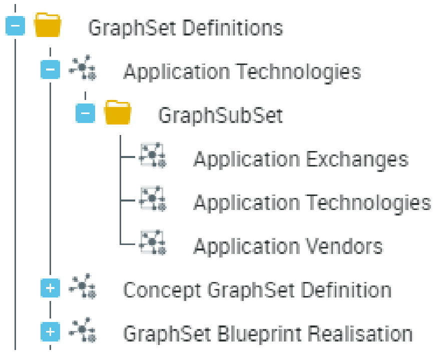
GraphSubSet

A **GraphSubSet** defines a logical set of arc (and/or path) and node definitions.

A **GraphSubSet** can be reused in any **GraphSet Definition**.

A **GraphSet Definition** is made up of one or several **GraphSubSets**.

```
Example:
The "Application Technologies" GraphSet Definition is made up of
"Application Exchanges", "Application Technologies", and "Application
Vendors" GraphSubSets.
```



GraphSet

A **GraphSet** is an instance of a **GraphSet Definition**, defining all parameter values necessary to generate the graph data structure.

A **GraphSet** is made up of nodes and arcs that are grouped together in one or several layers corresponding to the GraphSubSets of its **GraphSet Definition**.

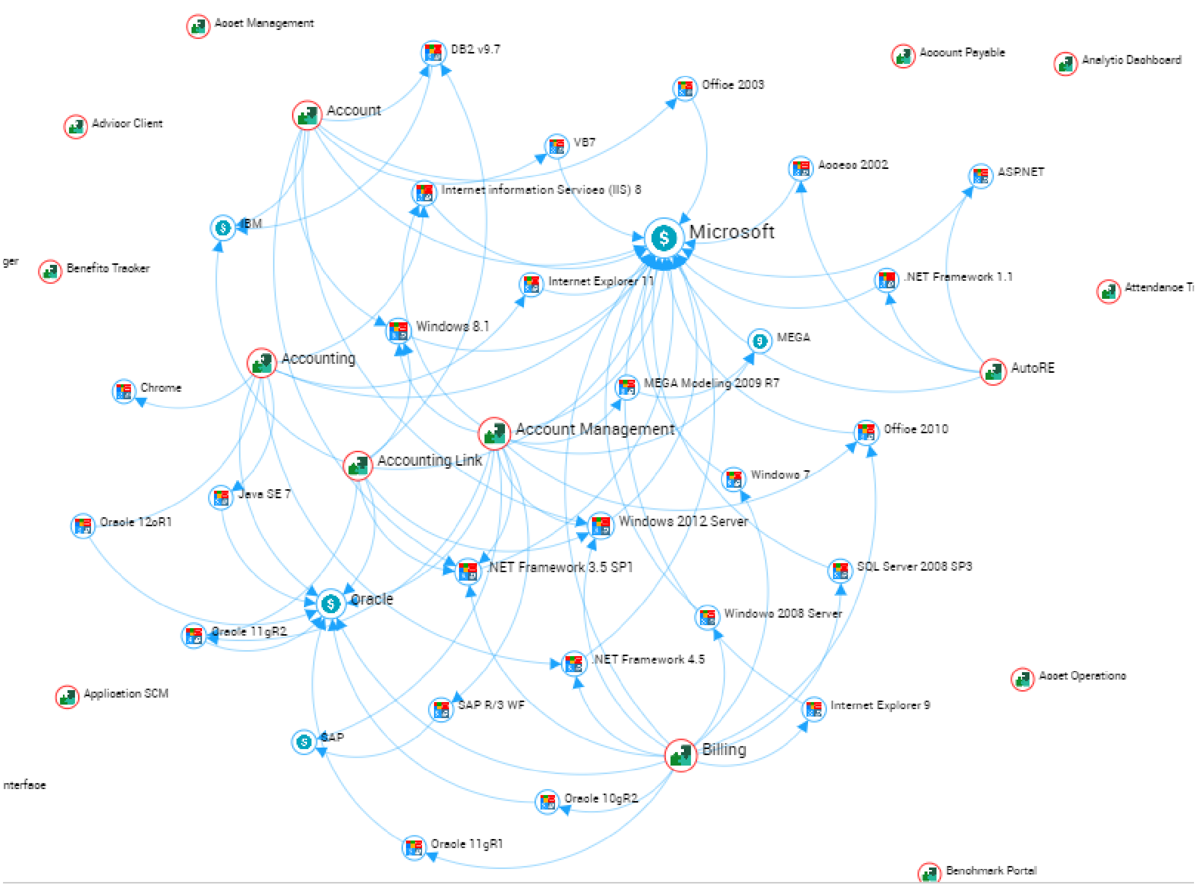
In the **GraphSet Definition** properties, the **Preview** page shows the default display of the **GraphSet** content.

A **GraphSet**, whose **GraphSet Definition** includes several GraphSubSets, shows by default all data of all its layers.

```
Example:
```

The GraphSet whose **GraphSet Definition** is "Application Technologies" is made up of the following layers (corresponding to each GraphSubSet of the **GraphSet Definition**):

- Application Exchanges
- Application Technologies
- Application Vendors

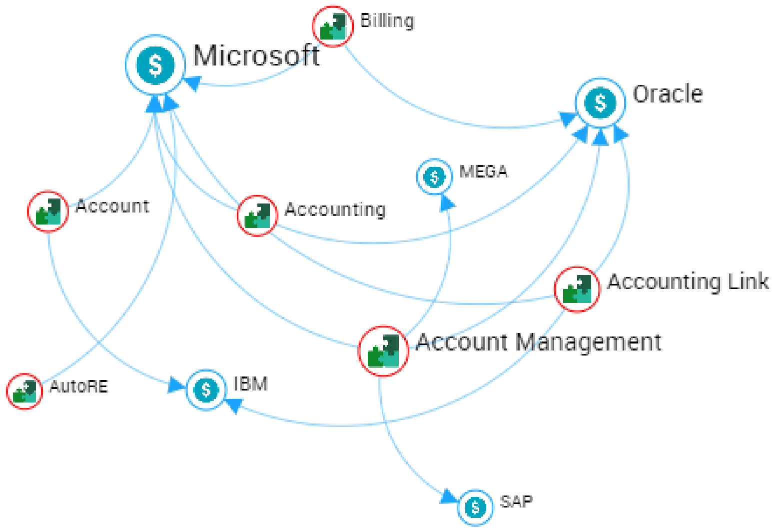


Depending on the amount of nodes and arcs included in the **GraphSet**, its complete display might be too crowed.

To focus on a specific analysis axis, the end-user can hide/display one or several of the **GraphSet** layers.

`Example:`

`For the GraphSet whose` **`GraphSet Definition`** `is "Application Technologies" you can display its "Application Vendors" layer only.`



Node

A node is an HOPEX object based on a MetaClass (concrete or abstract).

`For example:`

 `nodes are Application MetaClass-based nodes`

 `nodes are Technology MetaClass-based nodes`

The node can be included in several layers but is displayed only once in the GraphSet when all of the layers are displayed.

A node can carry fields based on MetaAttributes, TaggedValues, LegAttributes from the corresponding HOPEX object, or computed with a macro.

☛ *See Defining the nodes of the GraphSubSet.*

Arc

An arc links two nodes either based on a MetaAssociationEnd or on a query.

An arc is defined by rules to discover the links between nodes and discover new nodes.

An arc can carry fields based on MetaAttributes, TaggedValues, LegAttributes from the corresponding HOPEX Object, or computed with a macro.

☛ *See Defining the arcs of a node.*

Path

A path is a kind of arc, which includes nodes. The nodes that are part of the path have two arcs only, and are named intermediate nodes.

The path enables to collect fields on intermediate nodes and display them on the arc, or display the details of these intermediate nodes.

If an object is both part of a path and discovered by another arc in the **GraphSet,** then it is displayed twice in the graph.

Intermediate node

An intermediate node is a node, which exists within a path. Intermediate nodes are part of a path, and can be displayed or hidden.

Creating and Defining a GraphSet Definition: the Big Picture

To create and define a **GraphSet Definition**:

1. Define the object type, i.e. the **GraphSet** entry points.
   The **GraphSet** entry point can be:
   - an object,
   - an object collection,
   - a query

   ☛ *See Creating a GraphSet Definition.*
2. Define the GraphSubSets that constitute the **GraphSet**.
   You can:
   - create new GraphSubSets
   - reuse GraphSubSets already created

   ☛ *See Adding a GraphSubSet to a GraphSet Definition.*
3. Define each GraphSubSet:
   - its nodes
   - its arcs and/or paths

   ☛ *See Defining the Data that Feeds the GraphSubSet.*
4. (Optional) Add fields at node and/or arc level.

   ☛ *See Adding Fields to Nodes/Arcs.*

   ☛ *See Adding Fields to Intermediate Nodes.*
5. Preview the **GraphSet**.

   ☛ *See Previewing the GraphSet.*
6. On the **GraphSet**, perform a **Save as Report** to create a **GraphSet** report.
   The **GraphSet Definition** is available for any user to create a **GraphSet.**

   ☛ *See Saving the Graph Report.*
7. Customize the **GraphSet** report.

GraphSet Definition Best Practices

When you create a **GraphSet Definition** you should follow the best practices.

Define a **GraphSet Definition** to create focused **GraphSets**, i.e.:

- The GraphSet should answer a single issue.

  `For examples: a report, an export of specific data.`
- Do not build GraphSets with huge amount of data that you would use for different purposes.

A GraphSet size is limited to:

- 1000 nodes
- 10 000 arcs

☛ *These limits are defined in the following **Options** (**Options > Tools > Documentation > Reports**):*

Fix the limit for GraphSet Nodes

Fix the limit for GraphSet Arcs

A "GraphSet volume alert" warning is generated when a GraphSet exceeds these recommended limits.

Supervising GraphSet Events

In the HOPEX Supervision console, you can check the GraphSet-specific events (e.g.: "GraphSet Generate" informative event):

- **Node Count** indicates the number of nodes in the GraphSet.
- **Internal Node Count Limit** indicates the recommended maximum number of nodes in the GraphSet.
- **Arc Count** indicates the number of arcs in the GraphSet.
- **Internal Arc Count Limit** indicates the recommended maximum number of arcs in the GraphSet.

For detailed information regarding GraphSet related supervision events, see **HOPEX Administration > Technical Articles > Supervision Event Description > ReportDataSet, TreeSet, and GraphSet**.

GRAPHSET DEFINITION CREATION

The **GraphSet Definition** creation includes:

> ☛ *To customize the **GraphSet**, see Previewing the GraphSet.*

Accessing the GraphSet Definitions

You can access :

- all the GraphSet Definitions
- the custom GraphSet Definitions only
  Custom GraphSet Definitions are the GraphSet Definitions not provided by **MEGA**.

Accessing the GraphSet Definitions

To access the GraphSet Definitions:

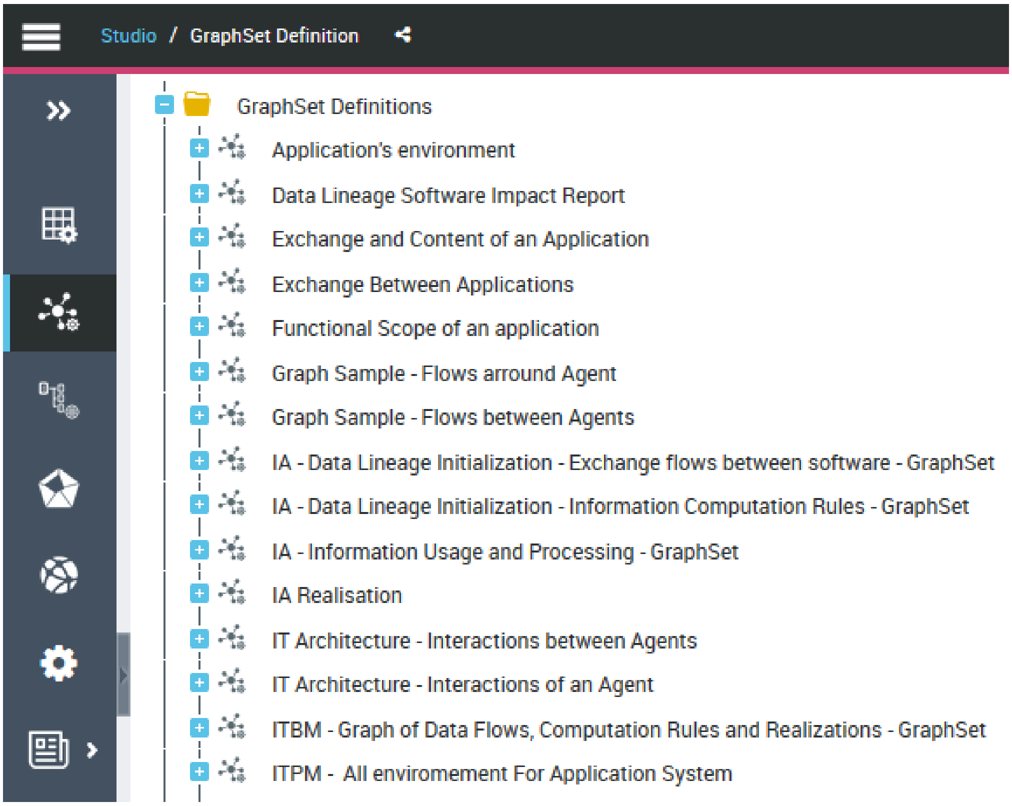
1. Connect to **HOPEX Studio** desktop.

   > ☛ *See Connecting to HOPEX Studio Desktop.*

2. Click the navigation menu and select **Report Studio > GraphSet Definition** .

   In the Edit area, the **Browse** view displays the **GraphSet Definitions** tree.

3. Expand the **GraphSet Definitions** folder.
   All the GraphSet Definitions available are accessible.



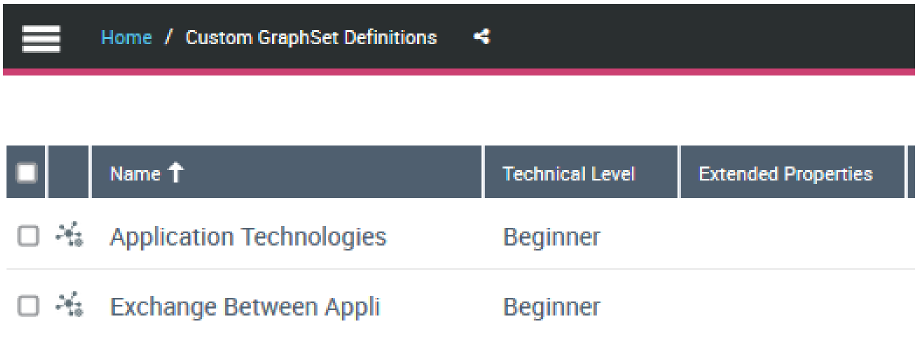
Accessing the custom GraphSet Definitions (Web Front-End)

To access the custom GraphSet Definitions:

1. Connect to **HOPEX Studio** desktop.

   ☛ *See Connecting to HOPEX Studio Desktop.*

2. In the **Home** page, click the **Custom GraphSet Definitions** tile.
   In the Edit area, the **Browse** view lists the custom GraphSet Definitions only.



Accessing the GraphSet Definitions from HOPEX (Windows Front-End)

To access the GraphSet Definitions from HOPEX (Windows Front-End):

1. Connect to **HOPEX** with HOPEX Customizer profile.
2. In **HOPEX** menu bar, select **View > Navigation Windows > Utilities**.
3. In the repository tree, expand **GraphSet Definition** folder.
   All the GraphSet Definitions available are accessible.

   ☛ *Right-click the GraphSet Definition and select **Properties** to access the GraphSet Definition properties.*

Displaying the GraphSet Definition properties

To display the GraphSet Definition properties:

1. Access the GraphSet Definition.

   ☛ *See Accessing the GraphSet Definitions.*

2. Select the GraphSet Definition.
   The GraphSet Definition properties are displayed in the **Multi-view** window.

Displaying the GraphSubSet properties

To display the GraphSubSet properties:

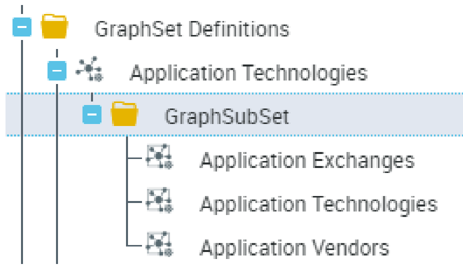
1. Access the GraphSet Definitions.

   ☛ *See Accessing the GraphSet Definitions.*

   In the Edit area, the **Browse** view displays the following folders:

- the **GraphSet Definitions**, which are made up of GraphSubSets
- the **GraphSubSets Without Definition**, which are GraphSubSets that are not included in any GraphSet Definition.

2. Expand the **GraphSet Definitions** folder, then the GraphSet Definition concerned and its **GraphSubSet** folder.
   The GraphSubSet included in the GraphSet Definition available are accessible.

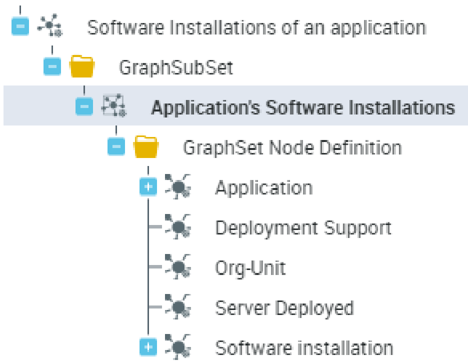


3. Display the **Properties** window .
4. Select the GraphSubSet concerned.
   The GraphSubSet properties are displayed in the **Properties** window.

Displaying the GraphSet arc/node properties

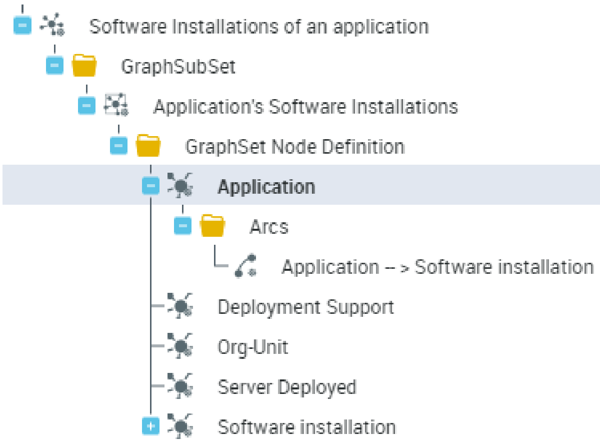
To display the GraphSet arc/node properties:

1. Access the GraphSubSet concerned.
   ☛ *See Displaying the GraphSubSet properties.*
2. Expand the **GraphSet Node Definition** folder.



3. Roll the mouse over the node concerned and in its row click **Properties**.
   Its properties are displayed in the **Properties** window.

4. Expand a node with a to access its arc(s).



5. Roll the mouse over the arc concerned and in its row click **Properties**.
   Its properties are displayed in the **Properties** window.

Creating a GraphSet Definition

A **GraphSet Definition** is MetaClass-specific.

The **GraphSet Definition** is based either on:

- an object or an object collection (an occurrence or a set of occurrences of the source MetaClass)
  The **GraphSet Definition** creation consists in connecting a source MetaClass (concrete or abstract) to the **GraphSet Definition**.
- a query (a set of occurrences)
  The **GraphSet Definition** creation consists in defining the query that collects the entry points.
  If needed, you can create a query parameter for this purpose.

Creating an object collection-based GraphSet Definition

When you create a GraphSet Definition based on an object or an object collection, you need to define a Source MetaClass.

In that case the following items are automatically created:

- a Graphset Entry Point Collection (collection type parameter) of the same type as the source MetaClass.
- a GraphSubSet with the same name as the GraphSet Definition
  The GraphSubSet definition is automatically initialized with a node with the same MetaClass as the source MetaClass. This node is defined as entry point and its name is the MetaClass name.

To create a GraphSet Definition based on an object:

1. Access the **GraphSet Definitions** folder

   ☛ *See Accessing the GraphSet Definitions.*

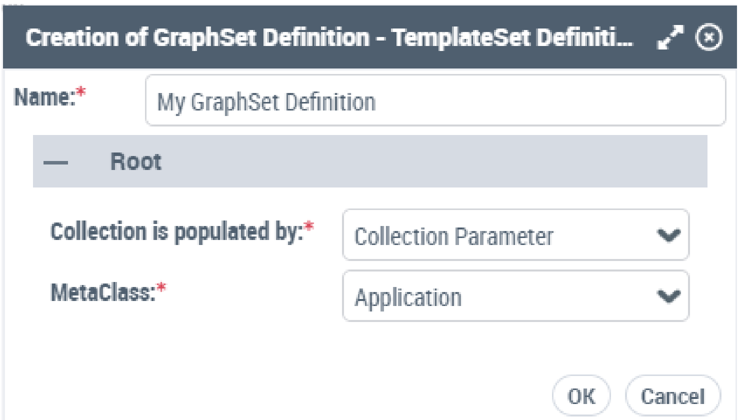
2. Right-click the **GraphSet Definitions** folder and select **New > GraphSet Definition**.
   The **Creation of GraphSet Definition** window appears
3. In the **Name** field, enter your GraphSet Definition name.

   ☛ *By default the GraphSet Definition name is: GraphSet Definition-x (x is a number).*

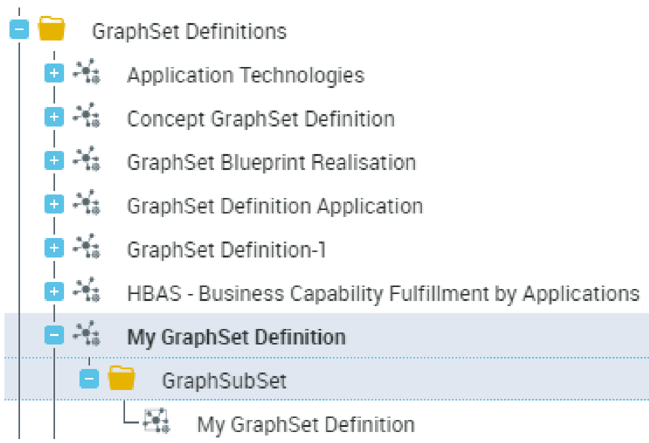
4. Define the **Root** section:
   - In the **Collection is populated by** field keep "Collection Parameter".
   - In the **MetaClass** field, click the arrow and in the drop-down menu select the MetaClass you want to define as the source MetaClass of your GraphSet Definition.

     ☺ *In the field, enter the first letters of the MetaClass for a quick access.*

   `Example: Application.`



5. Click **OK**.
   The selected MetaClass is defined as your GraphSet Definition source MetaClass.
   Your GraphSet Definition is created and added in the **GraphSet Definitions** folder.
6. In the **GraphSet Definitions** folder, expand your GraphSet Definition.
   A GraphSubSet is automatically created with the same name as your GraphSet Definition.



7. In your GraphSet Definition properties, display its **Characteristics** page.
   In the **Parameters** section a Collection type parameter of the same type as the source MetaClass is automatically created with value "N" as multiplicity.

   `Example: with "Application" as source MetaClass, the Collection Type is "Application" MetaClass.`
8. If you do not want a collection of objects but a single object only as entry point, in the **Collection Multiplicity** field, select "1".
9. In the **_GUIName** field, enter a name.

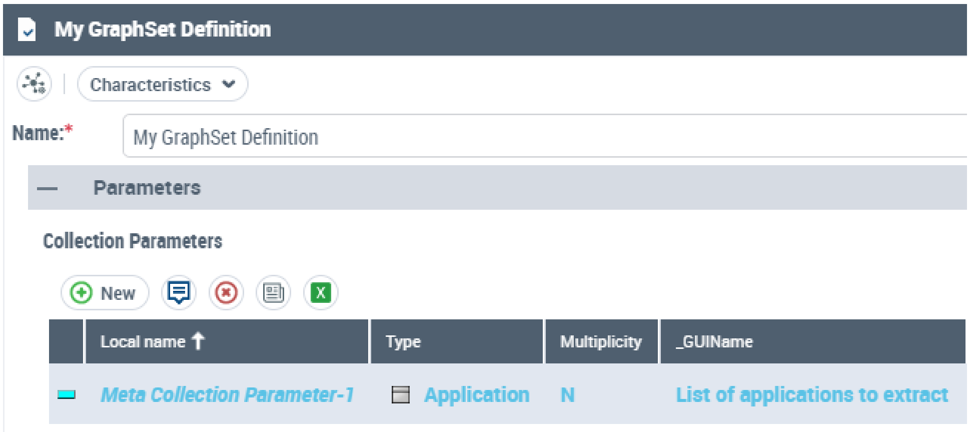
   `Example: "List of applications to extract".`

   This parameter is displayed in the GraphSet Definition **Preview** page.

   ☛ *See Previewing the GraphSet.*

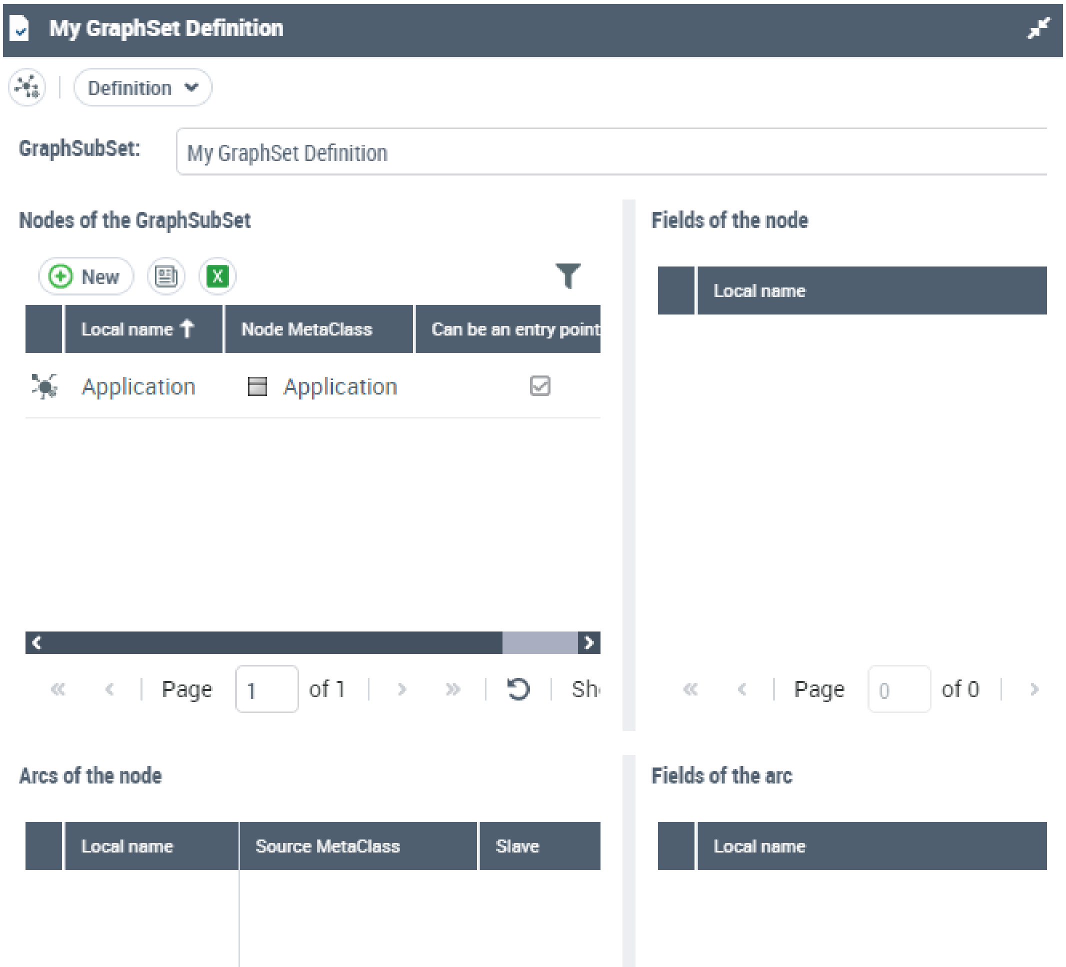
   You may need to add other entry point collections.

   ☛ *See Adding an entry point collection to the GraphSet Definition.*



10. In your GraphSet Definition properties, display its **Definition** page.
The **GraphSubSet**, automatically created, already includes a Node, defined as entry point.

```
Example: with "Application" as source MetaClass, the node "Application" is created.
```



11. Define the GraphSubSet (its nodes and arcs and their fields).

☛ *See Defining the Data that Feeds the GraphSubSet.*

Creating a query-based GraphSet Definition

When you create a **GraphSet Definition** based on a query, you need to define:

- its root MetaClass
- its populating query based on the root MetaClass selected

In the GraphSet Definition you can also define parameters that are used in any query used in the GraphSet Definition.

☛ *See Defining query parameters.*

To create a query-based GraphSet Definition:

1. Access the **GraphSet Definition** folder

   ☛ *See Accessing the GraphSet Definitions.*

2. Right-click the **GraphSet Definitions** folder and select **New > GraphSet Definition**.
   The **Creation of GraphSet Definition** window appears
3. In the **Name** field, enter your GraphSet Definition name.

   `Example: "Appli from Portfolio"`

   ☛ *By default the GraphSet Definition name is: GraphSet Definition-x (x is a number).*

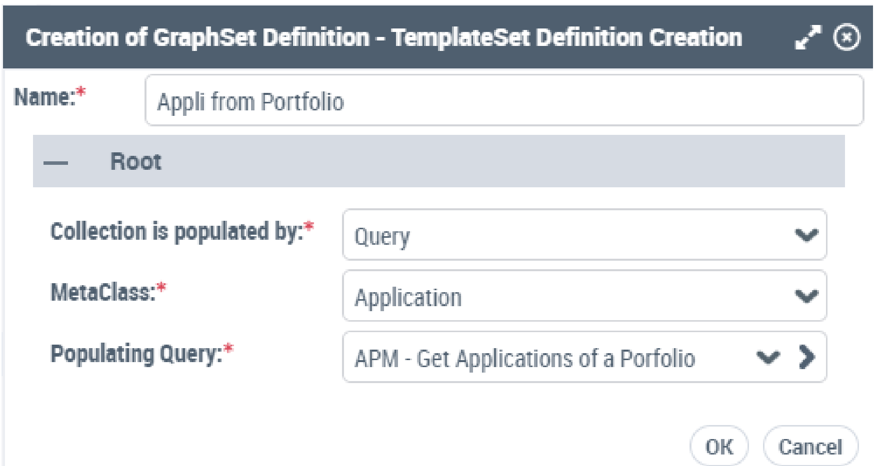
4. In the **Root** section:
   - In the **Collection is populated by** field, select "Query".
   The **Populating Query** field appears.
   - In the **MetaClass** field, click the arrow and in the drop-down menu select the MetaClass you want to define as the root MetaClass of your GraphSet Definition.

     ☺ *In the field, enter the first letters of the MetaClass for a quick access.*

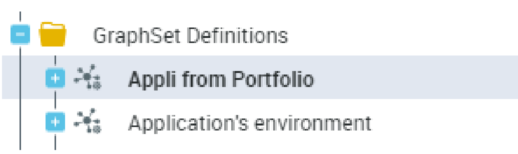
     `Example: Application.`
   - In the **Populating Query** field, use the drop-down menu to select the query used to collect the entry point.

     ☛ *Only the root MetaClass related queries are available.*

     `Example: "APM - Get Applications of a Portfolio".`



5. Click **OK**.
   You report GraphSet Definition is created and added to the **GraphSet Definition** folder.

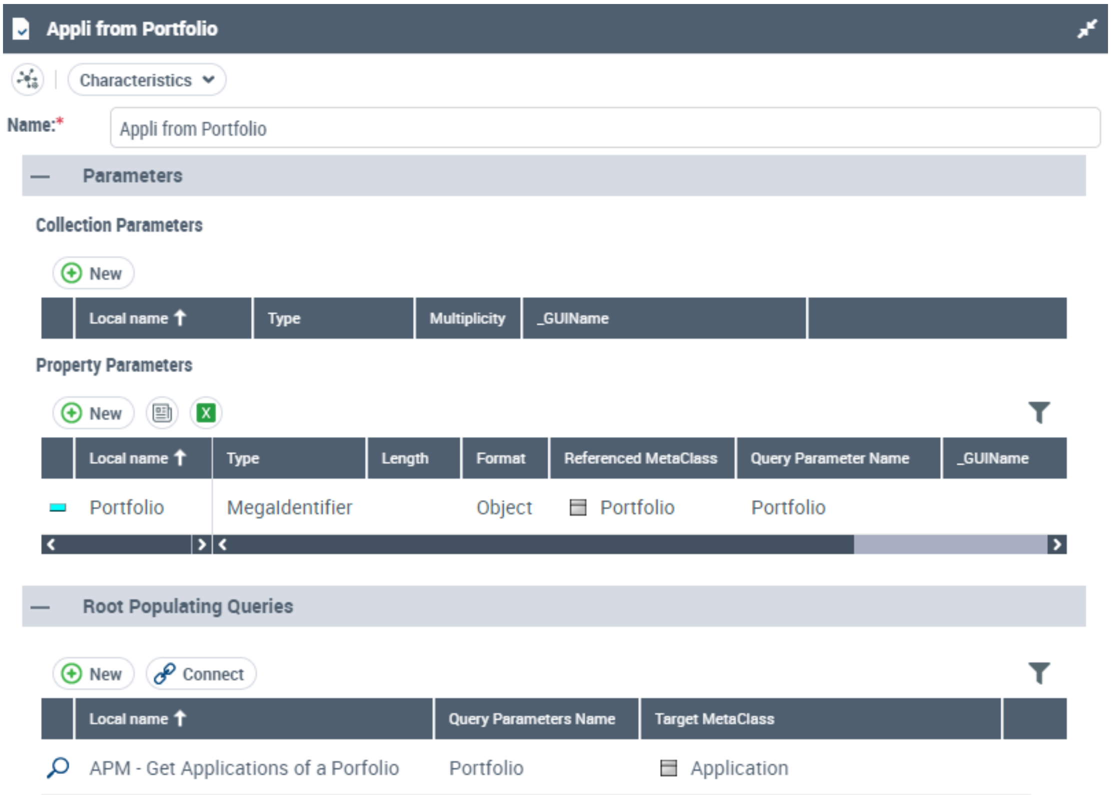


6. In your GraphSet Definition properties, display its **Characteristics** page:

- The **Root Population Queries** section displays the query used to populate the GraphSet.
- The **Parameters** section displays the **Property Parameter** automatically created when used in the populating query.

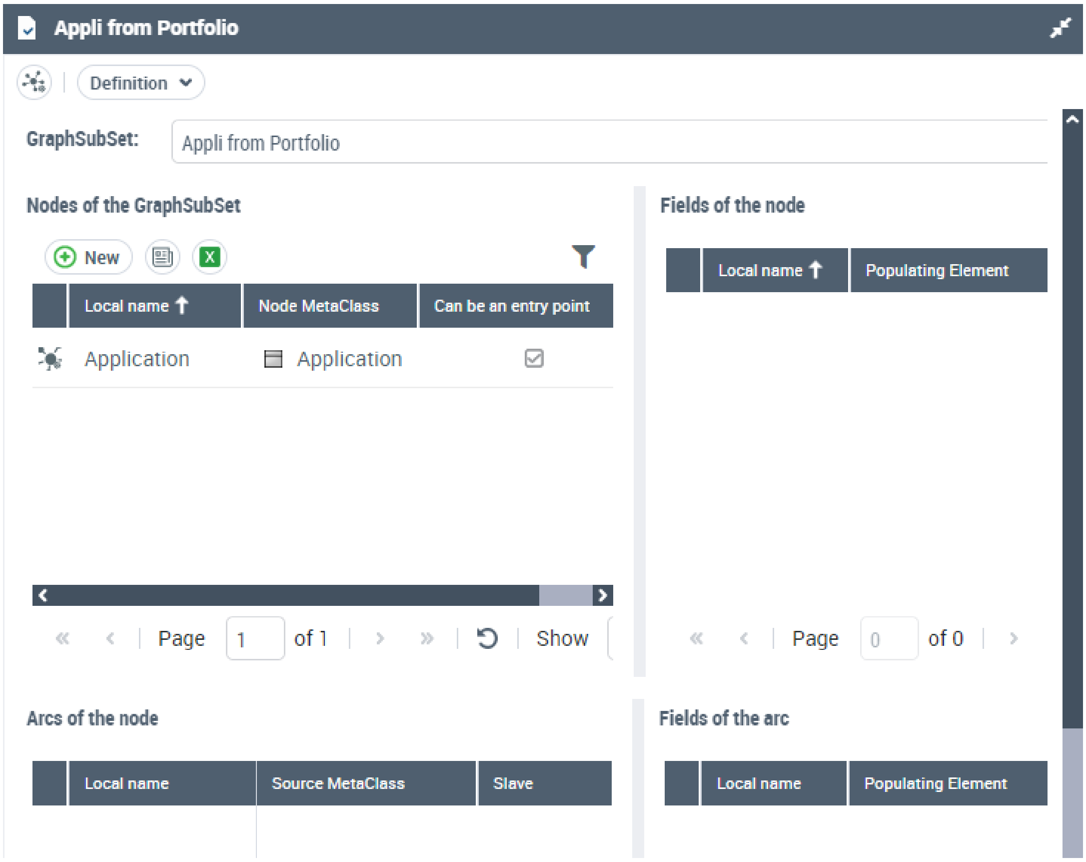
  `Eg.: here Portfolio.`

  `Select Application Where Initiative.Owner Portfolio =` ***`&Portfolio`*** `Or Deployed Object:Software Installation.Initiative.Owner Portfolio =` ***`&Portfolio`***



7. In your GraphSet Definition properties, display its **Definition** page.
The **GraphSubSet**, automatically created, already includes a Node, defined as entry point.

    Example: with "Application" as source **MetaClass**, the node "Application" is created.

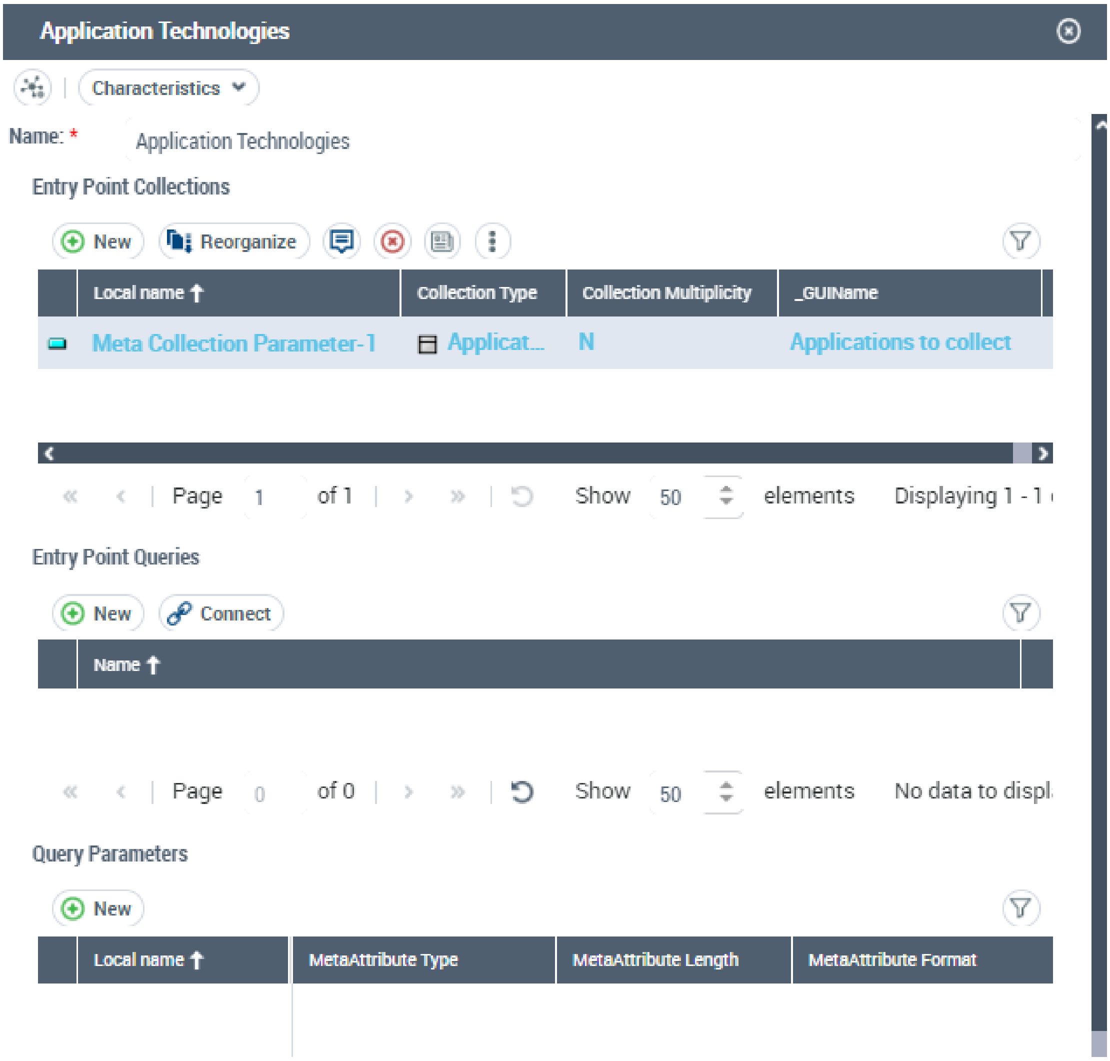


8. Define the GraphSubSet (its nodes and arcs and their fields).
   ☛ *See Defining the Data that Feeds the GraphSubSet.*
9. Display the GraphSet Definition **Preview** page and test the report.
   ☛ *See Previewing the GraphSet.*

Adding GraphSet Definition Entry Points

In the GraphSet Definition properties, the **Characteristics** page enables to:

- define how to collect the GraphSet entry points, via:
  - collections
  - queries
- create parameters that you can use in any query used in the GraphSet Definition (for example in GraphSet Definition entry point queries or in GraphSet Arc Definition queries).



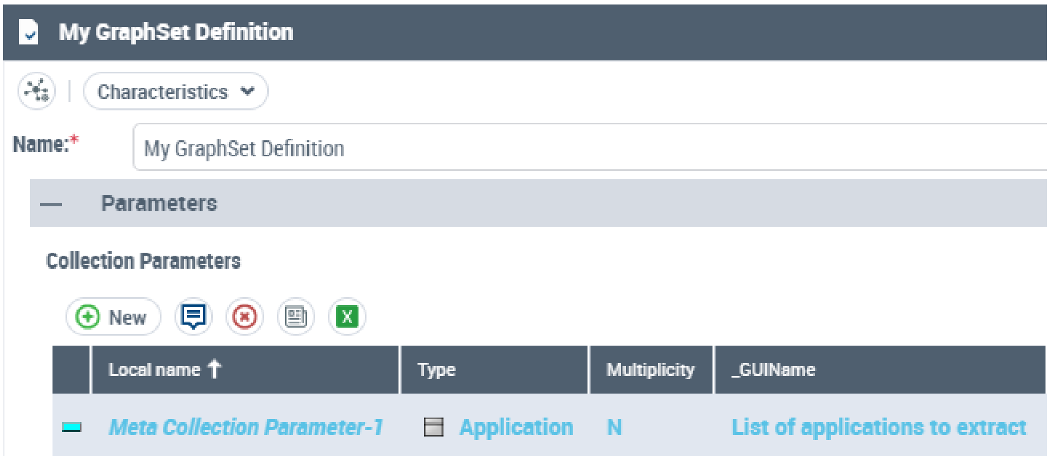
Adding an entry point collection to the GraphSet Definition

By default, an **Entry Point Collection** is made up by a collection of objects (**Collection Multiplicity** value is "N"). If needed, you can define the **Entry Point Collection** as a single object (**Collection Multiplicity** value is "1").

GraphSet Definitions with source MetaClass defined include by default an entry point collection based on the source MetaClass.

You may need to add entry point collection(s) to your GraphSet Definition (whether object-based or query-based).

At GraphSet creation the user is asked to enter the **Entry Point Collection** values, which are taken into account to build the GraphSet.



To feed the GraphSet Definition with an entry point collection:

1. Access the GraphSet Definition properties.

   ☛ *See Accessing the GraphSet Definitions.*

2. Display its **Characteristics** page.
3. In the **Entry Point Collections** section, click **New** .

   ☛ *You can add as many entry point collections as needed.*

   The **Meta Collection Parameter** creation window is displayed.
4. (Optional) In the **Local Name** field, enter a name for the collection parameter.
5. (if needed) Use the drop-down lists to modify the default Meta Collection Parameter characteristics:
   - **MetaAttribute Type**

     `Examples: "X" (String, default value), "D" (DateTime).`
   - **MetaAttribute Length**

     `Not defined by default.`
   - **MetaAttribute Format**

     `Examples: "S" (Standard, default value), "D" (Duration).`

   ☛ *For detailed information regarding MetaAttribute characteristics, see HOPEX Studio > Customizing the Metamodel : Managing the Metamodel : MetaAttributes : MetaAttribute Characteristics documentattion.*

6. Click **OK**.
   The entry point collection is added in the **Entry Point Collections** list.
7. Click its **Collection Type** field and select **Connect MetaClass**.

   `Example: Software Technology MetaClass.`
8. Use the **Connecting** window to select the MetaClass, and click **Connect.**
9. (If you want to define the entry point as a unique object) In its **Collection Multiplicity** drop-down list, select "1".
10. In the **_GUIName** field, enter a name.

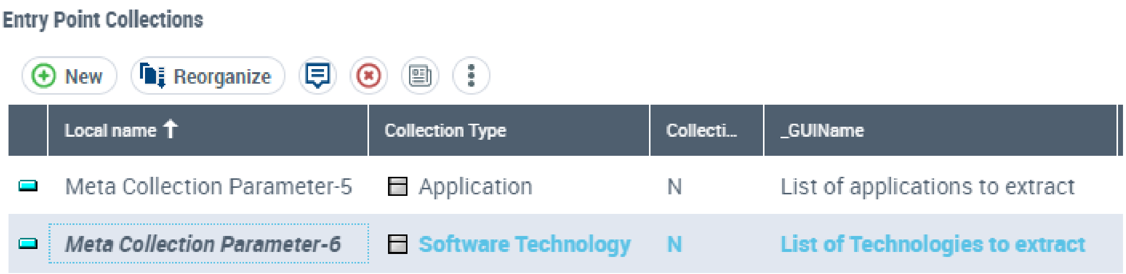
    `Example: List of Technologies to extract.`

    This parameter is displayed in the GraphSet Definition **Preview** page.

    ☛ *See Previewing the GraphSet.*

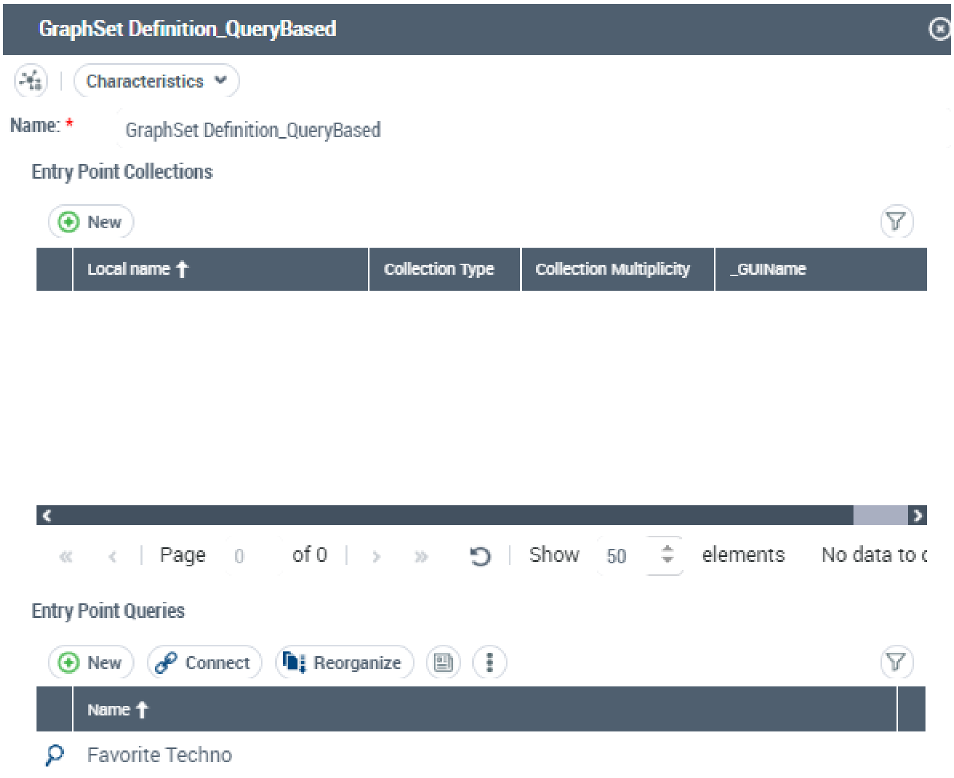
Example:

```
You can add the "List of Technologies to extract" Collection Parameter
with the "Software technology" Collection Type, so that at GraphSet
creation the user is asked to select the Technologies to feed the
GraphSet with this collection.
```



Adding an entry point query to the GraphSet Definition

Your query is already created. If needed, you can create query parameters.



To connect a query to a GraphSet Definition:

1. Access the GraphSet Definition properties.

   ☛ *See Accessing the GraphSet Definitions.*

2. Display its **Characteristics** page.
3. In the **Entry Point Queries** list click:
   - **Connect** to connect an existing query.

     `Example: connect the "ITPM - My applications" query to filter the Application collection to the applications owned by the end-user only.`

   - **New** to create a query.

     `Example: Select [Software Technology] where [Folder of Favorites).Absolute Identifier="<My favorite's IdAbs>" to filter the Technologies to your favorite ones.`

     ☛ *You can add as many queries as needed.*

Defining query parameters

In your GraphSet Definition, you can create and define **Query Parameters** that you can use in any query used in the GraphSet Definition.

```
Examples:
- in GraphSet Definition entry point queries, see Creating a query-based GraphSet Definition.
- in GraphSet Arc Definition queries, see Defining the arcs of a node.
```

At GraphSet creation the user is asked to enter the parameter values, which are taken into account in the query (for example to retrieve the entry points to build the GraphSet).

If no query parameters is specified the GraphSet operates on the entire repository occurrences.

To add query parameter in the GraphSet Definition:

1. Access the GraphSet Definition properties.

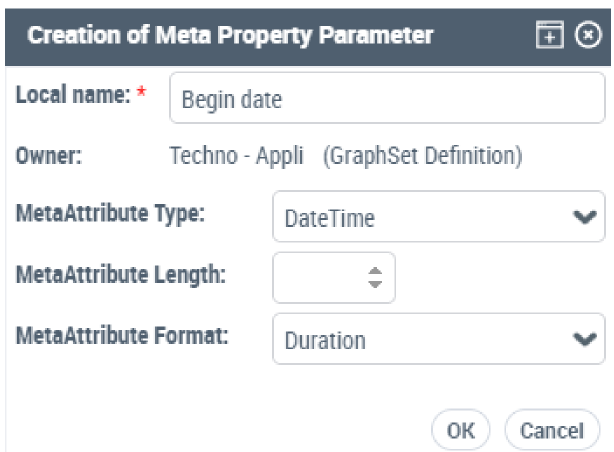
   ☛ *See Displaying the GraphSet Definition properties.*
2. Display the **Characteristics** page.
3. In the **Query Parameter** list, click **New** +.

   ☛ *You can add as many parameters as needed.*
4. In the **Meta Property Parameter** creation window:
   - enter a **Local Name** (name displayed in the GraphSet)

     `Example: Begin date`
   - (according to your needs) modify the default values for **MetaAttribute Type** ("String"), **MetaAttribute Length** and **MetaAttribute Format** ("Standard")

     `Example: MetaAttribute type "DateTime", MetaAttribute Format "Duration"`

     ☛ *For detailed information regarding MetaAttribute characteristics, see HOPEX Studio > Customizing the Metamodel : Managing the Metamodel : MetaAttributes : MetaAttribute Characteristics documentattion.*

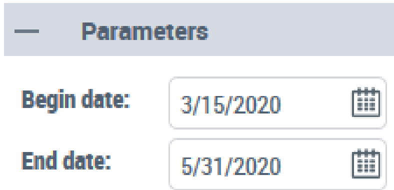


5. Click **OK**.
   The query parameter is added in the **Query Parameters** list.

6. In the **Parameter Name** field, enter a name for the parameter. This name is the one used in the query.

Example:

```
When you add "Begin Date" and "End Date" Query Parameters, at GraphSet
creation the user is asked to enter both dates, which filter the GraphSet
result according to the these dates.
```



Adding a GraphSubSet to a GraphSet Definition

With object-based GraphSet Definitions a GraphSubSet is automatically created (you still need to define it). You may need to add one or several GraphSubSets to the GraphSet Definition.

A GraphSubSet can be reused in GraphSet Definitions, so that you can:

- reuse an existing GraphSubSet
- add a new GraphSubSet
- remove a GraphSubSet

Reusing a GraphSubSet in a GraphSet Definition

With object-based GraphSet Definitions a GraphSubSet is automatically created. you may need to add one or several GraphSubSets to the GraphSet Definition.

To add an existing GraphSubSet to your GraphSet Definition:

1. Access the GraphSet Definition properties.

   ☛ *See Displaying the GraphSet Definition properties.*

2. Display its **Definition** page.
3. In the **GraphSubSet** field, click the right-oriented arrow and select **Connect** .
4. Use the **Connecting** window to select the GraphSubSet.

   ☛ *You can select as many* GraphSubSets as needed.

5. Click **Connect**.
   The GraphSubSet is added to the GraphSet Definition.

Adding a new GraphSubSet to a GraphSet Definition

To add a new GraphSubSet to a GraphSet Definition:

1. Access the GraphSet Definition properties.

   ☛ *See Displaying the GraphSet Definition properties.*

2. Display its **Definition** page.

3. In the **GraphSubSet** field, click the right-oriented arrow and select **New** to create a GraphSubSet.
4. Enter a **Name** to the GraphSubSet.
5. Click **OK**.
   The GraphSubSet is added to the GraphSet Definition.
6. Define the GraphSubSet.
   ☛ *See Defining the Data that Feeds the GraphSubSet.*

Removing a GraphSubSet from a GraphSet Definition

To remove a GraphSubSet from a GraphSet Definition:

1. Access the GraphSet Definition folders.
   ☛ *See Accessing the GraphSet Definitions.*
2. Expand the GraphSet Definition concerned.
3. Expand the **GraphSubSet** folder.
4. Right-click the GraphSubSet you want to remove from the GraphSet Definition and select **Remove** .
   A confirmation dialog box prompts you to confirm your choice to remove the GraphSubSet from the GraphSet Definition.
   ☛ *if you want to delete the GraphSubSet from the HOPEX repository, select* ***Delete*** *and confirm your choice in the next dialog box.*
5. Click **Remove**.
   The GraphSubSet is removed from the GraphSet Definition, but still available in the HOPEX repository.
   If the GraphSubSet is not used in another GraphSet Definition it is added in the **GraphSubSets Without Definition** folder.

Defining the Data that Feeds the GraphSubSet

The GraphSet Definition is made up of at least one GraphSubSet. You have to define how to populate each of these GraphSubSets.

A GraphSubSet is made up of nodes, for each of which you can define arcs and/or paths.

See GraphSubSet.

According to your needs, you can also add fields on:

- nodes
- arcs
- nodes of paths (intermediate nodes)

By default, these fields are hidden in the GraphSet, but displayed as tooltips when the end-user roll the mouse over each of them.

☛ *See Adding Fields to Nodes/Arcs and Adding Fields to Intermediate Nodes.*

A **GraphSubSet** can be reused in any other GraphSet Definition.

Defining the nodes of the GraphSubSet

The GraphSubSet must include at least one node.

See the node definition, section Node.

For each of the nodes, you must define:

- its name
- the MetaClass on which it is based
- whether it **Can be an entry point** or not
  A Node that can be an entry point is discovered from parameters or queries. The **Can be an entry point** parameter enables to instantiate nodes that are not discovered from another one.

To define the nodes of the GraphSubSet:

1. Access the GraphSet Definition properties.

   ☛ *See Displaying the GraphSet Definition properties.*
2. Display its **Definition** page.
3. In the **GraphSubSet** drop-down list, select the GraphSubSet you want to define.
4. In the **Nodes of the GraphSubSet** section, click **New** ➕.
5. In the **Local name** field, enter a name for the node.

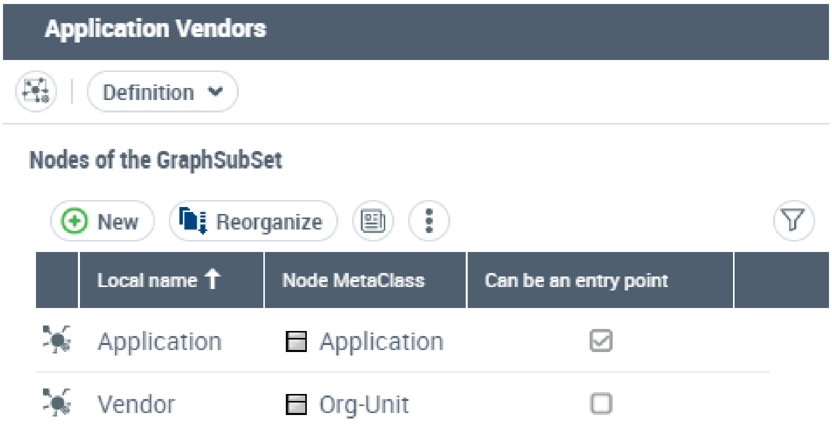
   `Example: Vendor`
6. In the **Node MetaClass** drop-down list, select the MetaClass (concrete or abstract) on which is based the node.

   `Example: Org-Unit`
7. (If you do not want these nodes to be used as entry points in the GraphSet) Clear the **Can be an entry point** option.

   ☛ *By default the node is an entry point.*
8. Click **OK**.

9. (If needed) Repeat steps 4 to 8 to add other nodes.

```
For example, the "Application Vendors" GraphSubSet includes:
- the "Application" node based on the Application MetaClass
- the "Vendor" node based on the Org-Unit MetaClass.
Only the Application node is defined as entry point.
```



Defining the arcs of a node

In the GraphSubSet, you must define at least one arc (or path, see Defining the paths) for one of the nodes, otherwise the GraphSet corresponding layer would display nodes only.

See the arc definition, section Arc.

An arc links two nodes, which can be from:

- different sources

  ```
  Example: an arc can link the technologies used by an application.
  ```

- similar source

  ```
  Example: an arc can link the applications that are connected via a
  message flow.
  ```

The arc provider can be based on:

- a MetaAssociationEnd, or
- a query

The arc is defined through the following attributes:

- **Behavior** to define the propagation behavior:
  - ***Deep*** (default): the opposite node is added in the graph and the exploration goes on.
  - ***Standard***: the opposite node is added in the graph but the exploration stops.
  - ***Link***: the arc is added only if the opposite node exists in the graph.
  - ***Computed***: the propagation is managed through a macro and can take "Deep", "Standard", or "Link" values.

    ☛ *For information regarding **Behavior** values, see Propagation.*
- **Condition** enables to consider an arc when it starts from an entry point node only or not.

  `By defaut: "No condition".`

To define the arcs of a node:

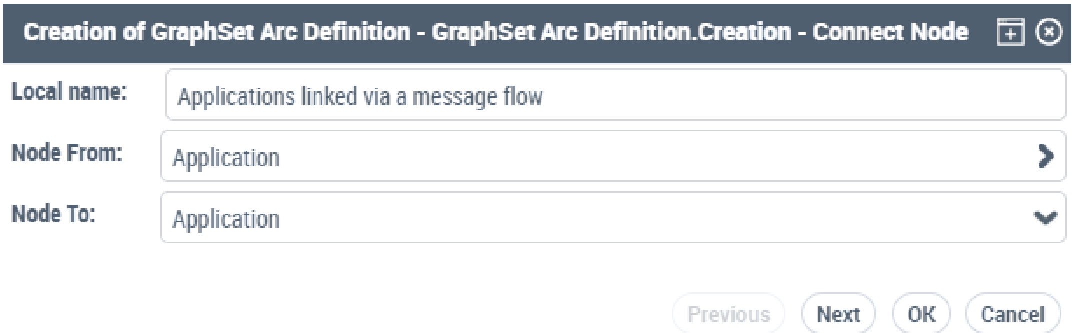
1. Access the GraphSet Definition properties.

   ☛ *See Displaying the GraphSet Definition properties.*
2. Display its **Definition** page.
3. In the **GraphSubSet** drop-down list, select the GraphSubSet concerned.

   ☛ *Else directly access the GraphSubSet properties, see Displaying the GraphSubSet properties.*
4. In the **Nodes of the GraphSubSet** pane, select the row of the node from which you want to define the arcs.
5. In the **Arcs of the node** pane, click **New** +.
6. In the **Local Name** field, enter a name
   The **Node From** field is already defined with the node selected.
7. In the **Node To** field, use the drop-down list to select the node at the opposite side of the arc.

   ☛ *The drop-down list lists the nodes included in the **Nodes of the GraphSubSet** section.*

   ☛ ***Node From** and **Node to** can be of different or similar source.*



8. Click **Next**.
9. (If your arc provider is query-based) Select **Query**.
   By default the arc provider is based on a **MetaAssociationEnd**.
10. Depending on the arc provider:
    - in the **MetaAssociationEnd** drop-down list, select the MetaAssociationEnd that provides the arc.
    - in the **Query** drop-down list, select the query that provides the arc.

11. Click **Next**.
12. Define the arc attributes:
    - **Behavior**

      `By defaut: "Deep".`
    - **Condition**

      `By defaut: "No condition".`
13. Click **OK**.
    The defined arc is added in the **Arcs of the node** section.

Defining the paths

In the GraphSubSet, you must define at least one path (or arc, see Defining the arcs of a node) between two nodes, otherwise the GraphSet corresponding layer would display nodes only.

See the path definition, section Path.

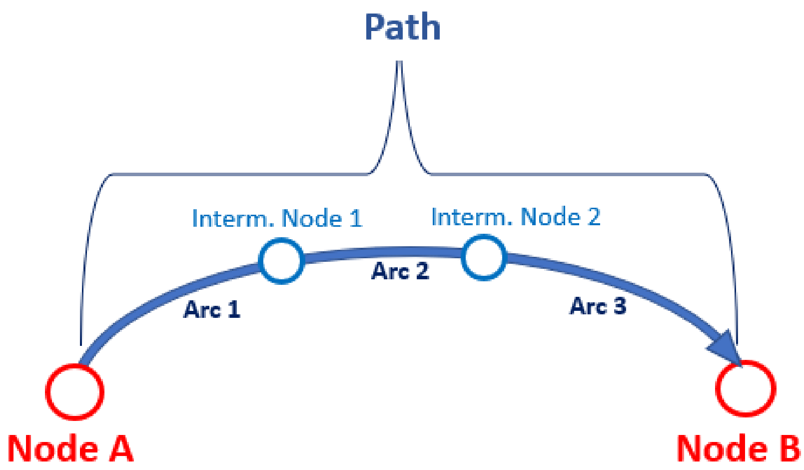
A path links two nodes through arcs and intermediate nodes.

The path definition includes the following phases:

- ***Phase 1***
  Definition of its extremity nodes.
- ***Phase 2***
  Definition of its intermediate nodes.
- ***Phase 3***
  Definition of each arc, which is part of the path, starting from the starting node of the path to the opposite node of the path.

  ```
  For example:
  1: Definition of Arc 1, starting from Node A to Intermediate Node 1.
  2: Definition of Arc 2, from Intermediate Node 1 to Intermediate Node 2.
  3: Definition of Arc 3, from Intermediate Node 2 to Node B.
  ```

  Each arc provider can be based on:
  - a MetaAssociationEnd, or
  - a query



- ***Phase 4***: Definition of the fields on the intermediate nodes of the path.

To define the paths:

1. Access the GraphSet Definition properties.

   ☛ *See Displaying the GraphSet Definition properties.*
2. Display its **Paths** page.
3. In the **GraphSubSet** drop-down list, select the GraphSubSet concerned.

   ☛ *Else directly access the GraphSubSet properties, see Displaying the GraphSubSet properties.*
4. In the **Paths** pane, click **Add** ➕.
   The GraphSet Path creation wizard: **Path Definition** appears.
5. In the **Name** field, enter the name of your path.

   `For example: Path_ReceivedFlows.`

6. In the **Path From** field, use the drop-down list to select the node from which you want to start the path.

   ☛ *The drop-down list lists the nodes included in the **Nodes of the GraphSubSet** section.*

7. In the **Path To** field, use the drop-down list to select the node at the opposite end of the path.

   ☛ *The drop-down list lists the nodes included in the **Nodes of the GraphSubSet** section.*

   ☛ ***Path From** and **Path to** can be of different or similar source.*

8. In the **GraphSet Path Intermediate Nodes**, click **Add Node** as many times as you need intermediate nodes in your path.
   Intermediate nodes are created with default names.

   ```
   For example, if you click twice: "InterNode1" and "InternNode2".
   ```

9. (Optional) In the **Name** fields, modify the intermediate node default names.
10. For each intermediate node, click its **Node MetaClass** field and use the drop-down list to select its object type.

11. (If needed) Modify the intermediate node order: click **Reorganize** and drag and drop the intermediate nodes to get the order required.

    ☛ *Else you can use the **Alphabetical order**.*

12. Click **Next**.
    The first arc definition window is displayed, with both of its two nodes already entered according to the intermediate node order you previously defined.
13. In the **Name** field, enter the first arc name.
14. (If your arc provider is query-based) Select **Query**.
    By default the arc provider is based on a **MetaAssociationEnd**.
15. Depending on the arc provider:
    - in the **MetaAssociationEnd** drop-down list, select the MetaAssociationEnd that provides the arc.
    - in the **Query** drop-down list, select the query that provides the arc.
16. Click **Next**.
    The second arc definition window is displayed, with its two nodes.
17. Define the second arc (and the next ones if any) as above:
    - its **Name**
    - its arc provider.
18. Click **OK**.
    The full path is created. In the **Paths** page, for the selected path:
    - the **Ordered Intermediate Nodes** pane displays the intermediate nodes of the path
    - the **Arcs** pane details the arcs building the path.

    Once created you can no longer modify the path (neither ts intermediate nodes nor its arcs).

    In the GraphSet Definition (**Definition** page), the path is added in the **Arcs of the node** and **Is Path** attribute is selected.

    ☛ *If you need to modify the path definition, see Modifying a path definition.*

19. According to your needs, add fields on intermediate nodes.

    ☛ *See Adding Fields to Intermediate Nodes.*

Modifying a path definition

Once the path is created, you may want to add an intermediate node or change the node order of the path.

To modify the path definition:

1. Access the GraphSet Definition properties.
   ☛ *See Displaying the GraphSet Definition properties.*
2. Display its **Paths** page.
3. In the paths pane, select the path and click **Edit** .
4. Use the wizard to modify the path according to your need.

Adding Fields to Nodes/Arcs

A node can carry ***fields***, which are:

- based on values:
  - MetaAttributes
  - TaggedValues
  - Legattributes
- computed with a macro
- an object

These fields are displayed on the node, but can be hidden.

Field access

By default, the field is provided by the source object (**Access** section: "Direct").

You may want to provide the field by a MetaAssociationEnd of the source object (**Access** section: "MetaAssociationEnd").

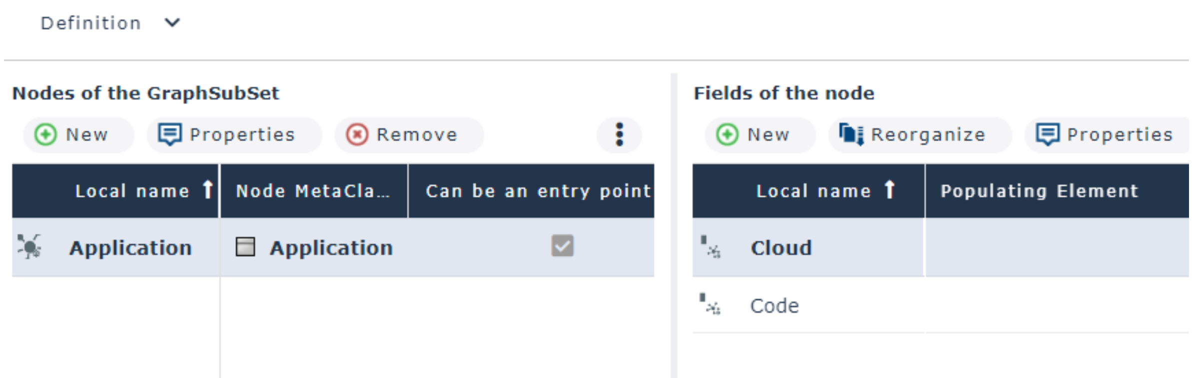
Adding value-based fields to a node

To add a value-based field to a node:

1. Access the GraphSet Definition properties.
   ☛ *See Displaying the GraphSet Definition properties.*
2. Display its **Definition** page.
3. In the **GraphSubSet** drop-down list, select the GraphSubSet concerned.
   ☛ *Else directly access the GraphSubSet properties, see Displaying the GraphSubSet properties.*
4. In the **Nodes of the GraphSubSet** pane, select the row of the node for which you want to define a field.

   `E.g.: Application.`
5. In the **Fields of the node** pane, click **New** .
   The **Creation of GraphSet Field Definition** windows is displayed.
6. In the **Local name** field, modify the default name.

7. (For a field from a MetaAssociationEnd) In the **Access** section:
   - in **Access type**: select "MetaAssociationEnd"
   - in **MetaAssociationEnd**: select the MetaAssociationEnd concerned
8. In the **Configuration** section, define the value type:
   By default the value is based on a **MetaAttribute**.
   If your value is TaggedValue-based, select **taggedValue**.
9. Depending on the value type:
   - in the **MetaAttribute** drop-down list, select the MetaAttribute you want to be displayed in the field.
   - in the **TaggedValue** field, click the right oriented arrow and connect the TaggedValue you want to be displayed in the field.
10. Click **OK**.



Adding macro-based fields to a node

Prerequisite:

For each macro-based field you have to create its implementing macro. This macro is based on the **GetAttributeValue** function:

GetAttributeValue(Object as MegaOject,AttributeID as Variant,Value As String)

Where:

Object is the object of which the attribute value is requested.

AttributeID is the absolute identifier of the attribute.

Value is the attribute value returned by the function, concerning this object.

☛ *See examples of macro-based field on intermediate nodes:*
Example 1: graph based on the "Exchange Between Applications" GraphSet Definition
Example 2: graph based on the "Exchange and Content of an Application" GraphSet Definition

To add a macro-based field to a node:

1. Access the GraphSet Definition properties.
   ☛ *See Displaying the GraphSet Definition properties.*
2. Display its **Definition** page.
3. In the **GraphSubSet** drop-down list, select the GraphSubSet concerned.
   ☛ *Else directly access the GraphSubSet properties, see Displaying the GraphSubSet properties.*

4. In the **Nodes of the GraphSubSet** pane, select the row of the node for which you want to define a field.
5. In the **Fields of the node** pane, click **New** ➕.
   The **Creation of GraphSet Field Definition** windows is displayed.
6. In the **Local name** field, modify the default name.
7. (For a field from a MetaAssociationEnd) In the **Access** section:
   - in **Access type**: select "MetaAssociationEnd"
   - in **MetaAssociationEnd**: select the MetaAssociationEnd concerned
8. In the **Configuration** section:
   - In **Property type** field: select "Computed"
   - In the **Implementation** field, click the right-oriented arrow and connect the macro that enables to compute the field value.

   Define the **MetaAttribute** characteristics:
   - **Length**
   - **Type** (default value "String)
   - **Format** (default value "Standard)
9. Click **OK**.

Adding object-based fields to a node

In case of "MetaAssociationEnd" **Access type**, you might want to add the object itself.

To add an object-based field to a node:

1. Access the GraphSet Definition properties.

   ☛ *See Displaying the GraphSet Definition properties.*
2. Display its **Definition** page.
3. In the **GraphSubSet** drop-down list, select the GraphSubSet concerned.

   ☛ *Else directly access the GraphSubSet properties, see Displaying the GraphSubSet properties.*
4. In the **Nodes of the GraphSubSet** pane, select the row of the node for which you want to define a field.
5. In the **Fields of the node** pane, click **New** ➕.
   The **Creation of GraphSet Field Definition** windows is displayed.
6. In the **Local name** field, modify the default name.
7. In the **Access** section:
   - in **Access type**: select "MetaAssociationEnd"
   - in **MetaAssociationEnd**: select the MetaAssociationEnd concerned
8. In the **Configuration** section:
   - in **Object property type**: select "Object"

Adding Fields to Intermediate Nodes

As a node, an intermediate node can also carry fields, which are either:

- based on values from the corresponding HOPEX object:
  - MetaAttributes
  - TaggedValues
  - Legattributes, or
- computed with a macro

These fields are displayed on the intermediate nodes, but can be hidden.

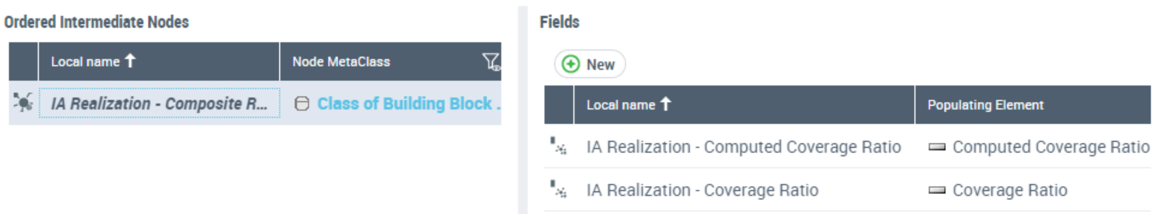
Adding value-based fields to an intermediate node

To add a value-based field on an intermediate node:

1. Access the GraphSet Definition properties.

   ☛ *See Displaying the GraphSet Definition properties.*
2. Display its **Path** page.
3. In the **GraphSubSet** drop-down list, select the GraphSubSet concerned.

   ☛ *Else directly access the GraphSubSet properties, see Displaying the GraphSubSet properties.*
4. In the **Path** pane, select the row of the path that includes the intermediate node.
5. In the **Ordered Intermediate Nodes** pane, select the intermediate node for which you want to define a field.
6. In the **Fields** pane, click **New** ✚.
   The **Creation of GraphSet Field Definition** windows is displayed.
7. In the **Local name** field, modify the default name.
8. In the **Field Definition Type** drop-down list, select "Value".
9. Define the value type:
   By default the value is based on a **MetaAttribute**.
   If your value is TaggedValue-based, select **taggedValue**.
10. Depending on the value type:
    - in the **MetaAttribute** drop-down list, select the MetaAttribute you want to be displayed in the field.
    - in the **TaggedValue** field, click the right-oriented arrow and connect the TaggedValue you want to be displayed in the field.
11. Click **OK**.



Adding macro-based fields to an intermediate node

Prerequisite:

For each macro-based field you have to create its implementing macro. This macro is based on the **GetAttributeValue** function:

GetAttributeValue(Object as MegaOject,AttributeID as Variant,Value As String)

Where:

Object is the object of which the attribute value is requested.
AttributeID is the absolute identifier of the attribute.
Value is the attribute value returned by the function, concerning this object.

`Example: the "Scenario Flow - Get Content" macro computes the "Content" field for application flows.`

```
'To Implement if you want to have a computed value for this attribute
Sub GetAttributeValue(Object as MegaObject,AttributeID as Variant,Value as String)
Value = ""
dim oContent
for each oContent in Object.GetCollection("Conveyed Work Product")
Value = oContent.GetProp("_IdAbs")
next
End Sub
```

With "***~ycuIb8buOn)E[Conveyed Work Product]***".

See examples:

- Example 1: graph based on the "Exchange Between Applications" GraphSet Definition.
- Example 2: graph based on the "Exchange and Content of an Application" GraphSet Definition.

To add a macro-based field on a node:

1. Display its **Path** page.
2. In the **GraphSubSet** drop-down list, select the GraphSubSet concerned.

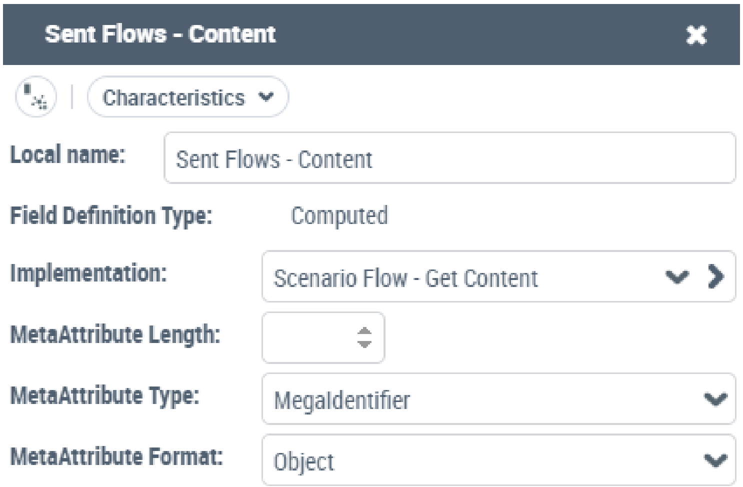
   ☛ *Else directly access the GraphSubSet properties, see Displaying the GraphSubSet properties.*
3. In the **Path** pane, select the row of the path that includes the intermediate node.
4. In the **Ordered Intermediate Nodes** pane, select the intermediate node for which you want to define a field.
5. In the **Fields** pane, click **New** ➕.
   The **Creation of GraphSet Field Definition** windows is displayed.
6. In the **Local name** field, modify the default name.
7. In the **Field Definition Type** drop-down list, select "Computed".
8. In the **Implementation** field, click the right-oriented arrow and connect the macro that computes the field value.
   The implementing macro is based on the **GetAttributeValue** function.

   `Example: the "Scenario Flow - Get Content" macro computes the "Content" field for application flows.`

9. Define the MetaAttribute characteristics:
   - (optional) MetaAttribute Length
   - MetaAttribute Type (default value "String)

     `Example: "MegaIdentifier"`
   - MetaAttribute Format (default value "Standard)

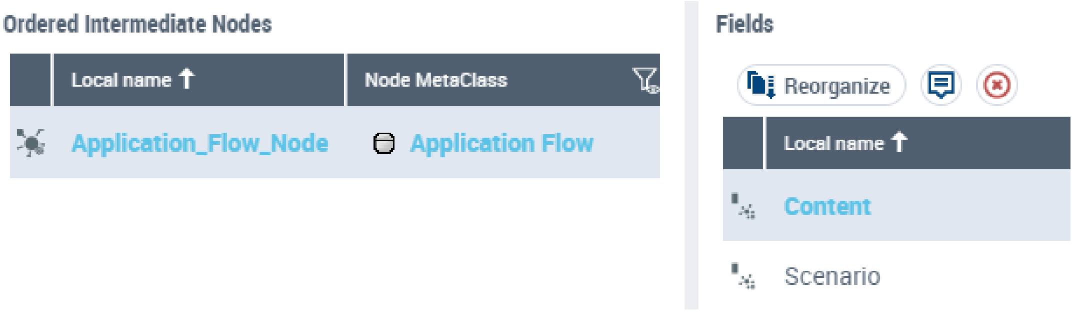
     `Example: "Object"`
10. Click **OK**.



Example 1: graph based on the "Exchange Between Applications" GraphSet Definition

In the "Exchange Between Applications" graph, two fields are added to the **Application Flow** intermediate nodes:

- "Content"
- "Scenario"



The macros used to implement the "Content" and "Scenario" fields are:

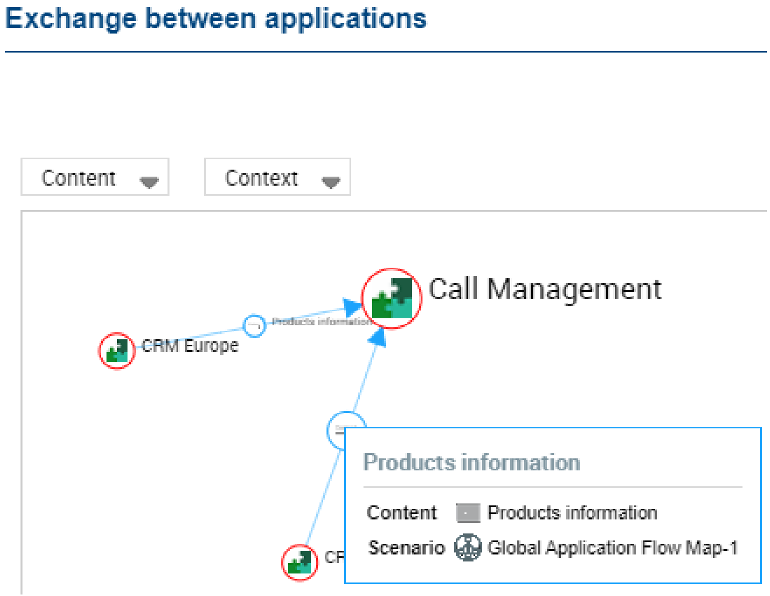
- the "Get Application Flow content.Implementation" macro

```
'To Implement if you want to have a computed value for this attribute
Sub GetAttributeValue(Object as MegaObject,AttributeID as Variant,Value
as String)
If Object.GetCollection("Content").Item(1).Exists Then
Value = Object.GetCollection("Content").Item(1).MegaField()
End If
End Sub
```

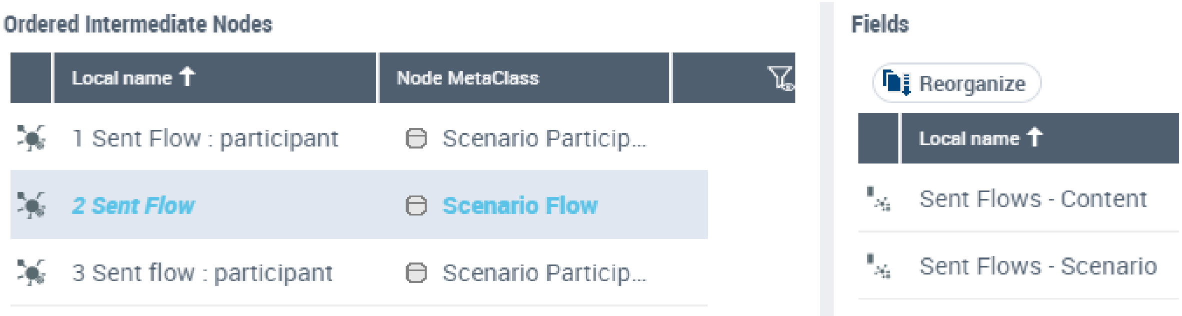
- the "Get Owner Ressource Scenario.Implementation" macro

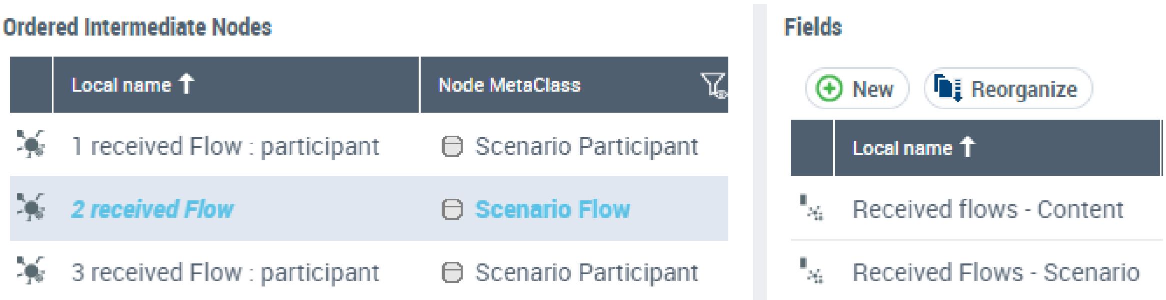
```
'To Implement if you want to have a computed value for this attribute
Sub GetAttributeValue(Object as MegaObject,AttributeID as Variant,Value
as String)
Dim mgobjRoot :set mgobjRoot = Object.GetRoot
If Object.GetCollection("Owner Resource Scenario").Item(1).Exists Then
Value = Object.GetCollection("Owner Resource
Scenario").Item(1).MegaField()
End If
End Sub
```

In the corresponding graph, the "Content" and "Scenario" fields can be displayed on each **Application Flow** intermediate node.



Example 2: graph based on the "Exchange and Content of an Application" GraphSet Definition





In the "Exchange and Content of an Application" graph, two fields ("scenario" and "content") are added to intermediate nodes of both "received flows" and "sent flows" paths between applications:

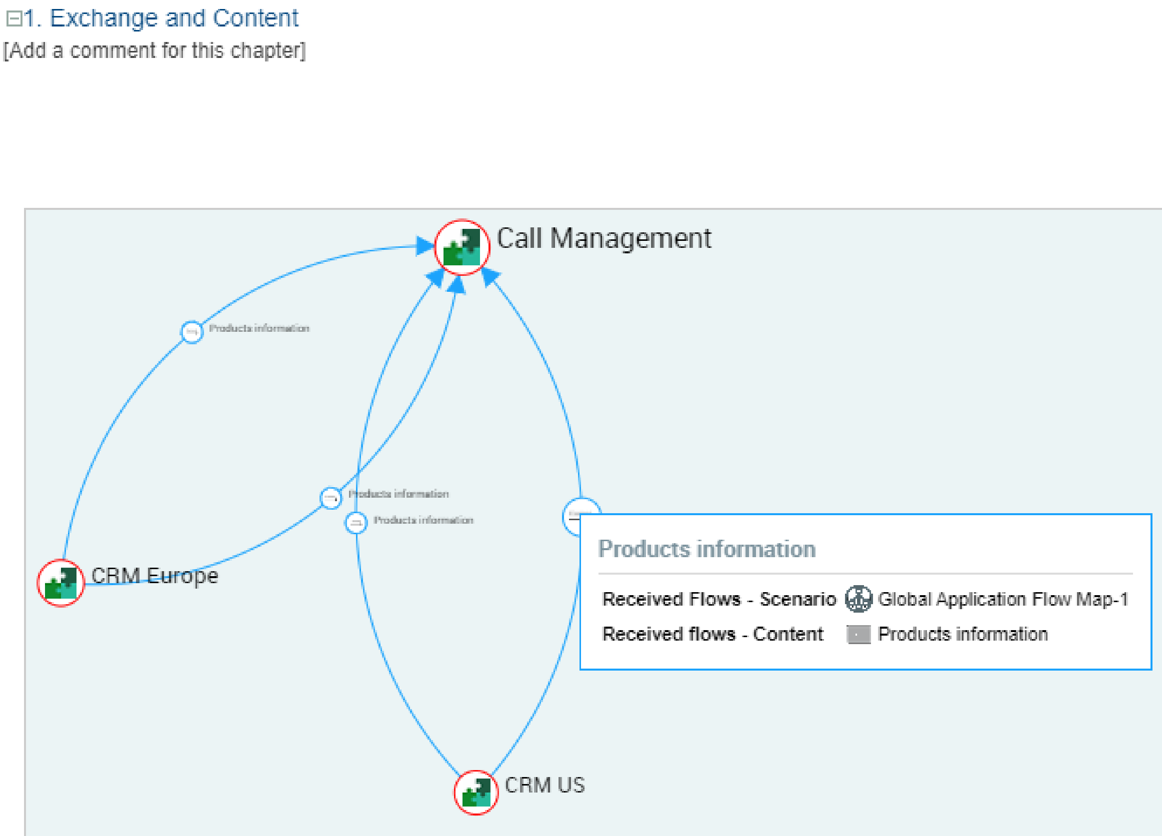
- The "Scenario Flow - Get Content" macro computes the values for the "Content" fields from the "Conveyed Work Product" collection.

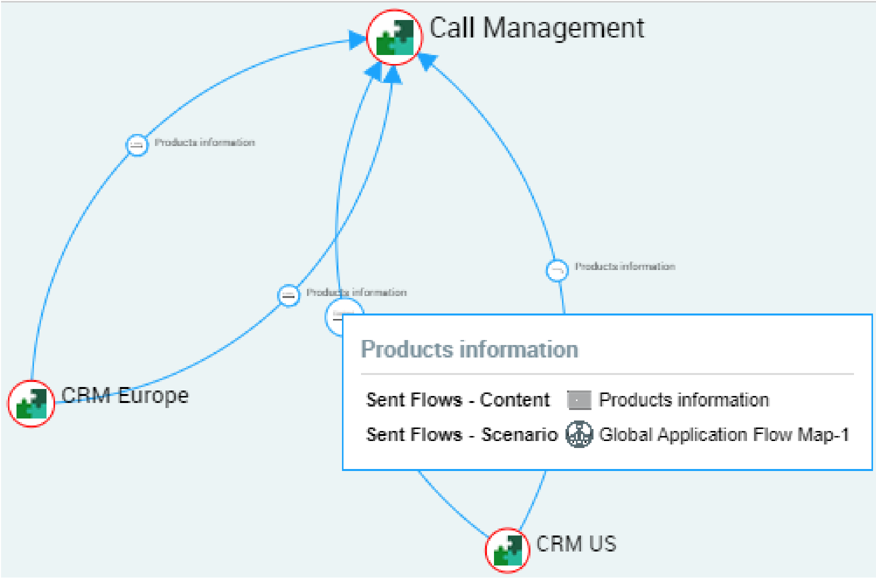
```
'To Implement if you want to have a computed value for this attribute
Sub GetAttributeValue(Object as MegaObject,AttributeID as Variant,Value as String)
Value = ""
dim oContent
for each oContent in Object.GetCollection("Conveyed Work Product")
Value = oContent.GetProp("_IdAbs")
next
End Sub
```

- The "Scenario Flow - Get Scenario" macro computes the values for the "scenario" fields from the "Flow owner" collection.

```
'To Implement if you want to have a computed value for this attribute
Sub GetAttributeValue(oFlow as MegaObject,AttributeID as Variant,Value as
String)
Value = ""
dim oScenario
for each oScenario in oFlow.GetCollection("Flow Owner")
Value = oScenario.GetProp("_IdAbs")
next
End Sub
```

The graph displays both "Content" and "Scenario" fields for each received/sent flow:





Previewing the GraphSet

From the GraphSet Definition properties you can test the GraphSet.

To preview the GraphSet:

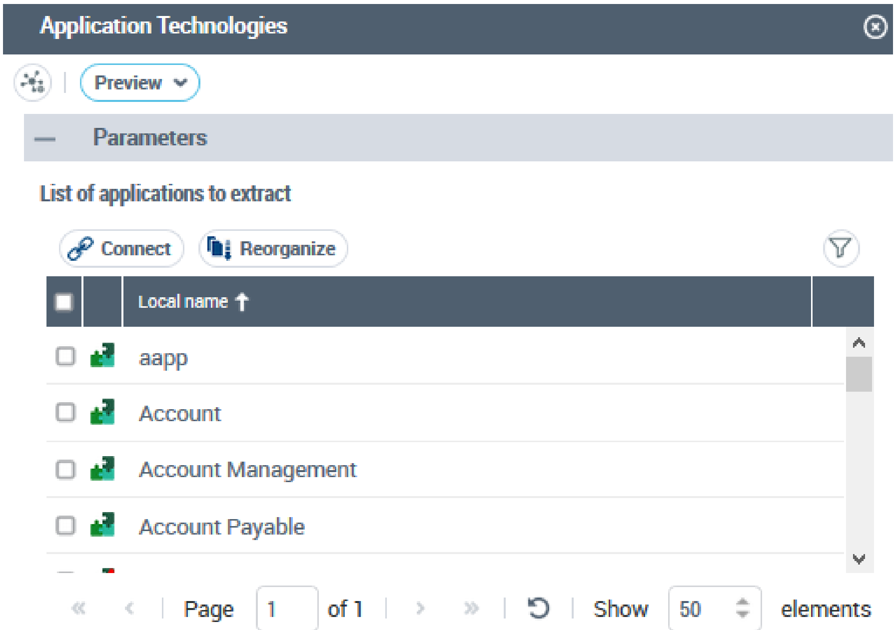
1. Access the GraphSet Definition properties.

    ☛ *See Accessing the GraphSet Definitions.*

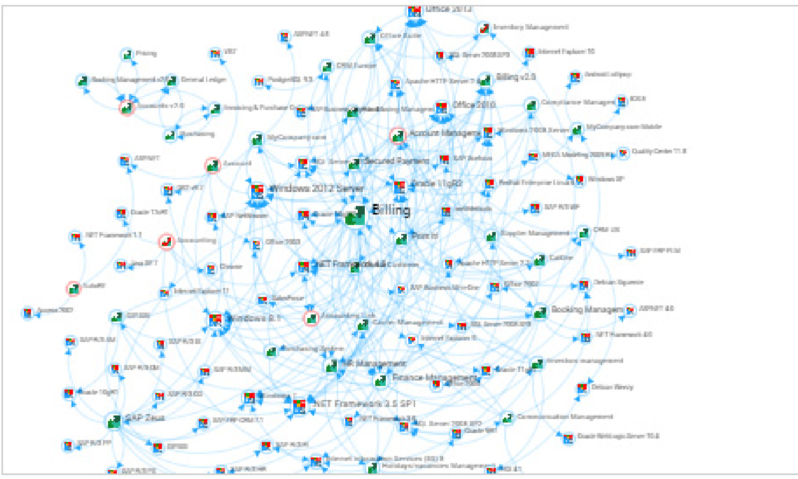
2. Display its **Preview** page.
3. Click **Create Preview**.

    ☛ *Once the preview has been generated once, you do not need to click **Create Preview** anymore. The GraphSet is automatically generated and displayed.*

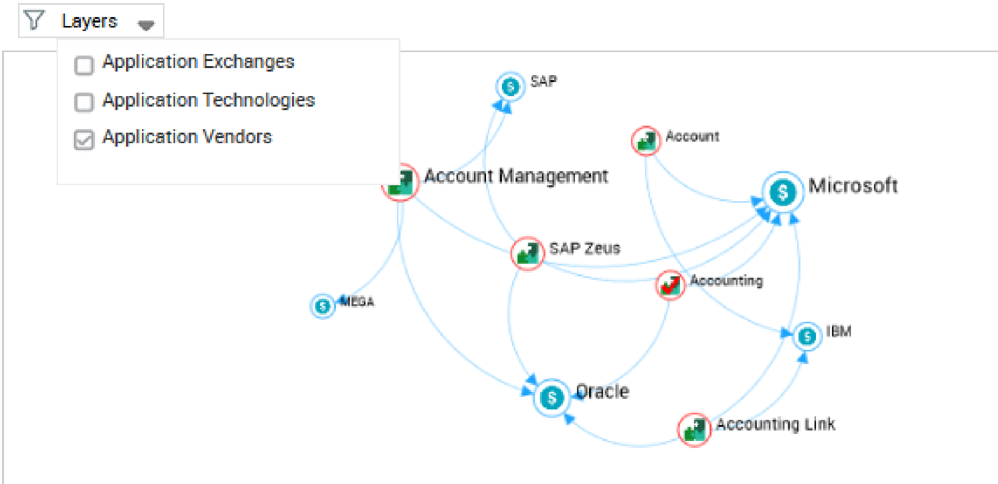
4. (If needed) In the **Parameters** section, define the parameters:
   - (if query parameters are used) enter the parameter values
     ☛ *See Defining query parameters.*
   - (if object collection-based GraphSet Definition) in the source MetaClass, click **Connect** and connect the MetaClass items you want.



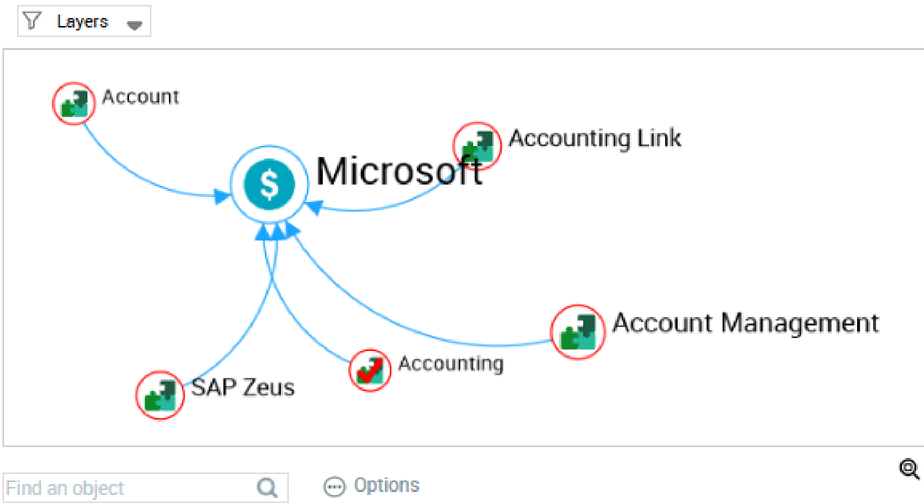
5. In the **Parameters** section title, click to reduce the section.
6. Click **Apply Parameters**.
   The GraphSet displays the nodes and arcs.
   ☛ *Objects from a Data Reading access not accessible to the user are not displayed.*



7. You can customize the preview display for a better visibility:
   - (GraphSet Definition including several GraphSubSets) Above the graph, click **Layers**, and clear layers to keep a single layer selected.



   - Double-click a node to display only this node, its arcs and opposite nodes.

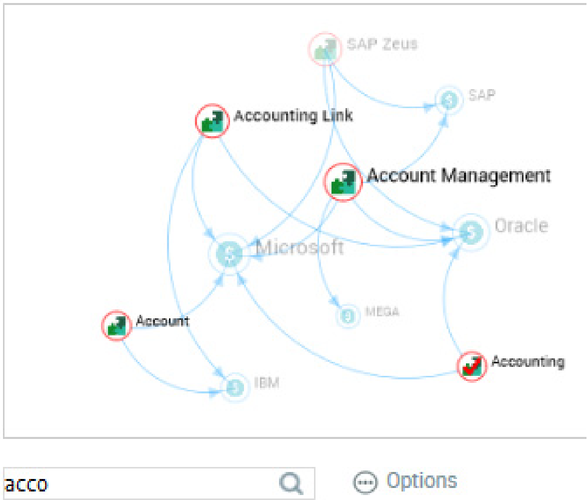


   - Bellow the graph, click **Options** to:

- reduce/increase the space between the nodes
- reduce the label length
- hide nodes without links

- In the **Find an object** field, enter characters to greyed all of the other nodes.

  `For example, enter "acco": "Account", "Account Link" and "Account Management" applications and "Accounting Technology" are highlighted (all of the other nodes are grayed out)`



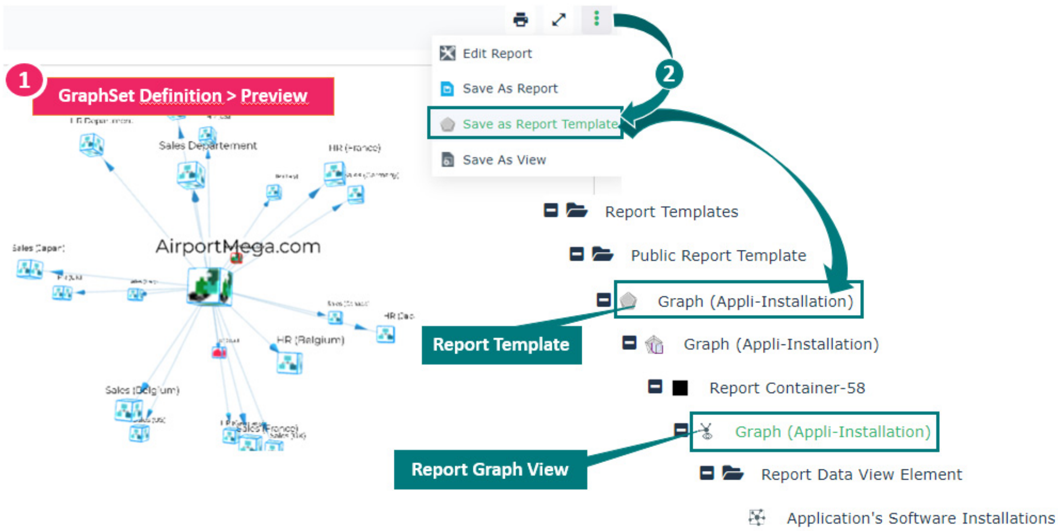
- Use the mouse wheel to zoom in (/out) the graph.

Saving the Graph Report

From the **Preview** page of the GraphSet Definition, you can save the graph as:

- a Report Template
- a Report Data View, to use it in a Report Template

> ☛ *Note that **Save as Report**, generates a public report and also a private **Report Template**, which you can access in **Private Report Template > My Report Templates** folder.*



Creating a graph-type Report Template

To create a Report Template from a graph report:

1. Access the GraphSet Definition properties.

   > ☛ *See Accessing the GraphSet Definitions.*

2. Display its **Preview** page.
3. In the graph, click **More** ⋮ **> Save As Report Template** .

   > ☛ *Note: if you customize the report display, this customization is not taken into account in the Report Template.*

4. In the **Name** field, enter your Report Template name.
5. Click **OK**.
   A public Report Template is created, with its public **Report Graph View**. They both have the same name.
   You can access the Report Template:
   - directly from your **Home** page, from **Custom Report Templates** tile, or
   - from the **Public Report Template** folder (**Report Studio > Report Template Definition > Report Templates**)

Creating a Report Graph View

To create a Report Graph View:

1. Access the GraphSet Definition properties.

   ☛ *See* *Accessing the GraphSet Definitions.*
2. Display its **Preview** page.
3. In the graph, click **More** ⋮ > **Save as View** .
4. Enter a **Name** to the view.
5. Click **OK**.
   The Report Graph View is created and accessible in the **Report Studio > Report Template Definition > Public Report Data Views** folder.
   Any HOPEX user can add the View in a report:
   - use the View to create a Report Template

     ☛ *See* *Creating a Report Template using a Report Data View.*
   - add the View as a report Chapter in a Report Template

     ☛ *See* *Adding a Report Chapter to a Report Template.*

TREESET DEFINITION

The following points are covered here:

INTRODUCTION TO TREESET DEFINITION

☛ ***TreeSet Definition** is only available with **HOPEX Power Studio** technical module.*

A **TreeSet Definition** creation is performed in **HOPEX** (Web Front-End) for users with **HOPEX Customizer** or **HOPEX Customizer Publisher** profile.

Creation and use of **TreeSets** are available in **HOPEX** (Web Front-End) for all users.

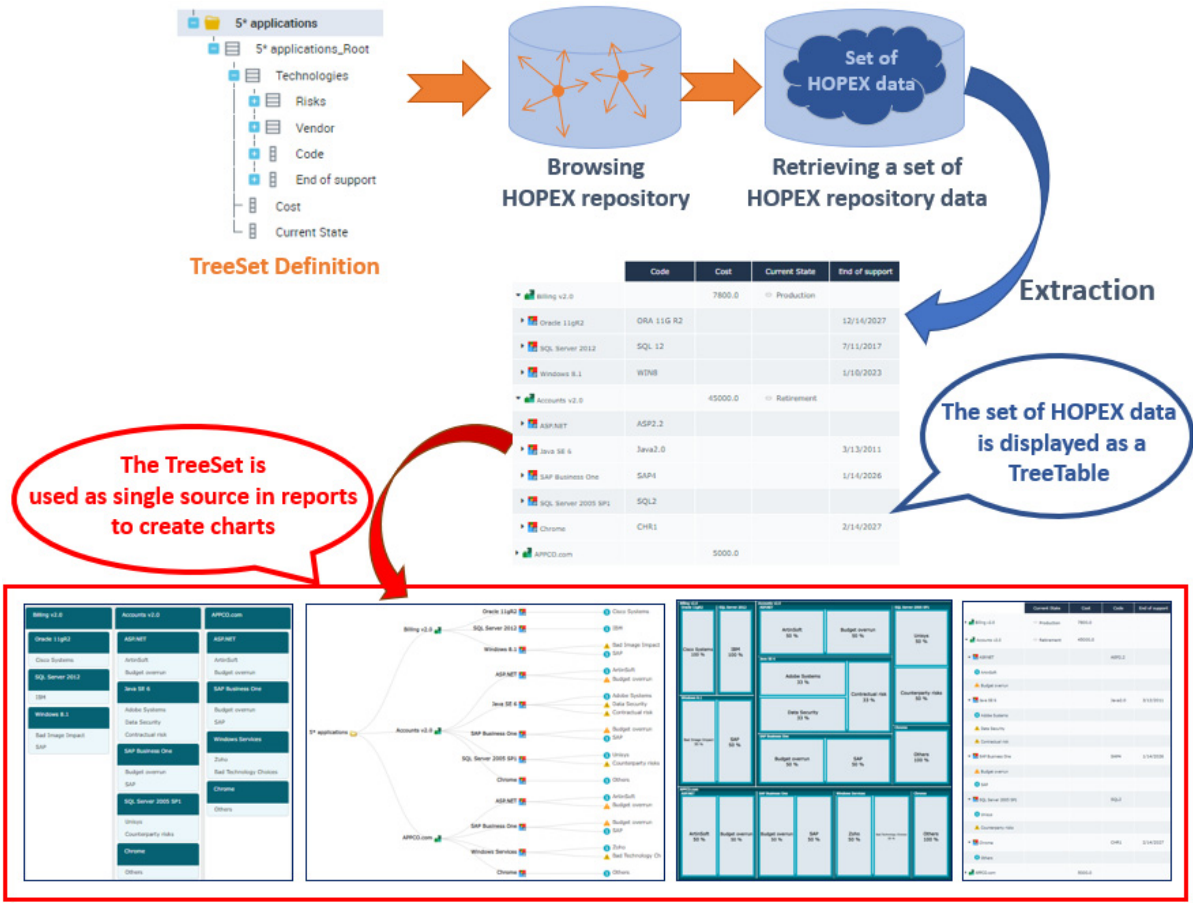
Once a **Report Template** is created from a **TreeSet Definition**, any user of a Solution can use it to query **HOPEX** repository and create **tree** reports.

TreeSet Definition and TreeSet Principle

TreeSet Definition

A **TreeSet Definition** enables to create tree reports as follows:

- ***Phase 1***: Extraction of a set of **HOPEX** data.
  This extraction consists in defining a root node (a single occurrence or a set of occurrences), from which the repository is browsed to retrieve a set of **HOPEX** data that will constitute the **TreeSet**.
- ***Phase 2***: Display of this set of **HOPEX** data as a tree (Breakdown, Dendrogram, TreeMap, and TreeTable).
- ***Phase 3***: Creation of tree reports from this display.



This data extraction constitutes the **TreeSet** data, with no style specification.

From this **TreeSet** you can generate **Instant Reports** (Breakdown, Dendrogram, TreeMap, and TreeTable).

TreeSet Definition structure

The **TreeSet Definition** defines how to build a hierarchical set of data.

The hierarchy is built from **TreeSet Collections**, which define how the repository is browsed. Additional data can be added to the structure thanks to TreeSet Properties.

The **TreeSet Definition structure** includes a **root node**, which holds the **root TreeSet Collection** with as many children as needed:

- TreeSet Properties
- TreeSet Collections with:
  - TreeSet Properties
  - TreeSet Collections

Example:

The "Application" **TreeSet Definition structure** is based on "5* Applications-Root" **root TreeSet Collection** with the "Technologies used" **TreeSet Collections** child and with "Application Type" and "Cost" ("Global expense" MetaAttribute) **TreeSet Properties**.

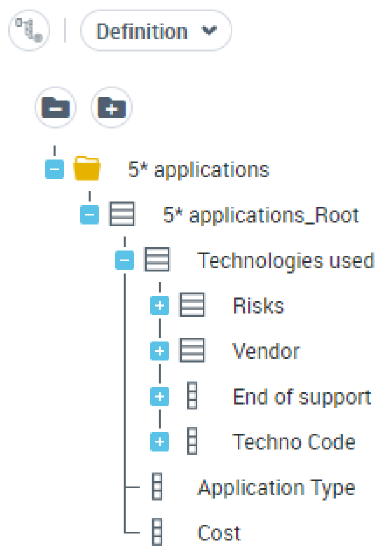
The "Technologies used" **TreeSet Collections** has four children:

- the "Risks" and "Vendor" **TreeSet Collections**

See *Adding a TreeSet Collection to a TreeSet Definition.*

- the "End of support" and "Techno code" **TreeSet Properties**

See *Adding TreeSet Properties to a TreeSet Collection.*



Root node

The **Root node** is a node visible in the tree display only when the structure includes several entry objects.

TreeSet Collection

A **TreeSet** Collection is MetaClass-specific. It allows to define a collection of child objects computed from its parent.

The **TreeSet Collection** children of the root node are the entry points of the structure.

A **TreeSet Collection** can carry TreeSet Properties based on MetaAttributes, TaggedValues, LegAttributes from the corresponding HOPEX object, or computed with a macro.

TreeSet

A **TreeSet** is an instance of a **TreeSet Definition**, which defines all parameter values necessary to generate the tree data structure.

A **TreeSet** is made up of nodes that are linked together as branch of the tree.

In the **TreeSet Definition** properties, the **Preview** page shows the **TreeSet** displayed as a TreeTable:

Preview

Report TreeSet

Apply Parameters | Instant Report | Limit the data to 50 nodes

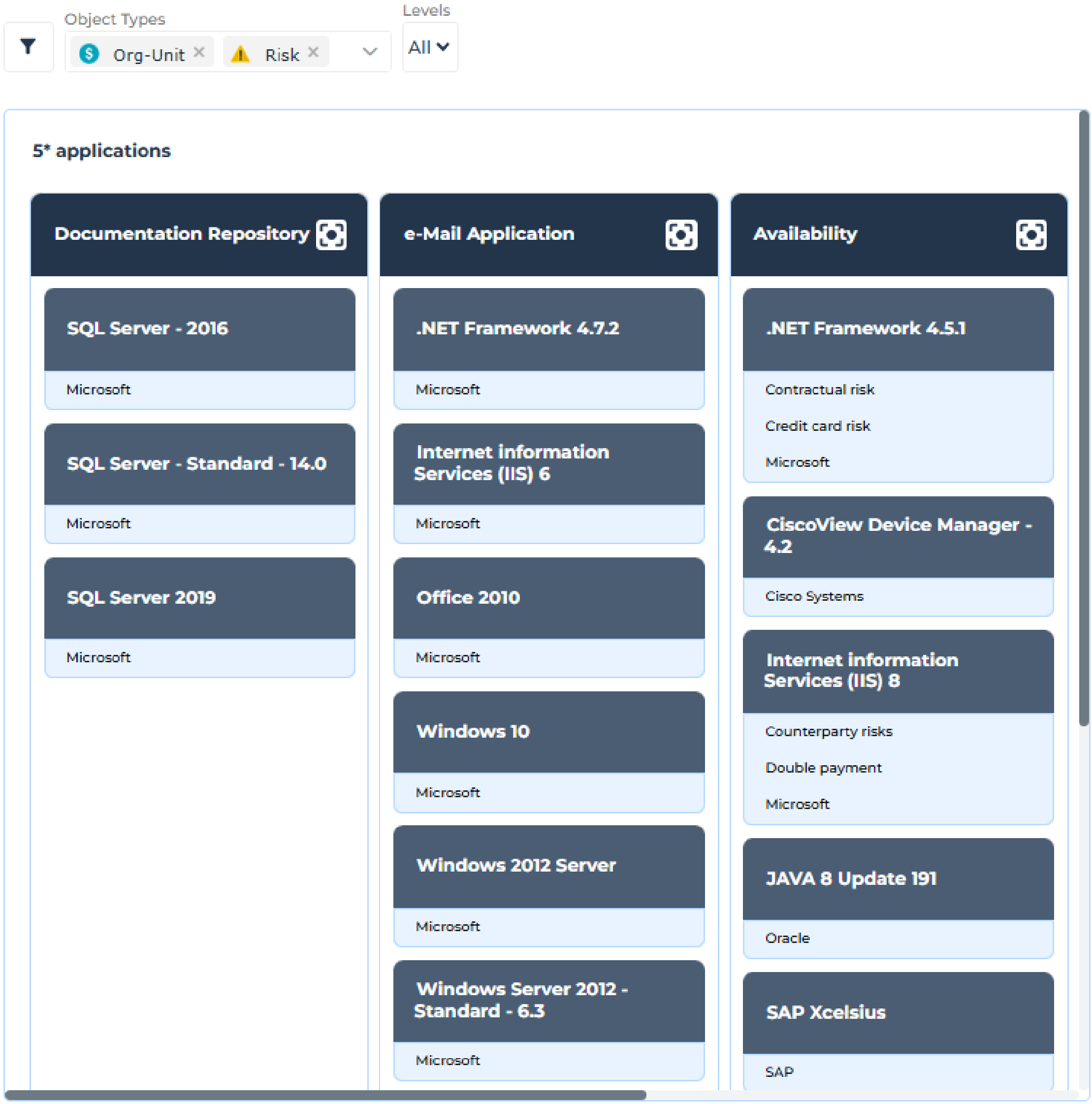
	Application Type	Cost	End of support	Techno Code
Documentation Repository	In House Application	€625,867.00		
SQL Server - 2016				sserver-2016
SQL Server - Standard - 14.0			10/11/2022	sserver-standard-14.0
SQL Server 2019				
Microsoft				
e-Mail Application	Office System	€488,194.00		
.NET Framework 4.7.2			12/31/2040	4.7.2
Internet information Services (IIS) 6			7/13/2010	iinformationservices(iis)6
Office 2010				o2010
Windows 10			10/13/2020	w10
Windows 2012 Server			1/9/2018	WIN2K12
Windows Server 2012 - Standard - 6.3			10/9/2018	wserver2012-standard-6.3
Availability	In House Application	€52,800.00		
Instant Messaging	In House Application	€548,335.00		
Relationship Selling	In House Application	€348,338.00		

Instant Reports

From the **TreeSet** you can launch Instant Reports, customize them and save them to create tree-type Report templates and Report Tree Views.

You can launch the following Instant Report types:

- ***Breakdown***



- ***Dendrogram***

5* applications (11)
e-Mail Application (6)
.NET Framework 4.7.2 (1)
Internet information Services ... (1)
Office 2010 (1)
Windows 10 (1)
Windows 2012 Server (1)
Windows Server 2012 - Standard... (1)
Availability (9)
.NET Framework 4.5.1 (3)
Contractual risk
Credit card risk
Microsoft
CiscoView Device Manager - 4.2 (1)
Cisco Systems
Internet information Services ... (3)
Counterparty risks
Double payment
Microsoft
JAVA 8 Update 191 (1)
SAP Xcelsius (1)
SQL Server - Enterprise - 15.0 (1)
Windows 7 (1)
Windows Server 2012 - Standard... (1)
Windows Server 2016 - Essentia... (1)
Instant Messaging (2)
Office 2013 (1)
SQL Server - Enterprise - 13.0... (1)
Relationship Selling (4)
Office 2013 (1)
SQL Server - Enterprise - 13.0... (1)
SQL Server - Enterprise - 15.0 (1)
Windows Server 2016 - Essentia... (1)
Supplies purchase (2)
.NET Framework 4.7.2 (1)
Angular - 9.1 (1)

- ***Treemap***


Number of displayed levels 3
Display criterion Number of components
5* applications
Documentation Repository
SQL Server - 2016
Microsoft 100 %
SQL Server 2019
Microsoft 100 %
SQL Server - Standard - 14.0
Microsoft 100 %
e-Mail Application
.NET Framework 4.7.2
Microsoft 100 %
Internet Information Services (...
Microsoft 100 %
Office 2010
Microsoft 100 %
Windows 2012 Server
Microsoft 100 %
Windows 10
Microsoft 100 %
Windows Server 2012 - Standar...
Microsoft 100 %
Availability
.NET Framework 4.5.1
Contractual risk 33 %
Credit card risk 33 %
Microsoft 33 %
Internet Information Services (IIS) 8
Counterparty risks 33 %
Double payment 33 %
Microsoft 33 %
JAVA 8 Update 191
Oracle 100 %
SAP Xcelsius
SAP 100 %
Windows 7
Microsoft 100 %
Windows Server 2012 ...
Microsoft 100 %
CiscoView Device Manager - 4.2
Cisco Systems 100 %
SQL Server - Enterprise - 15.0
Microsoft 100 %
Windows Server 2016 - Essentials - 10.0
Microsoft 100 %
Instant Messaging
Office 2013
Microsoft 100 %
SQL Server - Enterprise - 13.0 - SP2
Microsoft 100 %
Relationship Selling 4 %

- ***TreeTable***

	Application Type	Cost	Techno Code	End of support
Documentation Repository	In House Application	€625,867.00		
e-Mail Application	Office System	€488,194.00		
Availability	In House Application	€52,800.00		
.NET Framework 4.5.1			.NET 4.5	1/9/2018
Contractual risk				
Credit card risk				
Microsoft				
CiscoView Device Manager - 4.2			cdevicemanager-4.2	3/25/2020
Internet information Services (IIS) 8			iinformationservices(iis)8	10/9/2018
Counterparty risks				
Double payment				
Microsoft				
JAVA 8 Update 191			j8update191	2/16/2021
SAP Xcelsius			sxcelsius	12/31/2015
SQL Server - Enterprise - 15.0			sserver-enterprise-15.0	1/7/2030
Windows 7			w7	1/13/2015
Windows Server 2012 - Standard - 6.3			wserver2012-standard-6.3	10/9/2018

Creating and Defining a TreeSet Definition: the Big Picture

To create and define a **TreeSet Definition**:

1. Define the **root TreeSet Collection**, i.e. the **TreeSet** entry points.
   The **TreeSet** entry point can be:
   - an object,
   - an object collection,
   - a query

   ☛ *See Creating a TreeSet Definition.*

2. Define the **TreeSet Collections** that constitute the **TreeSet**.

   ☛ *See Adding a TreeSet Collection to a TreeSet Definition.*

3. (optional) Add a loop to a **TreeSet Collection**.

   ☛ *See Defining a TreeSet Collection.*

4. (Optional) Add **TreeSet Properties** to a **TreeSet Collection**.

   ☛ *See Adding TreeSet Properties to a TreeSet Collection.*

5. Preview the **TreeSet**.

   ☛ *See Previewing the TreeSet.*

6. From the **TreeSet**, generate Instant Reports of type:
   - Breakdown
   - Dendrogram
   - TreeMap
   - TreeTable

   ☛ *See Generating Instant Reports from a TreeSet.*

7. Customize the Instant Report and perfom:
   - a **Save as Report** to create a **tree report** and get its **Report Template**.

     ☛ *See Creating a tree-type Report Template.*

   - a **Save as View** to create a **Report Tree View**.

     ☛ *See Creating a Report Tree View.*

8. Customize the tree report.

TreeSet Definition Best Practices

When you create a **TreeSet Definition** you should follow the best practices.

Define a **TreeSet Definition** to create focused **TreeSets**, i.e.:

- The TreeSet should answer a single issue.

  `For examples: a report, an export of specific data.`

- Do not build TreeSets with huge amount of data that you would use for different purposes.

A TreeSet size is limited to:

- 50.000 nodes
- 50 properties

  ☛ *These limits are defined in the following options (**Options > Tools > Documentation > Reports**):*

  Fix the limit for TreeSet Nodes

  Fix the limit for TreeSet properties

A "TreeSet volume alert" warning is generated when a TreeSet exceeds these recommended limits.

Supervising TreeSet Events

In the HOPEX Supervision console, you can check the TreeSet-specific events (e.g.: "TreeSet Generate" informative event):

- **Node Count** indicates the number of nodes of the TreeSet.
- **Internal Node Count Limit** indicates the recommended maximum number of nodes in the TreeSet.
- **Property Count** indicates the number of nodes of the TreeSet.
- **Internal Node Count Limit** indicates the recommended maximum number of nodes in the TreeSet.

For detailed information regarding TreeSet related supervision events, see **HOPEX Administration > Technical Articles > Supervision Event Description > ReportDataSet, TreeSet, and GraphSet**.

TREESET DEFINITION CREATION

The **TreeSet Definition** creation includes:

- Accessing the TreeSet Definitions
- Creating a TreeSet Definition

Accessing the TreeSet Definitions

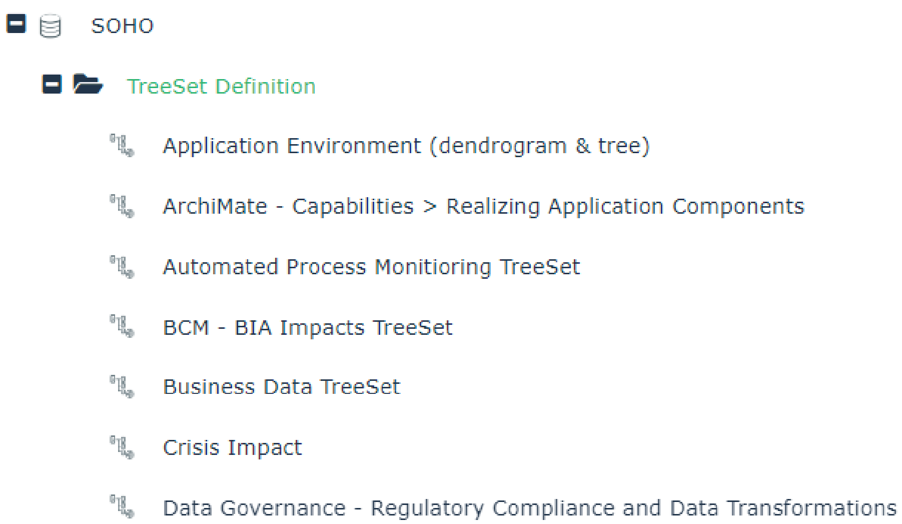
You can access:

- all the TreeSet Definitions
- the custom TreeSet Definitions only
  Custom TreeSet Definitions are the TreeSet Definitions not provided by **MEGA**.

Accessing the TreeSet Definitions (Web Front-End)

To access the TreeSet Definitions:

1. Connect to **HOPEX Studio** desktop.
   ☛ *See Connecting to HOPEX Studio Desktop.*
2. Click the navigation menu and select **Report Studio > TreeSet Definition** .
   In the Edit area, the **Browse** view displays the **TreeSet Definition** tree.
3. Expand the **TreeSet Definition** folder.
   All the TreeSet Definitions available are accessible.



☛ *Select a TreeSet Definition to display its properties in the **Multi-view** window.*

Accessing the custom TreeSet Definitions (Web Front-End)

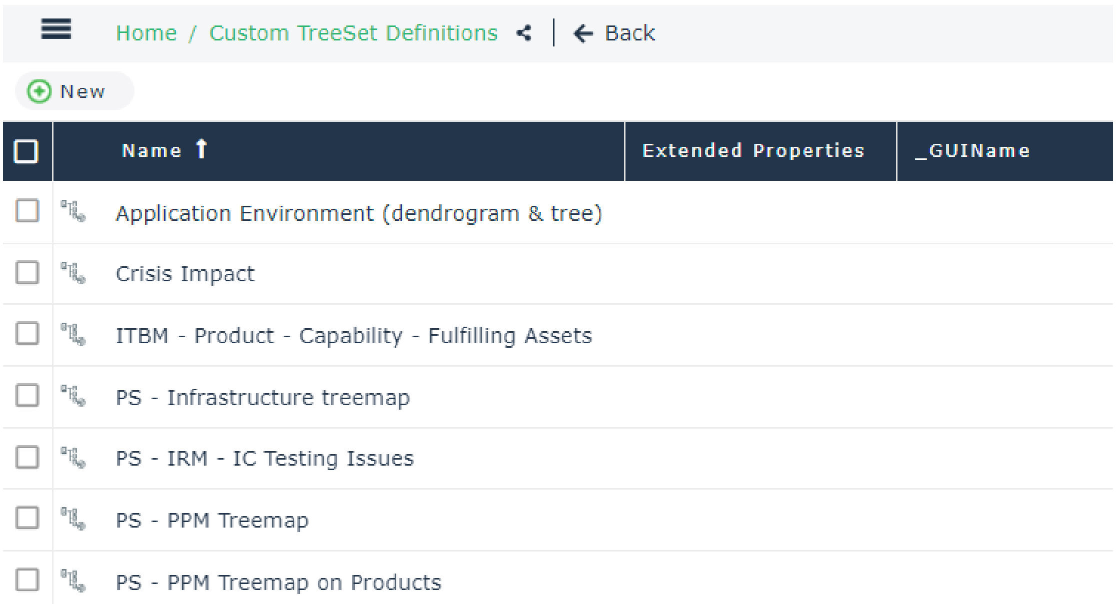
To access the custom TreeSet Definitions:

1. Connect to **HOPEX Studio** desktop.

   ☛ *See Connecting to HOPEX Studio Desktop.*

2. In the **Home** page, click the **Custom TreeSet Definitions** tile.

   In the Edit area, the **Browse** view lists the custom TreeSet Definitions only.



☛ *Select a TreeSet Definition to display its properties in the* ***Multi-view*** *window.*

Accessing the TreeSet Definitions from HOPEX (Windows Front-End)

To access the TreeSet Definitions:

1. Connect to **HOPEX** with **HOPEX Customizer** or **HOPEX Customizer Publisher** profile.
2. In **HOPEX** menu bar, select **View > Navigation Windows > Utilities**.
3. In the repository tree, expand **TreeSet Definition** folder.
   All the TreeSet Definitions available are accessible.

   ☛ *Right-click the TreeSet Definition and select* ***Properties*** *to access the TreeSet Definition properties.*

Creating a TreeSet Definition

A **TreeSet Definition** is MetaClass-specific.

The **TreeSet Definition** is based either on:

- a collection parameter (an occurrence or a set of occurrences of the source MetaClass)
  The **TreeSet Definition** creation consists in defining the root MetaClass (concrete or abstract) of the **TreeSet Definition**.
- a query (a set of occurrences)
  The **TreeSet Definition** creation consists in defining the root MetaClass and a query that collects the entry points.
  In that case, you might need to use Property Parameters.

The **TreeSet Definition Structure** includes a **root TreeSet Collection** with as many items as needed:

- TreeSet Properties
- TreeSet Collections with:
  - TreeSet Properties
  - TreeSet Collections

Creating a Collection Parameter - based TreeSet Definition

When you create a **TreeSet Definition** based on a Collection Parameter, you need to define its root MetaClass.

To create a TreeSet Definition based on a Collection Parameter:

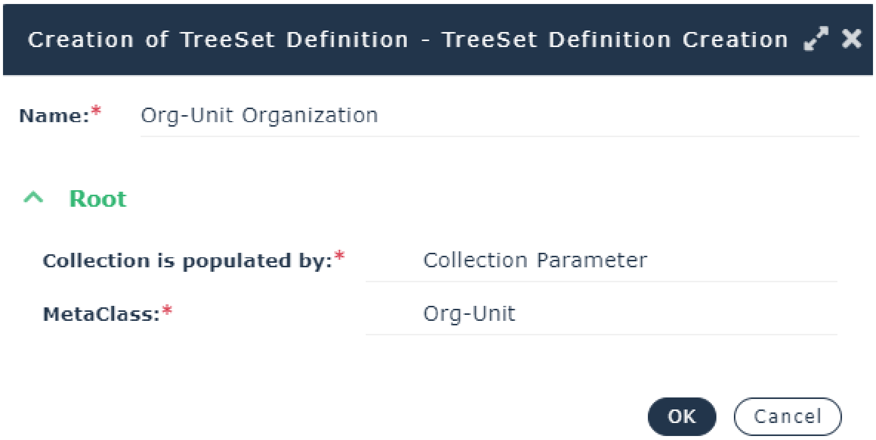
1. Access the **TreeSet Definition** folder
   ☛ *See Accessing the TreeSet Definitions.*
2. Right-click the **TreeSet Definition** folder and select **New > TreeSet Definition**.
   The **Creation of TreeSet Definition** window appears.
3. In the **Name** field, enter your TreeSet Definition name.
   ☛ *By default the TreeSet Definition name is: TreeSet Definition-x (x is a number).*

   `Example: Org-Unit Organization.`

4. Define the **Root** section:
   - In the **Collection is populated by** field keep "Collection Parameter".
   - In the **MetaClass** field, click the arrow and in the drop-down menu select the MetaClass you want to define as the root MetaClass of your TreeSet Definition.

     ☺ *In the field, enter the first letters of the MetaClass for a quick access.*

   `Example: Org-Unit.`



5. Click **OK**.
   The selected MetaClass is defined as your TreeSet Definition root MetaClass.
   Your TreeSet Definition is created and added to the **TreeSet Definition** folder.

6. In your TreeSet Definition properties, display its **Characteristics** page.
It shows:

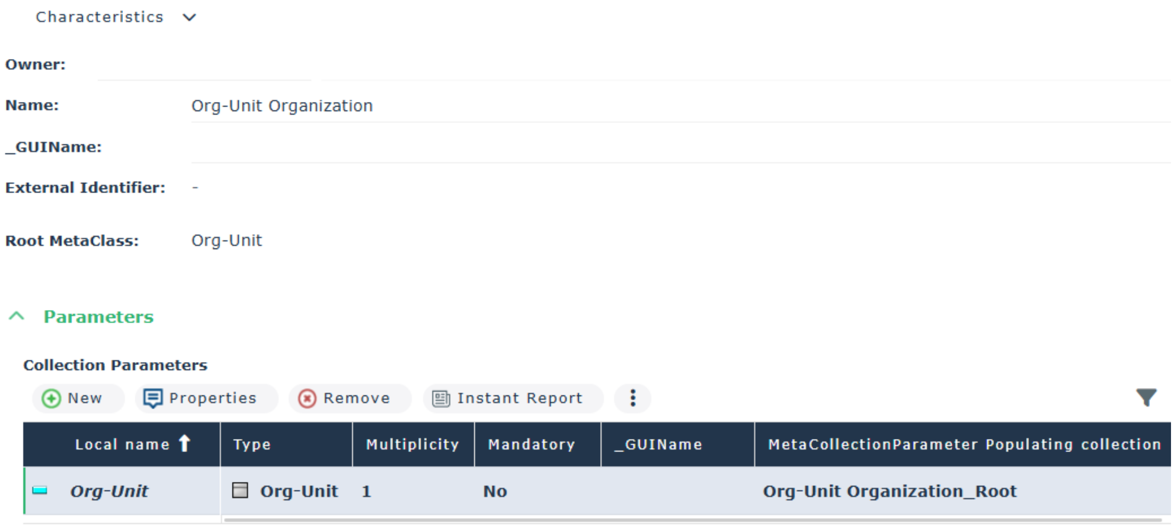
- the **Root MetaClass**
- the **Collection Parameter** characteristics.

For example, with "Org-Unit" as root MetaClass, the Collection parameter characteristics are:

- **Type**: "Org-Unit" MetaClass

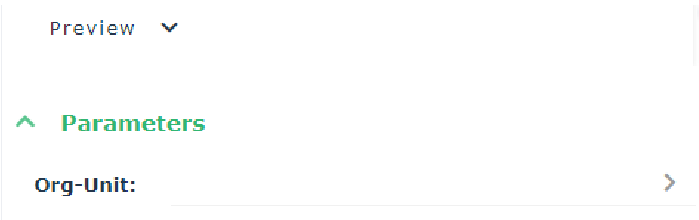
- **Multiplicity**: "1" (default), i.e. the root TreeSet Collection is populated by a single object

- **Populating**: "Org-Unit Organisation_root", i.e. this Collection parameter populates the root node of the "Org-Unit Organization" TreeSet Definition.

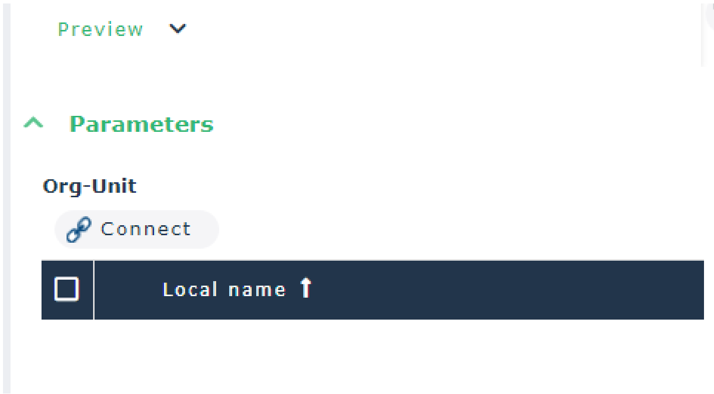


7. (If needed) In the **_GUIName** field, enter a GUI name for the Collection Parameter.
This parameter is displayed in the TreeSet Definition **Preview** page.
If you do not define a GUI name, the local name is used.

☛ *See Previewing the TreeSet.*



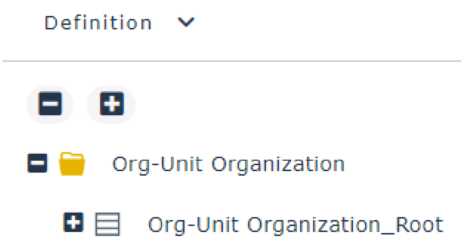
8. (If you want to collect several objects) In the **Multiplicity** field, select "N".
In that case, at TreeSet creation the user can enter several objects (Entry Point Collection values), which are taken into account to build the TreeSet.



☛ *For an example, see How to Add Several Objects to the Root Collection.*

9. In your TreeSet Definition properties, display its **Definition** page.
The TreeSet Definition structure is created and includes the root TreeSetCollection.

```
Example: The "Org-Unit Organization" TreeSet Definition is created and
includes its root node.
```



10. Define the TreeSet Definition structure:

- you can add ***TreeSet Collections***

  Example: you can add the "Org-Unit Component" TreeSet Collection

  In the **Collection Provider** section:

  Collection is populated by: "MetaAssociationEnd"

  Populating MetaAssociationEnd: "Component".

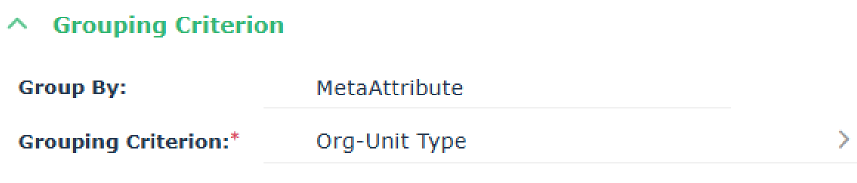
  See Adding a TreeSet Collection to a TreeSet Definition.

- you can add a ***grouping criterion*** on a TreeSet Collection

  Example: you can add a grouping criterion on the "Org-Unit Component" TreeSet Collection, in the **Grouping Criterion** section:

  Group by: "MetaAttribute", **Grouping Criterion**: "Org-Unit Type"

  See Adding automatic folders to a TreeSet Collection.



- you can add a ***loop*** on a TreeSet Collection

  Example: you can add a loop on the "Org-Unit Component" TreeSet Collection, in the **Recursive Definition** section:

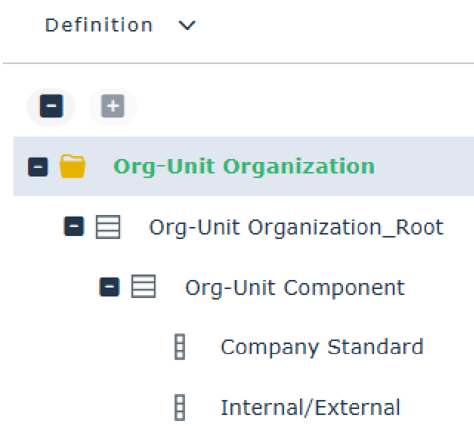
  TreeSet Collection loop: "Org-Unit Component" with a **Maximum loop depth** of "3".

  See Adding a loop to a TreeSet Collection.

- you can add ***TreeSet Properties***

  `Example: you can add two TreeSet Properties to the "Org-Unit Component" TreeSet Collection: two MetaAttribute values of the Org-Unit Component.`

  ☛ *See Adding TreeSet Properties to a TreeSet Collection.*



11. Display the TreeSet Definition **Preview** page and test the report.
    The TreeSet is displayed as a TreeTable. It shows its first three levels expanded, with objects in rows and corresponding Properties in columns.

    ☛ *See Previewing the TreeSet.*

Creating a query-based TreeSet Definition

When you create a **TreeSet Definition** based on a query, you need to define:

- its root MetaClass
- its populating query, which is based on the root MetaClass selected

You can use a provided query or create your own query.

`For example you can create a query that retrieves all the Applications with "5 stars" ranking, get their Software Technologies, and for each Technology its Vendor and risks.`

To create a query-based TreeSet Definition:

1. Access the **TreeSet Definitions** folder

   ☛ *See Accessing the TreeSet Definitions.*
2. Right-click the **TreeSet Definitions** folder and select **New > TreeSet Definition**.
   The **Creation of TreeSet Definition** window appears
3. In the **Name** field, enter your TreeSet Definition name.

   ☛ *By default the TreeSet Definition name is: TreeSet Definition-x (x is a number).*

   `Example: 5-star Appli.`
4. In the **Root** section, in the **Collection is populated by** field, select "Query".
   The **Populating Query** field appears.

5. In the **MetaClass** field, click the arrow and in the drop-down list select the MetaClass you want to define as the root MetaClass of your TreeSet Definition.

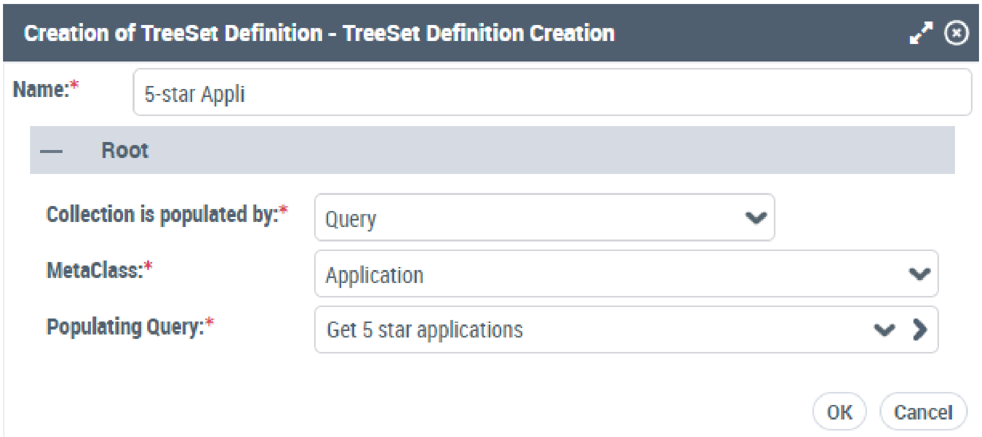
   ☺ *In the field, enter the first letters of the MetaClass for a quick access.*

   `Example: Application.`

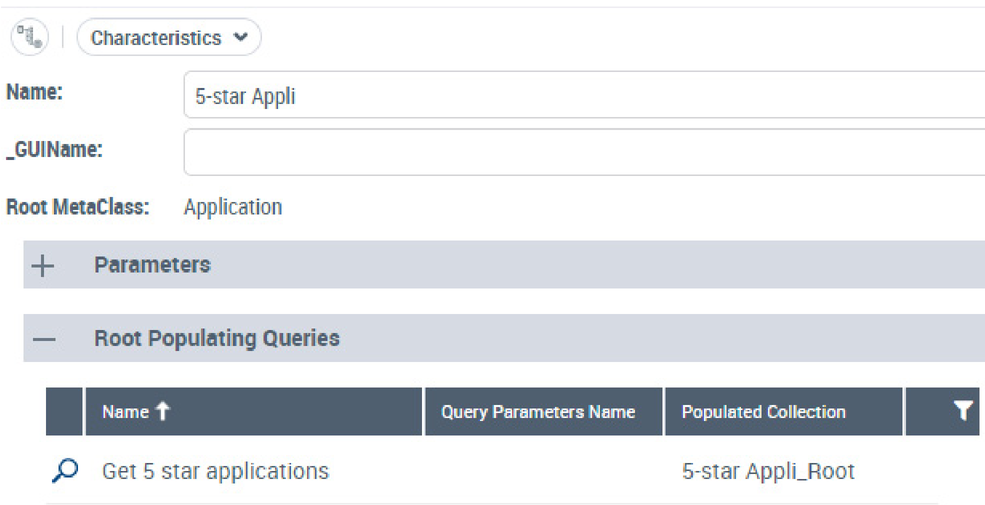
6. In the **Populating Query** field use the drop-down menu to select the query used to collect the entry point.

   ☛ *Only the root MetaClass related queries are available. You can also create your own query.*

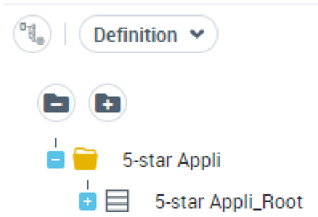
   `Example: "Get 5 star Applications".`



7. Click **OK**.
   The selected MetaClass is defined as your TreeSet Definition root MetaClass.
   Your TreeSet Definition is created and added to the **TreeSet Definition** folder.
8. In your TreeSet Definition properties, display its **Characteristics** page.
   It displays the **Root MetaClass**, and how to populate the TreeSet:
   The **Root Population Queries** section displays the query used to populate the TreeSet.



9. In your TreeSet Definition properties, display its **Definition** page.
The TreeSet Definition structure is created and includes the root node.



10. Define the TreeSet Definition structure:

- you can add TreeSet Collections

  ☛ *See Adding a TreeSet Collection to a TreeSet Definition.*

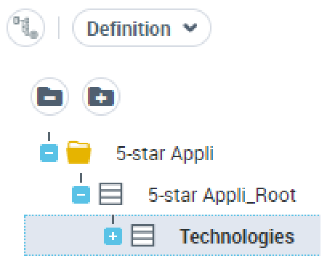
  Example: you can add the "Software Technology" TreeSet Collection.

  In the **Collection Provider** section:

  Collection is populated by: "MetaAssociationEnd"

  Populating MetaAssociationEnd: "Supporting Software Technology"

  Target MetaClass: "Software Technology"

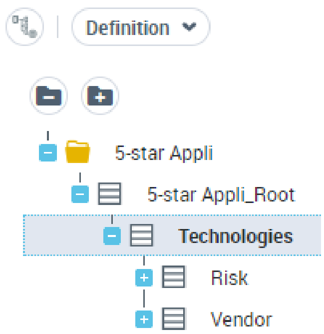


  Example: from Technologies, you can add the "Risk" and "Vendor" TreeSet Collections.

  In the **Collection Provider** section:

  Collection is populated by: "MetaAssociationEnd"

  Populating MetaAssociationEnd: "Risk" and "Vendor"



- you can add a loop to a TreeSet Collection

  ☛ *See Adding a loop to a TreeSet Collection.*

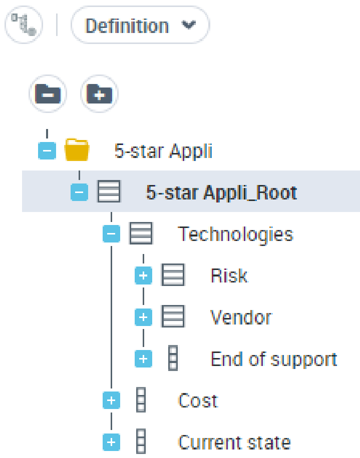
- you can add TreeSet Properties

  ☛ *See Adding TreeSet Properties to a TreeSet Collection.*

  Example: you can add:

  - one TreeSet Property to the "Technologies" TreeSet Collection: the "End of suport" MetaAttribute value of the Technology.

  - two TreeSet Properties to the "5-star Appli-Root" TreeSet Collection: the "Cost" and "Current state" MetaAttribute value of the Application.


Definition
5-star Appli
5-star Appli_Root
Technologies
Risk
Vendor
End of support
Cost
Current state

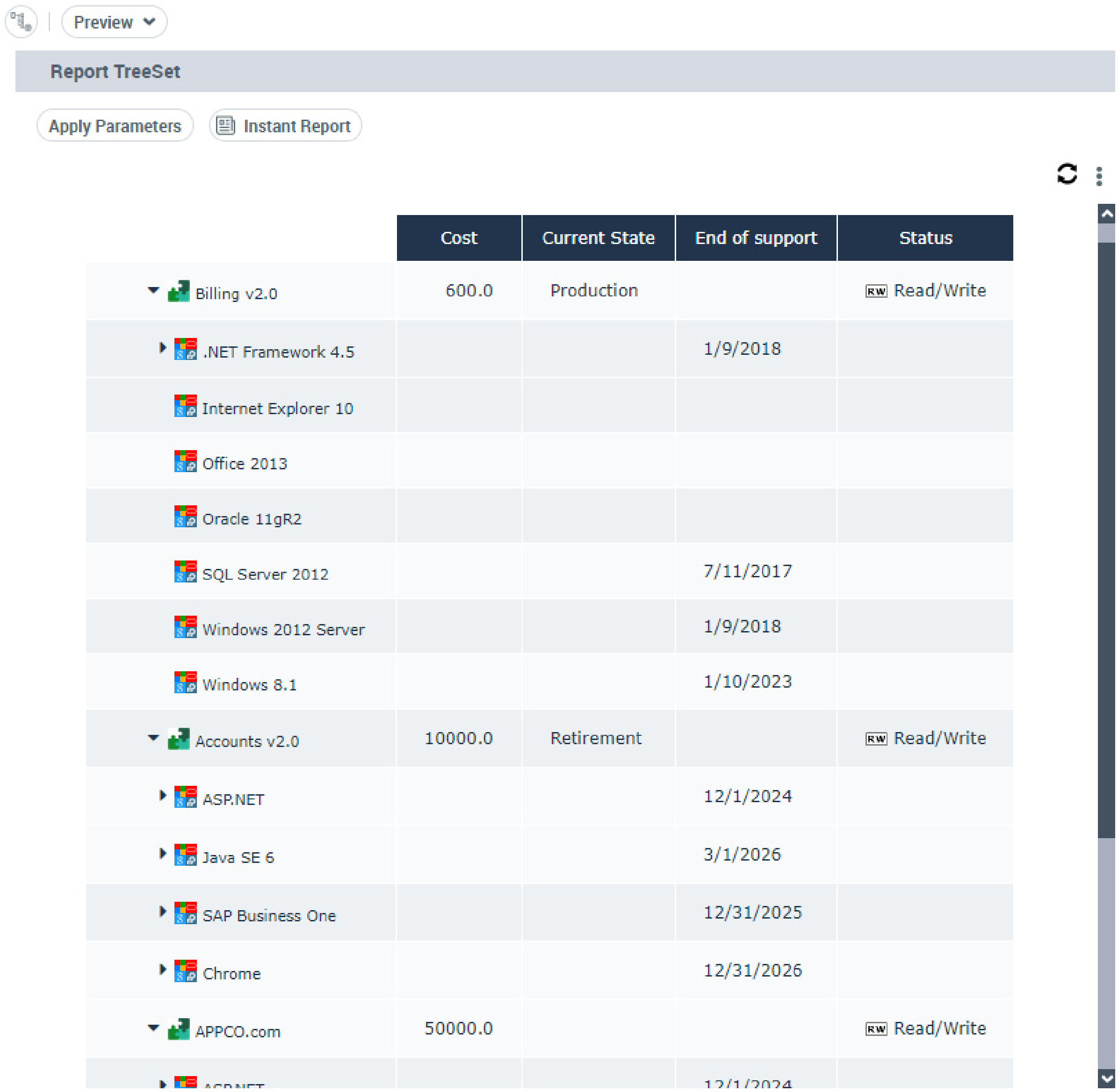
11. Display the TreeSet Definition **Preview** page and test the report.
The TreeSet is displayed as a TreeTable. It shows its first three levels expanded, with objects in rows and corresponding Properties in columns.

☛ *Note that folders (if any) count for levels two.*

Exemple: the TreeSet includes all of the applications with a 5-star ranking, and for each application its Software Technologies and their corresponding property values.

In this case there is a root folder (see Root node), so that only two levels are expanded, i.e. Risks and vendors are hidden.

☛ *See Previewing the TreeSet.*



12. (If you have more than 3 levels) Click the arrow next to each Object to display (/hide) one more level.

    Example: Vendors and Risks are displayed.

	Cost	Current State	End of support	Status
Billing v2.0	600.0	Production		RW Read/Write
Accounts v2.0	10000.0	Retirement		RW Read/Write
ASP.NET			12/1/2024	
ArtinSoft				
Insufficient market awareness				
Insufficient budget				
Java SE 6			3/1/2026	
Adobe Systems				
Contractual risk				
SAP Business One			12/31/2025	
Chrome			12/31/2026	
APPCO.com	50000.0			RW Read/Write

13. Click **Instant Report** to create your Tree type report as a:
    - Breakdown
    - Dendrogram
    - TreeMap
    - TreeTable

    With the same TreeSet, you can create Tree reports of each type.

    ☛ *See Displaying a Breakdown, a Dendrogram, a TreeMap, or a TreeTable.*

Creating property parameters

In your TreeSet Definition, you can create and define **Property Parameters** that you can use in any query used in the TreeSet Definition.

At TreeSet creation the user is asked to enter the parameter values, which are taken into account in the query.

To add a property parameter to the TreeSet Definition:

1. Access the TreeSet Definition properties.

   ☛ *See Accessing the TreeSet Definitions.*

2. Display the **Characteristics** page.
3. In the Parameters section, in the **Property Parameters** list, click **New** ✚.

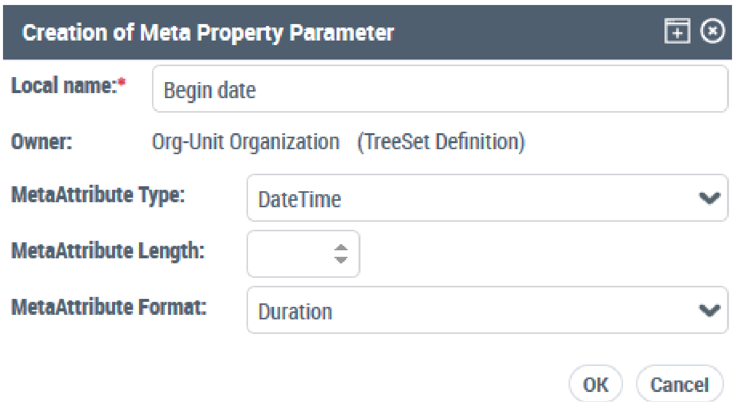
   ☛ *You can add as many parameters as needed.*

4. In the **Meta Property Parameter** creation window:
   - enter a **Local Name** (name displayed in the TreeSet)

     `Example: Begin date`

   - (according to your needs) modify the default values for **MetaAttribute type** ("String"), **MetaAttribute Length** and **MetaAttribute Format** ("Standard")

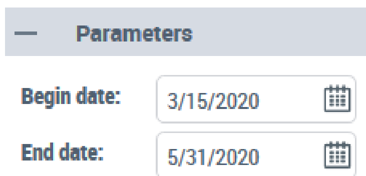
     ☛ *For detailed information regarding MetaAttribute characteristics, see HOPEX Studio > Customizing the Metamodel : Managing the Metamodel : MetaAttributes : MetaAttribute Characteristics documentattion.*



5. Click **OK**.
   The property parameter is added to the **Property Parameters** list.
6. In the **Query Parameter Name** field, enter a name for the parameter. This name is the one used in the query.

Example:

`When you add "Begin Date" and "End Date" Query Parameters, at TreeSet creation the user is asked to enter both dates, which filter the TreeSet result according to the these dates.`



DEFINING THE DATA THAT FEEDS THE TREESET DEFINITION

See:

- Introduction to TreeSet Definition Properties
- Adding a TreeSet Collection to a TreeSet Definition
- Removing a TreeSet Collection from a TreeSet Definition
- Defining a TreeSet Collection
- Adding TreeSet Properties to a TreeSet Collection

Introduction to TreeSet Definition Properties

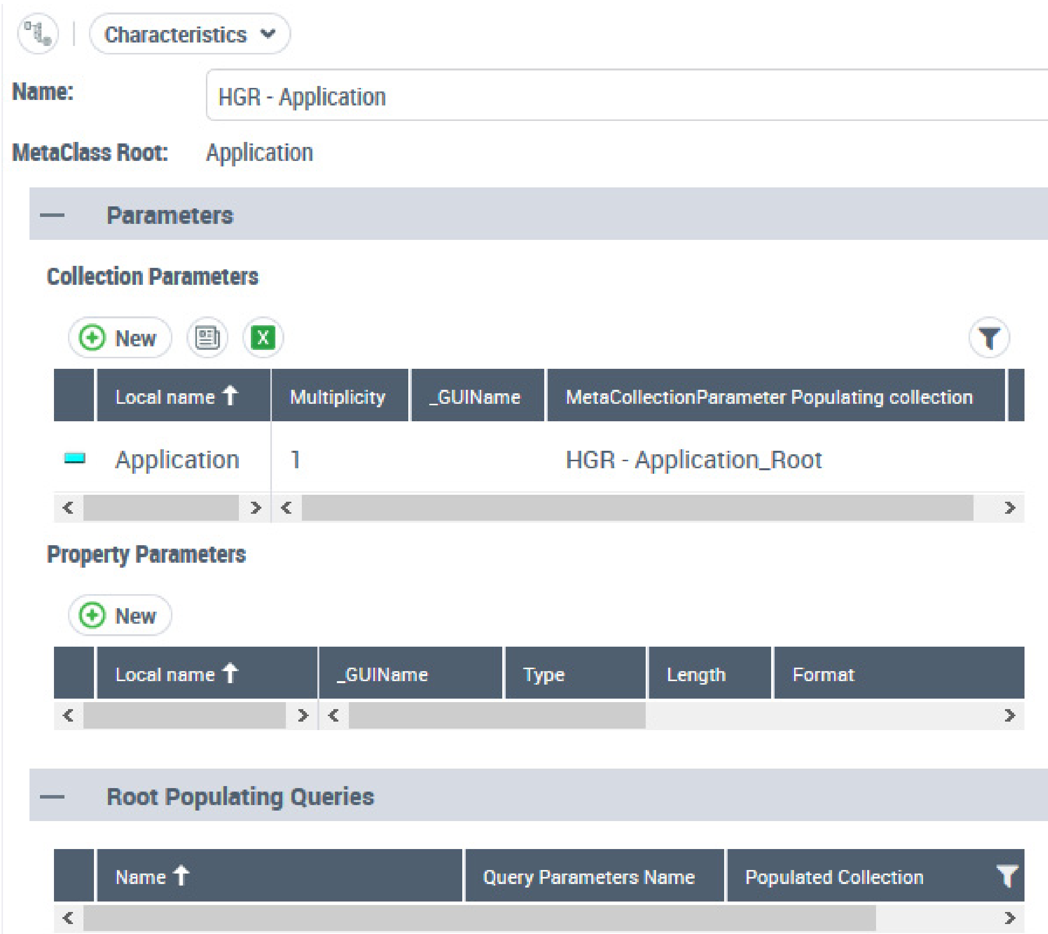
Characteristics page

In the TreeSet Definition properties, the **Characteristics** page sums up the entry points used in the TreeSet Definition:

- the root MetaClass

  `Example: Application`
- how data is collected, via:
  - a collection, or

    `Example: Application collection`
  - a query
- it enables to create property parameters that you can use in queries

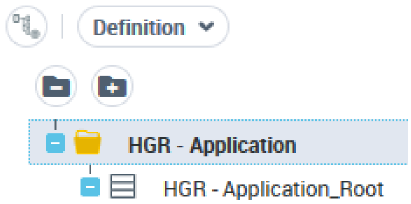


Definition page

In the TreeSet Definition properties, the **Definition** page displays the structure of the TreeSet Definition.

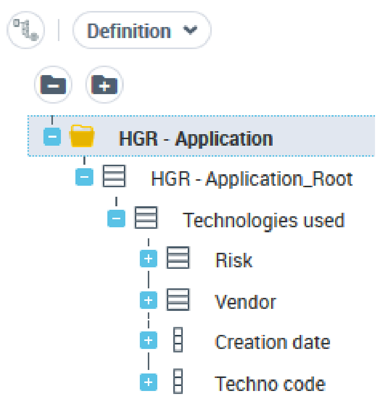
At creation it displays:

- the root TreeSet node (folder)
- the root TreeSet Collection



It enables to:

- configure the root TreeSet Collection
  - ☛ *See Defining a TreeSet Collection.*
- add TreeSet Collections
  - ☛ *See Adding a TreeSet Collection to a TreeSet Definition.*
- configure each TreeSet Collection
  - add TreeSet properties to the TreeSet Collection
  - hide the TreeSet Collection itself
  - hide its nodes with no child
  - (recursive hierarchy) add a loop on the TreeSet collection
  - ☛ *See Defining a TreeSet Collection.*
- add TreeSet Properties
  - ☛ *See Adding TreeSet Properties to a TreeSet Collection.*



Adding a TreeSet Collection to a TreeSet Definition

At creation a TreeSet Definition includes a single TreeSet Collection: the root TreeSet Collection. You may need to add one or several TreeSet Collections to the TreeSet Definition structure.

To add a TreeSet Collection you must define:

- where you want to add the TreeSet Collection in the TreeSet Definition structure
- how you want to populate the TreeSet Collection, with:
  - a parameter
  - a MetaAssociationEnd
  - a query
  - a TreeSet Collection reference

Adding a TreeSet Collection populated by a MetaAssociationEnd

To add a TreeSet Collection populated by a MetaAssociationEnd:

1. Access the TreeSet Definition properties.

   ☛ *See Accessing the TreeSet Definitions.*

2. Display its **Definition** page.
3. In the TreeSet Definition, right-click the TreeSet Collection (or the root) under which you want to add a TreeSet Collection, and select **New > TreeSet Collection**.
4. Enter a **Local name** to the TreeSet Collection.
5. In the **Collection is populated by** drop-down list, keep "MetaAssociationEnd".
6. In the **Populating MetaAssociationEnd** drop-down list, select the MetaAssociationEnd.

   ☛ *The list includes available MetaAssociationEnds only, i.e. the MetaAssociationEnds of the parent TreeSet Collection.*

7. Click **OK**.
   The TreeSet Collection is added to the TreeSet Definition under the parent TreeSet Collection.
8. (If needed) Define the TreeSet Collection.

   ☛ *See Defining a TreeSet Collection.*

Adding a TreeSet Collection populated by a query

You can use a provided query or create your own query.

To add a TreeSet Collection populated by a query:

1. Access the TreeSet Definition properties.

   ☛ *See Accessing the TreeSet Definitions.*

2. Display its **Definition** page.
3. In the TreeSet Definition, right-click the TreeSet Collection (or the root) under which you want to add a TreeSet Collection, and select **New > TreeSet Collection**.
4. Enter a **Local name** to the TreeSet Collection.
5. In the **Collection is populated by** drop-down list, select "Query".
   The **Populating Query** field is displayed and includes the list of available queries only, i.e. the queries related to the parent TreeSet Collection.
6. In the **Populating Query** drop-down list, select the query.

   ☛ *Else click the right-oriented arrow to create a query.*

7. Click **OK**.
   The TreeSet Collection is added to the TreeSet Definition under the parent TreeSet Collection.
8. (If needed) Define the TreeSet Collection.

   ☛ *See Defining a TreeSet Collection.*

Adding a TreeSet Collection populated by a parameter

To add a TreeSet Collection populated by a parameter:

1. Access the TreeSet Definition properties.

   ☛ *See Accessing the TreeSet Definitions.*

2. Display its **Definition** page.
3. In the TreeSet Definition, right-click the TreeSet Collection (or the root) under which you want to add a TreeSet Collection, and select **New > TreeSet Collection**.
4. Enter a **Local name** to the TreeSet Collection.
5. In the **Collection is populated by** drop-down list, select "Parameter".
   The **Populating Parameter** field is displayed and includes the list of available parameters only, i.e. the parameters related to the parent TreeSet Collection.
6. In the **Populating Parameter** drop-down list, select the parameter (which is a Collection Parameter).
7. Click **OK**.
   The TreeSet Collection is added to the TreeSet Definition under the parent TreeSet Collection.
8. (If needed) Define the TreeSet Collection.

   ☛ *See Defining a TreeSet Collection.*

Adding a TreeSet Collection reusing part of the definition

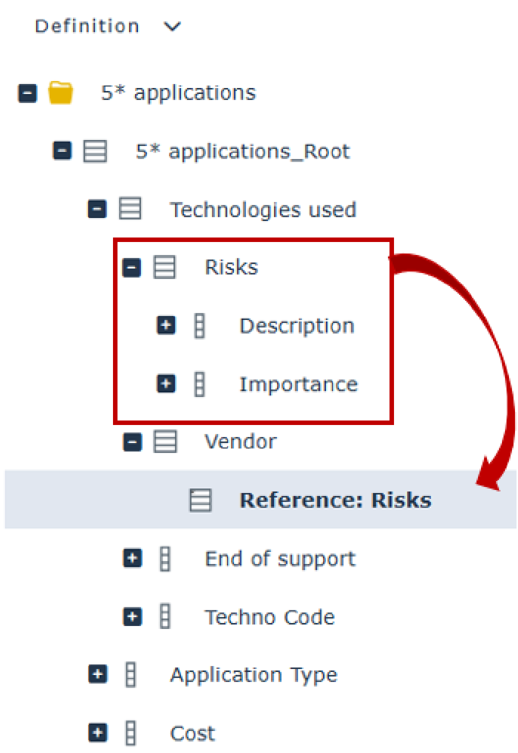
You can add a TreeSet Collection which reuses a TreeSet Collection as a reference.

This Referenced TreeSet Collection can be either:

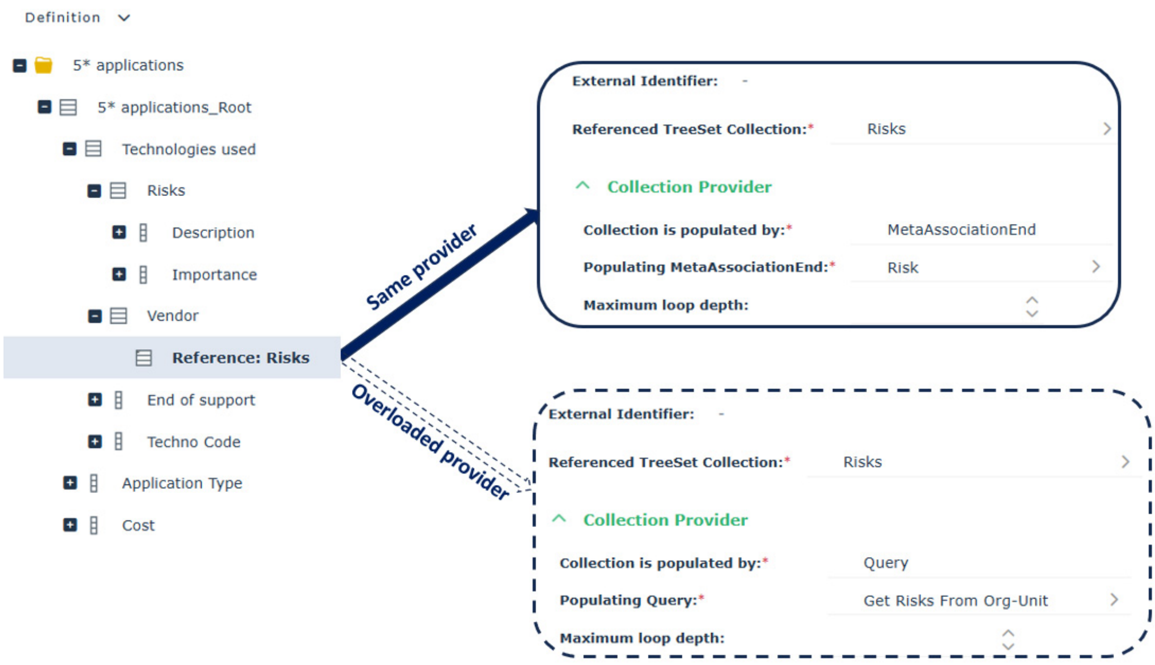
- a parent TreeSet Collection, or
- a same level TreeSet Collection

When adding a referenced TreeSet Collection:

- it automatically adds the TreeSet Collection and its children (properties and/or TreeSet Collections). Thus, you do not need to describe this part of the definition again.



- you can feed it with the same collection provider or overload it.



To add a TreeSet Collection reference:

1. Access the TreeSet Definition.

   ☛ *See Accessing the TreeSet Definitions.*

2. Display its **Definition** page.
3. In the TreeSet Definition, right-click the TreeSet Collection (or the root) under which you want to add the TreeSet Collection reference, and select **New > TreeSet Collection Reference**.
4. In the **Referenced TreeSet Collection** field, select the TreeSet Collection.
5. In the **Collection Provider** section, set its settings as described in the above sections (collection populated by a parameter, a MetaAssociationEnd, or a query).
6. Click **OK**.

Removing a TreeSet Collection from a TreeSet Definition

When you remove a TreeSet Collection from a TreeSet Definition, you also remove all its child TreeSet Collections and TreeSet Properties.

You can choose to remove the TreeSet Collection from the TreeSet Definition only or delete it from HOPEX repository.

To remove a TreeSet Collection from a TreeSet Definition:

1. Access the TreeSet Definition folders.

   ☛ *See Accessing the TreeSet Definitions.*

2. Expand the TreeSet Definition concerned.

3. Right-click the TreeSet Collection you want to remove from the TreeSet Definition and select **Remove** .
   A confirmation dialog box prompts you to confirm your choice to remove the TreeSet Collection from its TreeSet Collection parent only or delete it from HOPEX repository.

   ☛ *if you want to delete the TreeSet Collection from the HOPEX repository, select **Delete** and confirm your choice in the next window.*
4. Click **Remove**.
   The TreeSet Collection is removed from the TreeSet Definition, but still available in the HOPEX repository.

Defining a TreeSet Collection

By default a TreeSet Collection is displayed in the tree, and all of its nodes are visible. If needed, you can hide:

- the entire TreeSet Collection
  This is useful for example to define tree items that are used as filters but do not need to appear in the tree display.
- the nodes of the TreeSet Collection that do not have any child

You can add folders to a TreeSet Collection:

- automatically
- manually

You can define a TreeSet Collection with a loop, which enables to add a recursive definition.

Hiding a TreeSet Collection

☛ *See example How to Hide a TreeSet Collection.*

To hide a TreeSet Collection:

1. Access the TreeSet Definition.

   ☛ *See Accessing the TreeSet Definitions.*
2. Display its **Definition** page.
3. Select the TreeSet Collection you want to hide.
4. In the right pane set **Visible** parameter to "No.

Hiding nodes with no child

By default, any node of the collection is displayed, even if a node has no child. You can hide nodes with no child.

To hide the node with no child of a TreeSet Collection:

1. Access the TreeSet Definition.

   ☛ *See Accessing the TreeSet Definitions.*
2. Display its **Definition** page.
3. Select the TreeSet Collection for which you want to hide nodes with no child.
4. In the right pane set **Must have a child** parameter to "Yes".

Adding automatic folders to a TreeSet Collection

You can add automatic folders to a TreeSet Collection. In that case you cannot change the name of the folders as when manually adding folders.

These folders are created by adding a grouping criterion on a TreeSet Collection.

You can group by:

- Object Type
- a MetaAttribute value
- a MetaAssociationEnd, or
- a Query

To add a folder to a TreeSet Collection:

1. Access the TreeSet Definition.

   ☛ *See Accessing the TreeSet Definitions.*

2. Select the TreeSet Collection concerned by the grouping, and in the **Grouping Criterion** section define the grouping:
   - in **Group by**, select the grouping type
   - if you selected "MetaAttribute", "MetaAssociationEnd" or "Query", select the **Grouping Criterion**

Adding folders to a TreeSet Collection

You can add a folder to a TreeSet Collection. You can add the folder as you create the Definition or once the Definition is created.

☛ *See example How to Add a Folder to a TreeSet Definition.*

To add a folder to a TreeSet Collection:

1. Access the TreeSet Definition.

   ☛ *See Accessing the TreeSet Definitions.*

2. Right-click the TreeSet Collection under which you want to add a folder and select **New > TreeSet Collection**.
3. Define the folder:
   - **Local name**: enter the name of the folder
   - **Is folder**: yes
4. (If needed) Drag and drop the required TreeSet Collection(s) in the folder.

Adding a loop to a TreeSet Collection

With a recursive hierarchy, you can add a collection loop to the TreeSet Collections.

To add a loop to a TreeSet Collection:

1. Access the TreeSet Definition.

   ☛ *See Accessing the TreeSet Definitions.*

2. Display its **Definition** page.
3. Select the TreeSet Collection for which you want to add a loop.
4. In the right pane, in the **Recursive Definition** section define the loop:
   - in the **TreeSet Collection loop** field, select the "<TreeSet Collection name>".
   - (optional) in the **Maximum loop depth** enter the number of levels you want to add.

Adding TreeSet Properties to a TreeSet Collection

According to your needs, you can add TreeSet Properties to TreeSet Collections (even to the root one).

These properties are displayed in the **TreeTable** renderer only, and can be hidden if needed.

☛ *Properties are not visible in Breakdown type reports.*

The ***TreeSet Property*** can be:

- a value
  - MetaAttribute value
  - abstract property value
- a count
  - MetaAssociationEnd
  - query
- a computation (with a macro)
- an object

TreeSet Property access

By default, the TreeSet property is provided by the source object (**Access** section: "Direct").

You may want to provide the property by a MetaAssociationEnd of the source object (**Access** section: "MetaAssociationEnd").

Adding a value-based property to a TreeSet Collection

You can add value-based properties to a TreeSet Collection.

To add a value-based property to a TreeSet Collection:

1. Access the TreeSet Definition properties.

   ☛ *See Accessing the TreeSet Definitions.*
2. Display its **Definition** page.
3. Expand the TreeSet Definition structure.
4. Right-click the TreeSet Collection concerned and select **New > TreeSet Property**.
   The **Creation of TreeSet Property** windows is displayed.
5. In the **Local name** field, modify the default name.
   In the **Configuration** section, configure the property.
6. In the **Property Type** drop-down list:
   - keep "Value (MetaAttribute)", to base the value on a MetaAttribute, or
   - select "Value (Abstract Property)" to base the value on an Abstract Property.
7. Depending on the value type:
   - in the **MetaAttribute** drop-down list, select the MetaAttribute you want to be displayed in the property.
   - in the **TreeSet Abstract Property** field, select the Abstract Property you want to be displayed in the property.

   ☛ *Else click the right oriented arrow and connect the Abstract Property.*
8. Click **OK**.

Adding a count-based property to a TreeSet Collection

You can add count-based properties to a TreeSet Collection.

The property is collected locally for the TreeSet, with the count retrieved from a query or a MetaAssociationEnd.

To add a count-based property to a TreeSet Collection:

1. Access the TreeSet Definition properties.

   ☛ *See Accessing the TreeSet Definitions.*
2. Display its **Definition** page.
3. Expand the TreeSet Definition structure.
4. Right-click the TreeSet Collection concerned and select **New > TreeSet Property**.
   The **Creation of TreeSet Property** windows is displayed.
5. In the **Local name** field, modify the default name.
   In the **Configuration** section, configure the property.
6. In the **Property Type** drop-down list, select the count type:
   - "count (MetaAssociationEnd)", or
   - "count (Query)".
7. In the:
   - **Counted MetaAssociationEnd** field, select the MetaAssociationEnd in the list, or
   - **Counted Query** field, select the query in the list.

Adding a macro-based property to a TreeSet Collection

You can add computed properties to a TreeSet Collection. The property is computed locally, with a macro, for the TreeSet.

Prerequisite:

For each macro-based property you have to create its implementing macro. This macro is based on the **GetAttributeValue** function:

GetAttributeValue(Object as MegaOject,AttributeID as Variant,Value As String)

Where:

Object is the object of which the attribute value is requested.

AttributeID is the absolute identifier of the attribute.

Value is the attribute value returned by the function, concerning this object.

To add a macro-based property to a TreeSet Collection:

1. Access the TreeSet Definition properties.

   ☛ *See Accessing the TreeSet Definitions.*
2. Display its **Definition** page.
3. Expand the TreeSet Definition structure.
4. Right-click the TreeSet Collection concerned and select **New > TreeSet Property**.
   The **Creation of TreeSet Property** windows is displayed.
5. In the **Local name** field, modify the default name.
   In the **Configuration** section, configure the property.
6. In the **Property Type** drop-down list, select "Computed".

7. Define the MetaAttribute characteristics:
   - **MetaAttribute Type** (default value "String)
   - **MetaAttribute Length**
   - **MetaAttribute Format** (default value "Standard)

   ☛ *For detailed information regarding MetaAttribute characteristics, see HOPEX Studio > Customizing the Metamodel : Managing the Metamodel : MetaAttributes : MetaAttribute Characteristics documentattion.*
8. In the **Implementation** field, click the right-oriented arrow and connect the macro that enables to compute the field value.
9. Click **OK**.

Adding an object-based property to a TreeSet Collection

In case of "MetaAssociationEnd" **Access type**, you might want to add the object itself.

To add an object-based property to a TreeSet Collection:

1. Access the TreeSet Definition properties.

   ☛ *See Accessing the TreeSet Definitions.*
2. Display its **Definition** page.
3. Expand the TreeSet Definition structure.
4. Right-click the TreeSet Collection concerned and select **New > TreeSet Property**.
   The **Creation of TreeSet Property** windows is displayed.
5. In the **Local name** field, modify the default name.
6. Expand the **Access** section, and select:
   - in **Access type**: select "MetaAssociationEnd"
   - in **MetaAssociationEnd**: select the MetaAssociationEnd concerned
7. In the **Configuration** section, select:
   - **Object property type**: Object

HANDLING A TREESET

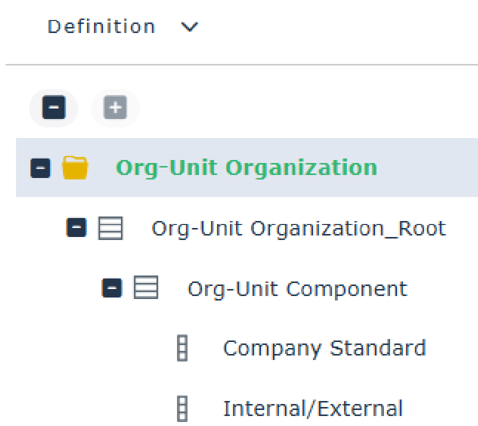
See:

- Previewing the TreeSet
- Generating Instant Reports from a TreeSet
- Saving the Tree Report

Previewing the TreeSet

From the TreeSet Definition properties you can test the TreeSet.

For example, you can test the Org-Unit Organization TreeSet Definition.



To preview the TreeSet:

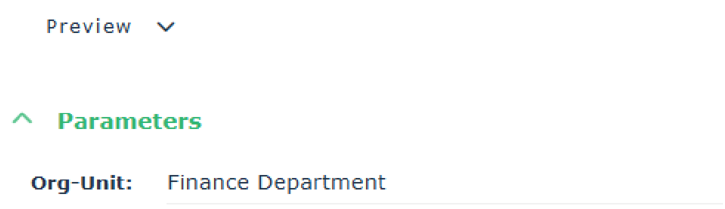
1. Access the TreeSet Definition properties.

   ☛ *See Accessing the TreeSet Definitions.*

2. Display its **Preview** page.

3. (If needed) In the **Parameters** section, define the parameters:
   - (if collection parameter-based TreeSet Definition) in the **<Collection Parameter GUIName>** field, click **Connect** and connect the MetaClass item required.

     `Example: in the` **`Org-Unit`** `field, which is the` **`Collection Parameter GUIName`**`, connect "Finance Department".`

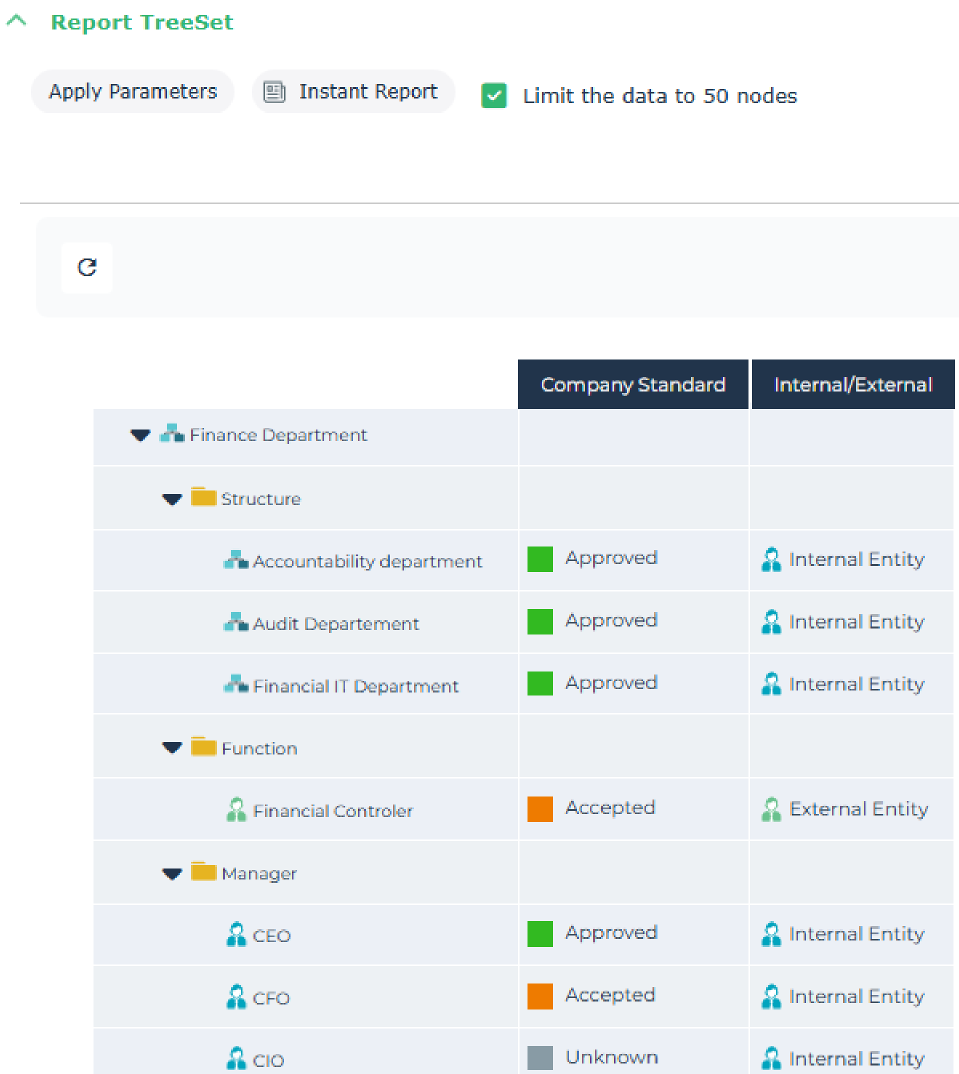
   - (if property parameters are used) enter the parameter values.

     ☛ *See* *Creating property parameters.*
4. (If needed, to get more space for the report) In the **Parameters** section title, click to reduce the section.

5. Click **Apply Parameters**.
The TreeSet is displayed as a **TreeTable** renderer type.
Three levels of the tree are expanded by default.
The display is limited to 50 nodes by default (clear **Limit the data to 50 nodes** to display more data).

`Ex.: this TreeSet is based on the Finance Department` **`Org-Unit`** `and displays for each Org-Unit child its` **`Company Standard`** `and` **`Internal/External`** `values.`

`Org-Unit children are grouped by type in the corresponding` **`Org-Unit Type`** `folder.`



6. In the **Report TreeSet** section, click **Instant Report** to create instant report of the following types:
   - Breakdown
   - Dendrogram
   - TreeTable
   - TreeMap

Generating Instant Reports from a TreeSet

From the **Preview** page of the TreeSet Definition, you can generate instant reports of the following rendering types:

- Breakdown
- Dendrogram
- TreeTable
- TreeMap

Objects from a Data Reading access not accessible to the user are not displayed in the report.

The Instant report is generated with a **Configuration** pane, which enables to modify the report configuration.

Breakdown

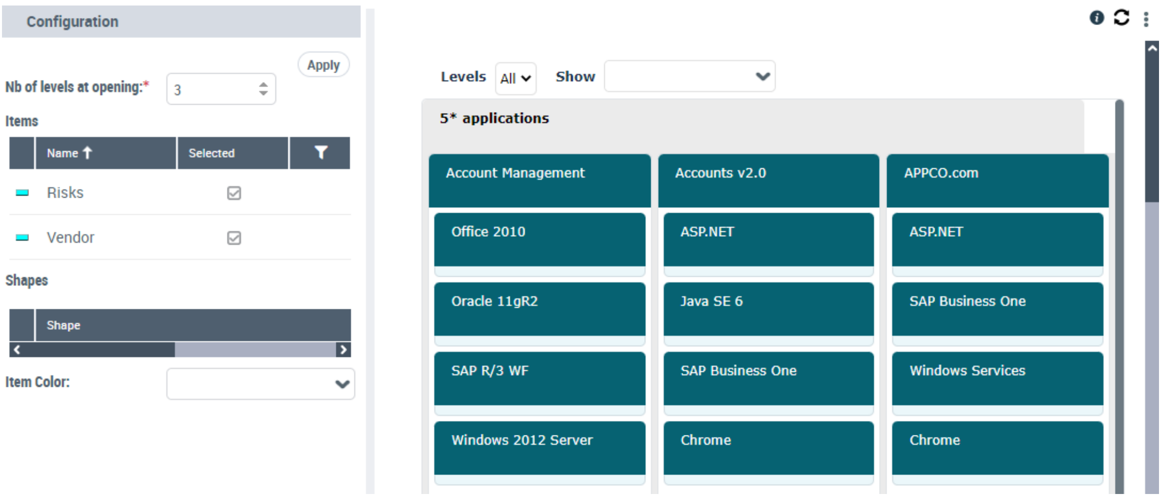
With the Breakdown type report, properties (when defined) are not displayed.

To generate a breakdown type report:

1. Access the TreeSet Definition.

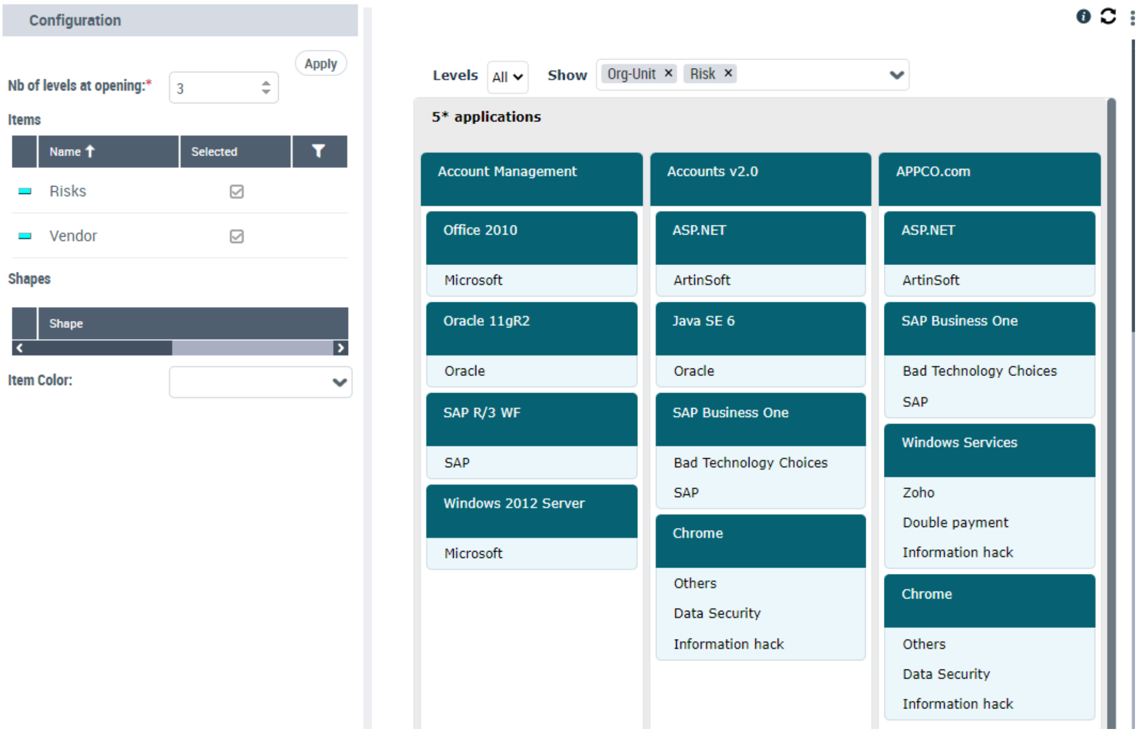
2. From its **Preview** page, click **Instant Report** and select **Breakdown**.
   By default, at opening, the breakdown map displays:
   - a maximum of three levels of Components
   - no items

   ```
   E.g.: this Breakdown map is based on "Account Management", "Accounts
   V2.0", and "APPCO.COM" applications, which all have a ranking of 5 stars.
   For each of them, its Technologies are displayed.
   ```



3. To handle the breakdown map:
   - In the **Levels** field, select the required level.
   - In the **Show** field, click the arrow and select the items you want to display.

     `E.g.: select` **`Org-Unit`** `and` **`Risk`**`.`



4. In the **Configuration** section, you can modify the **Properties** displayed:
   - for the property you do not want to display in the TreeMap, clear its **Selected** check box.
   - Click **Apply**.

     ☛ *In the Breakdown report, click **Refresh** to see your modifications.*
5. Save the Breakdown map to create a Report Template or a Tree View.

   ☛ *See Saving the Tree Report.*

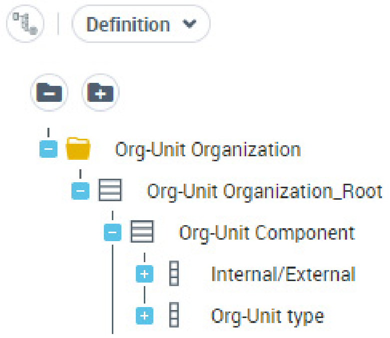
Dendrogram

With the Dendrogram type report, properties (when defined) are displayed on object in a pop-up window.

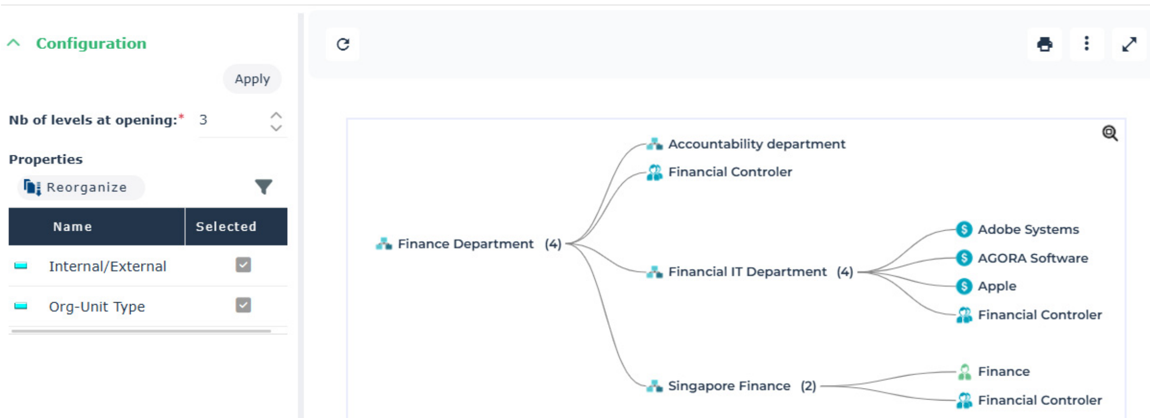
To create a Dendrogram type report:

1. Access the TreeSet Definition.

   E.g.: this TreeSet Definition is based on an **Org-Unit** and displays for each Org-Unit child its **Internal/External** and **Org-Units Type** values.

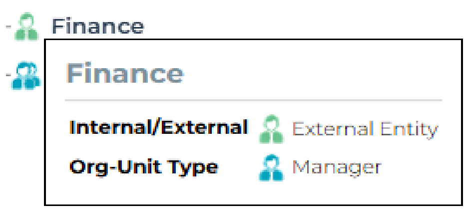


2. From its **Preview** page, click **Instant Report** and select **Dendrogram**.
   By default, the dendrogram displays three levels.



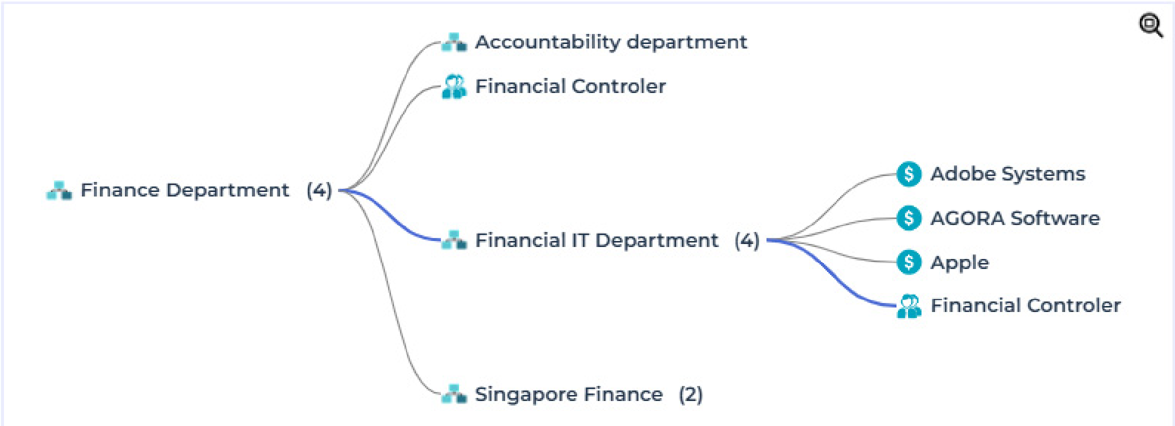
3. To view the Properties (if any): hover the cursor over a node.

   Example: hover the cursor over "Marketing Departement", its "Internal/External" and "Org-Unit Type" values are displayed in a pop-up window.



4. In the **Configuration** section, you can modify the display:
   - Modify the number of expanded levels at opening: in the **Nb of levels at opening**, select the required number.
   - Reorganize the property order in the pop-up window: click **Reorganize**, and drag and drop the properties so as to get another display order, and click **OK**.
   - Click **Apply** and **Refresh** the report.
5. To handle the dendrogram:
   - Click a node to hide its daughter branches.

     `Example: click "Singapore Finance" component, its two daughter branches are hidden.`



   - Below the graph, click **Options** to:
   - reduce/increase the space between the nodes
   - reduce the label length
   - Use the mouse wheel to zoom in (/out) the graphic.
6. Save the Dendrogram to create a Report Template or a Tree View.

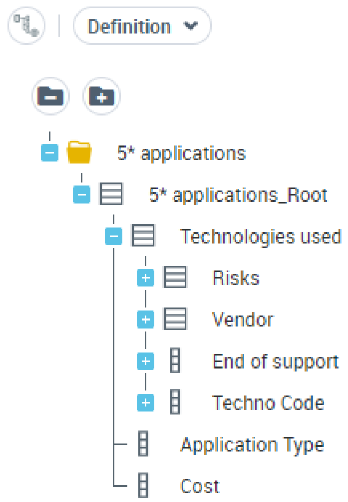
   ☛ *See Saving the Tree Report.*

TreeMap

With the TreeMap type report, properties (when defined) are displayed on object in a pop-up window.

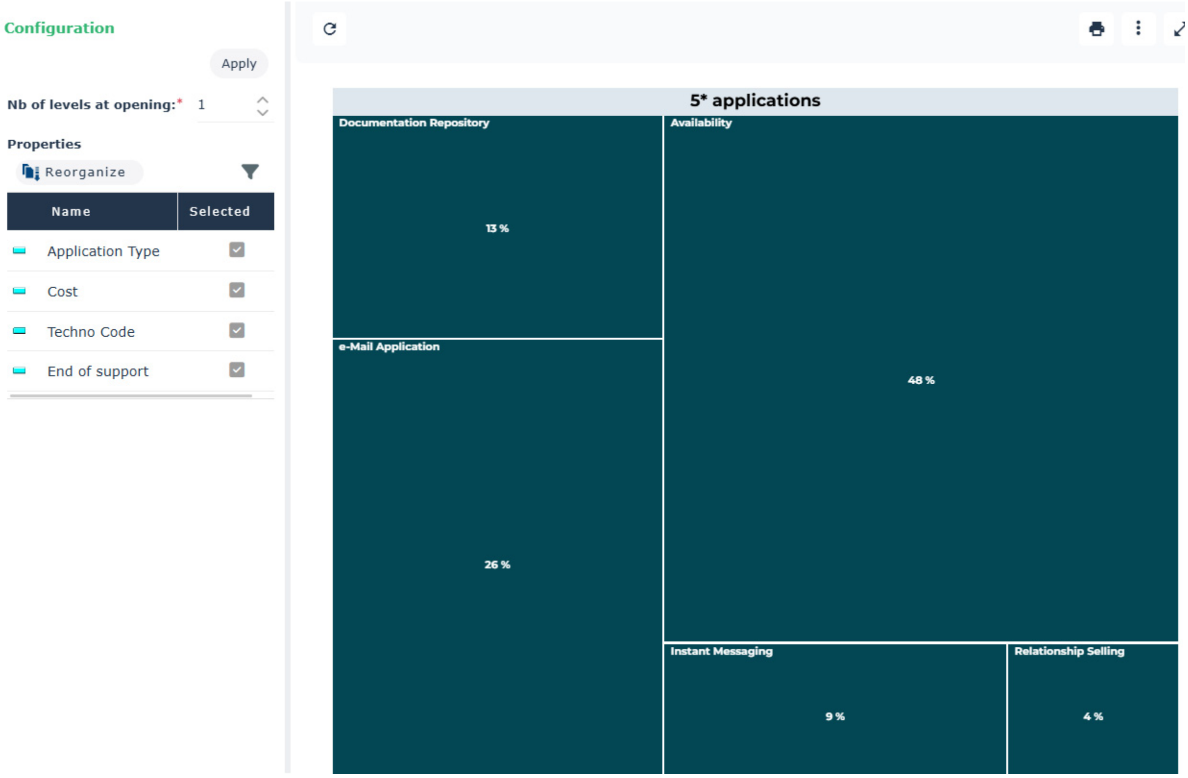
To create a TreeMap type report:

1. Access the TreeSet Definition.



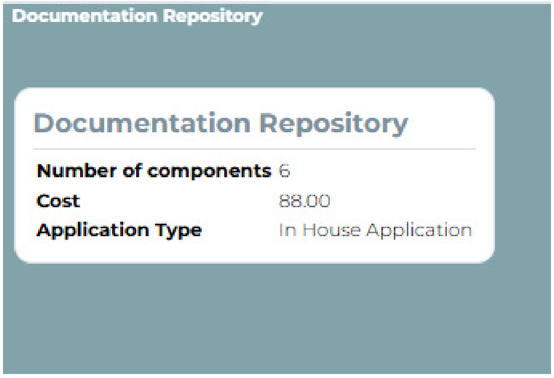
2. From its **Preview** page, click **Instant Report** and select **TreeMap**.
By default, the TreeMap displays only one level at opening.

    E.g.: this TreeMap is based on five applications, which all have a ranking of 5 stars.



3. To view the Properties (if any): hover the cursor over a box.
Its properties are displayed in a pop-up window.

    Example: hover the cursor over "Documentation Repository" Application, its "Cost" and "Application Type" values are displayed in a pop-up window.



4. In the **Configuration** pane, you can modify:
   - the number of levels displayed at opening: in the **Nb of levels at opening**, select the required number.
   - the **Properties** displayed: for the property you do not want to display in the TreeMap, clear its **Selected** check box.
   - Click **Apply**.

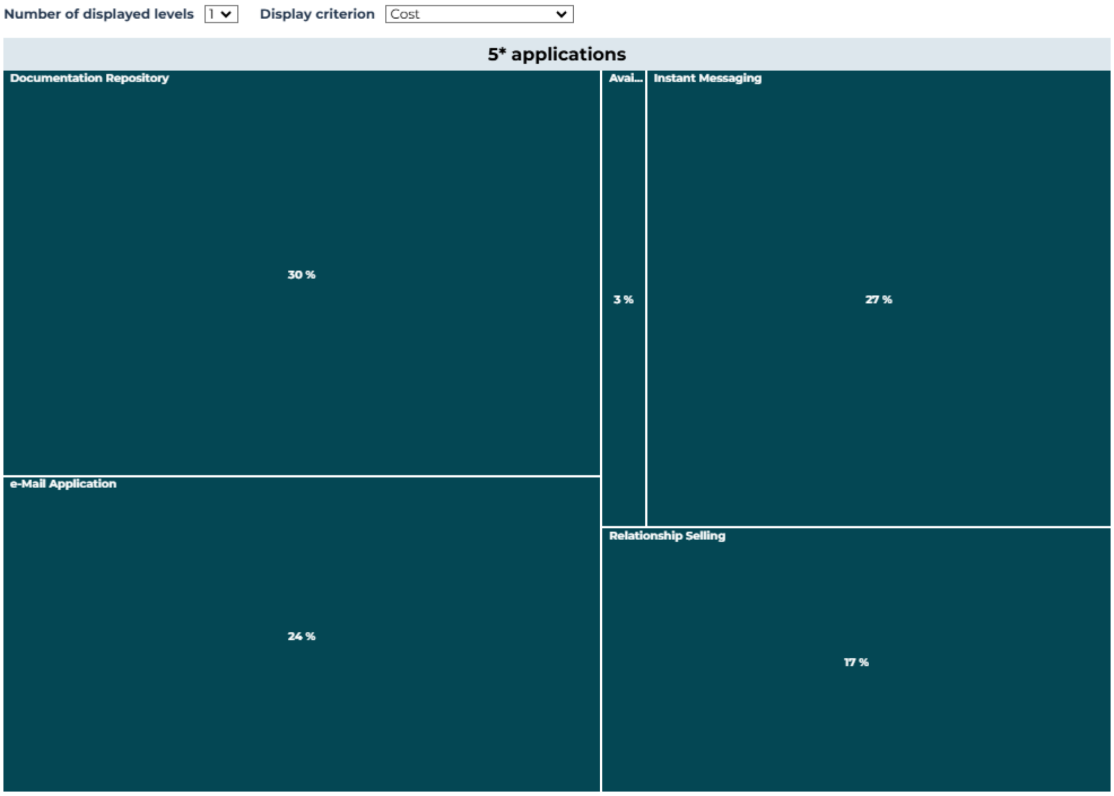
   ☛ *In the TreeTable report, click* ***Refresh*** *to see your modifications.*

5. To handle the TreeTable, above the TreeTable:

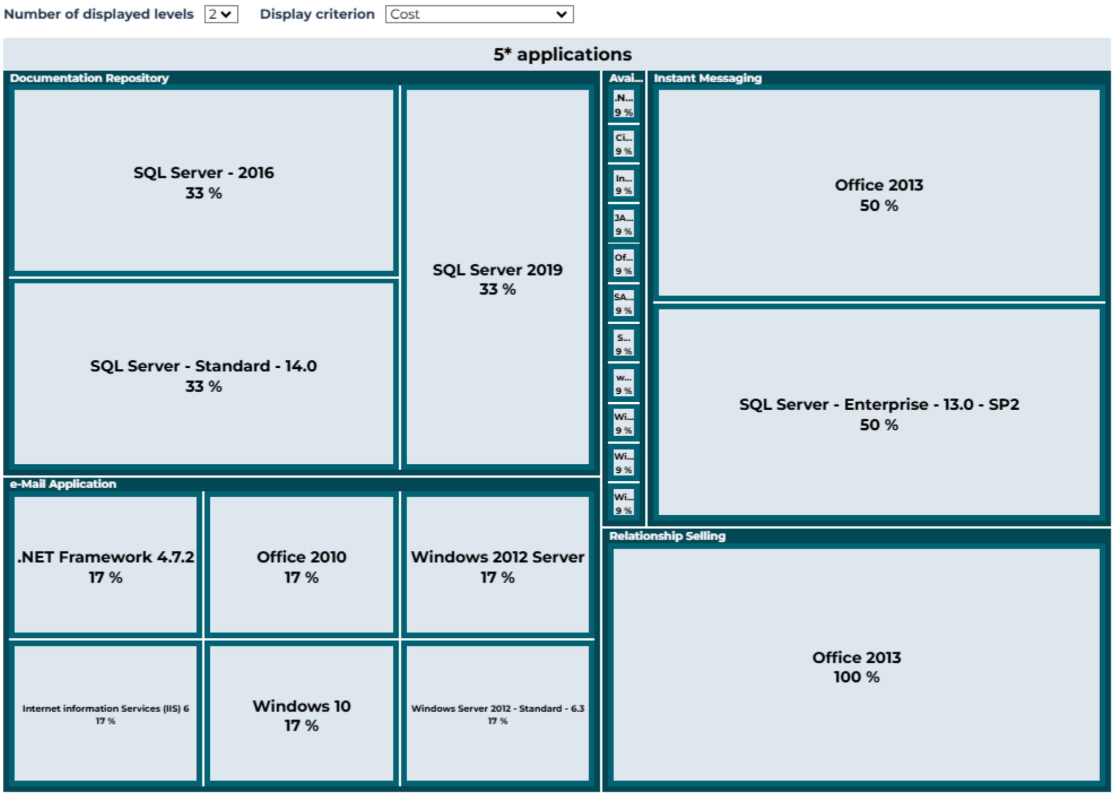
- modify the **Number of displayed levels**
- modify the **Display criterion** (when available)

  For example: display Applications according to their "Cost" ("Accounts V2.0": 45.000, "Account Management": 1.000, "APPCO.com": 5.000)

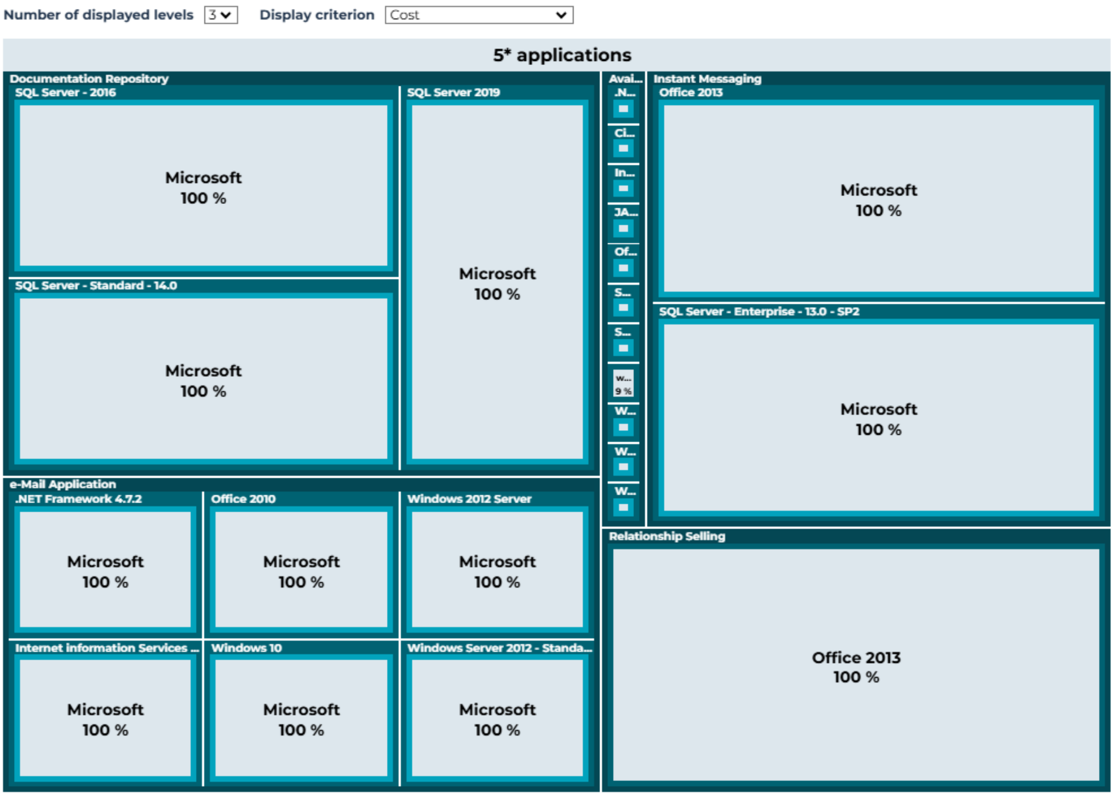
  - with 1 displayed level (so that Applications are displayed).



  - with 2 displayed levels (so that their Technologies are displayed).



- with 3 displayed levels (so that the Vendors are displayed).



6. Save the TreeMap to create a Report Template or a Tree View.
   ☛ *See Saving the Tree Report.*

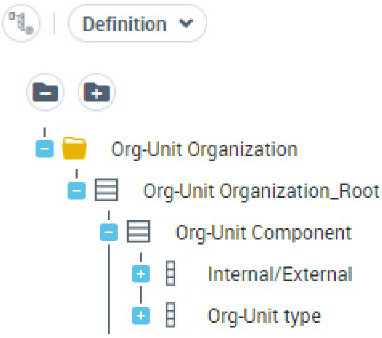
TreeTable

With the TreeTable type report, properties (when defined) define the Column headers of the table.

To create a TreeTable type report:

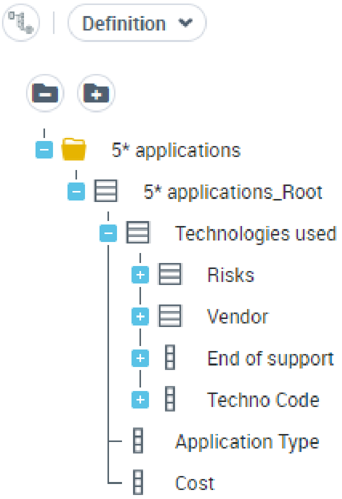
1. Access the TreeSet Definition.

   Ex. 1: this TreeSet Definition is based on an **Org-Unit** and displays for each Org-Unit child its **Internal/External** and **Org-Units Type** values.

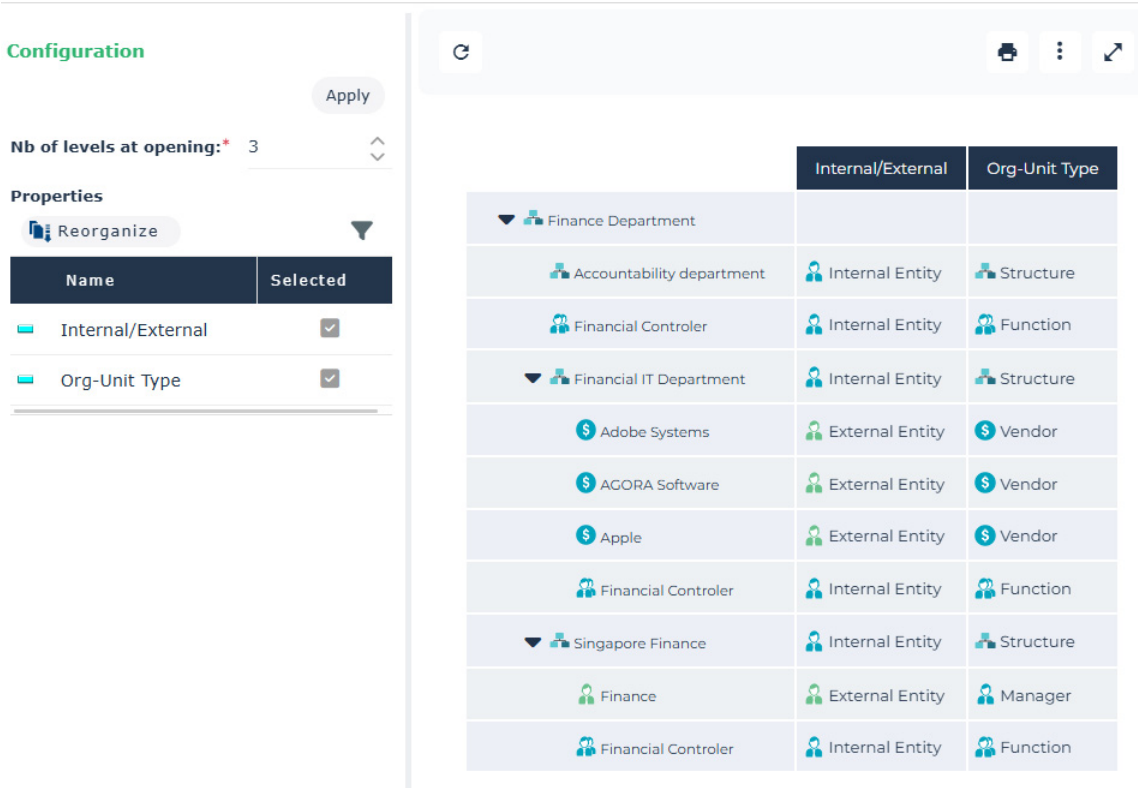
   Ex. 2: this TreeSet Definition is based on the **Applications** that have a 5-star ranking, and displays for each of them its **Technologies**, and for each of its Technologies its **Vendor** and **Risks**.

   It also displays Properties: "Application Type" and "Cost" values of each application, and "Techno Code" and "End of support" date of each technology.

2. From its **Preview** page, click **Instant Report** and select **TreeTable**.
   By default, the TreeTable displays three levels.
   It displays the objects in rows and the properties in columns.

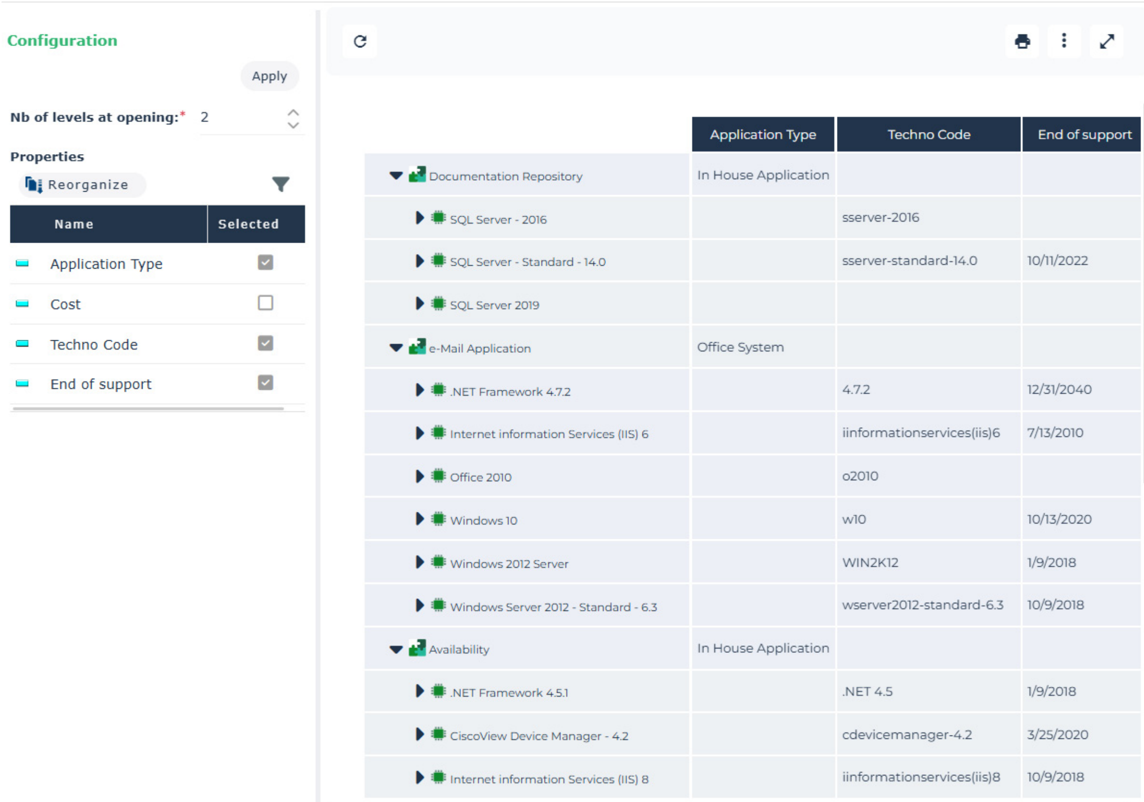
   E.g. 1: this TreeSet is based on the Finance Department **Org-Unit**.



3. In the **Configuration** pane, you can modify:
   - the number of levels displayed at opening: in the **Nb of levels at opening**, select the required number.
   - the **Properties** displayed: for the property you do not want to display in the TreeTable, clear its **Selected** check box.
   - Click **Apply**.

     ☛ *In the TreeTable report, click **Refresh** to see your modifications.*

   E.g. 2: this TreeSet is based on several applications, which all have a ranking of 5 stars. The **Nb of displayed level at opening** is set to 2 and the "Cost" property is hidden.



4. Save the TreeTable to create a Report Template or a Tree View.

   ☛ *See Saving the Tree Report.*

Saving the Tree Report

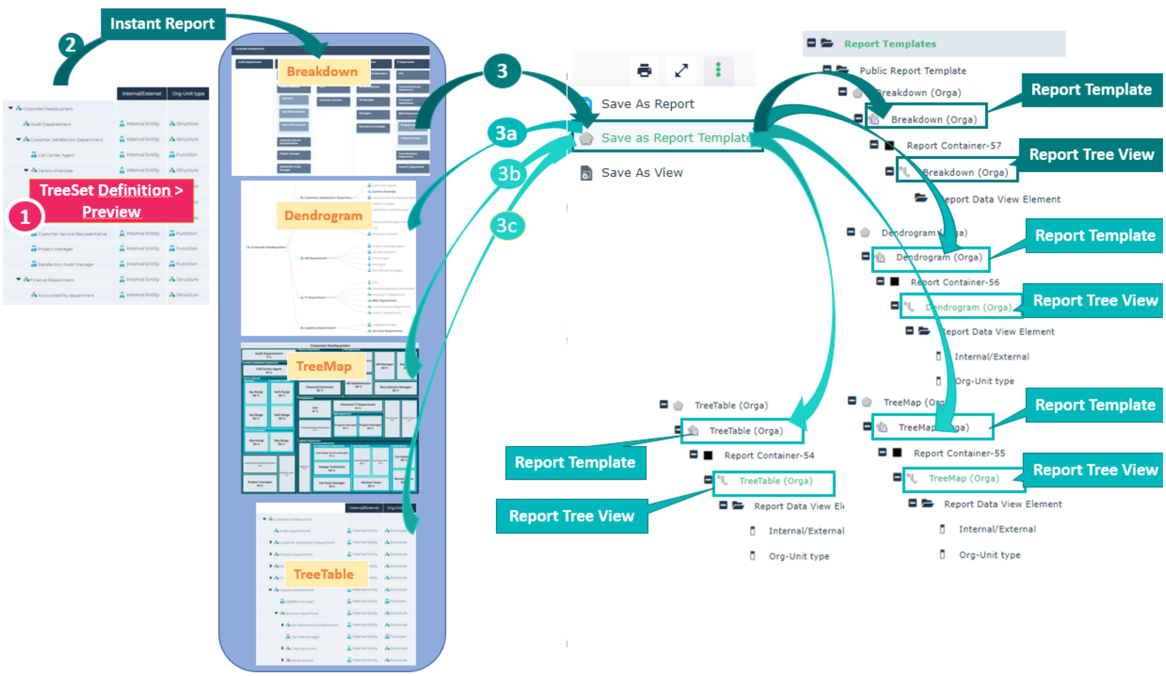
From the **Preview** page of the TreeSet Definition, you can save the tree as:

- a Report Template
- a Report Data View, to use it in a Report Template

> ☛ *Note that **Save as Report**, generates a public report and also a private **Report Template**, which you can access in **Private Report Template > My Report Templates** folder.*

You can save only one tree report at a time:

- a breakdown
- a dendrogram
- a Treemap
- a TreeTable



Creating a tree-type Report Template

To create a Report Template from a tree report:

1. Access the TreeSet Definition properties.

   ☛ *See Accessing the TreeSet Definitions.*

2. Display its **Preview** page.
3. Click the renderer type you want to save.

   `E.g.: Breakdown, Dendrogram, TreeMap, or TreeTable.`

4. In the graphic, click **More** > **Save As Report Template** .

   ☛ *Note: if you customize the report display, this customization is not taken into account in the Report Template.*

5. In the **Name** field, enter your Report Template name.
6. Click **OK**.
   A public Report Template is created, with its public **Report Tree View**. They both have the same name.
   You can access the Report Template:
   - directly from your **Home** page, from **Custom Report Templates** tile, or
   - from the **Public Report Template** folder (**Report Studio > Report Template Definition > Report Templates**)

Creating a Report Tree View

You can save each renderer type of your TreeSet as a Report Tree View.

To create a Report Tree View:

1. Access the TreeSet Definition properties.

   ☛ *See Accessing the TreeSet Definitions.*

2. Display its **Preview** page.
3. Click the renderer type you want to save.
4. In the graphic, click **More** > **Save as View** .
5. Enter a **Name** to the view.
6. Click **OK**.
   The Report Tree View is created and accessible in the **Report Studio > Report Template Definition > Public Report Data Views** folder.
   Any HOPEX user can add the View in a report:
   - use the View to create a Report Template

     ☛ *See Creating a Report Template using a Report Data View.*

   - add the View as a report Chapter in a Report Template

     ☛ *See Adding a Report Chapter to a Report Template.*

HOW TO

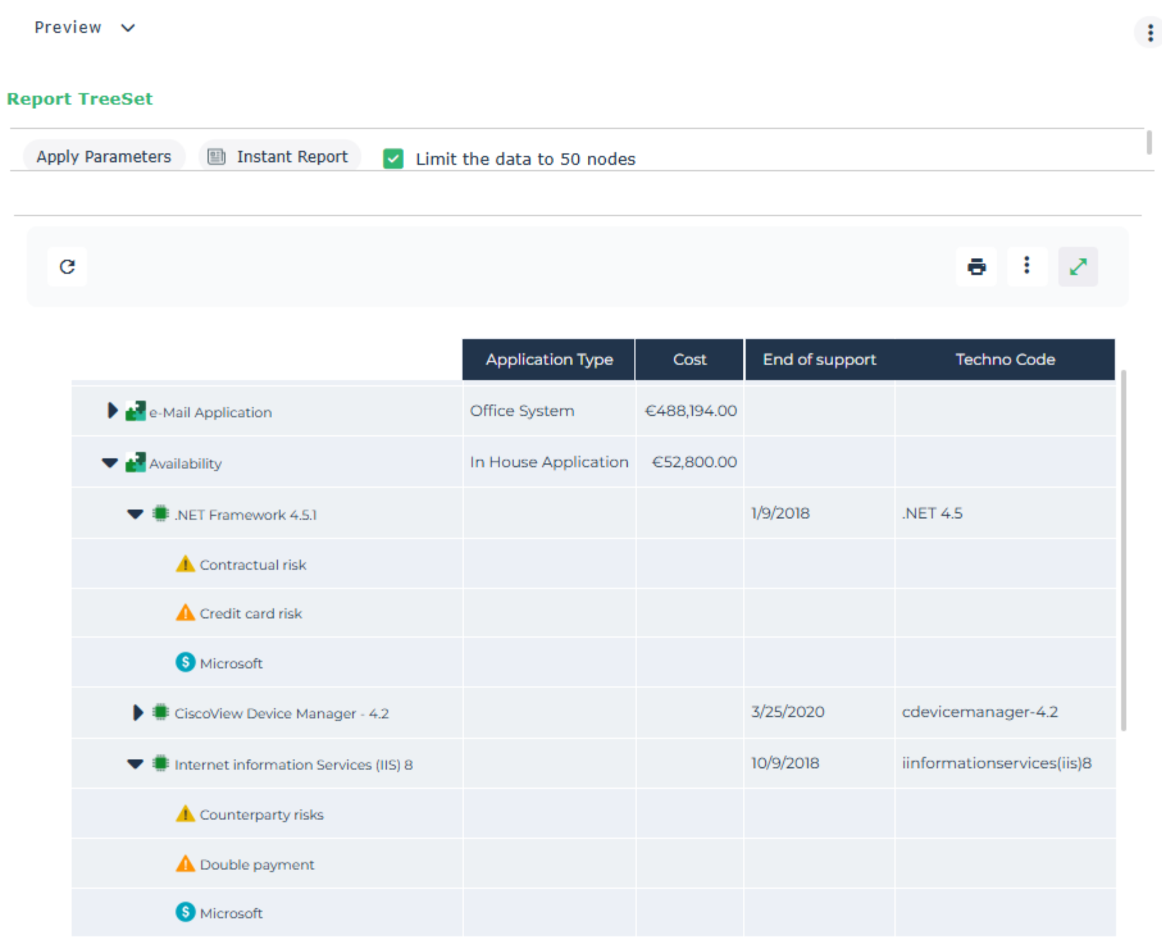
See:

- How to Hide a TreeSet Collection
- How to Automatically Add Folders to a TreeSet Definition (grouping)
- How to Add a Folder to a TreeSet Definition
- How to Reorganize Folders or Collections in a TreeSet Definition
- How to Define a Loop in a TreeSet Definition
- How to Define a Loop in a TreeSet Definition without Duplicating the Collection Declaration
- How to Add Several Objects to the Root Collection

How to Hide a TreeSet Collection

You may want to hide a TreeSet Collection from your TreeSet.

> ***Example:***
>
> The TreeSet Definition displays the Technologies used by an application, and for each Technology it displays its Vendor and its risks (if any).
>
> You want to hide the "Risk" items of the Technologies.



To hide a TreeSet Collection in the TreeSet:

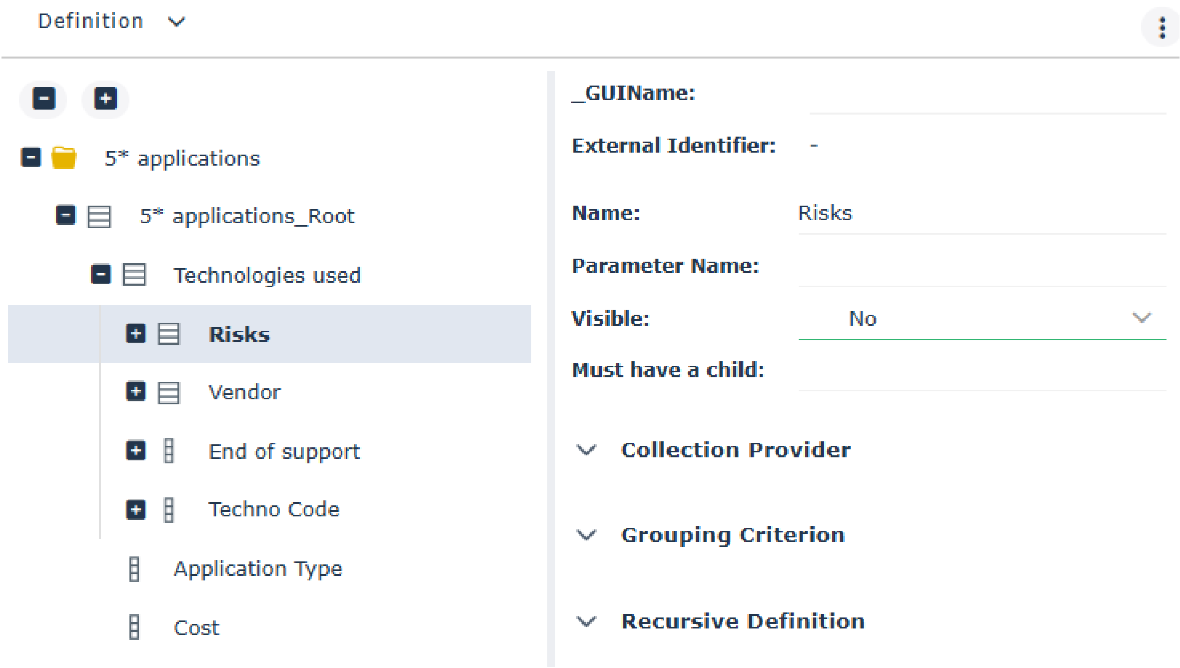
1. Access the TreeSet Definition properties.

   ☛ *See Accessing the TreeSet Definitions.*

2. Display its **Definition** page.
3. In the TreeSet Definition structure, select the TreeSet Collection you want to hide.

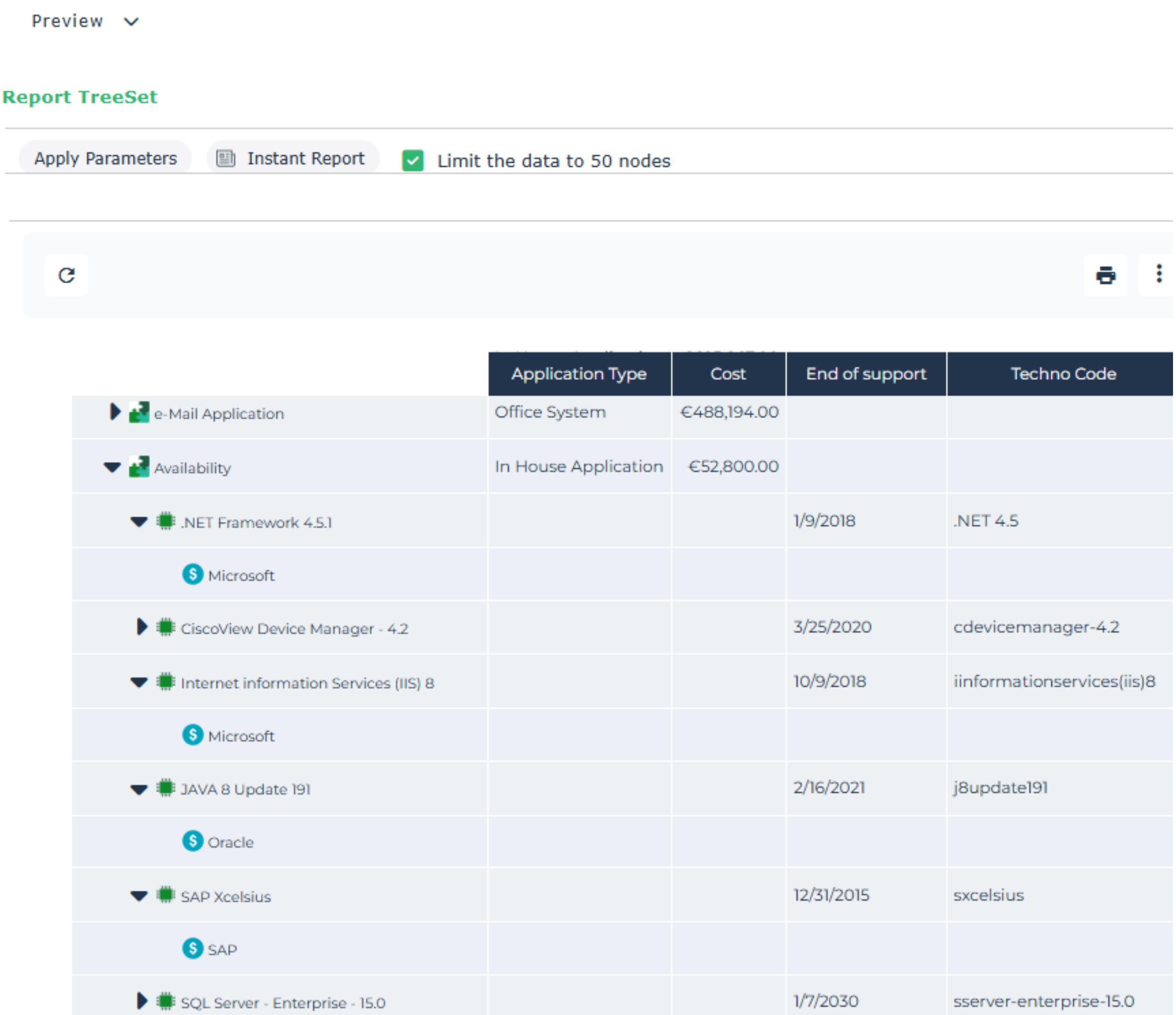
```
Example: "Risk"
```

4. In the right pane, in the **Visible** field, select "No".



5. In the TreeSet Definition **Preview** page.

6. Click **Apply Parameters**.



	Application Type	Cost	End of support	Techno Code
e-Mail Application	Office System	€488,194.00		
Availability	In House Application	€52,800.00		
.NET Framework 4.5.1			1/9/2018	.NET 4.5
Microsoft				
CiscoView Device Manager - 4.2			3/25/2020	cdevicemanager-4.2
Internet information Services (IIS) 8			10/9/2018	iinformationservices(iis)8
Microsoft				
JAVA 8 Update 191			2/16/2021	j8update191
Oracle				
SAP Xcelsius			12/31/2015	sxcelsius
SAP				
SQL Server - Enterprise - 15.0			1/7/2030	sserver-enterprise-15.0

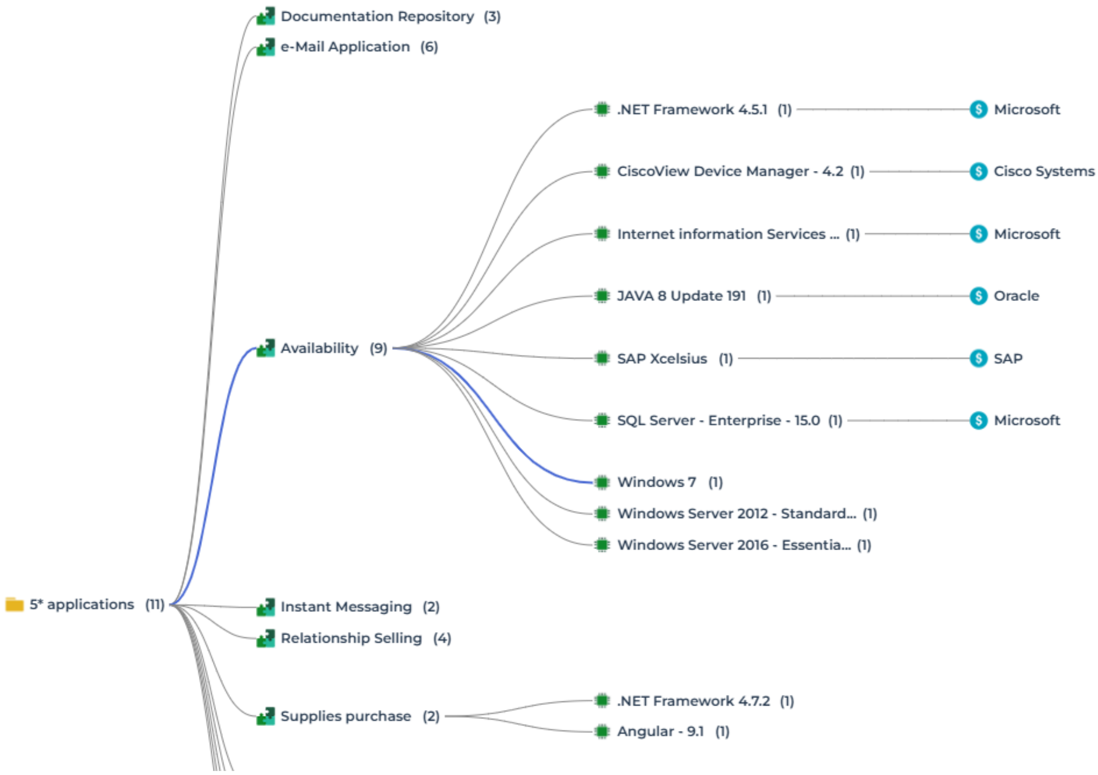
7. From the TreeSet, generate an Instant Report.

```
Ex: Dendrogram type Instant Report.
```

The TreeSet Collection concerned is hidden.

```
Example: "Risk" values of the Technologies are hidden.
```

☛ *In case your modification was not taken into account, click* ***Refresh*** *.*



How to Automatically Add Folders to a TreeSet Definition (grouping)

Instead of manually adding folders (see How to Add a Folder to a TreeSet Definition), in some cases you can add a grouping criterion on a TreeSet Collection. This grouping automatically adds the folders in the TreeSet.

Example: Grouping the Org-Units in folders according to their type"

The TreeSet Definition based on:

- Root MetaClass: Org-Unit
- Root TreeSet Collection: collection populated by the "Org-Unit" parameter
- added TreeSet Collection: "Org-Unit Component"
- added Properties: Company Standard, Internal/External, and Org-Unit Type of the Org-Units

	Company Standard	Internal/External	Org-unit Type
Finance Department			
Accountability department	Approved	Internal Entity	Structure
Audit Departement	Approved	Internal Entity	Structure
CEO	Approved	Internal Entity	Manager
CFO	Accepted	Internal Entity	Manager
CIO	Unknown	Internal Entity	Manager
Financial Controler	Accepted	External Entity	Function
Financial IT Department	Approved	Internal Entity	Structure
IT Department			
CIO	Unknown	Internal Entity	Manager
Financial Business Department	Unknown	Internal Entity	Structure
Financial IT Department	Approved	Internal Entity	Structure
Travel Business Department	Prohibited	Internal Entity	Structure
Travel IT Department	Accepted	Internal Entity	Structure

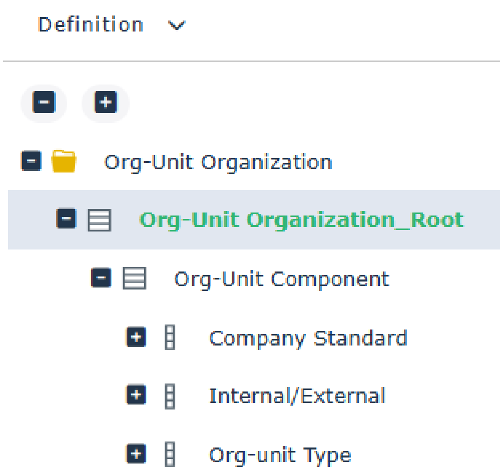
Instead of showing the Org-Unit Type values in a TreeSet column, you can group the Org-Unit according to their type.

To group the Org-Unit according to their type in folders:

1. Access the TreeSet Definition properties.

☛ *See Accessing the TreeSet Definitions.*

2. Display its **Definition** page.



3. In the TreeSet Definition structure, select the "Org-Unit Component" TreeSet Collection.
4. In its **Grouping Criterion** section, select:
   - **Group by** "MetaAttribute"
   - **Grouping Criterion** "Org-Unit Type"
5. In this case, you can remove the Org-unit Type property from the Org-Unit Component TreeSet Collection:
   in the TreeSet definition structure, right-click **Org-unit Type** TreeSet Property and select **Remove**.

6. Display the TreeSet Definition **Preview** page.
   Org-Units are now grouped in folders according to their type (e.g.: Structure, Function, Manager).

	Company Standard	Internal/External
Finance Department		
Structure		
Accountability department	Approved	Internal Entity
Audit Departement	Approved	Internal Entity
Financial IT Department	Approved	Internal Entity
Function		
Financial Controler	Accepted	External Entity
Manager		
CEO	Approved	Internal Entity
CFO	Accepted	Internal Entity
CIO	Unknown	Internal Entity
IT Department		
Structure		
Financial Business Department	Unknown	Internal Entity
Financial IT Department	Approved	Internal Entity
Travel Business Department	Prohibited	Internal Entity
Travel IT Department	Accepted	Internal Entity
Manager		
CIO	Unknown	Internal Entity

How to Add a Folder to a TreeSet Definition

You can add folders to a TreeSet Definition. A folder is added after a TreeSet Collection.

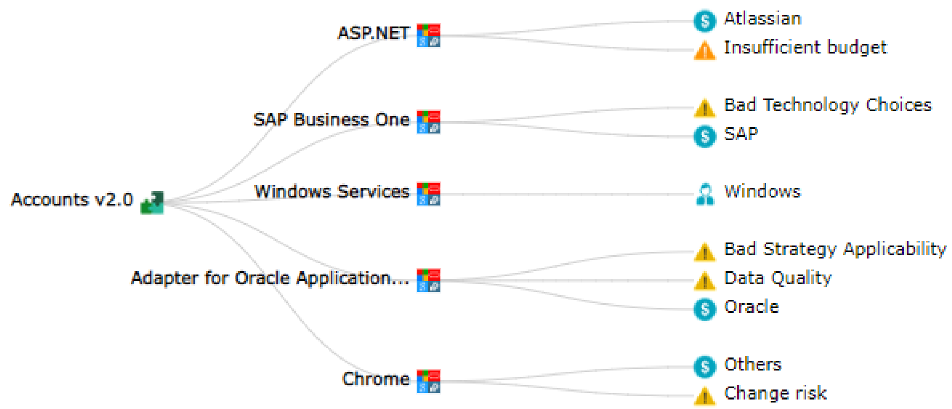
☛ *Check if you can use the automatic way instead, see How to Automatically Add Folders to a TreeSet Definition (grouping)*

☛ *See also How to Reorganize Folders or Collections in a TreeSet Definition*

Example: Adding "Risks and Vendors" folder after each Technology

The TreeSet Definition based on:

- MetaClass Root: Application
- Root TreeSet Collection: collection populated by the "Application" parameter
- added TreeSet Collections: "Technology" and its "Risk" and "Vendor"
- added Properties: Creation date and Code of the Technologies, and Cost of the Application.

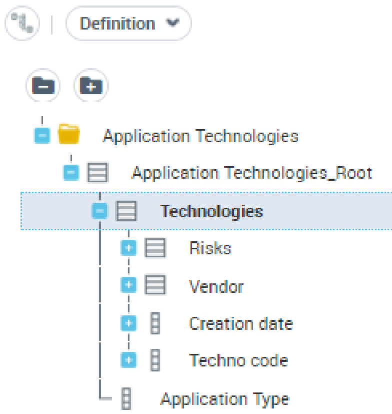


To add after each Technology a folder that includes the its risks and its Vendor:

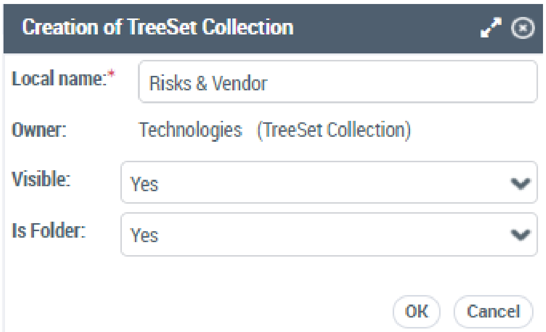
1. Access the TreeSet Definition properties.

   ☛ *See Accessing the TreeSet Definitions.*

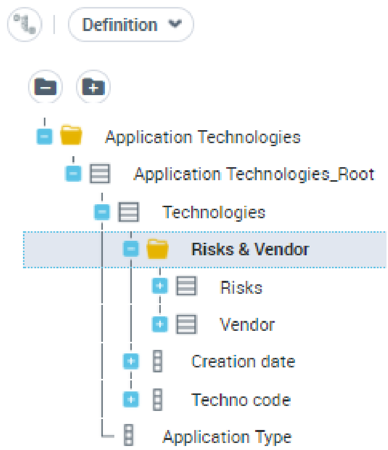
2. Display its **Definition** page.



3. In the TreeSet Definition structure, right-click the "Technologies" TreeSet Collection. and select **New > TreeSet Collection**.
4. Define the folder:
   - **Local name**: enter "Risks and Vendor"
   - **Is folder**: "yes"

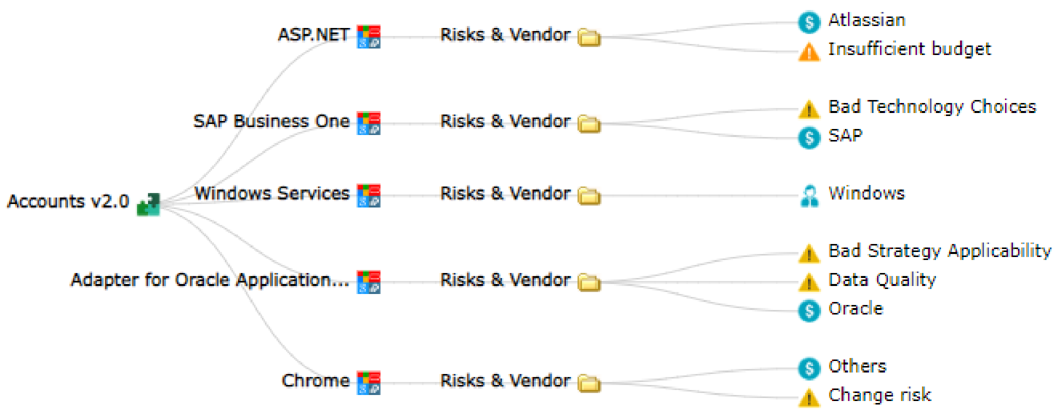


5. Drag and drop the "Risks" and "Vendor" TreeSet Collections into the "Risks and Vendor" folder.



6. From the **Preview** page, generate a **Dendrogram** type Instant Report and click .
   Your folder is added after each Technology and includes its Risks and Vendor.

   ☛ *With the added folder, you might want to add one more level at opening.*

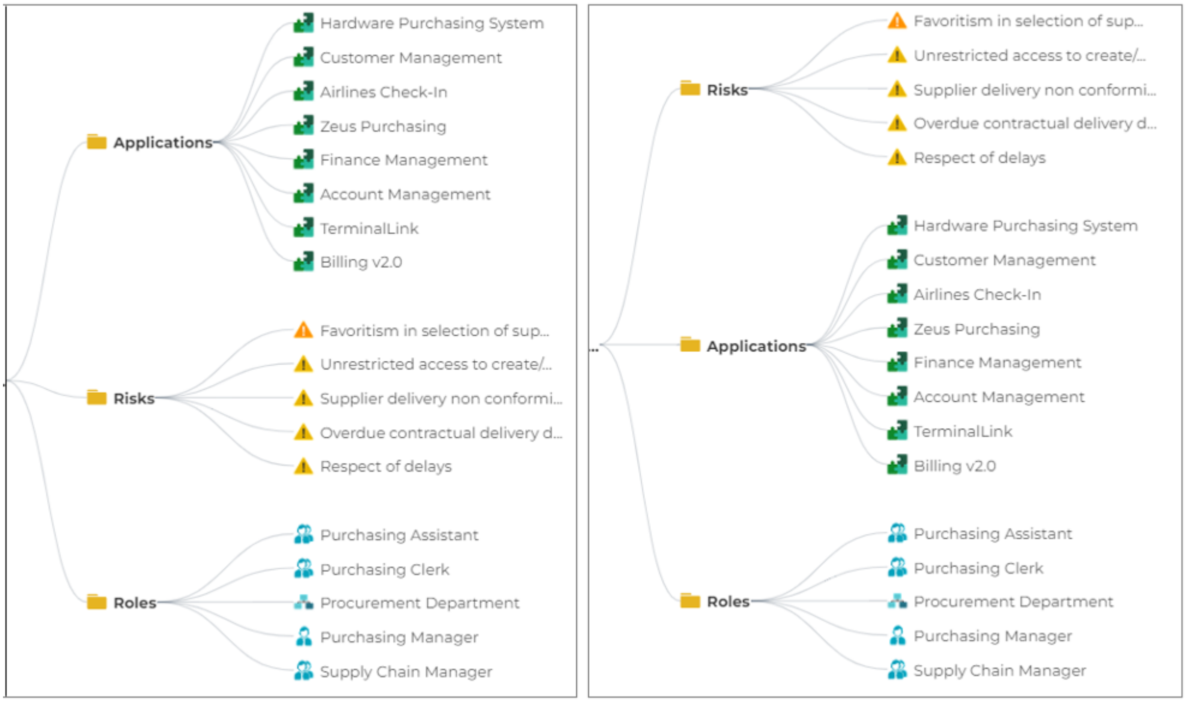


7. (If you want to hide the added folder) In the TreeSet Definition, select the folder and in the right pane set **Visible** to "No".
   The folder is hidden.

How to Reorganize Folders or Collections in a TreeSet Definition

In a TreeSet Definition, you can modify the display order of TreeSet Collections and/or folders, so as to get another display in the report showing a Dendrogram, TreeMap, Breakdown, or TreeTable.

Example: Displaying Risks folder above Applications folder in a dendrogram

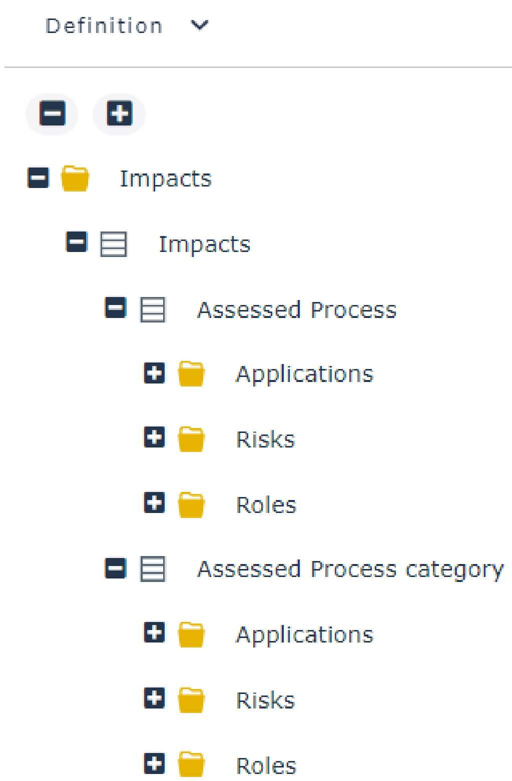


To reorganize folders or Collections:

1. Access the TreeSet Definition properties.

   ☛ *See Accessing the TreeSet Definitions.*

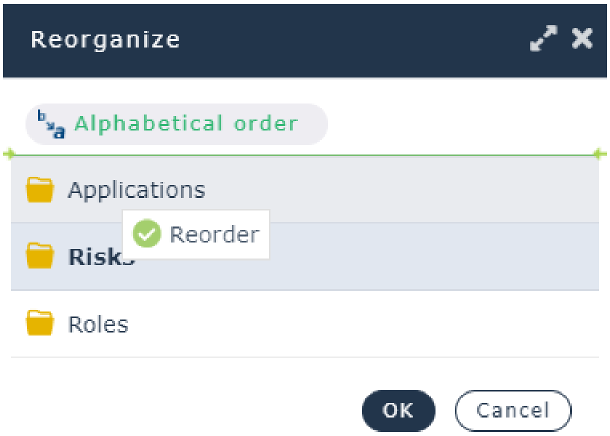
2. Display its **Definition** page.



3. In the TreeSet Definition structure, right-click the TreeSet Collection (or folder) concerned and select **Reorganize**.

   `E.g.: Assessed Process TreeSet Collection.`

4. Drag and drop the TreeSet Collection (or folder) so as to get another display order, and click **OK**.



How to Define a Loop in a TreeSet Definition

When a hierarchy is recursive, you can define a collection loop on any TreeSet Collection of the TreeSet Definition.

Example: Defining a loop on "Org-Unit Component" TreeSet Collection

The TreeSet Definition based on:

- MetaClass Root: Org-Unit
- Root TreeSet Collection: collection populated by the "Org-Unit" parameter
- added TreeSet Collection: "Org-Unit Component" collection populated by the "Component" MetaAssociationEnd

`For example, the "Finance Department" Org-Unit has four children.`

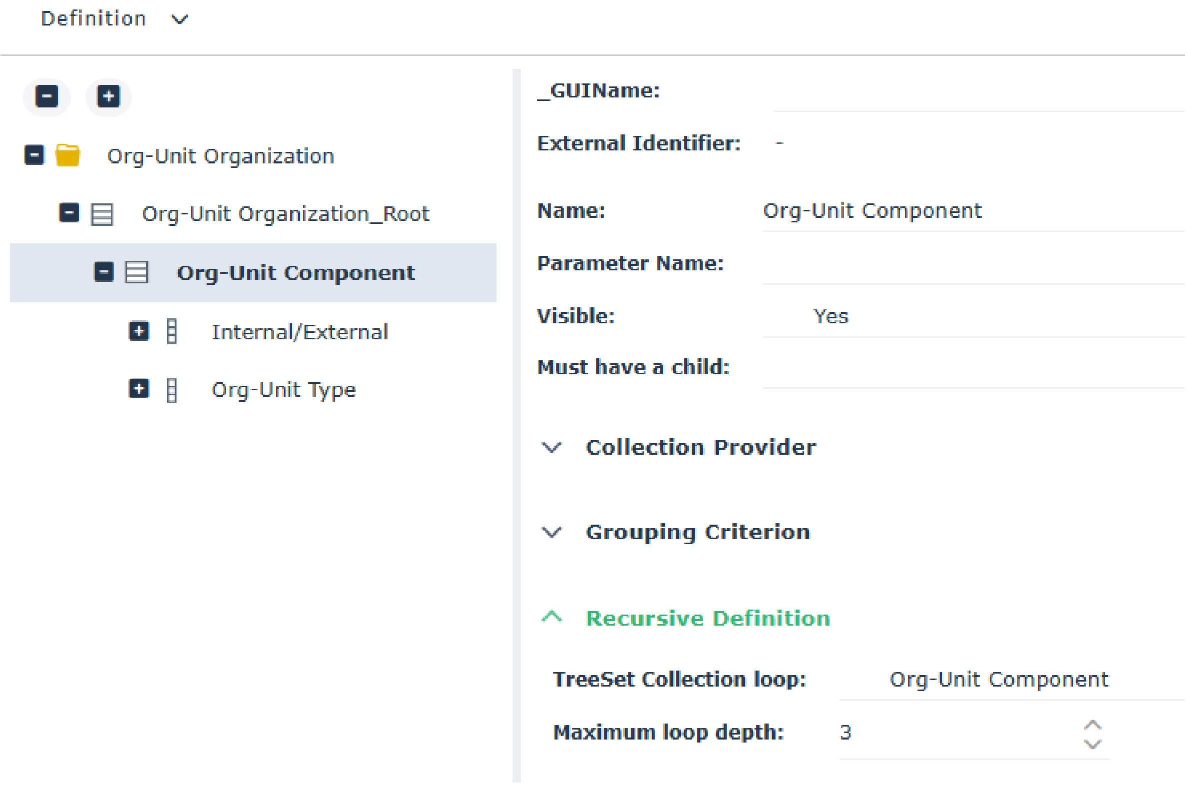
	Internal/External	Org-Unit Type
Finance Department		
Accountability department	Internal Entity	Structure
Financial Controler	Internal Entity	Function
Financial IT Department	Internal Entity	Structure
Singapore Finance	Internal Entity	Structure

You can define a loop on the "Org-Unit Component" TreeSet Collection.

To define a loop on the TreeSet Collection:

1. Access the TreeSet Definition properties.
   ☛ *See Accessing the TreeSet Definitions.*
2. Display its **Definition** page.
3. In the TreeSet Definition structure, select the "Org-Unit Component" TreeSet Collection.

4. In the **Recursive Definition** section, define the loop:
   - In the **TreeSet Collection loop**, select "Org-Unit Component" TreeSet Collection.
   - In the **Maximum loop depth**, enter "3" (the depth represents the number of displayed levels)



5. Display the **Preview** page, and click **Apply Parameters**.

   `Now two levels of Org-Unit components are added to the "Finance department".`

	Internal/External	Org-Unit Type
Finance Department		
Accountability department	Internal Entity	Structure
Financial Controler	Internal Entity	Function
Financial IT Department	Internal Entity	Structure
Adobe Systems	External Entity	Vendor
AGORA Software	External Entity	Vendor
Apple	External Entity	Vendor
Financial Controler	Internal Entity	Function
Singapore Finance	Internal Entity	Structure
Finance	External Entity	Manager
Financial Controler	Internal Entity	Function

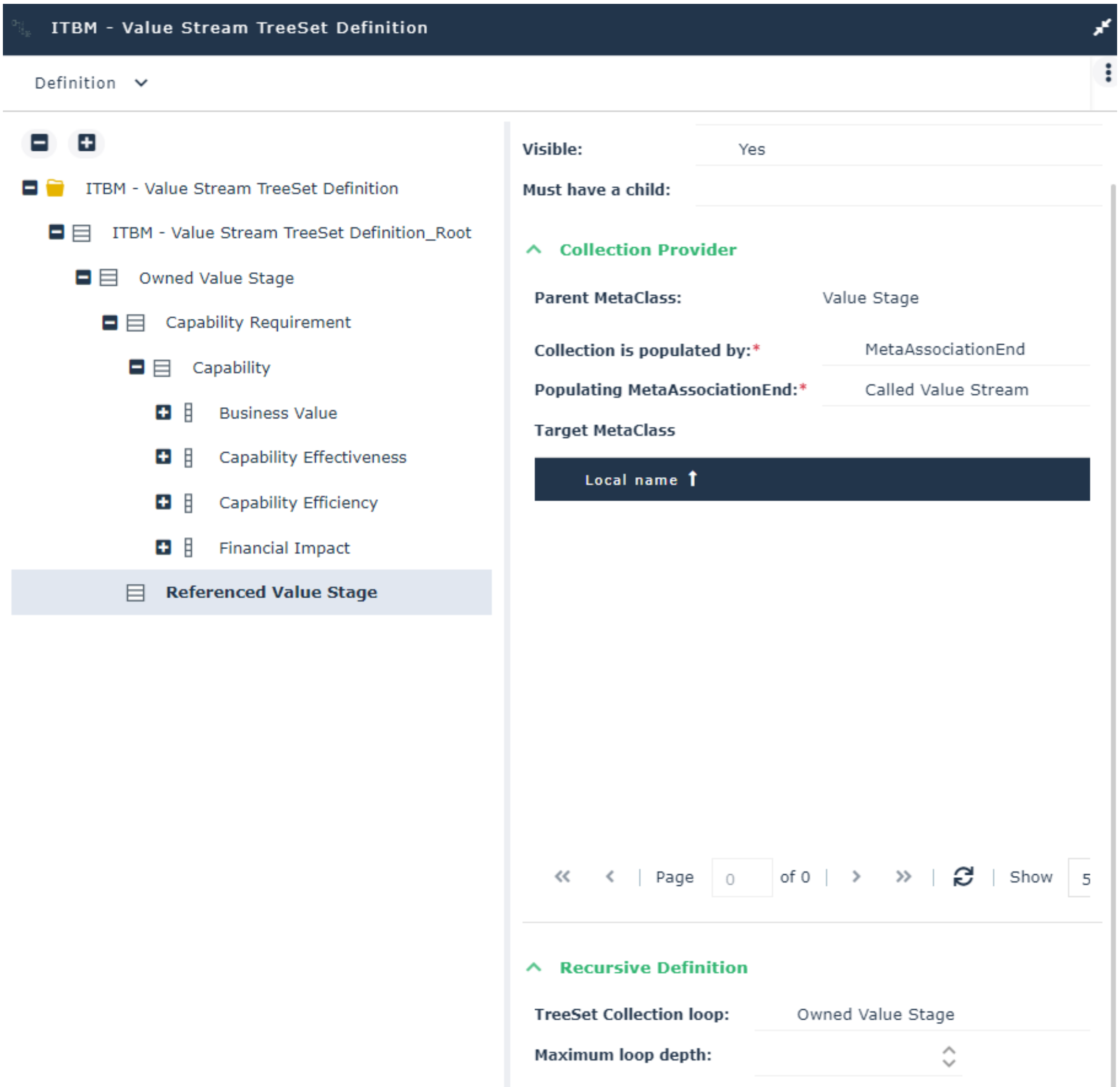
How to Define a Loop in a TreeSet Definition without Duplicating the Collection Declaration

In a TreeSet collection you can use the **Recursive Definition** section to reference another collection as a child.

Example:

In the "ITBPM - Value Stream Definition" TreeSet Definition, the "Referenced Value Stage" Collection defines how to find a set of "Value Stream" via the "Called value stream" MetaAssociationEnd.

Each value stream is used as a parent of the "Owned Value Stage" collection.



How to Add Several Objects to the Root Collection

If your TreeSet Definition is based on a Collection Parameter, at TreeSet Definition creation, the root Collection parameter is made up by a collection of objects with a Multiplicity value set to "1" by default. This means that at TreeSet creation the user can add a single object only.

☺ *Be careful in adding too many objects as TreeSet entry points, as the TreeSet might be two crowded*

If you still want the user to be able to add several objects as entry points at TreeSet creation:

- set the root Collection parameter multiplicity to "N"
- else create a TreeSet Definition based on a Query Parameter, with a query collecting several objects.

Note that, when the TreeSet has several objects as entry points, the **Root node** is displayed.

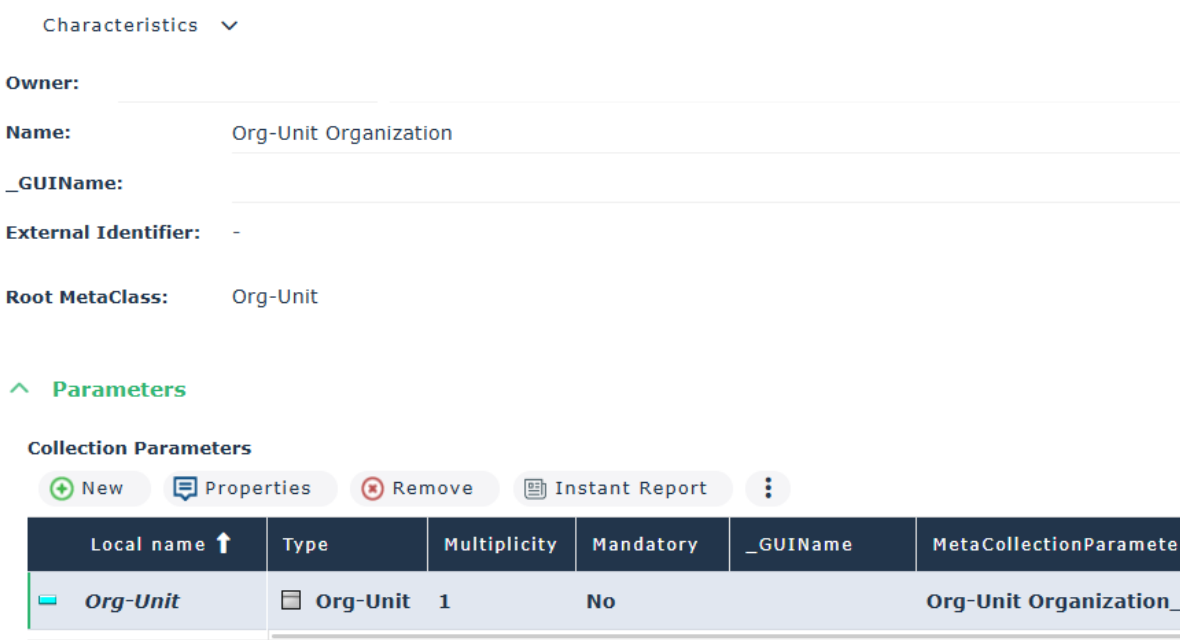
Example: Adding selected objects as root Collection parameter

To add several objects to your root node:

1. Access the Collection Parameter - based TreeSet Definition properties.

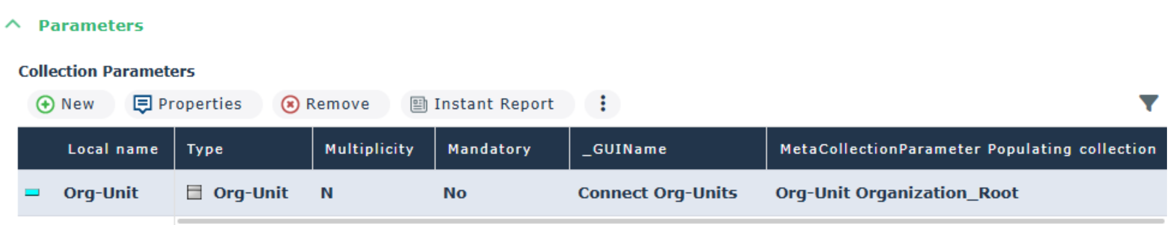
   ☛ *See Accessing the TreeSet Definitions.*

2. Display its **Characteristics** page.



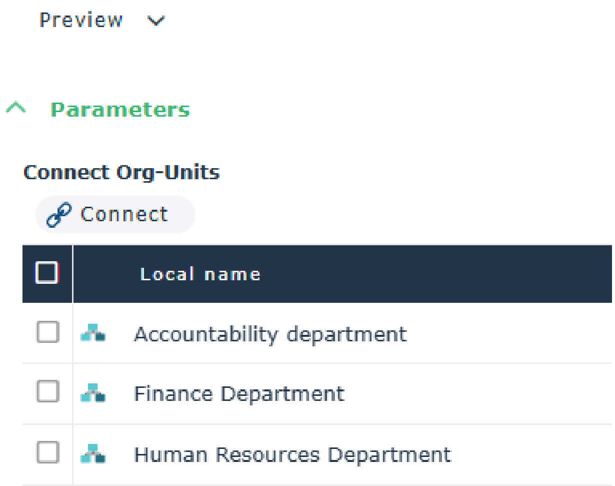
3. In the **Parameters** section, in the root Collection parameter row:
   - click the **Multiplicity** cell and select "N" value.
   - (Optional) In the **_GUIName** cell enter a more suitable name.

     `Example: "Connect Org-Units:".`



4. Display the TreeSet Definition **Preview** page.
   In the **Parameters** section the user is now able to connect several objects as entry points of the TreeSet.

   ```
   Example: add "Accountability departement", "Finance Department" and
   "Human Resources Department" Org-Units.
   ```

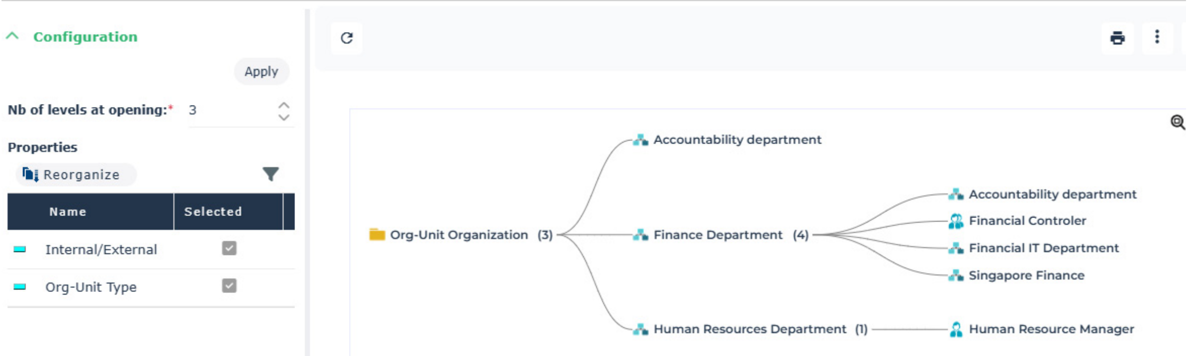


5. In the **Parameters** section title, click ^ to reduce the section (and get more space for the report).
6. Click **Apply Parameters**.

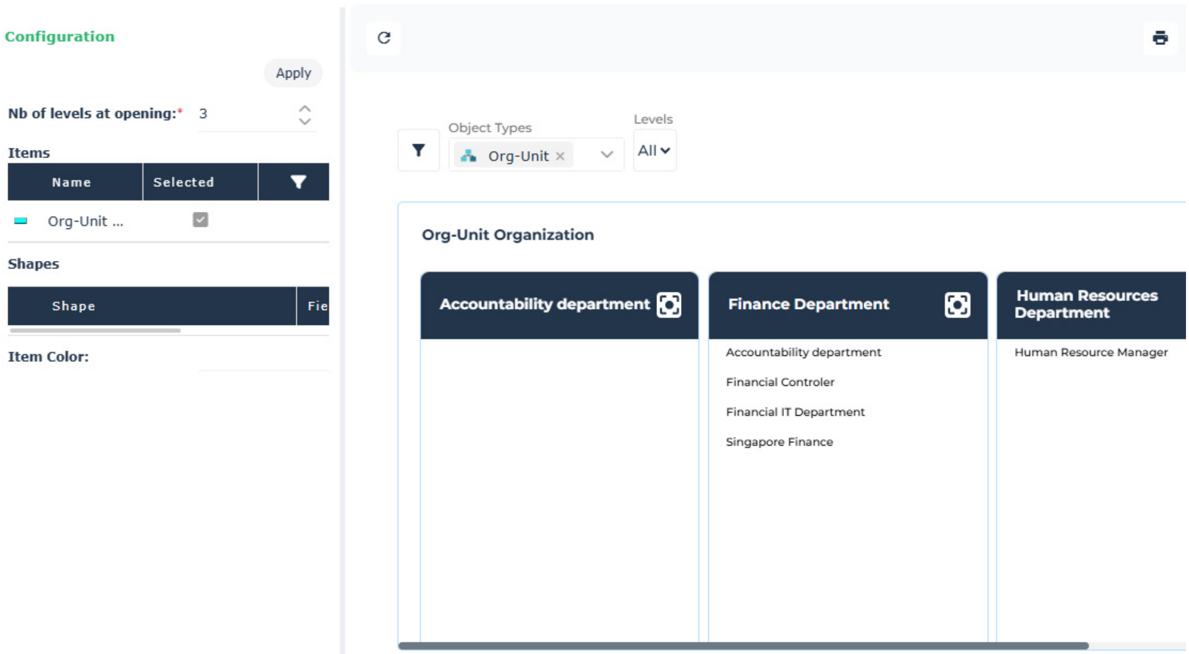
   ```
   The TreeSet is displayed with the 3 Org-Units connected to the "Org-Unit
   Organization" root folder.
   ```

7. Generate Instant Reports:

E.g.: Dendrogram, with its root node displayed.



E.g.: Breakdown



5

Report Template Definition

The following points are covered here:

Report Template Introduction

The **Report Template** describes how to build the report.

> ☛ *To build a **Report Template** you must be in "Expert" MetaModel access (Repository options).*

MEGA supplies predefined Report Templates. Each Report Template is specific to a HOPEX Solution (e.g.: Risk analysis). With **HOPEX Power Studio** you can create your own Report Templates.

Report Template creation is performed in both **HOPEX** Windows Front-End and Web Front-End for users with **HOPEX Customizer** or **HOPEX Customizer Publisher** profile.

> ☛ *Report Template definition is only available with **HOPEX Power Studio** technical module.*

Once a Report Template is created, any user can use it to query **HOPEX** repository and create a report. Data first shown in list form can then be handled and shown in graphical format using Instant Report feature.

> ☛ *For details on the use of reports, see Common Features > Generating documentation and more specifically Launching an Instant Report.*

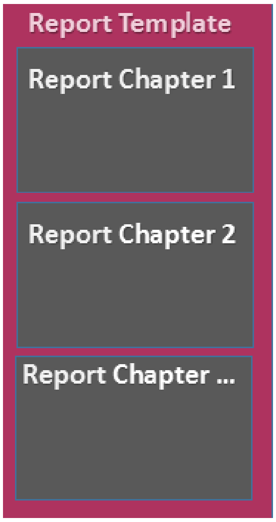
Dynamic reports enable to analyze repository data through different focus.

Each report is built from a Report Template, which defines parameters on which is based the report. Creation and use of reports is available in both **HOPEX** Windows Front-End and Web Front-End for all users.

Report Template and Report Chapters

A **Report Template** is made up of at least one Report Chapter. The **Report Template** defines:

- each Report Chapter
- (if needed) report parameters.



Report Chapter and Macro or Report Data Views

The Report Chapter creation is based on either:

- a macro, or
- Reports Data View(s)

A single **Report Template** can include both types of Report Chapters.

Report Chapter and macro

The Report Chapter creation can be based on:

- a macro, which refers to Java code that implement dedicated interfaces
  and
- if needed Report Parameters
  Parameter definition indicates the input data type the user must enter for the report calculation (e.g.: Applications).

> ☛ *To implement a chapter with a Java macro, see Writing Java Report Chapters Technical Article.*
>
> *For examples of Report Templates based on a macro see for example **Report Templates > HOPEX > MEGA** folder.*

Report Chapter and Report Data Views

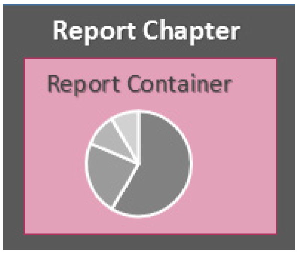
The Report Chapter creation can be based on:

- a Report Data View.
  The Report Data View is included in a Report Container.
- a set of Report Data Views.
  Each single Report Data View is included in a single Report Container.
  You can add a set of Report Containers in a Report Chapter.
  By default the Report Containers are displayed in a single column. You can modify the display of the Report Containers (e.g.: on a same row, in separate tabs) through the Report Container group.

> ☛ *See How to Group Report Data Views in a Report Chapter section.*

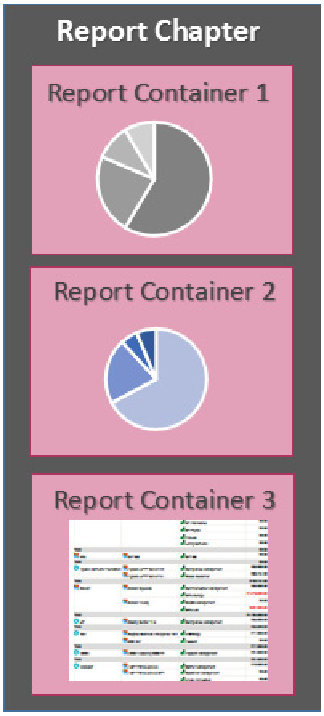
A **Report Chapter** may include:

- at least one Report Container including a single Report Data View.

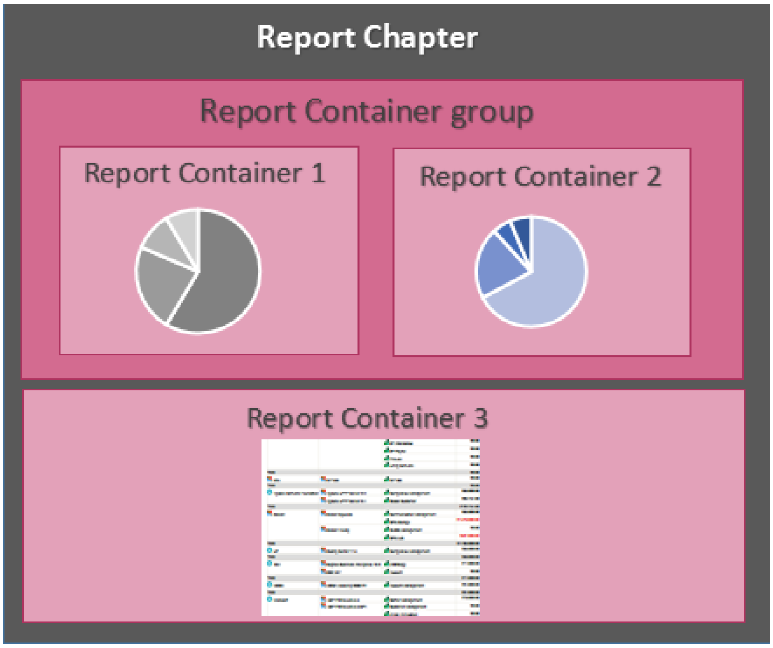


- several Report Containers, which can be grouped in Container Groups, so that the Report Data Views are displayed for example on the same row.

  Example 1: a Report Chapter with its Report Data Views displayed in a single column.



  Example 2: a Report Chapter with its first two Report Data Views displayed on the same row.



Report Data View types

The Report Data View types are:

- ***Table*** (Report Table View)
  A **Report Table View** is built from a set of columns of a Report DataSet, which defines the set of columns of the table.
- ***Matrix*** (Report Matrix View)
  A **Report Matrix View** is built from three columns of a Report DataSet:
  - two columns to define the matrix axes,
  - a third column to define the cells.
- ***Graph*** (Report Graph View)
  A **Report Graph View** is built from data provided by a Report Data Source of GraphSet Definition type.
- ***Tree*** (Report Tree View)
  A **Report Tree View** is built from data provided by a Report Data Source of TreeSet Definition type.
- ***Value*** (Report Value View)
  A **Report Value View** is built from data provided by a Report Data Source of query type.

Available displays depend on the Data View type.

☛ *See Report Renderer.*

Report Style

You can define the style of the reports generated from a **Report Template** at the **Report Template** level with a **Report Style**. In that case the style applies to all the chapters of the report.

> *See How to Customize a Report Template Style section.*

You can also apply:

- a **general style** at Report Data ***View*** level.
  > *See an example in How to Apply a General Style section.*
- a **conditional style** at Report Data ***View element*** level.
  > *See an example in How to Define the Filters Displayed in a Report section.*
  > *See an example in How to apply a Conditional Style to a Graph section.*

Report Renderer

The renderer defines how the dataset is displayed in the report.

Depending on the Report Data View, the following specific renderers are available:

- ***Report Matrix View:***
  - matrix and associated bar chart
  - matrix and associated radar chart

  > *See the drawing showing how to create a **Report Template** Displaying a Matrix-Bar Chart or a Matrix-Radar Chart.*
- ***Report Table View:***
  - radar chart
  - line chart
  - bar chart
  - pie chart
  - gauge
  - word cloud
  - table

  > *See the drawing showing how to create a **Report Template** Displaying a Table, a Radar/Line/Bar/Pie Chart, a Set of Gauges, or a Word Cloud.*
- ***Report Graph View:***
  - graph

  > *See the drawing showing how to create a **Report Template** Displaying a Graph.*
- ***Report Tree View:***
  - Breakdown
  - Dendrogram
  - TreeMap
  - TreeTable

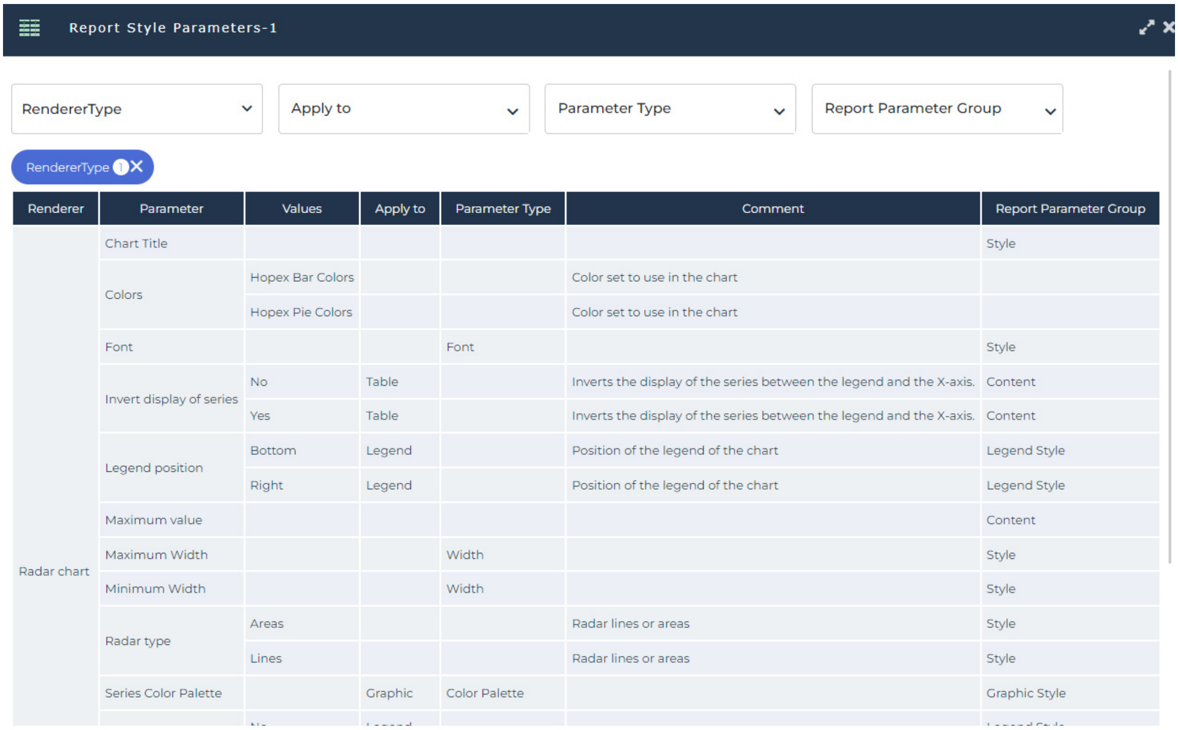
  > *See the drawing showing how to create a **Report Template** Displaying a Breakdown, a Dendrogram, a TreeMap, or a TreeTable.*
- ***Report Value View:***
  - gauge

  > *See the drawing showing how to create a **Report Template** Displaying a Gauge.*

Report renderer parameters

From the **Report Studio > Reports** navigation menu, use the "Report Style" Report Template to create a report displaying for each renderer type, all its available parameters and where they are used. Use the filters to narrow down the display.

> Example: in the **RendererType** filter select "Radar chart" to display the radar chart renderer information only.



Renderer	Parameter	Values	Apply to	Parameter Type	Comment	Report Parameter Group
Radar chart	Chart Title					Style
	Colors	Hopex Bar Colors			Color set to use in the chart	
		Hopex Pie Colors			Color set to use in the chart	
	Font			Font		Style
	Invert display of series	No	Table		Inverts the display of the series between the legend and the X-axis.	Content
		Yes	Table		Inverts the display of the series between the legend and the X-axis.	Content
	Legend position	Bottom	Legend		Position of the legend of the chart	Legend Style
		Right	Legend		Position of the legend of the chart	Legend Style
	Maximum value					Content
	Maximum Width			Width		Style
	Minimum Width			Width		Style
	Radar type	Areas			Radar lines or areas	Style
		Lines			Radar lines or areas	Style
	Series Color Palette		Graphic	Color Palette		Graphic Style

Report Template Definition

The following point are detailed:

Accessing the Report Templates and their Constituents

Report Templates are accessible in HOPEX (Web Front-End) and HOPEX (Windows Front-End).

Accessing the Report Templates and their constituents in HOPEX (Web Front-End)

To access the Report Templates in HOPEX (Web Front-End):

1. Connect to **HOPEX Studio** desktop.

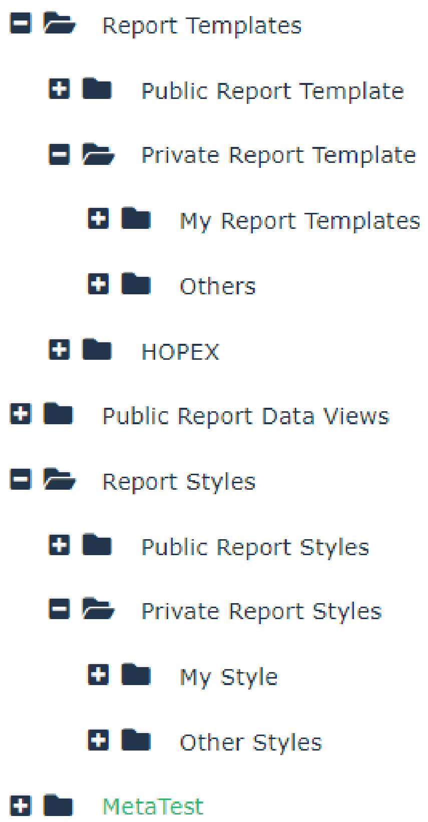
   ☛ *See Connecting to HOPEX Studio Desktop.*

2. Click the navigation menu and select **Studio > Report Template Definition**.
   In the Edit area, the **Browse** view displays the following folders:
   - **Report Templates**, where the Report Templates are classified by their sharing characteristics (**Private** or **Public**). You can add sub-folders for a more precise Report Template classification. The **HOPEX** folder includes standard Report Templates.
   - **Public Report Data Views**, which includes Report Data Views that can be reused. You can add sub-folders for a more precise Report Data View classification.

     ☛ *A Private Report Data View cannot be reused in another Report Templates, so that it is relevant to its Report Template only.*
   - **Report styles**, where Report Styles are classified by their sharing characteristics (**Private** or **Public**).

     ☛ *Report styles are used to add styles to Report Templates, see How to Apply a General Style, How to Define the Filters Displayed in a Report and How to apply a Conditional Style to a Graph sections.*
   - **MetaTest**, where the MetaTests are classified by MetaClass tested.

     ☛ *A MetaTest can be used in Report Style condition properties, see Report Style Condition Properties.*



Click a Report Template: its properties are automatically displayed in the Edit area (**Multi-View** window).

If you click other Report Templates, their properties are automatically displayed in new tabs in the Edit area (**Multi-View** window).

3. Expand a Report Template to access:
   - its Report Chapters .
   - (for Report Chapters based on macros) its Report Parameters .
4. Expand a Report Chapter to access:
   - (for a Report Chapter based on a macro) its macro .
   - (for a Report Chapters based on Report Data View) the containers or container group including its Report Data Views:

     For example:

     Report Table View

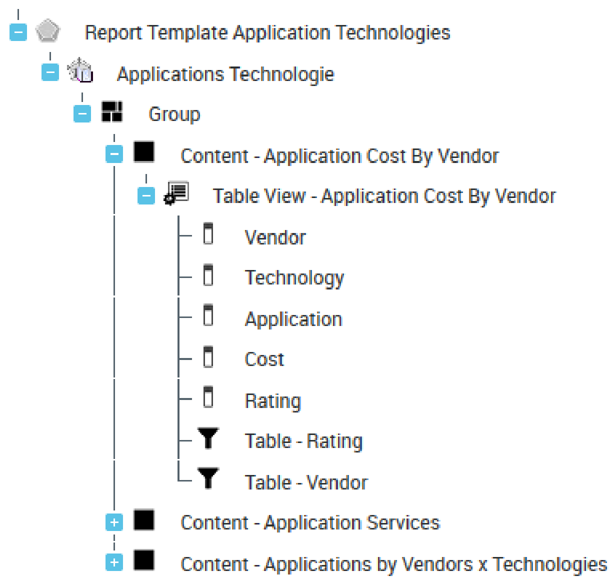
     Report Matrix View

     Report Graph View

     Report Value View

   Hover the cursor over the Report Data View row and click **Properties** : its properties are displayed in the **Properties** window.

   ☛ *See Report Template Examples.*
5. Expand each Report Data View to access its constituents:



   Hover the cursor over the Report Data View element row and click **Properties** : its properties are displayed in the **Properties** window.

Accessing the Report Templates in HOPEX (Windows Front-End)

To access the Report Templates in HOPEX (Windows Front-End):

1. Connect to **HOPEX** with **HOPEX Customizer** or **HOPEX Customizer Publisher** profile.

2. In **HOPEX** menu bar, select **View > Navigation Windows > Utilities**.
3. In the repository tree, expand **Report Templates** folder.
   All the Report Templates available are accessible.

> *Right-click a Report Template and select* ***Properties*** *to access the Report Template properties.*
> *Dedicated tabs enable access to input* ***Parameters*** *and report* ***Chapters****.*

Report Template Properties

> *To access the Report Template properties, see Accessing the Report Templates and their Constituents.*

From the Report Template properties, you can access information regarding:

- its **Characteristics**
- its input **Parameters**
- its **Chapters**
- its style (**Report Template Edition**)
- its **Permissions**
- its **Advanced** parameters
- the report print availability (**Extensions**)

Characteristics

The **Illustrations** section shows the **Renderers** used in the report and their corresponding illustration.

This section:

- is automatically filled for **HOPEX Studio** Report Templates.
- must be set manually for java reports

> *See How to Define the Report Template Renderer and Illustration section.*

The **Subjects & Categories** section enables to define the report related information which is used by the end-user to filter the Reports and Report Templates:

- **Categories** are domains considered in the report
  This field must be set manually.
- **Subjects** are MetaClasses that are displayed in the report
  This field:
  - is automatically filled for **HOPEX Studio** Report Templates.
  - must be set manually for java reports

> *See How to Define the Categories and Subjects Filters section.*

The **Report Template Sharing** parameter defines the Report Template availability:

- "public"
  Allows other users to access the Report Template.
  These users must be connected with one of the Profiles that are allowed to access the Report Template. The Report Template availability for each profile is defined in **Permissions** (see Permissions).
- "private" (default value, at Report Template creation)
  Only the Report Template creator can access the Report Template.

The **Display print button** parameter enables to manage the printing availability of the reports generated with this Report Template:

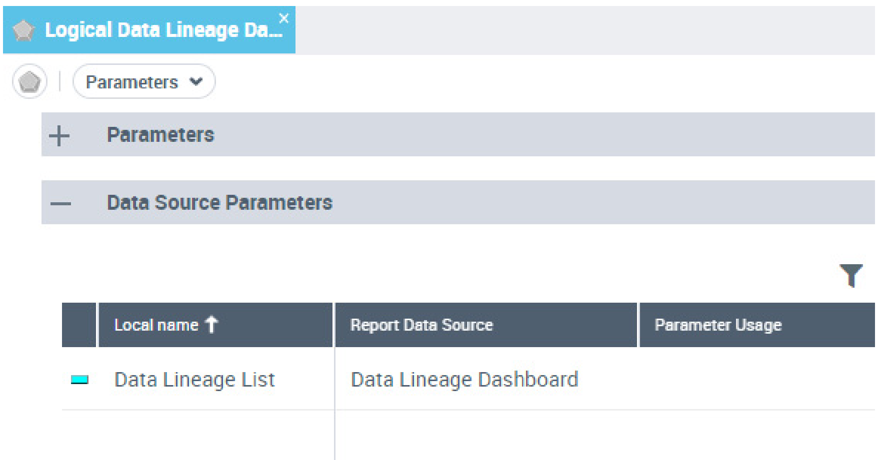
- "yes": the **Print** button is added to the report (default value).
- "No": the **Print** button is removed from the report.

In the **Parameters** section (for Report Chapters based on macros), you can:

- add new parameters to the Report Template.
  - ☛ *See How to Define Parameters in a Report Template section.*
- configure the parameter to define the values that the end-user can use at report creation. The list of values can be computed form another parameter value.
  - ☛ *See Configuring a Report Template parameter.*
- customize the parameter to:
  - add reports in object property pages.
    - ☛ *See How to Add Reports to an Object Property Pages section.*
  - add reports at the Instant Report creation from an object list.
    - ☛ *See How to Make a Report Creation Available to an Object List*
- customize the parameter display.
  - ☛ *See Customizing Parameter Display.*

In the **Data Source Parameters** section (for Report Chapters based on Report Data Views), you can:

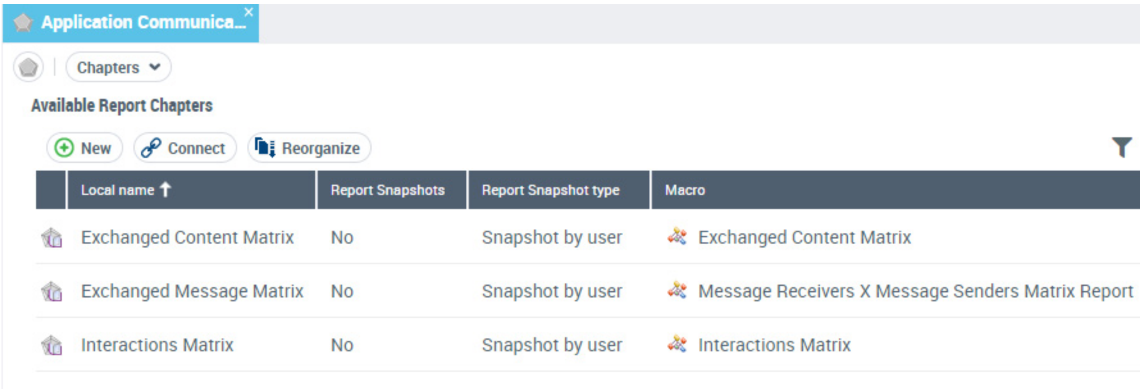
- define the parameters that have to be selected at report creation.
  - ☛ *See Defining parameters in a Report Template based on Data Views.*



Chapters

From the **Available Report Chapters** list, you can:

- add Report Chapters to the Report Template.
  - *See Adding a Report Chapter to a Report Template.*
- organize the Report Chapters
  - *See Organizing the Report Chapters.*
- manage the report snapshot
  - *See Managing a Report Snapshot.*



Report Template Edition

The **Report Template Edition** defines the default style of the reports generated with this Report Template.

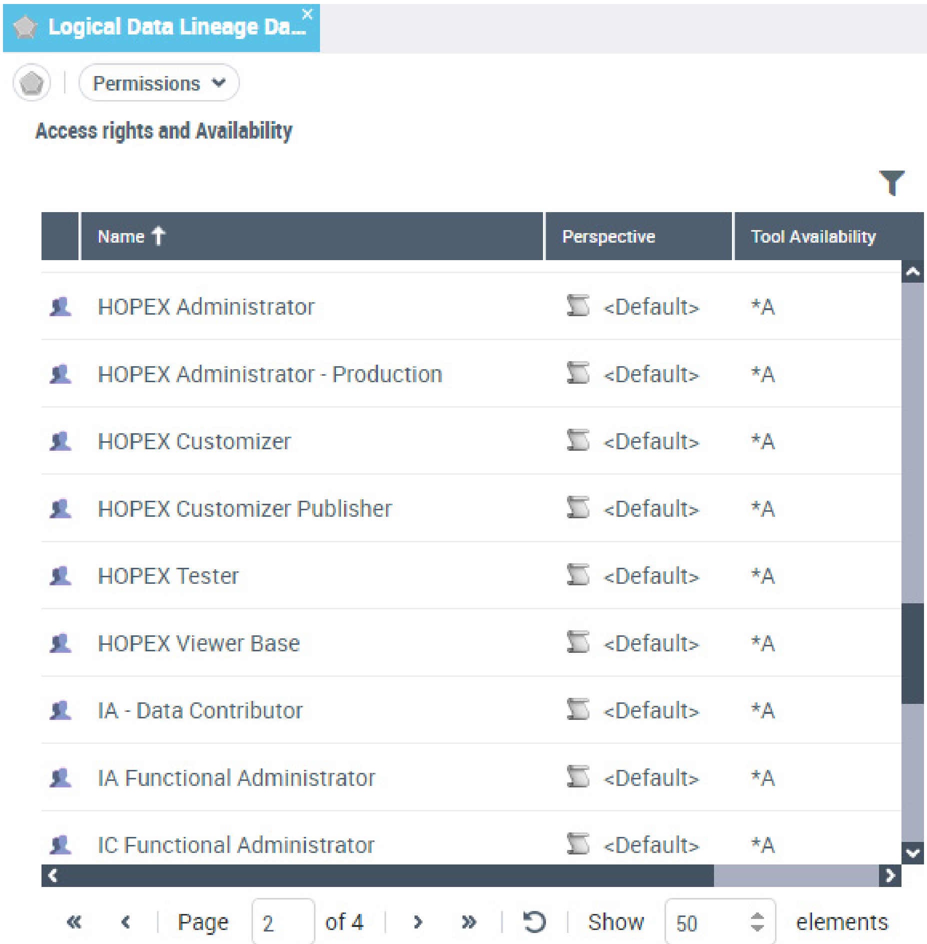
The **Report** field enables to associate the Report Template with a test report so as to define the style parameters that are applied to all the reports generated from this Report Template.

See How to Customize a Report Template Style section.

Permissions

In **Access rights and Availability**, the **Tool Availability** parameter defines for each profile, and associated **Perspective**, whether the Report Template is available to the user or not.

☛ *For details on permissions, see Managing UI Access (Permissions).*



Advanced

In **Advanced** properties you can select:

- **Automatic Report Creation**
  When selected, creating a report from this Report Template is available from any objects on which the Report Template applies.
- **RTF font enabled**
  When selected, fonts used in the RTF texts of objects (e.g.: comment) are kept in web site generation. Else, a CSS file should manage the font choice depending on each HTML tag class.

Extensions

By default the end-user can print its report. With the **Display print button** parameter you can disable the print feature on the report.

Report Chapter Properties

From the Report Chapter properties you can access information regarding:

- its **Characteristics**,
- its **Definitions** (specific to Report Chapters based on Report Data Views),
- its **Filters Binding** (specific to Report Chapters based on Report Data Views).

Characteristics

In **Characteristics**:

- You can define whether the Report Chapter can be exported in the following formats:
  - Excel (**Is compatible with Excel**)
  - RTF (**Is compatible with RTF**)
- **_GUIName**: you can define the name you want to be displayed for the Report Chapter.
- (for a Report Chapter based on a macro) **Implementation** section: defines on which macro the Report Chapter is based.
- **Snapshot** section: you can define a snapshot for a fast Report Chapter display. The snapshot is also used for export operations.

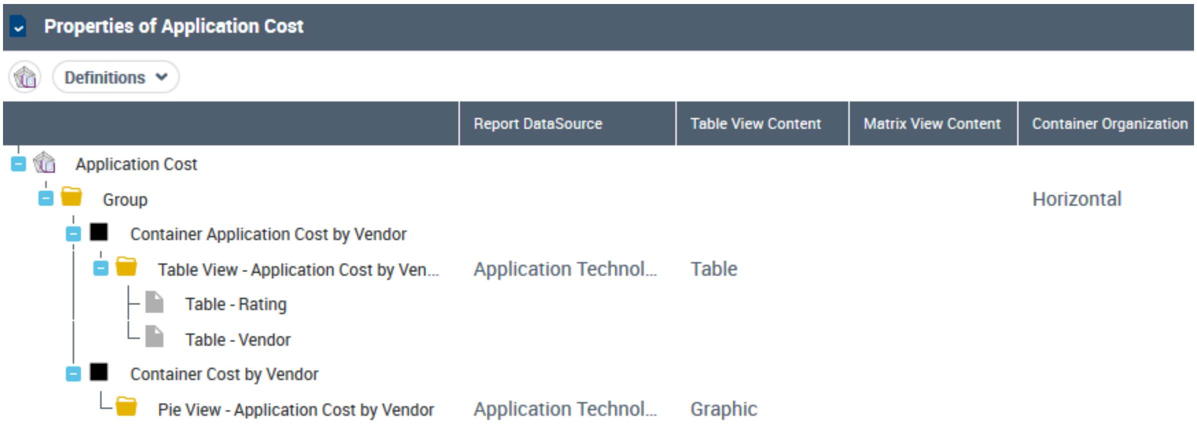
☛ *See Managing a Report Snapshot.*

Definitions (for Report Chapters based on Report Data Views)

Definitions shows:

- a global view of the Report Chapter constituents and structure (as a tree view)
- how each of its following constituents are displayed:
  - (if any) Container Group: horizontal, tab or vertical.
  - Report Data View: table, graphic, or table and graphic.

  For example, the "Application Cost" Report Chapter includes two Report Data Views showing a Table and a graphic on the same row (Group Container Organization: "horizontal")

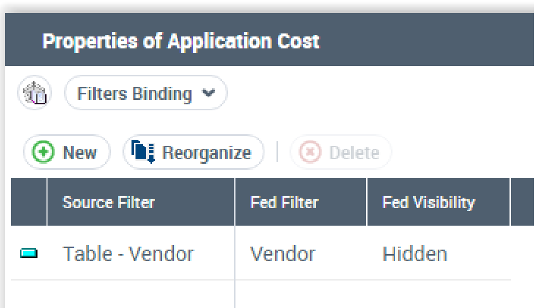


Filters Binding (for Report Chapters based on Report Data Views)

A filter enables to restrict a Report DataSet column values, which are considered in the report. This Report DataSet column is not necessarily part of the report.

Filters Binding enables to feed a filter content with another filter content. Both filters must be of the same type.

To do so you need to create a binding object.



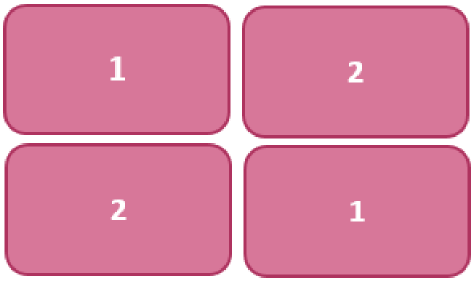
Report Container Properties

A Report Container is a part of a Report Chapter that contains a Report Data View (Table View, Matrix view).

Characteristics

In **Characteristics**:

- **Show All Filters** enables to define the filters displayed in the report:
  - when selected (default), all the Data View filters are displayed.
  - when cleared, you can select the required Data View filters from the available ones.
- **Data Source Instance Identifier**: is used only in case the same Report DataSet is used in several Data Views of the same Report Template.
  You can:
  - share the same Report DataSet instance with all the Report Data Views (default behavior).
  - define which Report DataSet instances are shared with the Report Data Views and which ones are distinct. In that case, you define the same numerical identifier to the Report Data Views that share the same Report DataSet instance.



Report Data View Properties: Matrix and Table

The following properties are common to both types of Report Content:

- **Report Matrix View**
- **Report Table View**

Characteristics

Report Matrix Views and Report Table Views show the same **Characteristics** page, including:

- **Sharing**:
  - "Public" (default): the Report Data View can be reused in several reports. The Report Data view is available at Report Template or Report Chapter creation.
  - "Private": the Report Data View cannot be reused.
- **General Style**, which defines the style to apply to the Report Data View. It defines the values for a set of Renderer-type Parameters.
- **Display**, which defines the display associated with the Report Data View:
  - "Table"
  - "Graphic"
  - "Graphic and table"
- **Chart Type**, which defines the graphic display type:
  - "Area Chart"
  - "Bar Chart"
  - "Line Chart"
  - "Radar Chart"
  - "Gauge Chart" (for a Report Table View)
  - "Set of gauges Chart" (for a Report Table View)
  - "Pie Chart" (for a Report Table View)
  - "Word Cloud" (for a Report Table View)

Filter

Report Matrix Views and Report Table Views include the same **Filter** page, which enables to define which filters can be displayed in a report for this Report Data View.

A filter enables to restrict a Report DataSet column values, which are considered in the report. This Report DataSet column is not necessarily part of the report.

☛ *See How to Define the Filters Displayed in a Report section.*

Characteristics

Each filter is defined by the following **Characteristics**:

- **Visibility** enables to hide the filter when it is fed by another filter or to display its value only.
- **Object Edition Mode** defines how objects are displayed and whether they are editable or not.

  `I.e.: Menu, In-Place Edit, No Edition.`
- **Input type** defines how the report user can define its condition:
  - "Single Value"
  - "Multiple Values" when the column includes a restricted number of values.
  - "Continuous" to define a range of values.

  The condition is also column data type dependent.
- **Behavior** defines the filter action on the report:
  - "Immediate Refresh" (for a report easily calculated): as soon as the user modifies the filter the report is updated.
  - "No Refresh": the user needs to click the Refresh button to update the report.
- **Filter Choice:** defines what can be selected: one or several objects, or contains a value.

  `I.e.: "Single Choice", "Multiple Choice", "Contains".`
- values:
  - **Initial Value:** defines the default value on report opening.
  - **Contains**: defines the values the user can use for a specific filter.
  - **Shows:** defines how to manage empty values in case of condition or not.

    `I.e.:`

    `"Add empty values": if no condition: displays empty values only, if a condition is defined: displays the elements matching the condition and also the empty values,`

    `"Display only empty values",`

    `"Display only non empty values", or`

    `"Consider all values": displays values matching the condition only.`
  - for numerical values: **Less than**, **Equals to**, **Greater than**
  - for character strings: **Begins with**, **Ends with**

Text

In **Text**, **_Settings** tab enables to define:

- an initial value
  An initial value is the default value on report opening.
  The user can modify the value.

  `paragraph [default Value]`
- a candidate value
  Candidate values are the values the user can use for a specific filter.

  `paragraph [Candidate]`

The paragraph content depends on the filtered data type.

For a filter of type:

- ***Occurrence***, the values are defined either by:
  - an object defined a conventional variable:

    `Value = &Current`

    `Example: Value=&CurrentUser`
  - an object list explicitly defined in field form:

    `Value = ~object1,~object2,…`
  - a set of occurrences obtained by applying a condition on the DataSource item occurrences

    `Value="Character string"`

    `All of the occurrences, whose short name includes "character string".`
  - a set of occurrences retrieved by a query execution

    ```
    Query = ~Query quoted : the values retrieved by the quoted query.
    If the query does not include a parameter it is executed straight away,
    else the included parameter is valued by the report parameter quoted in
    the Parameter variable.
    ```

    ☛ *Queries including more than one parameter are not exploitable.*

    `Parameter = ~Parameter`

    `Defines the report parameter which enables to value the query parameter.`
- ***Tabulated value attribute***, the values are defined either by:
  - a list of internal values explicitly quoted

    `Value=value1, value2, ...`

    `Example: certain values of a multivalued attribute.`
  - a set of values obtained by applying a condition on the attribute internal values (numerical values)

    `ValueInferiorTo="Numerical value"`

    `ValueSuperiorTo="Numerical value"`
- ***Date***, the values are defined either by:
  - an object defined by a conventional variable

    `Value = &CurrentDate`
  - a list of dates explicitly quoted (Mega internal format for dates)

    `Value = 2016/15/25,...`
  - a set of dates obtained by applying a condition on the DataSource item occurrences

    `ValueInferiorTo="date" : all the dates < to the date.`

    `ValueSuperiorTo="date" : all the dates > to the date.`
- ***Numerical***, the values are defined either by:
  - a list of explicit values

    `Value=12,200,3000`
  - a set of values obtained by applying a condition on the DataSource item occurrences

    `ValueInferiorTo="Numerical value"`

    `ValueSuperiorTo="Numerical value"`

Chart

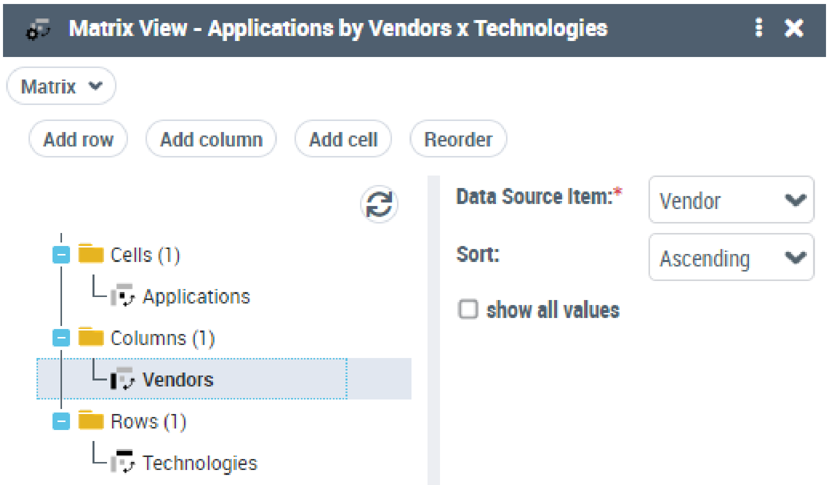
The **Chart** page is specific to Report Matrix Views and Report Table Views that include a chart. This page defines which element defines:

- the chart axes
- the chart legend (for Bar, Line, Area, Pie, or Radar charts)

Matrix (Report Matrix View)

The **Matrix** page is specific to Report Matrix Views. This page defines the matrix content, i.e. the data source columns on which is based the matrix:

- the **Rows** and **Column** folders:
  Each item defines the content of all the rows and columns displayed in the matrix. The displayed data is provided by a Report Data Source Element.
  - **Data Source Item** attribute defines the Report Data Source Element, which provides the data.
  - **Sort** attribute defines which sorting is applied to the row/column.
  - **Show all values** attribute defines if all the column possible values are displayed in the report. It is used only when rows or columns match a set a known values.

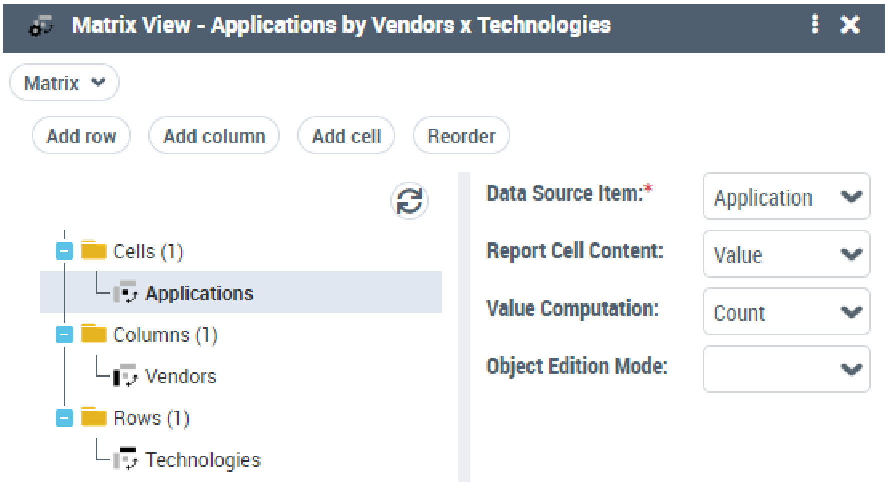


- the **Cells** folder:
  Each item defines the content of all the cells displayed in the matrix. The displayed data is provided by a Report Data Source Element.
  - **Data Source Item** attribute defines the Report Data Source Element, which provides the data.
  - **Report Cell Content** attribute defines what is displayed in the cell.

    `I.e.: Value, Check mark, or details of objects.`
  - **Value computation** attribute defines which computation is applied to the set of values grouped in the cell. It operates only for cells with numerical values.

    `I.e.: Average, Count, Max, Median, Min, or Sum.`
  - **Object Edition Mode** attribute defines how objects are displayed and whether they are editable or not.

    `I.e.: Menu, In-Place Edit, No Edition.`



This page also enables to **Add row**, **Add column**, **Add cell**, and also to **Reorder** the cells, columns, and/or rows in alphabetical order.

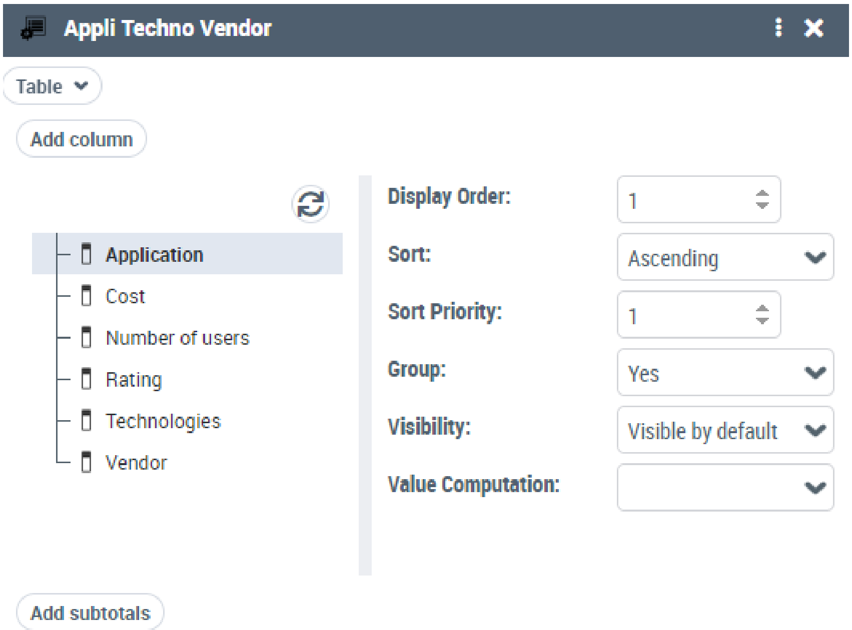
Table (Report Table View)

The **Table** page is specific to Report Table Views. This page defines the table content:

- **Columns Displayed** defines the data source columns on which are based the table.

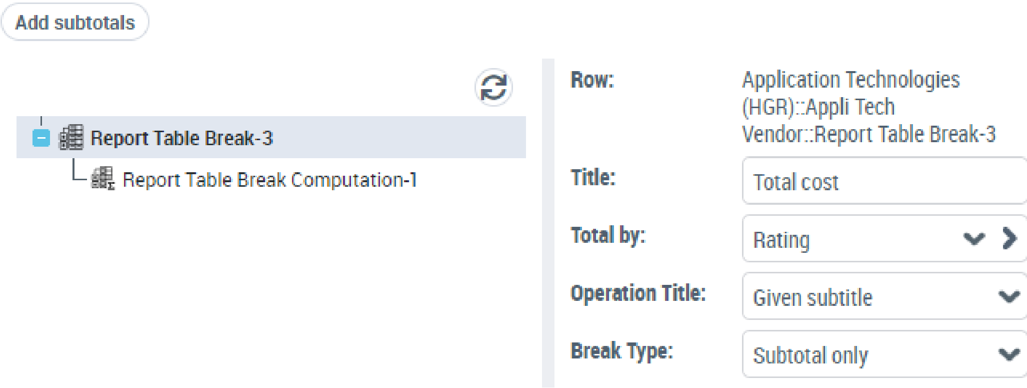
For each selected column item, the following parameters are defined:

- **Display order**: to define the column appearance order (if not specified, the row order is considered).
- **Sort** and **Sort priority**: to define the sorting direction and in which order it is applied.
- **Group**: to merge the pre-sorted set of cells showing the same value.
- **Visibility**: to define whether the column is visible by default or hidden by default, or never visible. When the column is visible by default or hidden by default the end-user can hide or display the column for his specific report using the report Edit mode.
- **Value Computation**: to define a computation when the tab includes a grouping.



You can add subtotals to grouped rows (**Group** value is set to "yes"). Each added **Report Table Subtotal** defines the sub-total rows that have to be added to the table:

- **Title**: defines the name displayed in the added row.
- **Total by**: defines to which grouped rows the row is added.
- **Operation Title**: to add a subtitle, regarding the column, in the break. You can define the title ("Given subtitle") or keep the automatic title ("Automatic subtitle" which is defined from the computation function name). Default value is "No subtitle".
- **Break type**: defines where the row appears ("Subtotal only" or "Subtotal and Grand total")



On the row you can display several computation results performed on each column including numerical values.

☛ *See How to Add Sub-totals in a Table Break section.*

Report Table Column Properties

Characteristics

- **Visibility** attribute defines if the column is available and/or visible in the displayed table. For "Visible" and "Hidden by default", the end-user can hide or display the column for his specific report using the report Edit mode.
- **Value Computation** attribute defines which computation is applied to the set of values grouped in the cell.
- **Display Order** attribute defines the position of the column in the table.
- **Group** attribute defines if identical values of a given column are merged.
- **Report Column Group Order** attributes defines in which order the columns are used to group the table rows.
- **Sort** attribute defines which sort is applied on the column.
- **Sort Priority** attribute defines in which order the columns are used to sort the table rows.
- **Column Type** attribute shows the data format of the Report table Column.

  `E.g.: string, long, float.`
- **Show all Values** attribute defines whether all the possible values for the column are displayed in the report or not.
- **Object Edition Mode** attribute defines how objects are displayed and whether they are editable or not.

  `E.g.: Menu, In-place edit, No edition.`
- **Data Source Item** attribute is a part of the Data Source used in the report.

Style (for conditional style)

The **Style** page is similar for the data view elements (line, column, and matrix cell) and enables to define a style when the condition is satisfied.

You can create a set of conditions and associate each of them with styles.

Conditions are considered one after another.

Report Data View Properties: Graph

A **Report Graph View** is used to display a graph in a Report Container. The data displayed in the graph is provided by a Report Data Source of GraphSet Definition type.

Characteristics

The **Characteristics** page includes the following properties:

- **Sharing**:
  - "Public" (default): the Report Graph View can be reused in several reports. The Report Graph view is available at Report Template or Report Chapter creation.
  - "Private": the Report Graph View cannot be reused.
- **Data Source** defines the GraphSet Definition on which the Report Graph View is based.
- **General Style** defines the style to apply to the Graph view (i.e. values for a set of renderer type parameters)

  `For example: node display style.`
- **3D illustration** defines whether 3D display is available or not
- **Default illustration** defines the default display when 3D illustration is available

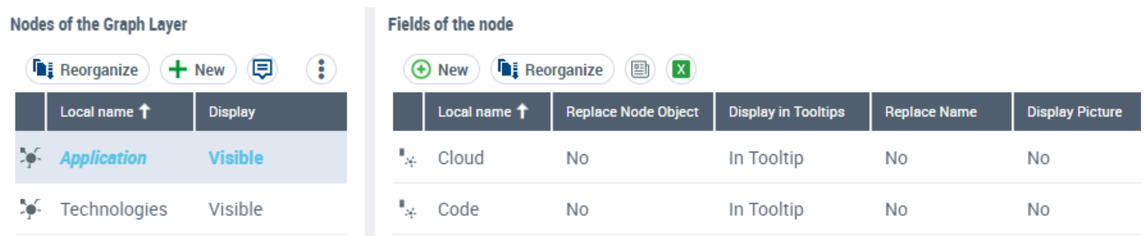
Graph

The **Graph** page defines each **Report Graph Layer** display:

- its **Visibility**

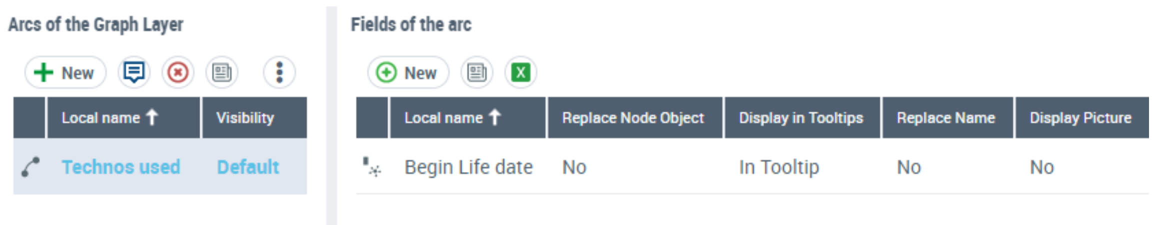
  `Values: Visible, Never visible, or Hidden by default.`
- its **Nodes**

  ☛ *For details on its properties and style, see Report Graph Node properties.*
- for each node its **Fields**
  A field can replace the node object, be displayed in tooltips (by default), or replace the object name.



- for each node its **Arcs**

  ☛ *For details on its properties and style, see Report Graph Arc properties.*
- for each arc its **Fields**



Filter

The **Filter** page enables to define which filters can be displayed in a report for this Report Graph View.

A filter enables to restrict the nodes or fields that are considered in the report.

☛ *See How to Define the Filters Displayed in a Report section.*

A filter is defined by the following **Characteristics**:

- **Filtering Group Id**
- **Input type** defines how the report user can define its condition:
  - "Single Value"
  - "Multiple Values"
  - "Continuous" to define a range of values.
- **Behavior** defines the filter action on the report:
  - "Immediate Refresh" (for a report easily calculated): as soon as the user modifies the filter the report is updated.
  - "No Refresh": the user needs to click the Refresh button to update the report.
- values:
  - **Contains**: defines the values the user can use for a specific filter.
  - **Shows:** defines how to manage empty values.

    I.e.: "Add empty values", "Display only empty values", "Display only non empty values", or "Consider all values"
  - for numerical values: **Less than**, **Equals to**, **Greater than**
  - for character strings: **Begins with**, **Ends with**

Report Graph Node properties

A node is defined by the following **Characteristics**:

- **Name Display** attribute enables to hide the name of node in the Graph.

  Values: Show Name (default) or Hide Name.
- **Display** attribute enables to hide the node in the Graph.

  Values: Visible (default) or Not visible.

You can also define a **Style** (or conditional style) on a node, see Style.

Report Graph Arc properties

An arc is defined by the following **Characteristics**:

- **Visibility** attribute enables to override the visibility of the GraphSet Arc Definition in the Graph.

  `Values: Visible (default) or Not visible.`
- **Arrow Direction** attribute enables to override the orientation of the GraphSet Arc Definition or to hide the arrow in the Graph.

  `Values: Default, Reverse, or No arrow.`
- **Grouping specifications** section enables to configure:
  - a grouping by object or by field, or grouping all of the arcs
  - the detailed list of objects

  `Values: Node, Object field of a node.`

You can also define a **Style** (or conditional style) on an arc, see Style.

Style

The **Style** page enables to define values for a set of renderer type parameters of the Report Graph View at graph element level (node, arc).

The **Style** page of a node/arc enables to define a style on the node/arc when the condition is satisfied.

If no condition is set, the style always applies. You can create a set of conditions and associate each of them with styles. Conditions are considered one after another.

The style can apply:

- to all the nodes or arcs,

  ☛ *See Report Style Properties.*

- on condition to certain nodes or arcs

  ☛ *See Report Style Condition Properties.*

  `For example, you can define:`

  `- a specific style on arcs.`

  `- a conditional style on nodes: the nodes "Applications" for which the "Cloud" field value is "Saas" are displayed with a blue background.`

☛ *See example in How to apply a Conditional Style to a Graph section.*

Report Data View Properties: Tree

A **Report Tree view** is used to display a tree in a Report Container. The data displayed in the tree is provided by a Report Data Source of TreeSet Definition type.

Characteristics

The **Characteristics** page includes the following properties:

- **Data Source** defines the TreeSet Definition on which the Report Tree View is based.
- **Display** defines the renderer type to apply to the Tree view.

  `E.g.: Dendrogram, TreeTable, or Treemap.`

- **General Style** defines the style to apply to the Tree View (i.e. values for a set of renderer type parameters).

Filter

The **Filter** page enables to define which filters can be displayed in a report for this Report Tree View.

A filter enables to restrict the nodes that are considered in the report.

☛ *See How to Define the Filters Displayed in a Report section.*

A filter is defined by the following **Characteristics**:

- **Input type** defines how the report user can define its condition:
  - "Single Value"
  - "Multiple Values"
  - "Continuous" to define a range of values.
- **Behavior** defines the filter action on the report:
  - "Immediate Refresh" (for a report easily calculated): as soon as the user modifies the filter the report is updated.
  - "No Refresh": the user needs to click the Refresh button to update the report.
- values:
  - **Contains**: defines the values the user can use for a specific filter.
  - **Shows:** defines how to manage empty values.

    `I.e.: "Add empty values", "Display only empty values", "Display only non empty values", or "Consider all values"`
  - for character strings: **Begins with**, **Ends with**

Report Data View Properties: Value

A **Report Value view** is used to display a gauge in a Report Container. The data displayed in the gauge is provided by a Report Data Source of query type.

The gauge report shows the query result value against the number of concerned occurrences in the repository.

Characteristics

The **Characteristics** page includes the following properties:

- **Sharing**:
  - "Public" (default): the Report Value View can be reused in several reports. The Report Value view is available at Report Template or Report Chapter creation.
  - "Private": the Report Value View cannot be reused.
- **Data Source** defines the query (without parameters) from which the Report Value View is built.
- **Display** defines the renderer type applied to the Tree view.

  `Value: Graphic.`
- **Chart Type** defines the chart type.

  `Value: Gauge.`

Chart

The **Chart** page defines how the gauge is built and customized:

- **Value** defines the gauge value, i.e. the (Data Source) query on which the chart is based.
- **Value Computation** defines the value computation mode. For example, it can be more efficient to perform a computation relating to a minimum value or to a maximum value.

  `Values: Default, Added to min, subtracted from max.`
- **Min value** defines the gauge minimum value. By default this minimum value is 0. You can define a query that sets the gauge minimum value.

  `Values: a query of the same source, or empty (in that case 0).`
- **Max value** defines the maximum value of the gauge. By default this maximum value is the number of occurrences of the object type that represents the gauge. You can define a query that sets the gauge maximum value.

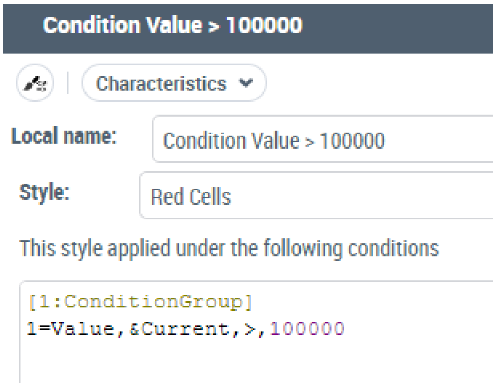
  `Values: a query of the same source, or empty (in that case the maximum number of occurrences concerned).`
- **Drill-down** defines whether a drill-down is available on the gauge or not.

  `Values: yes, no`
- **Specific Drill-down** defines the occurrences displayed in the drill-down. By default all the occurrences are displayed. This parameter enables to select another query (than the Data Source query) for the drill-down.

  `Value: a query of the same source, or empty (in that case the Data Source query is used).`
- **_Settings** automatically fed (and updated) while setting the chart parameters.

Report Style Condition Properties

The condition is defined in text format. If the text is empty the style is applied with no condition.



The text syntax is as follows:

```
[Paragraph]
<Order number>=<condition type>,<test subject>,<operator>,<value>
```

A paragraph can include one or several conditions. Each condition is expressed in a distinct line.

The set of conditions included in a paragraph is evaluated and if all of the conditions are TRUE, then the style is applied.

With several paragraphs, if the set of conditions of one paragraph is TRUE, then the style is applied.

☛ *See How to Define the Filters Displayed in a Report section.*

☛ *See How to apply a Conditional Style to a Graph section.*

Condition on value (Table/Matrix/Tree View element)

This condition is in the following form:

<order number>=**Value**,<report element>,<Operator>,<Value>

Where:

<report element> is expressed by one of the following:

- ***&Current***
  &Current is the report data view element linked to the condition.
  It enables to test for example:
  - the current value of a matrix cell, or
  - the current cell of a of a given column.

  `Example: for a Table each table cell, for a column each column cell value.`

  ☛ *For an example, see How to Define the Filters Displayed in a Report section.*
- ***&ColumnHeader***
  When the report data view element is a matrix cell, the &ColumnHeader enables to test the column header value.

  `Example: for a Table each column header value.`
- ***&RowHeader***
  When the report data view element is a matrix cell, the &RowHeader enables to test the row header value.
- ***~Column (~subtotal, ~cells, ~rows, ~columns)***
  ~Column is a field that identifies the column, which enables to test the current cell value.
  The field can also be ~subtotal, ~cells, ~rows, or ~columns.

  `Use case: applying a style on an element but testing another element value.`
- ***~Paramtype***
  ~Paramtype is a field that identifies a report parameter (or a DataSet parameter), which enables to test its value.
  Only multiplicity 1 parameters can be tested.

  `Example: a report parameter or a DataSet parameter.`
- ***&CurrentUser (&CurrentDate, etc.)***
  All of the conventional variables can be tested in this paragraph.

<OPERATOR> can be:

- <
- >
- =

<Value> is a character string.

Example: to add a condition style on values superior to 10,000 enter:

```
[1:ConditionGroup]
1=Value,&Current,>,10000
```

Condition on value (Report Graph View element)

In a graph, you can add conditions on **Value** on:

- node
- arc
- path

The conditional style applies to arcs or nodes.

For a node, the **Value** can concern:

- the object associated with the node,
- any field value of the node,

  For an example, see Applying a conditional style to a node.
- any field property if the field is an object.

For a path, the **Value** can concern:

- any field of nodes/arcs
- any node of the path

  The conditional style applies to each arc, which is part of the path.

For an arc, the **Value** can concern:

- (grouped arcs) the arc number
- (arc) any field of the arc

This condition is in the following form:

<order number>**=Value**,<report element>,<Operator>,<Value>

Where:

<report element> is expressed by one of the following:

- ***&Current***
  &Current is the node/arc/path linked to the condition.
- ***~Field***
  ~Field is any field of the node/arc, whose value can be compared with the given value.
- ***&ArcsNumber***
  &Arcs Number is the number of occurrences behind the grouped arcs.

<OPERATOR> can be:

- <
- >
- =

<Value> is expressed by one of the following:

- any conventional variables &CurrentUser (&CurrentDate, etc.)
- ***~MegaField***
  Any MegaObject.
- ***~Field***
  Any other Field value of the node/arc.
- a character string

> ***Example:*** to add a condition style on Application with cost values superior to 10,000 enter:
>
> `[1:`**`ConditionGroup`**`]`
>
> `1=`**`Value`**`,&ApplicationCost,>,10000`

Complex condition embedding a MetaTest

This condition is in the following form:

Metatest,<report element>,<~MetaTest>,<test result>

Where:

- <report element> is expressed in the same way as for a condition on a value.
  - ☛ *See Condition on value (Table/Matrix/Tree View element):*
- <~MetaTest> is the field that identifies the MetatTest to be executed on the occurrence corresponding to the report element quoted.
- <test result>: can take the following values:
  - TRUE
  - FALSE

Condition on a report element regardless of its content

This condition is in the following form:

id,<report element>,**IS**,<conventional value>

Where:

- <report element> can be either:
  - &CurrentColumn to designate the current column, or
  - &CurrentRow to designate the current row.
- <conventional value> can be:
  - ODD
  - EVEN
  - LAST
  - FIRST

Report Style Properties

Characteristics

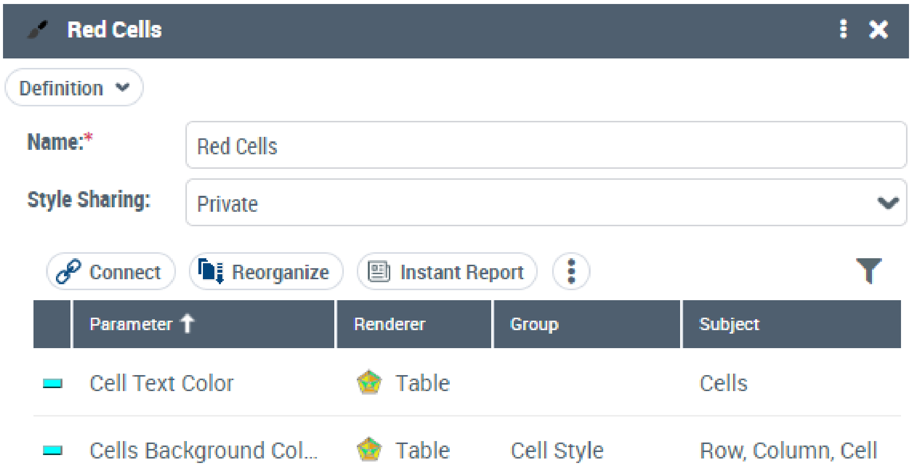
A style is a set of renderer parameters and their values. They include all the parameters available for report editing.

☛ *See How to Create a Report Style section.*

Definition

The **Definition** page enables to add render parameters that will be valued for this style.

Style Sharing: by default the style is "Private". "Public" value enables to make the style available to all users when editing reports.



REPORT TEMPLATE CREATION

The following point are details:

- Creating a Report Template
- Creating a Report Data View
- Adding a Report Chapter to a Report Template

Creating a Report Template

☛ *For a big picture see Report Template Creation: The Big Picture.*

You can create a Report Template from:

- a Report DataSet
- a GraphSet
- a TreeSet
- a query

Else you can create a Report Template using a Report Data View.

To answer specific needs, you can also implement a Report Template using macro.

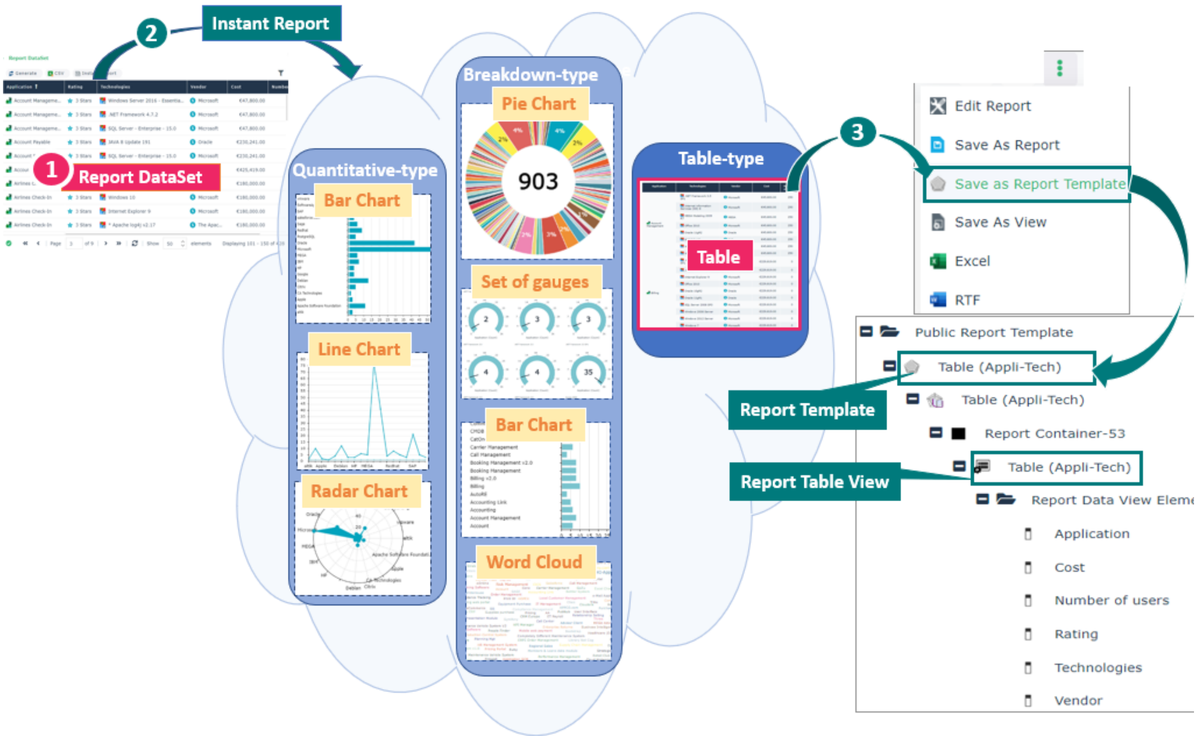
Creating a Report Template from a Report DataSet

In HOPEX Power Studio, you can create a Report Template from a Report DataSet.

From the **Data** page of the Report DataSet, you can launch an Instant Report, then save it as:

- a Report Template
- a Report Data View, to use it in a Report Template

> *Note: from the instant report,* ***Save as Report*** *generates a public report as well as its corresponding private* ***Report Template****, which you can access from* ***Private Report Template > My Report Templates*** *folder.*



To create a Report Template from a Report DataSet:

1. Access the Report DataSet properties (from Report **Studio > Report DataSets**).
2. Display its **Data** page.
3. In the **Report DataSet** section, click **Instant Report** .
4. Select the required instant report type.

   E.g.: ***Breakdown*** (pie chart, bar chart, set of gauges, word cloud), ***Matrix*** (bar chart, radar chart, ***Quantitative*** (bar chart, line chart, radar chart), ***Table***.
5. Click **OK**.
   The instant report is displayed in full page, **Report** section.
6. In the **Report** section, click **More** ⋮ **> Save As Report Template** .

   > *Note: if you customize the report display, this customization is not taken into account in the Report Template.*
7. In the **Name** field, enter your Report Template name.

8. Click **OK**.
   A public Report Template is created, with its public **Report Table View**. They both have the same name.
   You can access the Report Template:
   - directly from your **Home** page, from **Custom Report Templates** tile, or
   - from the **Public Report Template** folder (**Report Studio > Report Template Definition >Report Templates**)

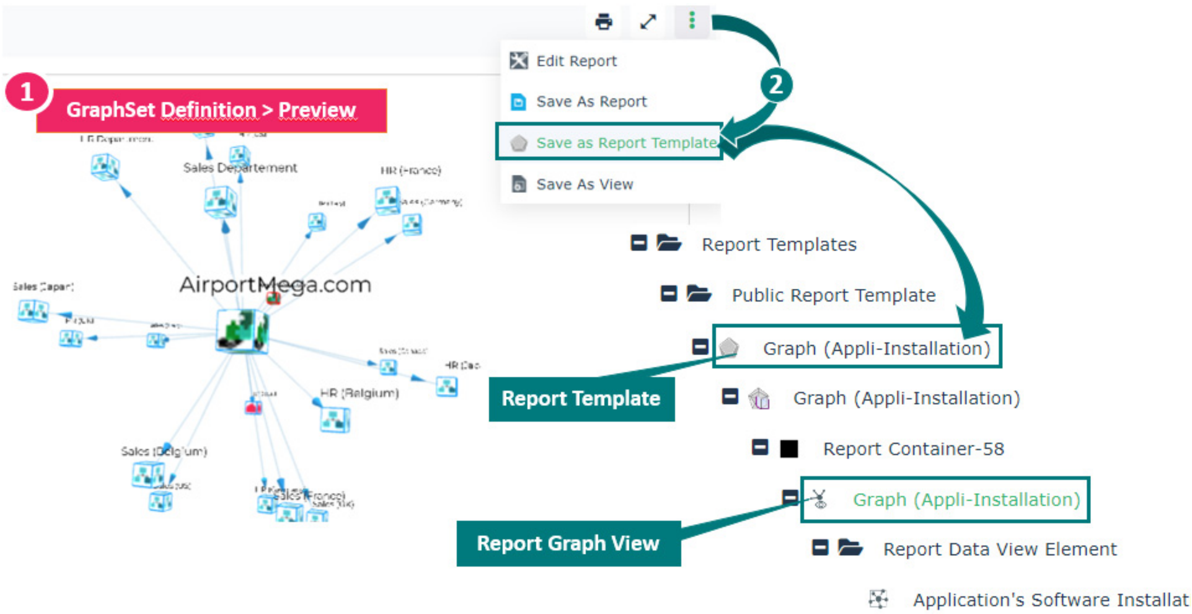
Creating a graph-type Report Template from a GraphSet

In HOPEX Power Studio, you can create a Report Template from a GraphSet.

From the **Preview** page of the GraphSet Definition, you can save the graph as:

- a Report Template
- a Report Data View, to use it in a Report Template

> ☛ *Note that **Save as Report**, generates a public report as well as its corresponding private **Report Template**, which you can access from **Private Report Template > My Report Templates** folder.*



To create a graph-type Report Template from a GraphSet:

1. Access the GraphSet Definition properties.
   > ☛ *See Accessing the GraphSet Definitions/Accessing the Report Templates and their Constituents.*
2. Display its **Preview** page.
3. In the graph, click click **More** ⋮ **> Save As Report Template** .
   > ☛ *Note: if you customize the report display, this customization is not taken into account in the Report Template.*
4. In the **Name** field, enter your Report Template name.

5. Click **OK**.
A public Report Template is created, with its public **Report Graph View**. They both have the same name.
You can access the Report Template:
   - directly from your **Home** page, from **Custom Report Templates** tile, or
   - from the **Public Report Template** folder (**Report Studio > Report Template Definition >Report Templates**)

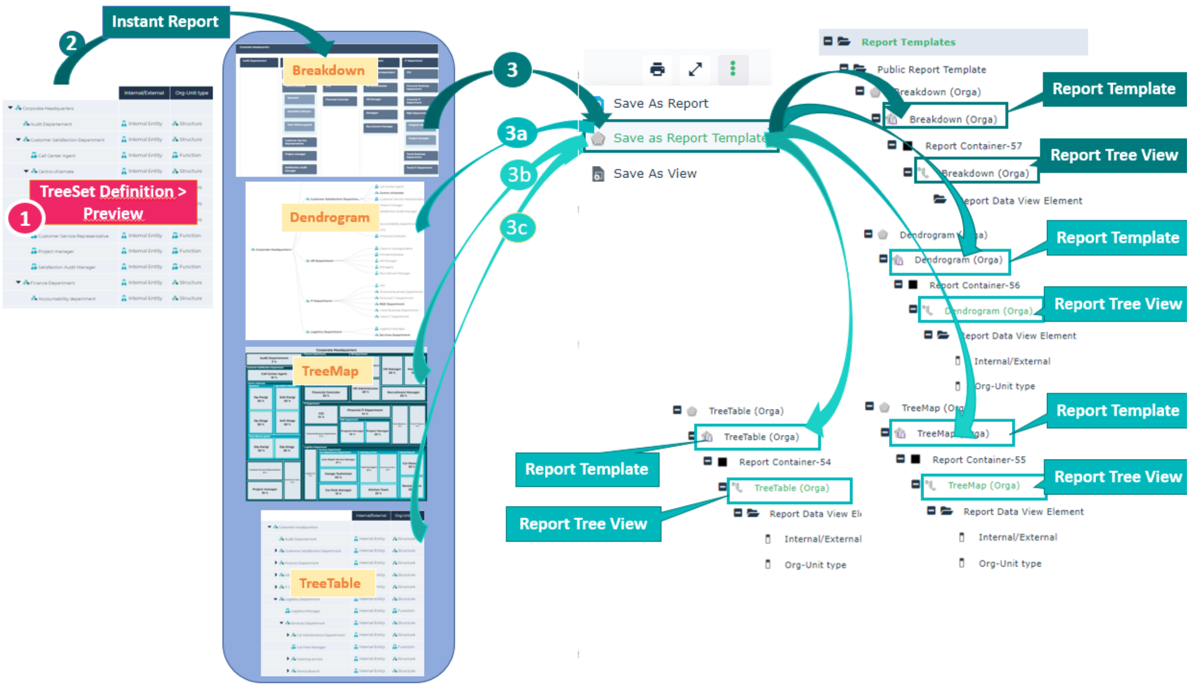
Creating a tree-type Report Template from a TreeSet

In HOPEX Power Studio, you can create a Report Template from a TreeSet.

From the **Preview** page of the TreeSet Definition, you can save the tree as:

- a Report Template
- a Report Data View, to use it in a Report Template

> ☛ *Note that **Save as Report**, generates a public report as well as its corresponding private **Report Template**, which you can access from **Private Report Template > My Report Templates** folder.*



You can save only one tree report at a time:

- a breakdown
- a dendrogram
- a TreeMap
- a TreeTabble

To create a Report Template from a tree report:

1. Access the TreeSet Definition properties.

> ☛ *See Accessing the Report Templates and their Constituents.*

2. Display its **Preview** page.
3. Click the renderer type you want to save.

   `E.g.: Breakdown, Dendrogram, TreeMap, or TreeTable.`

4. In the graphic, click click **More** ⋮ **> Save As Report Template** .

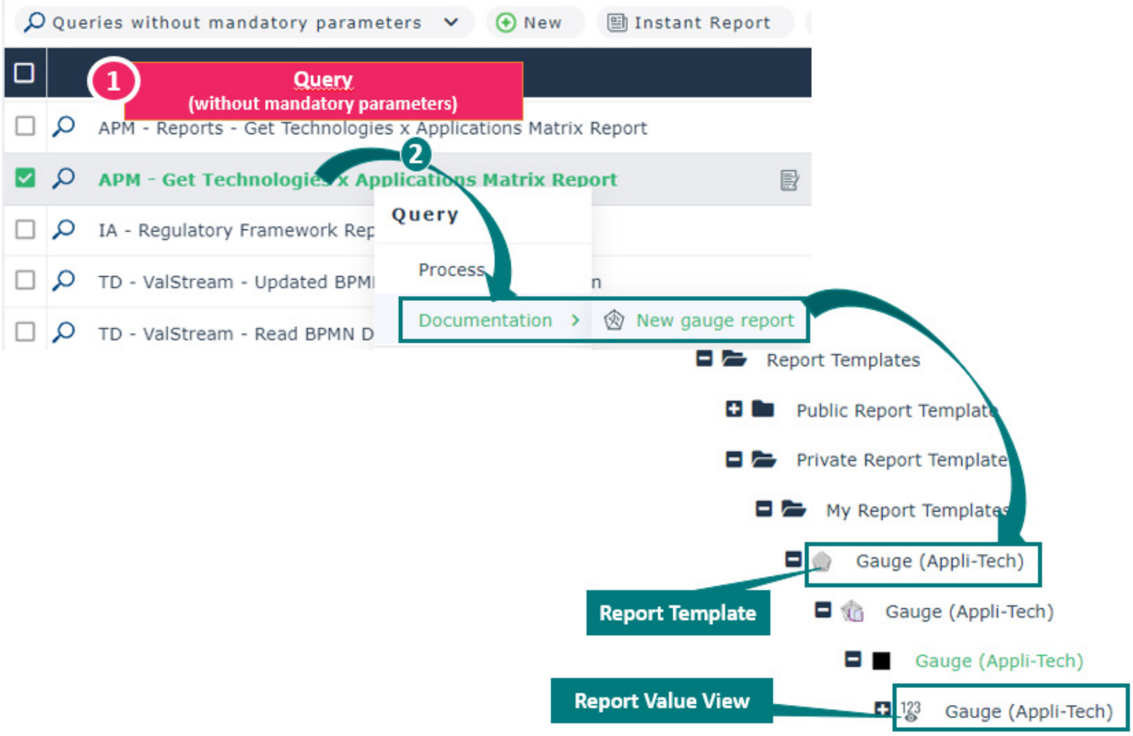
   ☛ *Note: if you customize the report display, this customization is not taken into account in the Report Template.*

5. In the **Name** field, enter your Report Template name.
6. Click **OK**.
   A public Report Template is created, with its public **Report Tree View**. They both have the same name.
   You can access the Report Template:
   - directly from your **Home** page, from **Custom Report Templates** tile, or
   - from the **Public Report Template** folder (**Report Studio > Report Template Definition >Report Templates**)

Creating a gauge-type Report Template from a query

In HOPEX Power Studio, you can create a Report Template from a query (without parameters). This Report Template enables to create gauge reports.

☛ *See use cases in How to Create and Customize a Gauge Report Template section.*



To create a gauge-type Report Template from a query:

1. Connect to **HOPEX Studio** desktop.

   ☛ *See Connecting to HOPEX Studio Desktop.*

2. Click **Report Studio > Queries** navigation menu.
3. In the **Queries without mandatory parameters** list, right-click a query and select **Documentation > New gauge report**.
4. In the **Name** field, enter the report name.
5. Click **OK**.
   The public gauge report is displayed and its corresponding private Report Template (with associated Report Value View) is also automatically created. They all have the same name.
   You can access the Report Template:
   - directly from your **Home** page, from **Custom Report Templates** tile, or
   - from the **Private Report Template** folder (**Report Studio > Report Template Definition >Report Templates**)

Creating a Report Template using a Report Data View

You can create a Report Template right away including a Report Chapter with a Report Data View saved in the **Public Report Data Views** folder.

Prerequisite: your Report Data View is created.

☛ *To create a Report Data View, see Creating a Report Data View.*

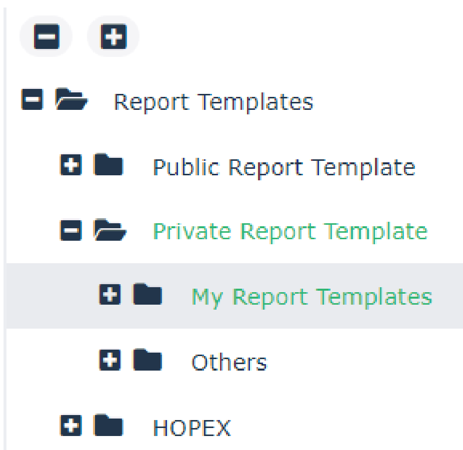
To create a Report Template using a Report Data View:

1. Access the **Report Templates** folders.

   ☛ *See Accessing the Report Templates and their Constituents.*

2. Right-click **My Report Templates** folder, and select **New > Report Template**.

   ☛ *As at creation a Report Template is "private" (see Report Template Properties), if you create a Report Template from another sub-folder, the Report Template is still saved in **My Report Template** folder.*



   Your Report Template **Basic Properties** window appears.

3. In the **Name** field, enter your Report Template name.

   ☛ *By default the Report Template name is: Report Template-x (x is a number).*

4. In the drop-down list select the Report Data View you want to fill your first chapter with.
5. Click **OK**.
   The Report Template is created with a single chapter based on the Report Data view selected.

   You can:

   - add other chapters.

     ☛ *See Adding a Report Chapter to a Report Template.*

   - customize the Report Template.

     ☛ *See Customizing a Table Report Template, Customizing a Graph Report Template, Customizing a Gauge Report Template.*

Implementing a Report Template using a macro

To build your report chapters, you can use an existing macro on condition that parameters are compatible.

To implement a Report Template using a macro:

1. Access the **Report Templates** folders.

   ☛ *See Accessing the Report Templates and their Constituents.*

2. Right-click **My Report Templates** folder, and select **New > Report Template**.

   ☛ *As at creation a Report Template is "private" (see Report Template Properties), if you create a Report Template from another sub-folder, the Report Template is still saved in **My Report Template** folder.*

   The **Basic Properties** page appears.
3. In the **Name** field, enter your Report Template name.

   ☛ *By default the Report DataSet Definition name is: Report Template-x (x is a number).*

   `Example: Application`
4. Click **Next**.
   The **Parameters** definition wizard appears.

   Parameter definition indicates the input data type the user must enter for the report calculation (e.g.: Applications).

   ☛ *Do not define any parameter if you want the report to be launched without requiring the user to select any input objects.*

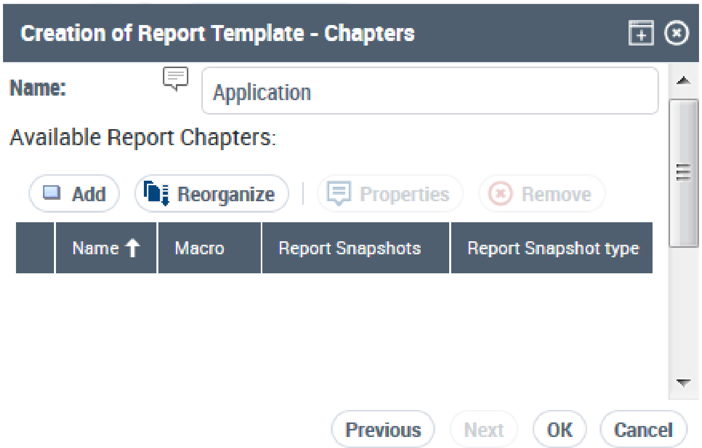
   For each parameter define the available object type that can be instantiated for the report (parameter with HOPEX objects or with simple types).

   ☛ *See How to Define Parameters in a Report Template section.*

5. Click **Next** to continue the Report Template definition:

   ☛ *Else click **OK** to terminate the Report Template definition later. You need to define at least one Report Template Chapter. See Adding a Report Chapter to a Report Template.*

   The **Chapters** definition wizard appears.



6. Click **Add** .
7. In the **Name** field, enter the Report Chapter name.

   ☛ *Else you can select an existing Report chapter in the drop-down list.*

   To build your report chapters, you can use an existing macro, provided that parameters are compatible.

   MEGA provides in particular the " Analysis - Set of Parameters" macro, which describes report template parameters and the way in which they are interpreted by the report chapter.

   Macros have a comment available in the form of a system note.
8. Click **Next**.
9. In the **Based on** field, select "Macro".
10. In the **Macro** field, click the right-oriented arrow and select **Add Macro**.

    ☛ *If the macro is already created, select the macro in the drop-down list and click **OK**.*
11. In the **Name** field, enter a name for the macro.
12. Click **Next**.
13. Select **Create a Java Macro** and click **Next**.
14. In the **Setup JAVA Macro**:
    - In the **Class Name** field, enter the class defined in Eclipse when developing the Java macro.
    - In the **Package Name** field, enter the macro path.

    `Example: "com.mega.modeling.analysis.reports" for those supplied by MEGA.`

    ☛ *To create the macro, see How to Define Parameters in a Report Template section.*

15. Click **OK**.
The macro name appears in the **Macro** field.

☛ *For such a Report Template example, see Report Template Example with Report Chapters Based on Macros.*

The macro appears in the Report Template tree, under the Report Chapter concerned.
ted.
You can:

- add other chapters.

  ☛ *See Adding a Report Chapter to a Report Template.*
- customize the Report Template.

  ☛ *See Report Template: How To.*

Creating a Report Data View

You can create a **Report Data View** from:

- the **Preview** page of a ***Report DataSet Definition*** properties, a ***GraphSet Definition*** properties, or a***TreeSet Definition*** properties.
- a query.

To create:

- a **Report Table View** or a **Report Matrix View** from a **Report DataSet Definition**, see Creating a Report Data View from the Report DataSet
- a **Report Graph View** from a **GraphSet Definition** properties, see Creating a Report Graph View.
- a **Report Tree View** from a **TreeSet Definition** properties, see Creating a Report Tree View.
- a **Report Value View** from a **Query**, see Creating a gauge-type Report Template from a query.

Adding a Report Chapter to a Report Template

At any time you can add a Report Chapter to a Report Template.

Prerequisite: your Report Data View or your macro is created.

☛ *To create a Report Data View, see Creating a Report Data View.*

To add a Report Chapter to a Report Template:

1. Access the Report Template concerned.

   ☛ *See Accessing the Report Templates and their Constituents.*
2. Right-click the Report Template and select **Add > Report Chapter**.

   ☛ *Else, in the **Creation of Report Template - Chapter** window, click **Add**.*
3. In the **Name** field, enter the Report Chapter name.
4. In the **Based on** field, select on what is based the report Chapter:
   - a Report Data View or
   - a Macro
5. In the **Report Data view / Macro** drop-down list select the Report Data view / Macro.

6. Click **OK**.

CUSTOMIZING A REPORT TEMPLATE

See:
- Managing a Report Snapshot
- Organizing the Report Chapters
- Customizing the Report Chapter Export Format
- Customizing Parameter Display
- Customizing the Report Container Group display

Managing a Report Snapshot

Report snapshot creation is available for reports based on macros as well as for reports based on Report Data View(s).

Report snapshot creation is useful for a fast Report Chapter display and is also used for export operations.

When the report snapshot creation is not activated the report chapter display is fully recalculated at each refresh, which can be time consuming.

For each Report Chapter, you can activate or deactivate the report snapshot creation and define its type.

To manage a report snapshot:

1. Access the Report Chapter properties: **Characteristics**.

   ☛ *To access the Report Chapter properties, see Accessing the Report Templates and their Constituents.*

   ☛ *See Report Chapter Properties.*
2. In the **Snapshot** section, from the **Report Snapshots** drop-down list select:
   - "Yes" to activate the report snapshot creation.
   - "By Format" to activate a report snapshot creation. In this case a report snapshot for each format is created (html, pdf, excel).
   - "No" to deactivate the report snapshot creation.
3. In the **Snapshot** section, from the **Report Snapshot type** drop-down list select:
   - "Global Snapshot" to define a single snapshot for all the users.
   - "Snapshot by profile" to define a single snapshot by profile.
   - "Snapshot by user" to define a single snapshot by user.

Organizing the Report Chapters

You can sort the Report Chapters of a Report Template:
- manually according to your needs, or
- automatically in alphabetical order.

To organize the Report Chapters of a Report Template:

1. Access the Report Template properties: **Chapters**.

   ☛ *See Accessing the Report Templates and their Constituents.*

2. In **Available Report Chapters**, click **Reorganize**.
3. To:
   - customize the Report Chapter order: drag and drop the chapters to their required position.
   - sort the Report Chapters in alphabetical order: click **Alphabetical order**.
4. Click **OK**.

Customizing the Report Chapter Export Format

You need to define the available report export formats according to the report specific purpose. For reports intended to:

- produce documents that need to be printed entirely (more than one chapter) and should fit in a printable format (A4, letter), the RTF export format must be available.
- produce dashboards or to be consulted online, the RTF export format should not be available.

Depending on the report purpose and use, you must define its Report Chapter export formats.

To define the available export format for a Report Chapter:

1. Access the Report Chapter properties: **Characteristics**.

   ☛ *See Accessing the Report Templates and their Constituents.*

2. According to the report specific purpose, modify the default export formats available for the Report Chapter:
   - (selected by default) **Is compatible with Excel**
   - (selected by default) **Is compatible with RTF**
3. Click **OK**.

   According to the export format selected, the Excel and RTF corresponding buttons are available in the report result.

   ☛ *Note that the **Print Report** button is always available on the reports and enables to print (or print in a PDF file) the report.*

Customizing Parameter Display

Grouping parameters (order field)

You can group the display of report parameters. A group of parameters corresponds to a MetaAttributeValue that is defined on the "Report Parameter" MetaClass and then connected to the parameters of report templates concerned.

Consider a Report Template "Org-Unit Characteristics", which contains three parameters:

"Status",

"Street Number", which can include one or no value, with a multiplicity of "0..1".

"City", which should display a mandatory value, with a multiplicity of "1".

For consistency and visibility, the following procedures detail how to display "Street Number" and "City" parameters in an "Address" group.

To create the "Address group":

1. Access the **Report Parameter** MetaClass properties.
2. In the **MetaAttribute** page, open the **Displayed in Group** MetaAttribute properties
3. Select **Characteristics - Standard** page.
4. In the **MetaAttributeValues for enumerated MetaAttribute only** section, click **Add** and create the MetaAttribute Value "Address".
5. Specify an **Internal Value** as well as values in each data language.
6. Exit HOPEX and compile the metamodel to take account of modifications.

To connect the "Address" group to the parameters of the "Org-Unit Characteristics" Report Template:

1. In the "Org-Unit Characteristics" Report Template, select the "Street Number" parameter and open its properties.
2. Select **Characteristics**.
3. In the **Displayed in Group** field, select the "Address" MetaAttributeValue.
4. Carry out the same procedure for the "City" parameter.

> ☛ *Note that if both parameters have the same order number, the "City" parameter appears in the report before the "Street Number" parameter, the multiplicity "1" being positioned before the multiplicity "0..1". For more details, see* *Defining object display order in generated reports.*

To position "Street Number" before "City", modify the MEGA order number of the "Street Number" parameter:

1. In the "Street Number" properties, select **General > Administration** page.
2. In the **Order** field, enter an order number lower than that of the "City" parameter.
3. Click **OK**.

Defining object display order in generated reports

Parameters without groups

For parameters not containing groups, display order depends on:

- **MEGA order number**: objects of a list have an order number taking value "9999" by default. You can modify this value for each object (from the object properties, **General > Administration** page).
- **Multiplicity**: for parameters with the same MEGA order number, display order depends on multiplicity:
  - 1. Multiplicity 1
  - 2. Multiplicity 0..1
  - 3. Multiplicity 1..*
  - 4. Multiplicity * and NONE (the two are equivalent)
- **Alphabetical order**: for the same multiplicity, it is alphabetical order that is taken into account.

Parameters with groups

The parameter group display order depends on group alphabetical order.

In each group, parameters are displayed in MEGA order.

If parameters of the group have the same order number, their display order depends on their multiplicity:

- 1. Multiplicity 1
- 2. Multiplicity 0..1
- 3. Multiplicity 1..*
- 4. Multiplicity * and NONE (both are equivalent)

For the same parameter multiplicity, it is alphabetical order that is taken into account.

Parameters with and without group

Parameters that are not in groups appear in first position. They are followed by groups.

Customizing the Report Container Group display

Customizing the Report Container group display includes modifying:

- the Report Data View display (by default they are displayed in a same row).

  For example, it is useful with a number of Report Data Views to modify the display for a tab view.
- the Report Data View display order.

To customize the Report Container Group display:

1. Access the Report Container Group properties.

   ☛ *See Accessing the Report Templates and their Constituents.*
2. Select **Characteristics**.

3. To define the Report Data View display, in the **HTML Layout** field, select:
   - "Horizontal" to display the Report Data Views on the same row.
   - "Tab", to display the Report Data Views in tabs.
   - "Vertical" to display the Report Data Views in a single column.
4. To modify the Report Data View display order, in the **Report Containers** section, click **Reorganize** and drag and drop the Report Containers to the required place.
5. Click **OK**.

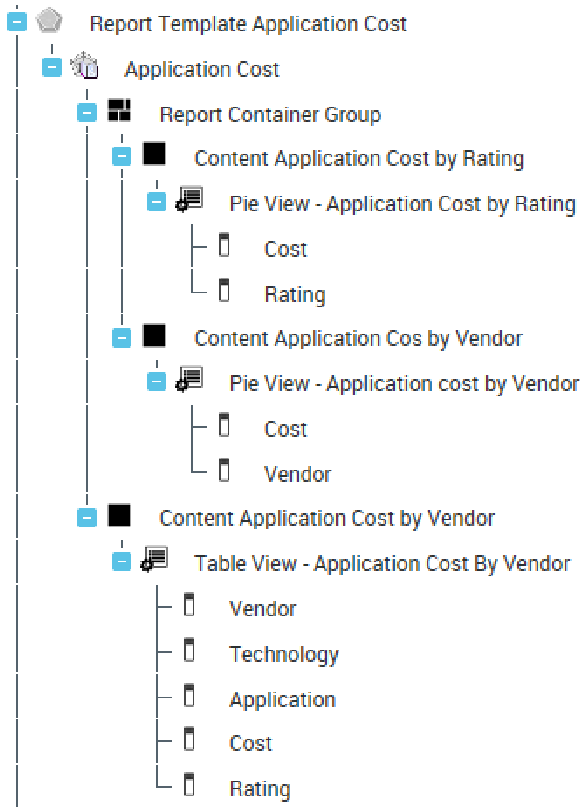
Report Template Examples

Report Template Example with a Report Chapter Based on a Report DataSet

Each **Report DataSet** column is an element that can be used in the Report DataSet Views.

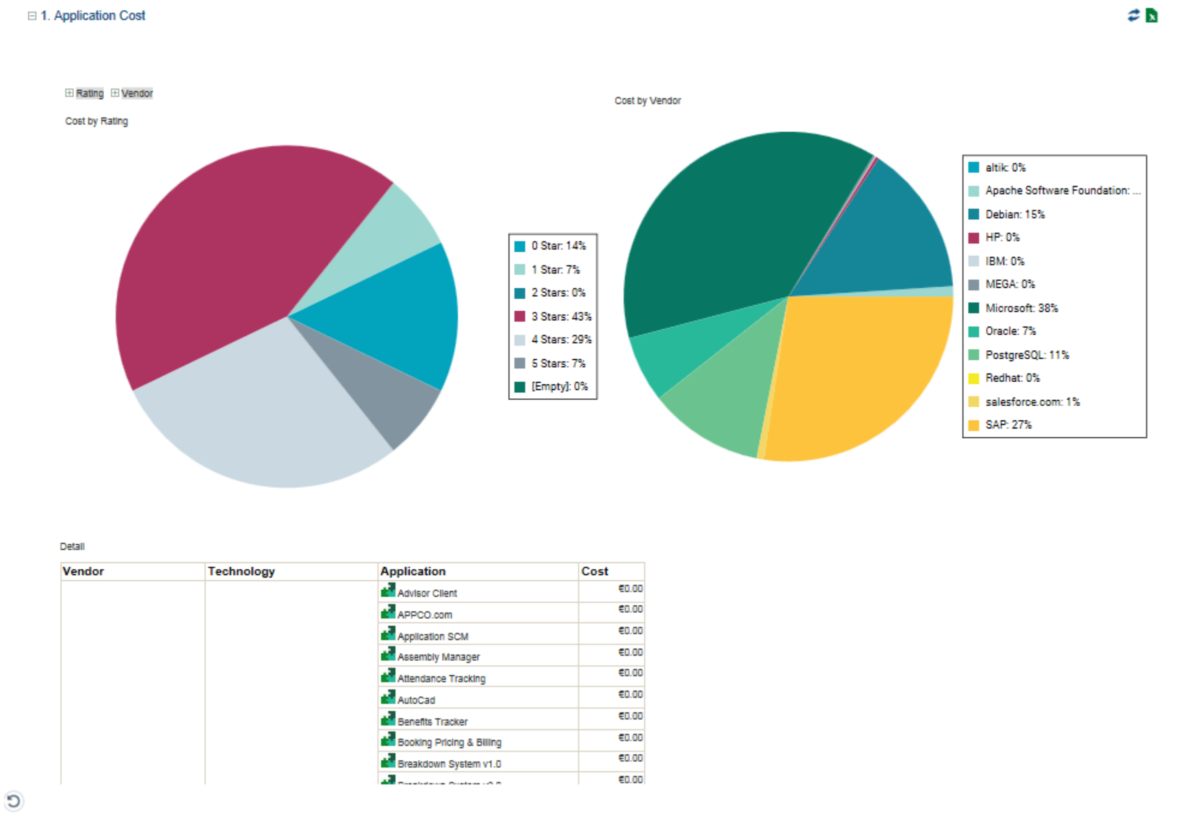
For example, in the **Report Studio samples** folder, the "Report Template Application cost" **Report Template** is based on the single "Application cost" **Report Chapter** with:

- a Container Group, which includes:
  - a Report Container showing a pie based on the "Application Technologies" **Report DataSet** elements ("Cost" and "Rating")
  - a Report Container showing a pie based on the "Application Technologies" **Report DataSet** elements ("Cost" and "Vendor")
- a Report Container showing a table based on the "Application Technologies" **Report DataSet** elements ("Vendor", "Technologies", "Application", "Cost", and "rating").



A report based on this "Report Template Application cost" **Report Template** shows the following two rows:

- the first row displays:
  - a pie chart showing the application "cost" by rating
  - a pie chart showing the application "cost" by vendor
- the second row displays a table with "Vendor", "Technologies", "Application", and "Cost" columns.

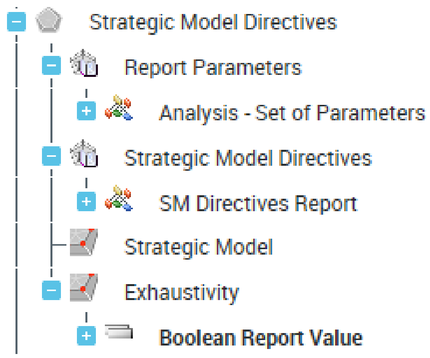


Report Template Example with Report Chapters Based on Macros

The Report Templates included in the **Public Report Templates > MEGA** folder are mostly based on macros.

For example, the "Strategic Model Directives" **Report Template**:

- generates the following **Report Chapters**:
  - "Report parameters" Report Chapter , which is based on the "Analysis - Set of parameters" macro .
  - "Strategic Model Directives" Report Chapter , which is based on the "SM Directive Report" macro .
- uses the following **Report Parameters**:
  - "Strategic Model" Report Parameters
  - "exhaustivity" Report Parameters



A report based on "Strategic Model Directives" **Report Template** presents the directives (business policies and business rules) of a strategic model.

6

REPORT TEMPLATE: HOW TO

The following points are covered here:

- ✓ Report Template Creation: The Big Picture
- ✓ Managing Report Templates
- ✓ Customizing a Report Template Style
- ✓ Customizing a Table Report Template
- ✓ Customizing a Graph Report Template
- ✓ Customizing a Gauge Report Template
- ✓ Customizing a Tree Report Template

Report Template Creation: The Big Picture

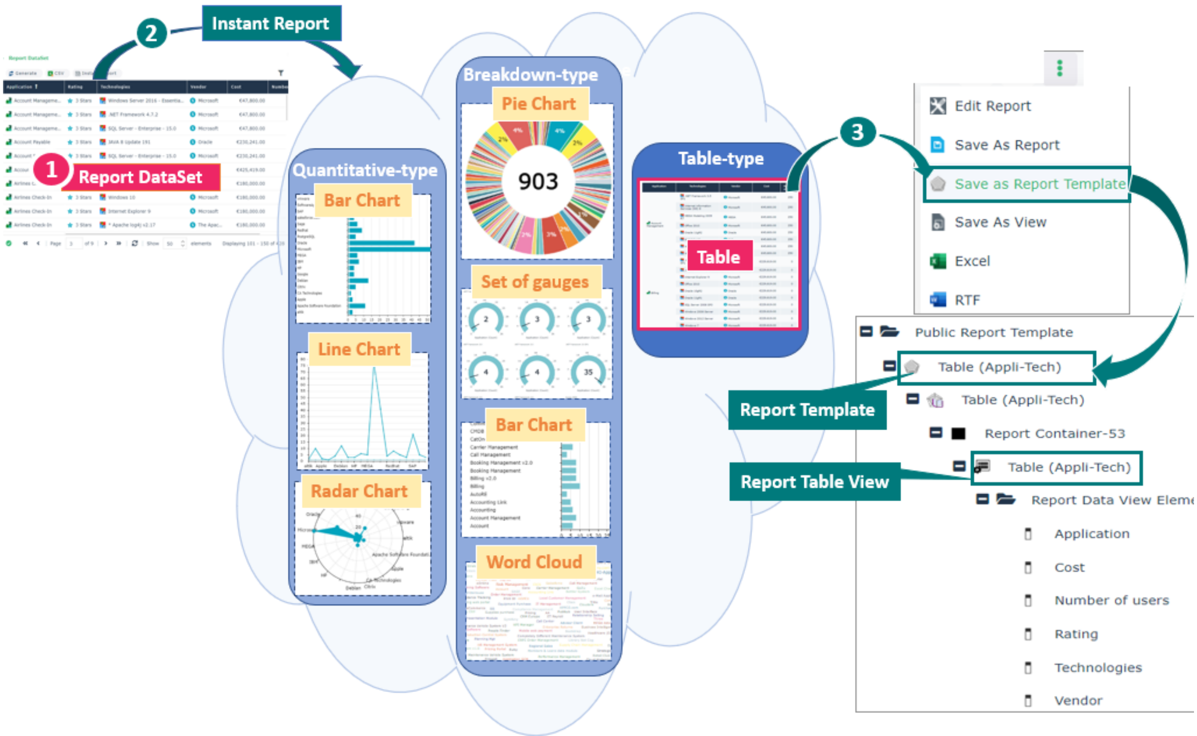
See:

- Displaying a Table, a Radar/Line/Bar/Pie Chart, a Set of Gauges, or a Word Cloud
- Displaying a Matrix-Bar Chart or a Matrix-Radar Chart
- Displaying a Graph
- Displaying a Gauge
- Displaying a Breakdown, a Dendrogram, a TreeMap, or a TreeTable

Displaying a Table, a Radar/Line/Bar/Pie Chart, a Set of Gauges, or a Word Cloud

You can create a **Report Template** showing a ***table***, a ***radar*** chart, a ***line*** chart, a ***bar*** chart, a ***pie*** chart, a ***set of gauges*** or a ***word cloud***:

- from a **Report DataSet**, or
- using a **Report Table View**



From a Report DataSet

To create a **Report Template** showing a ***table***, a ***radar*** chart, a ***line*** chart, a ***bar*** chart, a ***pie*** chart, or a ***word cloud***:

1. Access a **Report DataSet**.
2. From the Report DataSet, generate the **Instant Report** type corresponding to the chart type you want to display in the report:
   - **Breakdown** for a ***pie*** chart, a ***bar*** chart, a ***set of gauges***, or a ***word cloud***.
   - **Quantitative** for a ***bar*** chart, a ***line*** chart, or a ***radar*** chart.
   - **Table** for a ***table***.
3. In the **Report** section, click **More** **> Save As Report Template** .

   ☛ *Note: if you customize the report display, this customization is not taken into account in the Report Template.*
4. Enter a **Name** to the report and click **OK**.
   A public **Report Template** displaying the chart (***table***, ***radar*** chart, ***line*** chart, ***bar*** chart, ***pie*** chart, or ***word cloud***) is created, with its public **Report Table View**. They both have the same name.

   ☺ *You can access the **Report Template** in your **Home** page, from the **Custom Report Templates** tile. It is also stored in the **Public Report Template** folder.*

   ☛ *Note that **Save as Report**, generates a public report and also a private **Report Template**, which you can access in **Private Report Template > My Report Templates** folder.*

From a Report Table View

You can customize an instant report generated from a **Report DataSet** as your needs, and save it as a **Report Table View** so as to reuse it in a **Report Template**.

Once saved, you can still modify the Report Table View as needed.

To create a **Report Template** showing a ***table***, a ***radar*** chart, a ***line*** chart, a ***bar*** chart, a ***pie*** chart, or a ***word cloud***:

1. Access a **Report DataSet**.
2. From the Report DataSet, generate the **Instant Report** type corresponding to the chart type you want to display in the report:
   - **Breakdown** for a ***pie*** chart, a ***bar*** chart, a ***set of gauges***, or a ***word cloud***.
   - **Quantitative** for a ***bar*** chart, a ***line*** chart, or a ***radar*** chart.
   - **Table** for a ***table***.
3. In the **Report** section,, click **More** **> Save As View** .

   ☛ *Note: if you customize the report display, this customization is not taken into account in the Report Table View.*

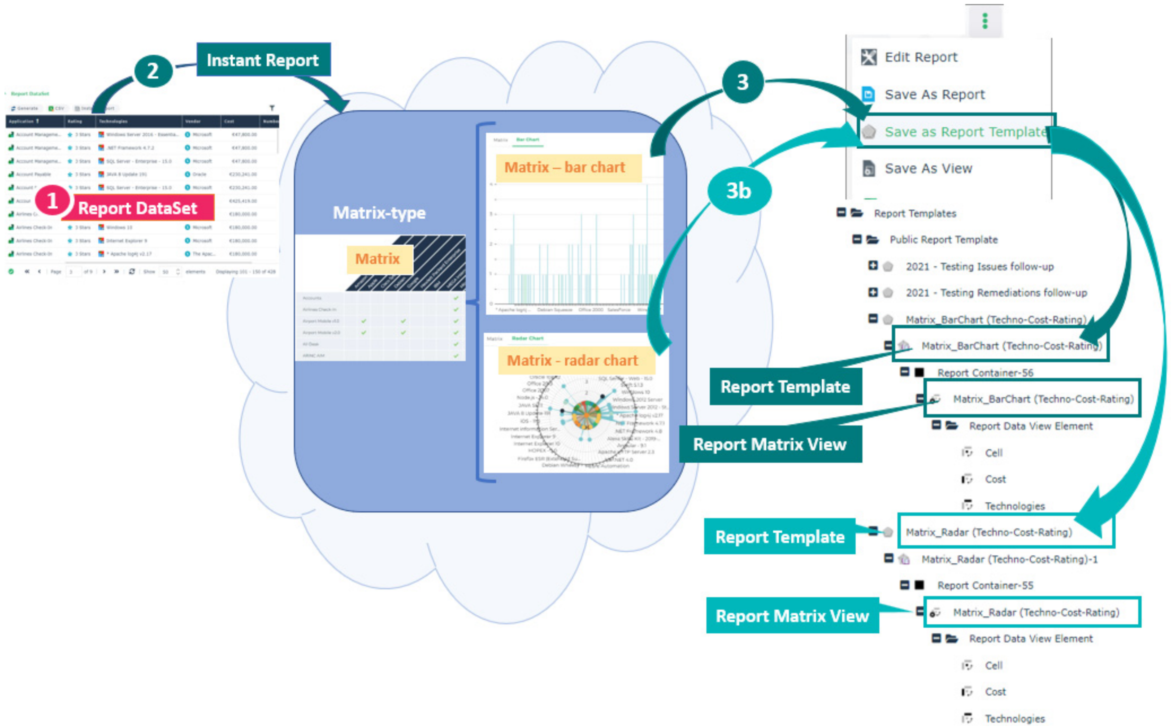
   A **Report Table View** is created.
4. You can:
   - Create a **Report Template** using the **Report Table View**.
   - Add the **Report Table View** to an existing **Report Template**.

Displaying a Matrix-Bar Chart or a Matrix-Radar Chart

A Report Template can show a matrix associated with its corresponding bar chart or radar chart.

You can create a **Report Template** showing a ***matrix-bar chart*** or a ***matrix-radar chart***:

- from a **Report DataSet**
- using a **Report Matrix View**



From a Report DataSet

To create a **Report Template** showing a ***matrix-bar chart*** or a ***matrix-radar chart***:

1. Access a **Report DataSet**.
2. From the Report DataSet, generate a **Matrix** type **Instant Report** .
3. Select the chart type: **Bar chart** or **Radar chart**.
4. In the **Report** section, click **More** **> Save As Report Template** .

☛ *Note: if you customize the report display, this customization is not taken into account in the Report Template.*

5. Enter a **Name** and click **OK**.
   A public **Report Template** displaying two tabs (the ***matrix*** and its associated ***bar/radar*** chart) is created, with its public **Report Matrix View**. They both have the same name.

   ☺ *You can access the **Report Template** in your **Home** page, from the **Custom Report Templates** tile. It is also stored in the **Public Report Template** folder.*

   ☛ *Note that **Save as Report**, generates a public report and also a private **Report Template**, which you can access in **Private Report Template > My Report Templates** folder.*

From a Report Matrix View

You can customize an instant report generated from a **Report DataSet** as your needs, and save it as a **Report Matrix View** so as to reuse it in a **Report Template**.

Once saved, you can still modify the Report Matrix View as needed.

To create a **Report Template** showing a ***matrix-bar chart*** or a ***matrix-radar chart***:

1. Access a **Report DataSet**.
2. From the Report DataSet, generate a **Matrix** type **Instant Report** .
3. Select the chart type: **Bar chart** or **Radar chart**.
4. In the **Report** section, click **More** **> Save As View** .

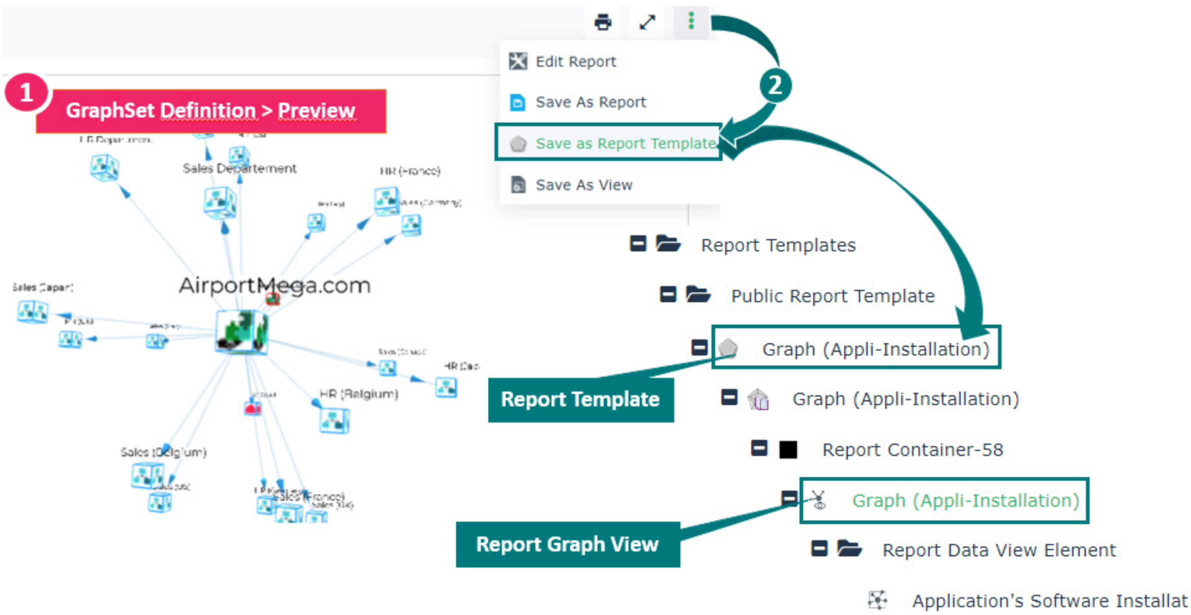
   ☛ *Note: if you customize the report display, this customization is not taken into account in the Report Matrix View.*

   A **Report Matrix View** is created.
5. You can:
   - Create a **Report Template** using the **Report Matrix View**.
   - Add the **Report Matrix View** to an existing **Report Template**.

Displaying a Graph

You can create a **Report Template** showing a ***graph***:

- from a **GraphSet Definition**, or
- using a **Report Graph View**



From a GraphSet Definition

To create a **Report Template** showing a ***graph***:

1. Access a **GraphSet Definition**.
2. From the **GraphSet Definition** properties, **Preview** page, click **More** **> Save As Report Template** .

   ☛ *Note: if you customize the report display, this customization is not taken into account in the Report Template.*
3. Enter a **Name** and click **OK**.
   A public **Report Template** displaying the ***graph*** is created, with its public **Report Graph View**. They both have the same name.

From a Report Graph View

You can preview a ***graph*** from a **GraphSet Definition**, and save it as a **Report Matrix View** so as to reuse it in a **Report Template**.

To create a **Report Template** showing a ***graph***:

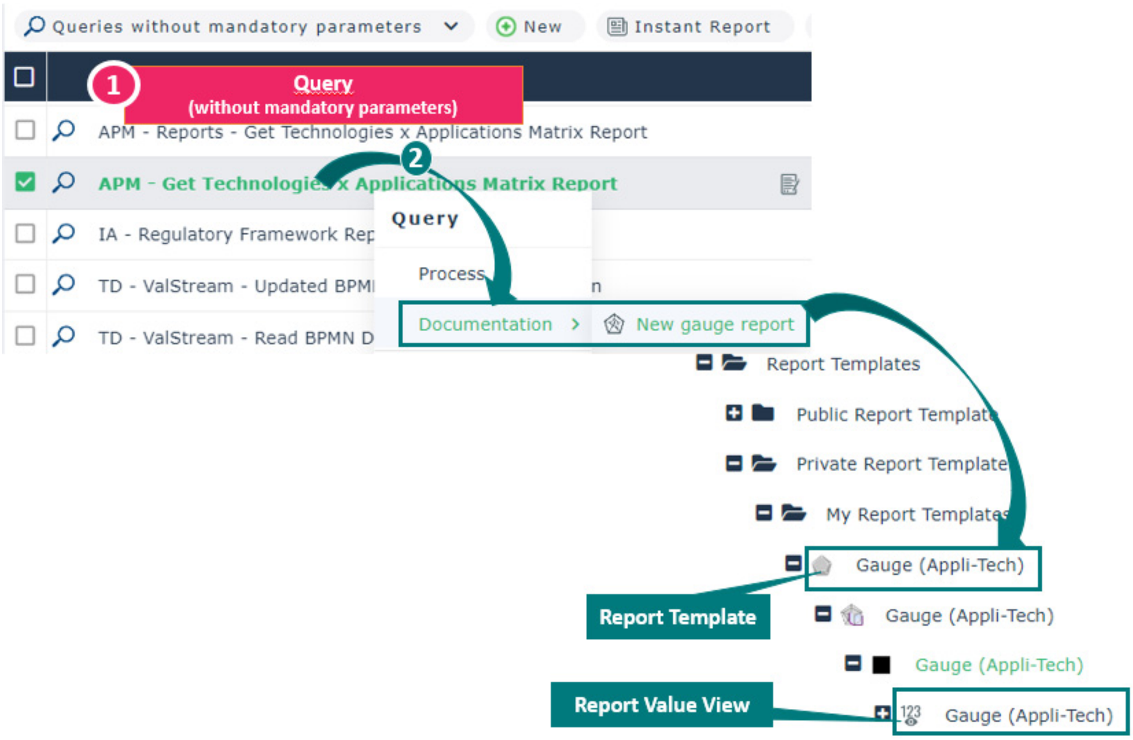
1. Access a **GraphSet Definition**.

2. From the **GraphSet Definition** properties, **Preview** page, click **More** ⋮ **> Save As View**.
   A **Report Graph View** is created.
3. You can:
   - Create a **Report Template** using the **Report Graph View**.
   - Add the **Report Graph View** to an existing **Report Template**.

Displaying a Gauge

You can create a **Report Template** showing a ***gauge***:

- from a **Query**, or
- using a **Report Table View**



From a Query

To create a **Report Template** showing a ***gauge***:

1. Access the **Queries without mandatory parameters** list.

2. Right-click a query and select **Documentation > New gauge report**.
   A public **report** displaying the ***gauge*** is created.
   A private **Report Template** is automatically created, with its private **Report Value View**.

   ☺ *You can access the **Report Template** in your **Home** page, from the **Custom Report Templates** tile. It is also stored in the **Private Report Template** folder.*

From a Report Value View

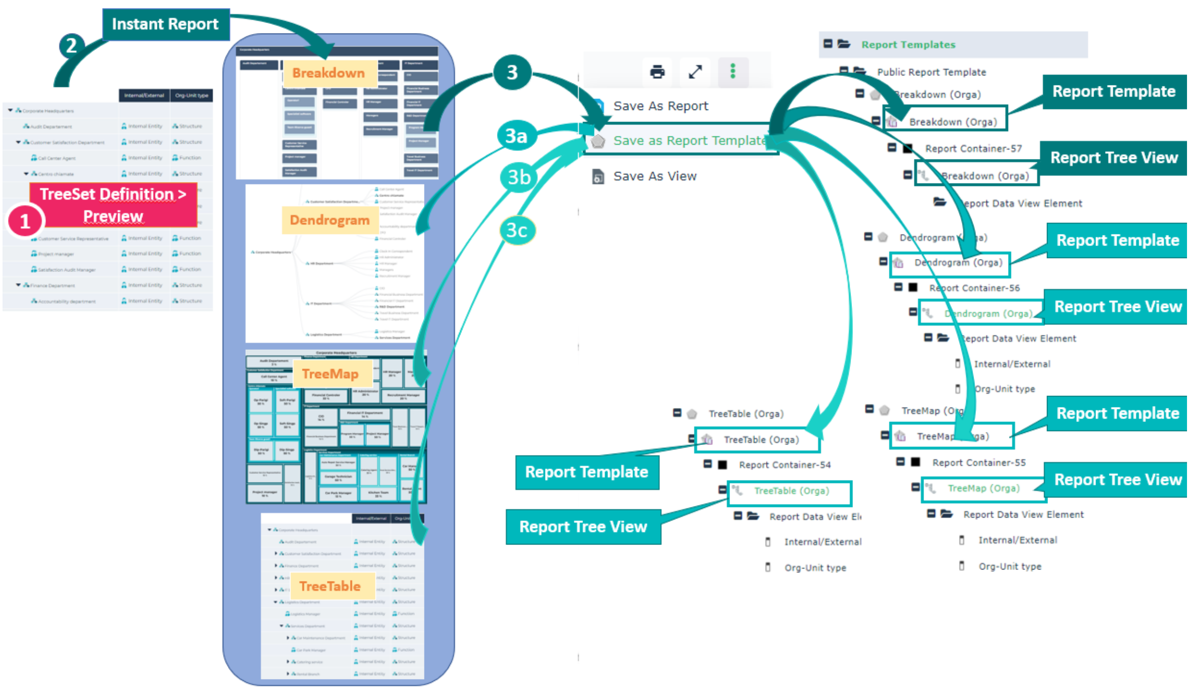
To create a **Report Template** showing a ***gauge***:

1. Access the **Queries without mandatory parameters** list.
2. Right-click a query and select **Documentation > New gauge report**.
   A **Report Template** is automatically created, with its **Report Value View**.
3. You can:
   - Create a **Report Template** using the **Report Value View**.
   - Add the **Report Value View** to an existing **Report Template**.

Displaying a Breakdown, a Dendrogram, a TreeMap, or a TreeTable

You can create a **Report Template** showing a ***Breakdown***, a ***Dendrogram***, a ***TreeMap*** or a ***TreeTable***:

- from a **TreeSet Definition**, or
- using a **Report Tree View**



From a TreeSet Definition

To create a **Report Template** showing a ***Breakdown***, a ***Dendrogram***, a ***TreeMap***, or a ***TreeTable***:

1. Access a **TreeSet Definition**.
2. From the **TreeSet Definition** properties, **Preview** page, click **Instant Report** and select the rendering type:

   E.g.: ***Breakdown, Dendrogram***, ***TreeMap***, or ***TreeTable***.

3. In the **Instant Report**, click **More** **> Save As Report Template** .

   ☛ *Note: if you customize the report display, this customization is not taken into account in the Report Template.*

   A public **Report Template** displaying the ***Breakdown*** map, ***Dendrogram***, ***TreeMap***, or ***TreeTable*** is created, with its public **Report Tree View**. They both have the same name.

   ☺ *You can access the **Report Template** in your **Home** page, from the **Custom Report Templates** tile. It is also stored in the **Public Report Template** folder.*

   ☛ *Note that **Save as Report**, generates a public report and also a private **Report Template**, which you can access in **Private Report Template > My Report Templates** folder.*

From a Report Tree View

To create a **Report Template** showing a ***Breakdown***, a ***Dendrogram***, a ***TreeMap***, or a ***TreeTable***:

1. Access a **TreeSet Definition**.
2. From the **TreeSet Definition** properties, **Preview** page, click **Instant Report** and select the rendering type:

   E.g.: ***Breakdown, Dendrogram***, ***TreeMap***, or ***TreeTable***.

3. Click **More** > **Save As View** .
   A **Report Tree View** is created.
4. You can:
   - Create a **Report Template** using the **Report Tree View**.
   - Add the **Report Tree View** to an existing **Report Template**.

Managing Report Templates

To create a **Report Template**, see:

- Report Template Creation: The Big Picture
- Creating a Report Template

To create a Report Data View:

- **Report Table View**, see:
  - Displaying a Table, a Radar/Line/Bar/Pie Chart, a Set of Gauges, or a Word Cloud
  - Creating a Report Data View from the Report DataSet
- **Report Matrix View**, see:
  - Displaying a Matrix-Bar Chart or a Matrix-Radar Chart
  - Creating a Report Data View from the Report DataSet
- **Report Graph View**, see:
  - Displaying a Graph
  - Creating a Report Graph View
- **Report Tree View**, see:
  - Displaying a Breakdown, a Dendrogram, a TreeMap, or a TreeTable
  - Creating a Report Tree View
- **Report Value View**, see:
  - Displaying a Gauge
  - Creating a gauge-type Report Template from a query

See:

- How to Define Parameters in a Report Template
- How to Group Report Data Views in a Report Chapter
- How to Modify a Report Data View
- How to Define the Filters Displayed in a Report
- How to Duplicate a Report Template
- How to Duplicate a Report Data View
- How to Add Reports to an Object Property Pages
- How to Make a Report Creation Available to an Object List
- How to Customize a Report Icon
- How to Manage Availability of Reports
- How to Make a Report Template Available at Report Creation
- How to Define the Report Template Renderer and Illustration
- How to Define the Categories and Subjects Filters
- How to Find Information Regarding JAVA Macros

See also

- Managing a Report Snapshot
- Organizing the Report Chapters
- Customizing the Report Chapter Export Format
- Customizing Parameter Display
- Customizing the Report Container Group display

Getting the Parameters available to Customize a Renderer Type

To help you with the parameters available to customize a report according to its renderer type, you can use the **Report Style Parameters** report template.

This report details for each renderer type all its available parameters, and where they are used.

☛ *See Report Renderer.*

To create the report:

1. From the navigation menu, select **Report Studio > Reports** and click **Create a report**.
2. In the **Objects** filter, select "RedererType".

   ☺ *Enter "renderer" in the search field.*
3. Hover the cursor over the **Report Style Parameters** tile and click **Create a report**.
4. Use the filters according to your needs:
   - **RendererType**: select the type(s) of rendering

     `E.g.: TreeTable and Table, Matrix.`
   - **Apply to**: select the item(s) you want to customize
   - **Parameter Type**: select the type of parameter you want to apply

     `E.g.: Background Color.`
   - **Report Parameter Group**: select the group of parameter you want to apply

   The report displays the parameters you can use to customize you report according to your selection.

RendererType 2 ×   Parameter Type : Background Color ×

Renderer	Parameter	Values	Apply to	Parameter Type	Comment	Report Parameter Group
Table, Matrix	Cell Background Color from Object	No	Row, Column, Cell	Background Color		Cell Style
		Yes	Row, Column, Cell	Background Color		Cell Style
	Cells Background Color		Row, Column, Cell	Background Color		Cell Style
	Column Header BackGround Color		Column Headers	Background Color		Header Style
	Row Header Background Color		Row Headers	Background Color		Header Style
	Table Background Color		Table	Background Color		Style
Tree Table	Cell Background Color from Object	No	Row, Column, Cell	Background Color		Cell Style
		Yes	Row, Column, Cell	Background Color		Cell Style
	Cells Background Color		Row, Column, Cell	Background Color		Cell Style
	Column Header BackGround Color		Column Headers	Background Color		Header Style
	Row Header Background Color		Row Headers	Background Color		Header Style
	Table Background Color		Table	Background Color		Style

How to Define Parameters in a Report Template

Report Template parameters enable to define required elements to build a report.

Defining parameters in a Report Template based on a macro

For each parameter you must specify:

- the object type that can be instantiated for the report:
  - **Parameter with MEGA objects**
  - **Parameter with simple types**: a parameter with simple types defines the types that can take values in the report (e.g.: boolean, date, string). **MEGA** provides, by default, a set of simple types, represented by tagged values.

    ☛ *Tagged values play the same role as MetaAttributes but do not belong to the MEGA metamodel.*
- its multiplicity.
  The Multiplicity of a report parameter enables definition of the number of values that can be associated with the parameter when creating a report. If multiplicity is 1 or 0..1, only one parameter value can be connected.

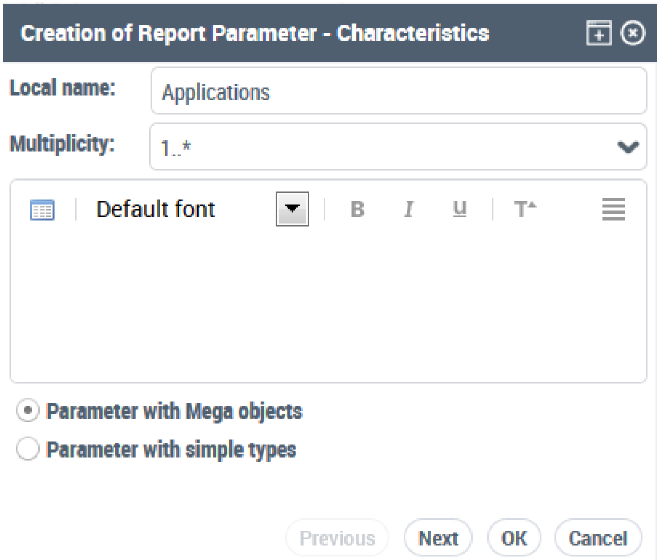
To create a parameter in a Report Template based on a macro:

1. Access the Report Template (based on a macro).

   ☛ *See Accessing the Report Templates and their Constituents.*
2. In the Report Template properties, select **Parameters**.
3. In the **Parameters** section, click **New** ✚ and for each parameter define its required **Characteristics**.
4. In **Local name** field, enter the parameter name.
5. In **Multiplicity** field, select the parameter multiplicity.
6. (Optional) In the **Comment** pane enter a description.

7. Select the parameter type:
   - **Parameter with MEGA objects** enables connection of MetaClasses that can define the parameter.

     `For example, for a parameter named "Application Architecture", you can connect an application folder, an application, a query returning applications.`



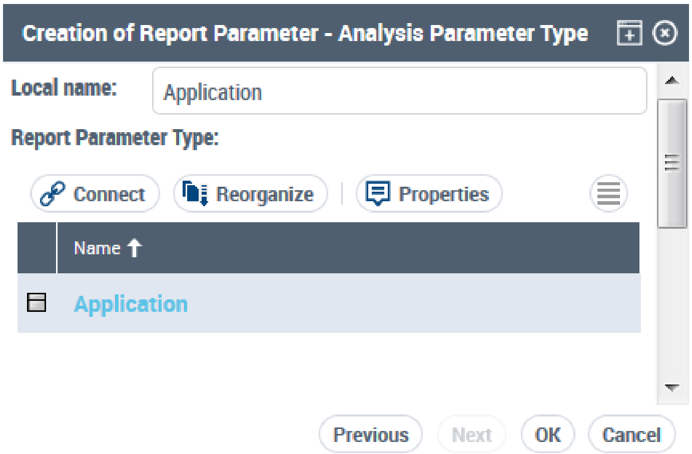
   - **Parameter with simple types** enables connection of "Tagged Values".

   Parameter types correspond to report values. Assigned MetaClasses inherit the "System report value" abstract MetaClass.

   ☛ *For details on abstract MetaClasses, see Abstract Metamodel.*
8. Click **Next**.

9. If you selected:
   - **Parameter with MEGA objects**, in the **Report Parameter Type** pane, click **Connect** and select the report parameter type(s).

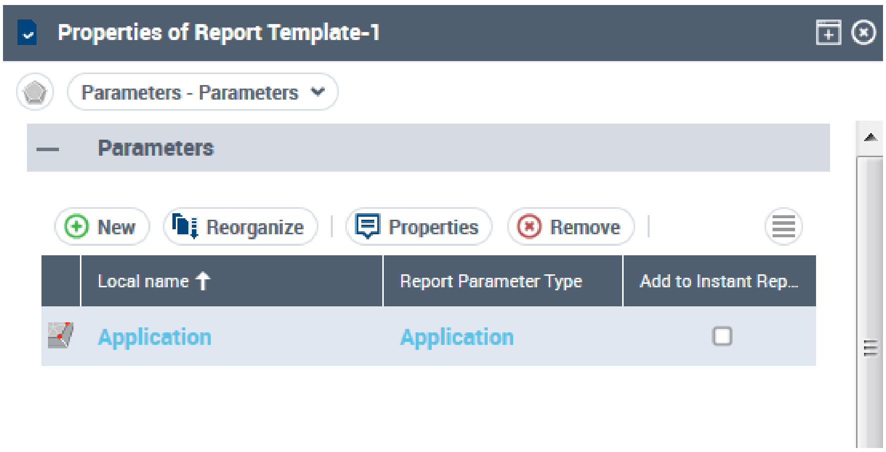
     `Example: Application.`



   - **Parameter with simple types**, in the **Tagged Value** pane, click **Connect** and connect the Tagged Value.

     `Example: to create a parameter that filters the display of certain objects in the report connect "Boolean Report Value" Tagged Value.`

10. Click **OK**.
    The input data type the user must enter for the report calculation is defined.



11. If needed:
    - configure the parameter to restrict the values the end-user can use

      ☛ *See Customizing Parameter Display.*
    - customize the parameter display

      ☛ *See Customizing Parameter Display.*

Configuring a Report Template parameter

If needed, you can define the values the end-user can use at report creation.

Values can be a closed or open list of values, which is defined by the **Value definition** field:

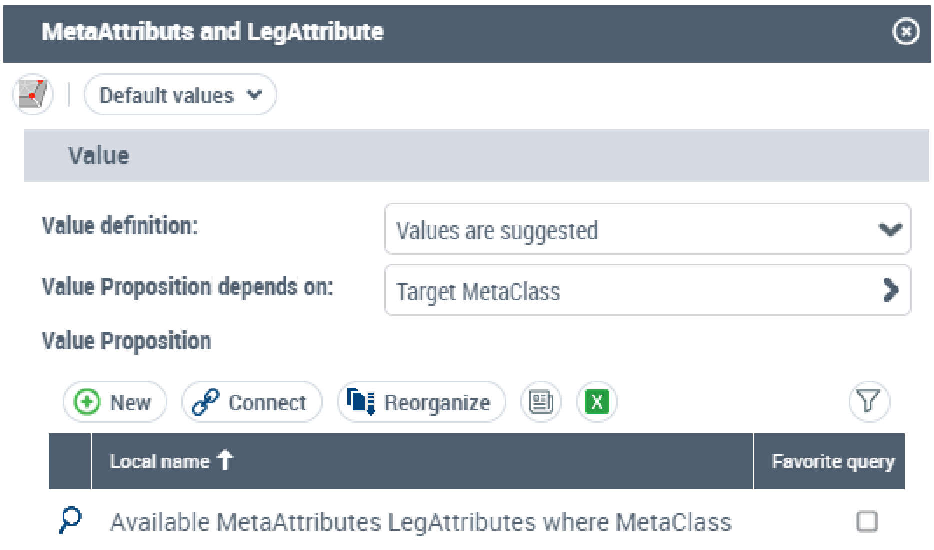
- ***Values are suggested*** (by default)
  The end-user can select values that are in the list, but can add other values if required.
- ***Values are proposed***
  The end-user can only select values that are included in the list.
- ***Values are fixed***
  The end-user cannot select or add any other values.
- ***Values are fixed and hidden***
  The end-user does not see the values.

Value proposition defines how the list of values is computed (through a macro or a query). This list of values can be computed from another parameter value, which is defined by the **Value Proposition depends on** field.

For example:

The parameters of the **List Of Objects** Report Template are "Target MetaClass" and "MetaAttributes and LegAttribute". This Report Template displays an object list and selected properties (attributes) of the analysis parameters, in a table format.

The list of attributes proposed depends on the MetaClass selected by the end-user.



To configure a Report Template parameter:

1. Access the Report Template.

   E.g.: The **List Of Objects** Report Template, whose parameters are "Target MetaClass" and "MetaAttributes and LegAttribute".

2. In the Report Template parameter properties, display its **Default values** page.

   E.g.: The "MetaAttributes and LegAttribute" parameter.
3. (If needed) Modify the **Value definition** field.

   E.g.: Values are suggested (default), Values are proposed, Values are fixed, Values are fixed and hidden.
4. (If the parameter value depends on another parameter) In the **Value proposition depends on** field, select **Connect Report Parameter** and connect the Report Template parameter on which the values depend on.

   E.g.: "Target MetaClass".
5. In the **Value Proposition**, click connect and connect the required query or macro.

   ☛ *You can connect several queries, in that case select "Favorite query" in the row of the query you want to be performed first.*

   For example: the query "Available MetaAttributes LegAttributes where MetaClass"
6. (If you want to display parameters ordered by group) In the **Displayed in Group** field, select the display.

   Values are: Parameters, Heatmap, Presentation.

Defining parameters in a Report Template based on Data Views

If the Report Template includes more than one Data View and these Data Views are based on:

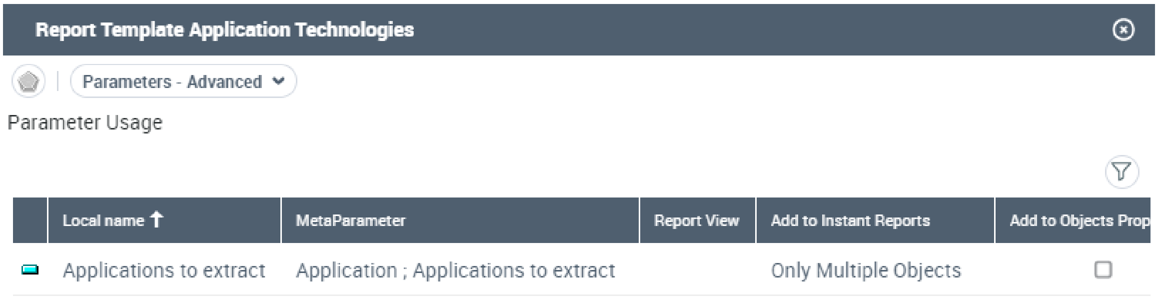
- the same Report DataSet Definition:
  By default a single Report DataSet instance is created for the report and associated parameters are valued at report creation.
- distinct Report DataSet Definitions:
  By default as many Report DataSet instances as Report DataSet Definitions are created for the report and associated parameters are valued at report creation.

  In case parameters are of the same type, you can use a **Parameter Usage** so that the user is asked to define the parameters only once.

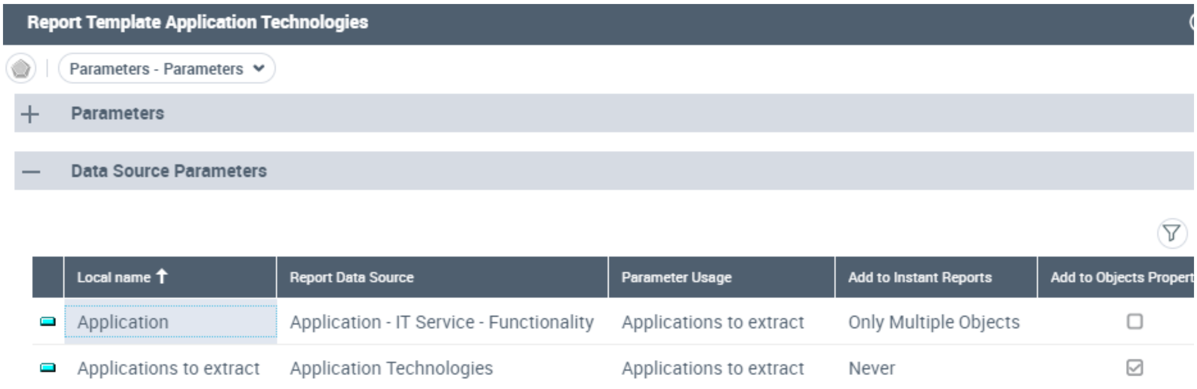
  Example: the "Application Technologies" Report Template, based on "Application - IT Service - Functionality" and "Application Technologies"

```
Report DataSet Definitions, uses the "Application to extract" Parameter
Usage.
```

In the Report Template properties > **Parameters - Advanced** page:



In the Report Template properties > **Parameters - Parameters** page:



How to Group Report Data Views in a Report Chapter

By default the Report Containers are displayed in a single column.

See Report Chapter and Report Data Views

To modify the display of the Report Containers (e.g.: on a same row, in separate tabs) you need to create a Report Container Group so as to group Report Data Views in the Report Chapter.

Grouping Report Data Views of a Report Chapter

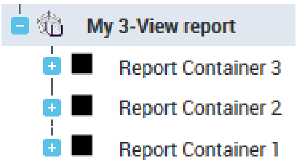
Prerequisite: the Report Data Views of the Report Chapter are created.

To create a Report Data View, see Creating a Report Data View.

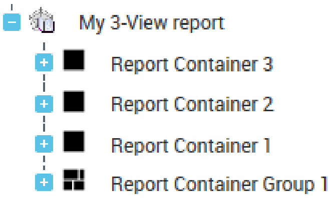
To group Report Data Views of a Report Chapter:

1. Access the Report Chapter of the Report Template (based on Data Views) concerned.

   ☛ *See* *Accessing the Report Templates and their Constituents.*

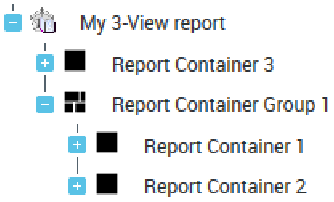


2. Right-click the Report Chapter and select **Add > Report Container Group**.
3. In the **Creation of Report Container Group**, enter its **Local name**.
4. Click **OK**.
   The Report Container Group is added in the report Chapter tree.

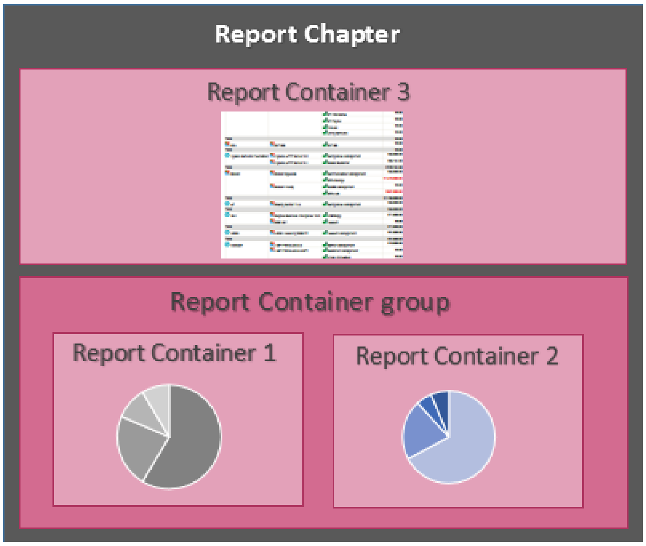


5. Drag and drop, in the Report Container group, each of the Report Containers you want to group.

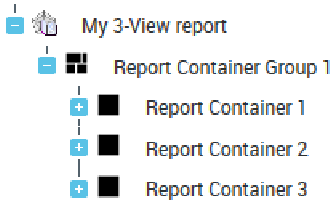
`Example 1: drag and drop two of the three Report Containers included in the Report Chapter.`



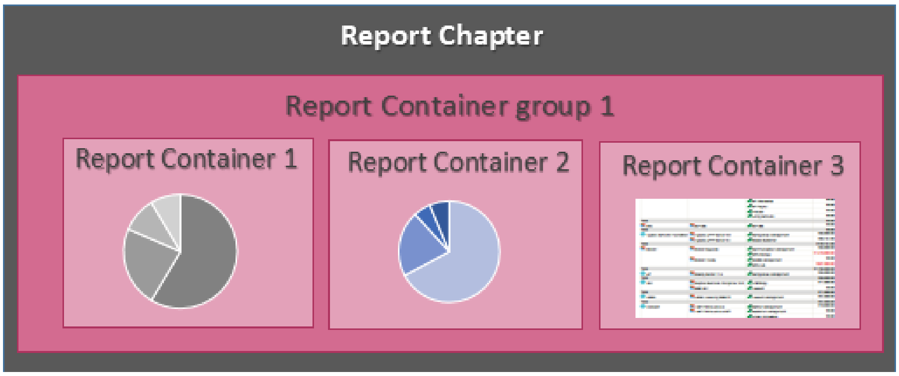
`This Report Chapter display looks like:`



```
Example 2: drag and drop all of the Report Containers included in the
Report Chapter.
```




```
This Report Chapter display looks like:
```



6. To modify the default Report Container Group display, see Customizing the Report Container Group display.

Creating a Report Chapter with grouped Report Data Views

Prerequisite: the Report Data Views you want to group in the Report Chapter are created.

☛ *To create a Report Data View, see Creating a Report Data View.*

To create a Report Chapter with grouped Report Data Views:

1. Access the Report Template.

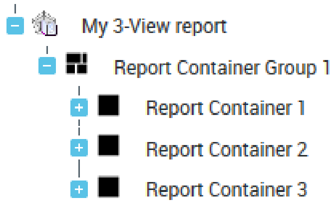
   ☛ *See Accessing the Report Templates and their Constituents.*
2. Add a Report Chapter based on Data Views to the Report Template concerned.

   ☛ *See Adding a Report Chapter to a Report Template.*
3. Right-click the Report Chapter and select **Add > Report Container Group**.
4. In the **Creation of Report Container Group** window, enter its **Local name**.
5. Click **OK**.
   The Report Container Group is added in the report Chapter tree.
6. Right-Click the Report Container Group and select **Add > Report Container**.
7. In the **Creation of Report Container** window, enter its **Local name**.
8. In the **Report Reusable Content** field, click the arrow and select **Connect**.
   The **Find object** window appears.
9. In the first drop-down menu, select the Report Data View type.

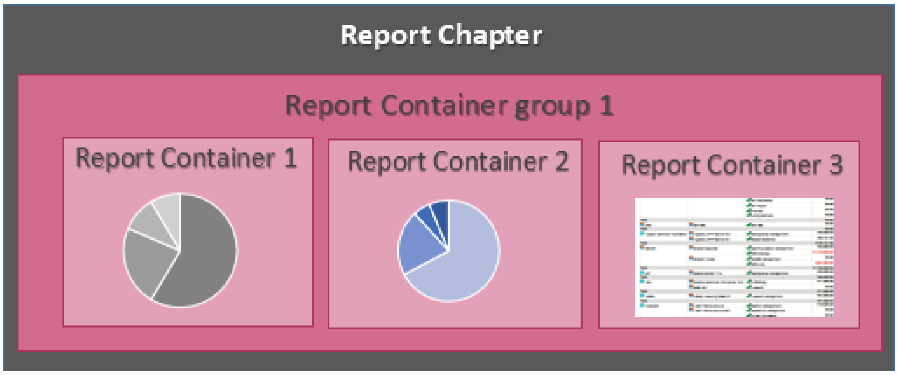
   ☛ *See Report Data View types.*

10. In the search result, select the Report Data View and click **OK**.
11. Click **Connect**.
    The Report Container is defined.
12. Click **OK**.
13. Repeat the Report Container creation steps to add the other Report Data Views.

    ```
    Example 2: drag and drop all of the Report Containers included in the
    Report Chapter.
    ```




    ```
    This Report Chapter display looks like:
    ```



14. To modify the default Report Container Group display, see Customizing the Report Container Group display.

How to Modify a Report Data View

To modify a Report Data View:

1. Access the **Report Data View** properties
   - *See Accessing the Report Templates and their Constituents.*
2. Modify the Report Data View properties according to your needs.
   - *See Report Data View Properties: Matrix and Table.*
   - *See Report Data View Properties: Graph.*
   - *See Report Data View Properties: Tree.*
   - *See Report Data View Properties: Value.*

3. For example depending on the Report Data View, in:
   - **Table**, you can modify the table column sort order and/or display order.
   - **Table**, you can delete a table column.
   - **Table**, you can add subtotals on breaks.
   - **Matrix**, you can modify the row or column sort order.
   - **Matrix**, you can modify the Cell value computation.
   - **Graph**, you can add/modify fields on nodes or arcs.
   - **Tree**, you can add/modify fields on nodes.
   - **Filters**, you can delete a filter.
   - **Characteristics**, you can modify its **Display**.

     For a **Report Table/Matrix View**: table, graphic, or table and graphic.

     For a **Report Tree View**: Breakdown, Dendrogram, TreeTable, TreeMap.
   - **Characteristics**, you can modify its **Sharing** (private or public, public by default).
   - **Characteristics**, you can define its **General Style**.

How to Define the Filters Displayed in a Report

You can define filters on Report Table Views, Report Matrix Views, and report Graph Views.

> ☛ *See Report Data View Properties: Matrix and Table and Report Data View Properties: Graph.*

To define the filters displayed in a report:

1. Access the Report Data View properties for which you want to define filters.

   > ☛ *See Accessing the Report Templates and their Constituents.*
2. In the **Filter** page, click **Add filter**.
   The list of available Data Source Elements is displayed.

   For example, a column for a Report Table View or a Report Matrix View, a node or a field for a Report Graph View.
3. Select the Data Source Element for which you want to add a filter.
4. In the right pane define the filter characteristics:
   - **_GUIName**
   - **Input type**
   - **Behavior** as required.
   - **Filter Choice**
   - **Shows**

   > ☛ *See Characteristics.*
5. (Optional) In the filter properties, define the **Object Edition Mode**.
6. (optional) In the filter properties, select **Texts > _Settings** to define an initial value or a candidate value.

   > ☛ *See Text.*

How to Duplicate a Report Template

The recommended way to create a new Report Template similar to an existing one is to duplicate a Report Template.

To duplicate a Report Template:

1. Access the Report Template you want to duplicate.

   ☛ *See Accessing the Report Templates and their Constituents.*

2. Right-click the Report Template and select **Manage > Duplicate**.
3. Define the Report Template name.
4. Select the duplication name format for your Report Template (Prefix or Suffix).
5. Click **OK**.

How to Duplicate a Report Data View

To duplicate a Report Data View:

1. Access the Report Data View you want to duplicate.

   ☛ *See Accessing the Report Templates and their Constituents.*

2. Right-click the Report Data View and select **Manage > Duplicate**.
3. Define the Report Data View name.
4. Select the duplication name format for your Report Data View (Prefix or Suffix).
5. Click **OK**.

How to Add Reports to an Object Property Pages

You can add object-based reports in the object property pages (**Reporting** page).

`Example: you can add the "Application costs", "List of Applications", "Scenario Comparison" reports in the property pages of all the Portfolios (e.g.: "CRM Transformation" portfolio).`

To add a report in an object property pages:

1. Access the Report Template (based on a macro) corresponding to the report you want to add to the object property pages.

   ☛ *See Accessing the Report Templates and their Constituents.*

   `Example: "Application costs" Report Template.`

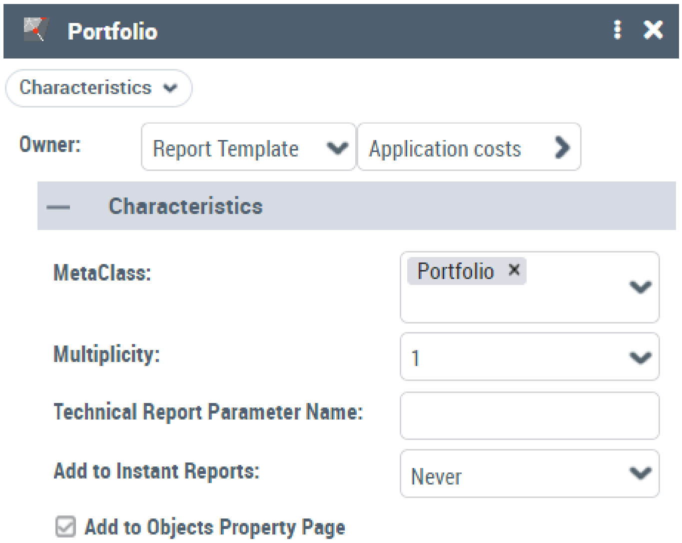
2. Access the Report Parameter properties.

   ☛ *See Accessing the Report Templates and their Constituents.*

   `Example: "Portfolio" Report Parameter.`

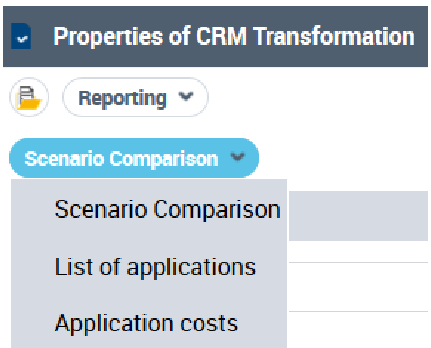
3. In the **Characteristics** page, select **Add to Objects Property Page**.
   A **Reporting** page including the corresponding report is added to all the object property pages.

   `Example: in the Property pages of all the Portfolios, the` **`Reporting`** `property page displays the "Application costs" report.`



4. You can add as many reports as available to the object **Reporting** property page.

   `Example: together with "Application costs" report you can add "List of Applications" and "Scenario Comparison" reports to the` **`Reporting`** `property page of all the portfolios.`



How to Make a Report Creation Available to an Object List

From an object list you can create Instant Reports. You can customize the list of Instant Reports available by adding specific reports, which are available on this object list.

```
Example: together with the Instant Reports available for a list of
Business Capabilities, Business Processes, or Organizational Processes,
you can add the "Execution and Performance HeatMaps" report.
```

To make a report creation available at Instant Report creation from an object list:

1. Access the Report Template (based on a macro) corresponding to the report you want to provide with the Instant Report list at Instant Report creation.

   ☛ *See Accessing the Report Templates and their Constituents.*

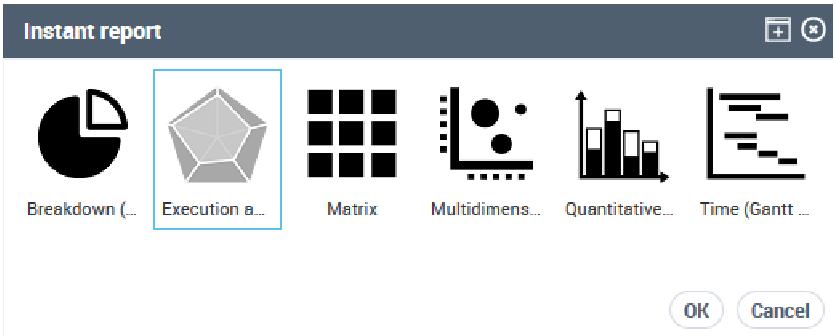
   `Example: "Execution and Performance Heatmaps" Report Template.`
2. Access the Report Parameter properties.

   ☛ *See Accessing the Report Templates and their Constituents.*

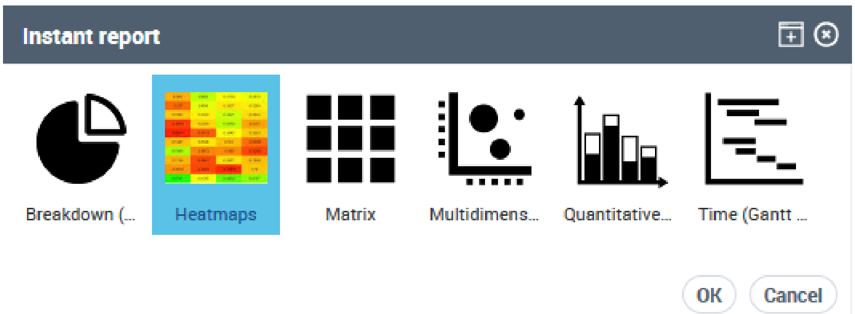
   ```
   Example: "Objects" Report Parameter, which is connected to the Business
   Capability, Business Process, and Organizational Process MetaClasses.
   ```

3. In the **Characteristics** page, in **Add to Instant Reports** field select "Always" (or "Only Multiple Objects" according to your needs).
The selection of the report creation corresponding to the Report Template concerned is available at Instant Report creation from the object list.
You can add as many reports as available on the object list.

```
Example: when you click Instant Report from a list of Organizational
Processes (or Business Capabilities or Business Processes) the "Execution
and Performance HeatMaps" report is also available.
```



You can customize the report icon: modify the report icon image and name.



☛ *See How to Customize a Report Icon.*

How to Customize a Report Icon

You can customize a report icon:

- modify the default report icon image
- modify the report icon name

To customize the report icon:

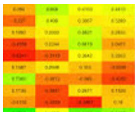
1. Access the Report Template properties.

☛ *See Accessing the Report Templates and their Constituents.*

2. In the **Characterictics** page:
   - click **Update Image** and select the image you want to be displayed.
   - in the **_GUIName** field enter the report icon name you want to be displayed.

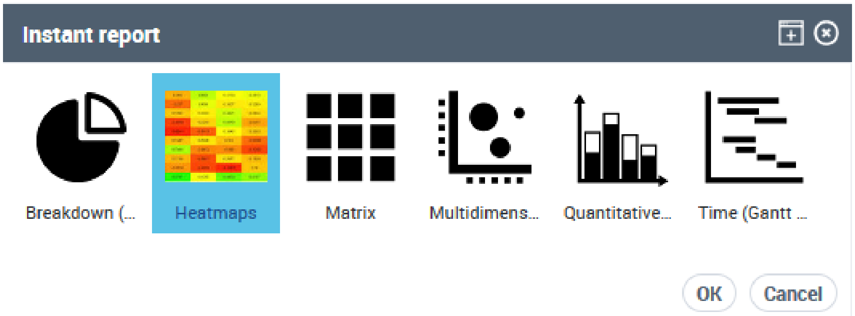
3. Click **OK**.

```
For example, you can customize the "Execution and Performance Heatmaps"
report added in the Instant Report list available as follows:
```

image: 

```
_GUIName: Heatmaps
```

☛ *See How to Make a Report Creation Available to an Object List.*



How to Manage Availability of Reports

You can use a Report parameter to:

- add reports to an object property pages (**Reporting** page).

  ☛ *See How to Add Reports to an Object Property Pages.*
- make the creation of reports available at Instant Report creation from an object list.

  ☛ *For an example, see How to Make a Report Creation Available to an Object List*

How to Make a Report Template Available at Report Creation

Only public Report Templates are available at report creation.

Saving an instant report as a Report Template, by default the Report Template created is public.

Saving an instant report as a report, by default the report created is public, but its Report Template automatically generated is private.

To make a Report Template available at report creation:

1. Access the Report Template properties.

   ☛ *See Accessing the Report Templates and their Constituents.*
2. In its **Characteristics** page, set its **Report Template Sharing** to "Yes".
   The Report Template is available at report creation.

How to Define the Report Template Renderer and Illustration

The renderer type and its illustration are automatically defined for HOPEX Studio reports, but have to be defined manually for Java reports.

Also if your Report Template includes several renderers (for example with several chapters), you might want to change its multi-renderer illustration for one of the renderers.

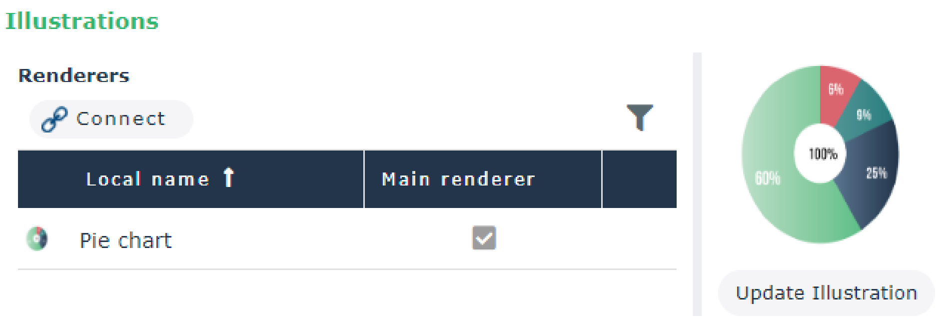
If can also update the illustration for another one.

To define the Report Template renderer and illustration:

1. Access the Report Template properties.

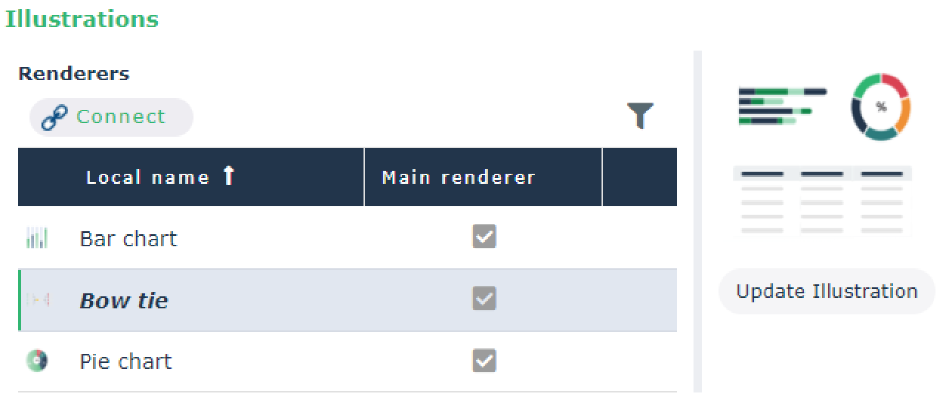
   ☛ *See Accessing the Report Templates and their Constituents.*

2. In the **Characterictics** page, **Illustration** section:
3. In the **Renderer** table, click **Connect** and select the renderer displayed in the report.
   The illustration is also automatically added.
   By default the renderer is selected as Main renderer.

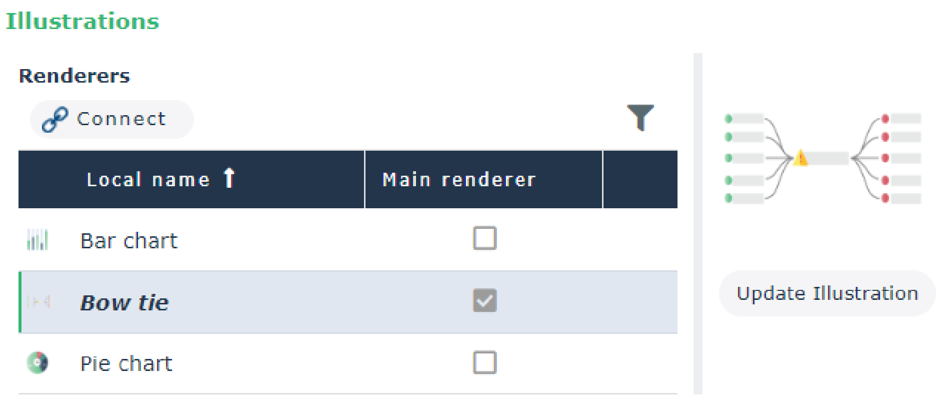


☛ *You can connect several renderers.*

4. (If several renderers) By default each render is selected as Main renderer, and the illustration is a multi-renderer one.



If you want to favor one renderer, clear the other Main renderers.



5. To change an **Illustration**, click **Update illustration** and select the MetaPicture.

```
The default Metapicture name format is:
-rt_<renderer type>
```

How to Define the Categories and Subjects Filters

The **Subjects** and **Categories** filters available to the end-user to filter the Reports and Report Templates (in the report access and creation pages) are defined in the Report Template properties:

☛ *See PLATFORM - Common Features > Documentation > Generating Documentation > Generating Reports documentation.*

☛ *See Report Template properties: Characteristics.*

- The **Categories** field has to be filled in manually.
- The **Subjects** field is automatically defined for HOPEX Studio reports, but has to be filled in manually for Java reports.

To define the **Categories** and **Subjects** fields:

1. Access the Report Template properties.

   ☛ *See Accessing the Report Templates and their Constituents.*

2. In the **Characterictics** page, add values to the corresponding fields:
   - **Categories** are domains considered in the report
   - **Subjects** are MetaClasses that are displayed in the report.

How to Find Information Regarding JAVA Macros

You can create macros either in VB Script or Java format.

To create a Java macro, see the following **HOPEX Studio** Technical Articles:

- Writing Java Report Chapters
- All about starting with APIs
- API - JavaDoc

☛ *The Java API help is also supplied with HOPEX in JavaDoc format.*

To read the Java API help:

1. In **<HOPEX Installation>\java**, expand **doc** file.
2. Unzip "mj-api.doc.zip" and "mj_anls.doc.zip in the shared repository.
3. Open "index.html" page.
4. Click **Ok**.

CUSTOMIZING A REPORT TEMPLATE STYLE

See:

- How to Customize a Report Template Style
- How to Create a Report Style
- How to Apply a General Style
- How to Customize a Report Color Palette

How to Customize a Report Template Style

You can customize a Report Template style using a report used as a style template. This customization is available for reports based on macros as well as for reports based on Report Data View(s).

Prerequisite: create a test report and customize its style. This test report will be associated with the Report Template so as to define the style parameters that are applied to all the reports generated from this Report Template.

☛ *To customize the test report, see Common Features: Customizing your Reports (Web Front-End) documentation.*

To customize a Report Template style:

1. Access the Report Template properties.

   ☛ *See Accessing the Report Templates and their Constituents.*

2. Select **Report Template Edition**.
3. In **Report** field click the arrow and select **Connect**.
4. Select the test report you created.
5. Click **OK**.

   ☛ *To cancel the customization, in **Report** field, click **Reinitialize** to delete the test report association with the* Report Template*.*

How to Create a Report Style

You might need, for example, to create a general Report Style to apply it at Report View level. This **general style** applies to a single Report Renderer type.

☛ *See Report Renderer.*

If you need to create a specific style or a **conditional style** to apply it at Report Data View element level (Report table/Matrix Column level or Report Graph Node/Arc level), see How to Define the Filters Displayed in a Report or How to apply a Conditional Style to a Graph dedicated sections.

To create a report style to be applied at Report View level::

1. Access the **Report Styles** folder.

   ☛ *See Accessing the Report Templates and their Constituents.*

2. Right-click the **Public Report Styles** folder and select **New > Report Style**.

   ☛ *A **general style** must be "public". You can create a "private" style, but to apply it as a general style, you need to change its sharing to "public".*

3. In the **Name** field enter a name that describes the style, so that it can be easily reused.

   `E.g.: Graph_ArcColor_LabelColor_NodeFillColor to create a general style, applying to a graph renderer, and defining arc, label, and node fill colors.`

4. Click **Connect**.
5. In the first drop-down menu select the **Renderer Type Parameter**.

   `E.g.: "Renderer GraphChart" Parameters if you want to create a general style that applies to a graph renderer.`

   The list of parameters is filtered according to the renderer type selected.

   `E.g.: graph parameters.`

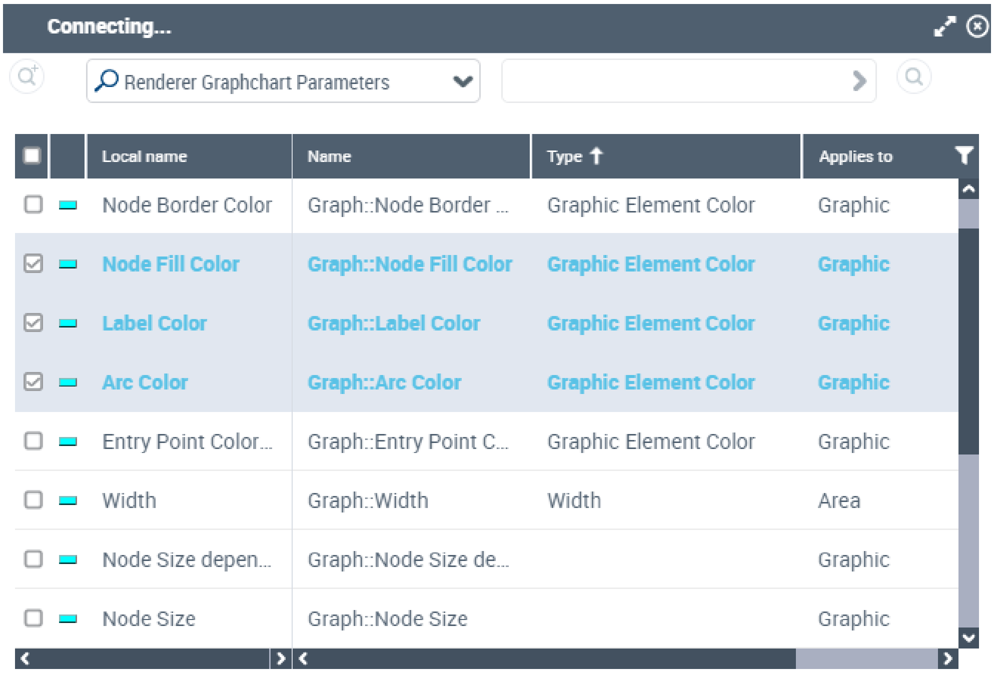
6. (If needed) Click the **Type** or **Applies to** column header to sort the parameters.

   E.g.: click the **Type** column header to get the set of "Graphic Element Color" type parameters.

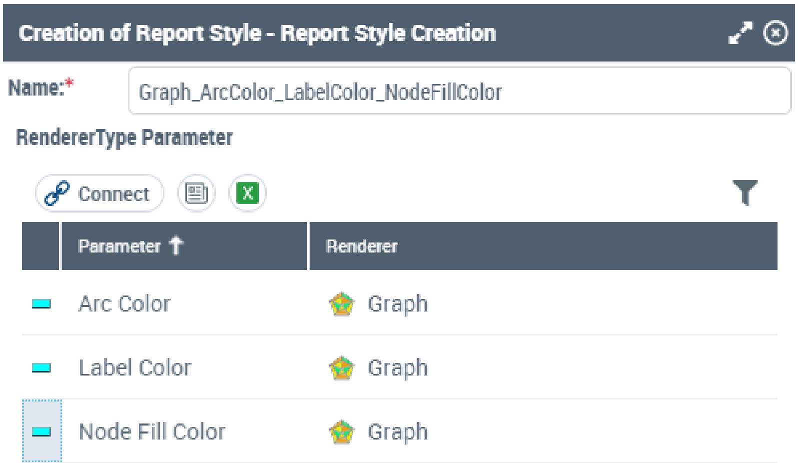
7. Select the parameter you want to add to the style.

   ☛ *You can select several parameters.*

   E.g.: select **Node Fill Color**, **Label Color** and **Arc Color** style parameters.

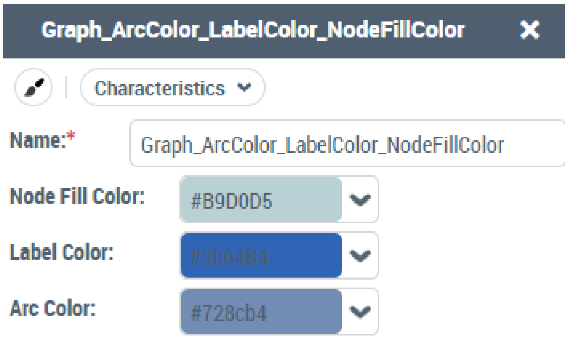


8. Click **Connect**.
   The selected parameters are connected to the style.



9. Click **OK**.
   The style is available in the **Public Report Styles** folder.
   Any user can add it as a general style to a Report Data View of the same renderer type.
10. Access the Report Style properties to configure its parameters.
11. In its **Characteristics** page define the parameters.

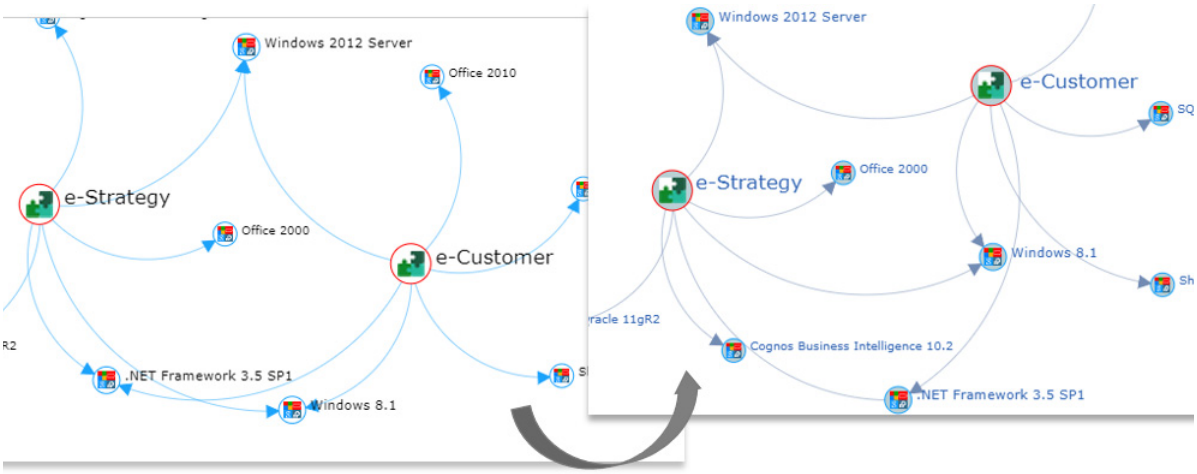
```
E.g.: define a color for the nodes, the labels, and the arcs.
```



12. You can apply the style at a Report Data View level.

☛ *See How to Apply a General Style section.*

For example, when you apply this style to a Report Graph View, the changes in the graph display are as follows.



If you want to apply a specific label color according to the node type, you need to apply a style at Report Graph View Node level, see How to apply a Conditional Style to a Graph section.

How to Apply a General Style

You can apply a **general style** at Report Data View level for ***Matrix***, ***Table***, ***Graph*** and ***Tree*** type Views.

☛ *For details on Properties of these Report Data Views, see Report Data View Properties: Matrix and Table, Report Data View Properties: Graph, Report Data View Properties: Tree.*

Particularly for a graph renderer, you can apply a **general style** at Report Graph View level. In that case the style applies to all of the graph layers of the View.

☛ *If you want to add a more specific style that you can apply (on condition or not) at Report Data View element level (Report table/Matrix Column level or Report Graph Node/Arc level), see How to Define the Filters Displayed in a Report or How to apply a Conditional Style to a Graph dedicated sections.*

Prerequisite: the Report Style is created, you can either select one of the available public Report Styles, or create a public Report Style, see How to Define the Filters Displayed in a Report section.

To apply a general style to a graph:

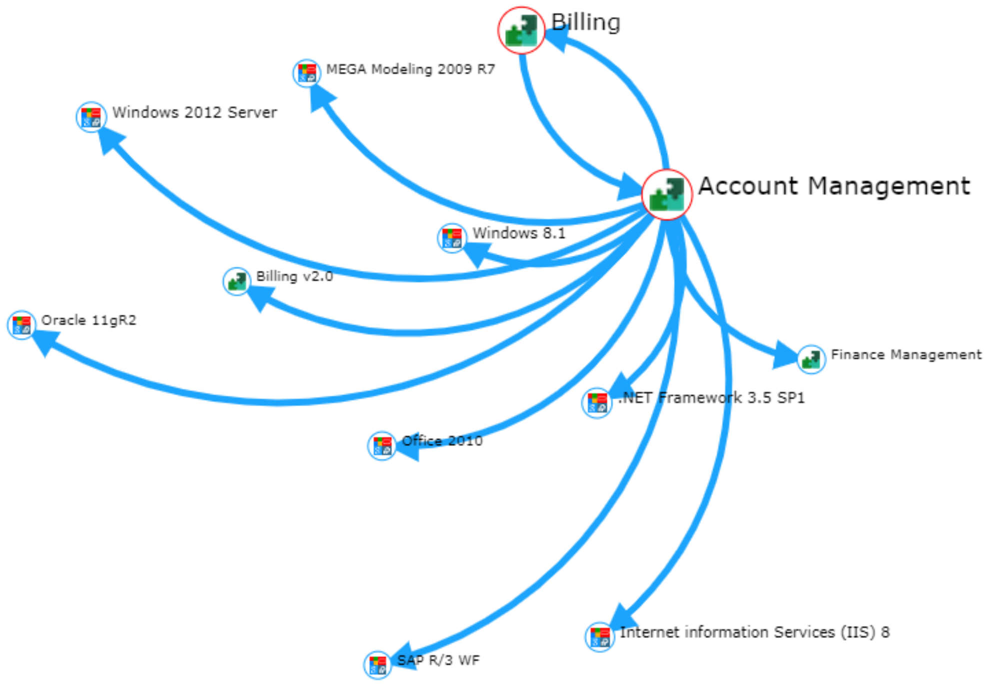
1. Access the Report Graph View properties.

☛ *See Accessing the Report Templates and their Constituents.*

2. In its **Characteristics** page > **General Style** field, use the drop-down list to select the style you want to apply to the graph.

   `E.g.: "Arc Thickness" style for thicker arcs.`

   `All the graphs belonging to the View display thicker arcs.`



How to Customize a Report Color Palette

You can create your own report color palette. For this purpose you have to duplicate an existing palette and modify it.

You can create a new palette for another library.

To create a color palette:

1. Connect to **HOPEX** (Windows Front-End) with **HOPEX Customizer** profile.
2. From **HOPEX** menu bar, select **View > Navigation Windows > Utilities**.
3. Expand **HOPEX Palette** folder.
4. Right-click the palette that will provide the basis for your palette and select **Manage > Duplicate**.
5. (If needed) In the **Library** pane, select the option for duplicating the palette in another library and from the drop-down list select the target library.
6. Select the duplication name format for your palette (Prefix or Suffix).
7. Click **OK**.
   The color palette is duplicated. You can customize it.
8. Access the color palette properties.
9. From the **Characteristics** tab, in the **Palette Type**, select "Report Palette".
10. From the **Color set** tab, modify each **HOPEX Palette Color** as needed.

11. Click **OK**.

CUSTOMIZING A TABLE REPORT TEMPLATE

See:

- How to Apply a Conditional Style on a Column
- How to Add Sub-totals in a Table Break
- How to Add a Grand Total Break (Use case)
- How to Apply a Conditional Style on a Pie Chart
- How to Customize the Empty Label in a Pie Chart

See also Customizing a Report Template Style.

How to Apply a Conditional Style on a Column

Conditional styles on columns apply to Report Table Views and Report Matrix Views.

This enables to apply a conditional style on a column.

To apply a conditional style on a column:

1. Access the Report Table Column (or Report Matrix Column) properties.

   ☛ *Access and expand the Report Table View or Report Matrix View concerned, see Accessing the Report Templates and their Constituents.*
2. In the Report Table/Matrix Column properties, display its **Style** page.

   ☛ *See Report Table Column Properties.*
3. Click **Add** +.
4. Enter a **Name** to your conditional style.
5. In the **Style Parameters** list, select the style you want to add on condition.

   ☺ *Only the style parameters applying to the Report Data View renderer are listed. You can sort them according to a specific column (for example, according to the* ***Applies to*** *column) or use the list filtering tool to filter the styles.*

   ☛ *You can select several styles.*
6. Click **Next**.
7. In the **Style Parameters** list, select the style parameter and configure it in the right part.

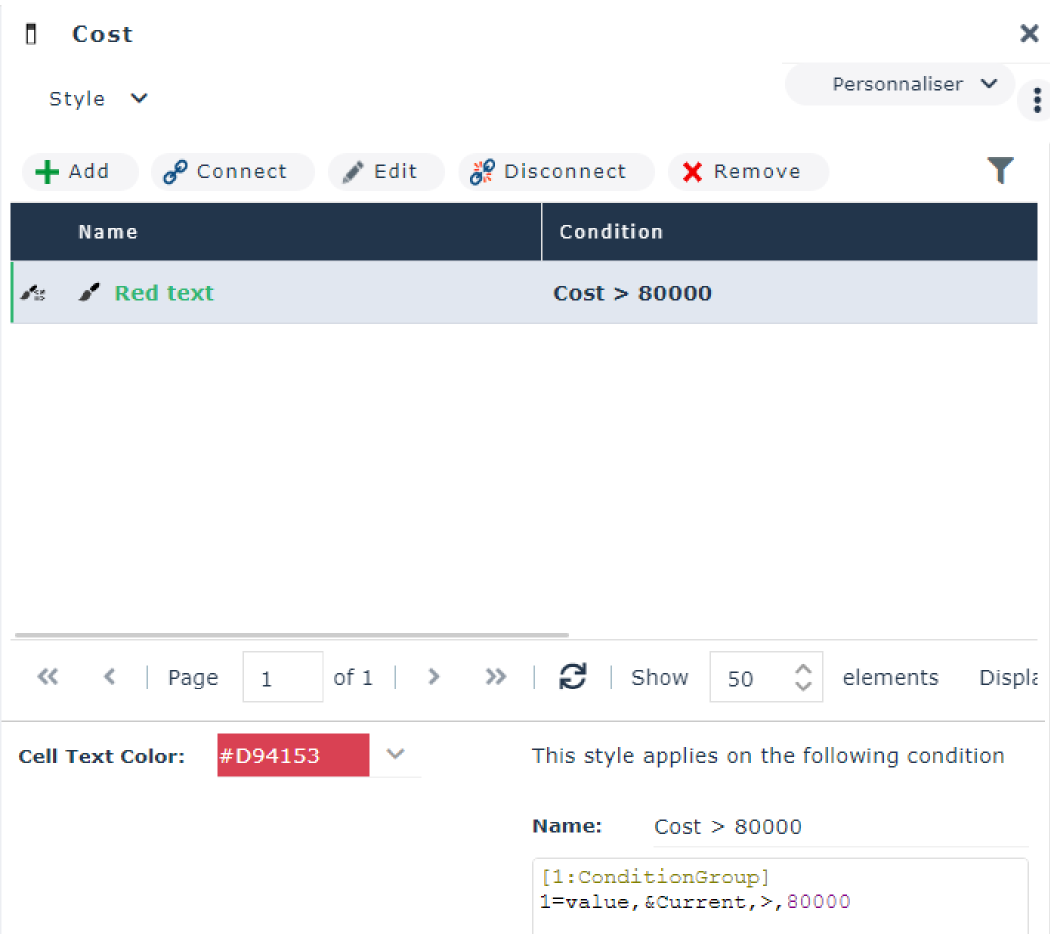
   `Example: for` **a Cell Background Color** `or a` **Series Color**`, in` **Value** `keep "Fixed" and select a color.`
8. Click **OK**.
9. In the **Style** page, select your conditional style row.

10. In **This style applies on the following condition**:

- in the **Name** field enter a name describing the condition
- in the pane enter the code for your condition.

☛ *For code syntax, see Report Style Condition Properties.*

```
Example:

For a column with numerical values, to add a condition style on values
superior to 80000 enter:

[1:ConditionGroup]

1=value,&Current,>,80000
```



Application	Rating	Technologies	Vendor	Cost	Number of users
* AirportMega.com	4 Stars	* Apache log4j v2.17	The Apache Software Foundation	€324,000.00	275
		.NET Framework 4.8	Microsoft	€324,000.00	275
		Application Load Balancer	Amazon Web Services	€324,000.00	275
		SQL Server - Enterprise - 15.0	Microsoft	€324,000.00	275
* MEGA BANK Mobile App	5 Stars	.NET Framework 4.8	Microsoft	€1,620,000.00	3000
		ASP.NET 4.5	Microsoft	€1,620,000.00	3000
		JAVA SE 11	Oracle	€1,620,000.00	3000
		Mobile SDK	Amazon Web Services	€1,620,000.00	3000
		Node.js - 14.0	Node.js Foundation	€1,620,000.00	3000
Account Management	4 Stars	.NET Framework 4.7.2	Microsoft	€47,800.00	250
		Office 2010	Microsoft	€47,800.00	250
		SAP R/3 WF	SAP	€47,800.00	250
		SQL Server - Enterprise - 15.0	Microsoft	€47,800.00	250
		Windows 10	Microsoft	€47,800.00	250
		Windows Server 2016 - Essentials - 10.0	Microsoft	€47,800.00	250

How to Add Sub-totals in a Table Break

In Table type Report Data Views, where a grouping is defined, you can add break rows in tables when the value of a given column changes.

☛ *See Table (Report Table View).*

In the Break rows, you can display computation results performed on each column including numerical values.

To add sub-totals in a table:

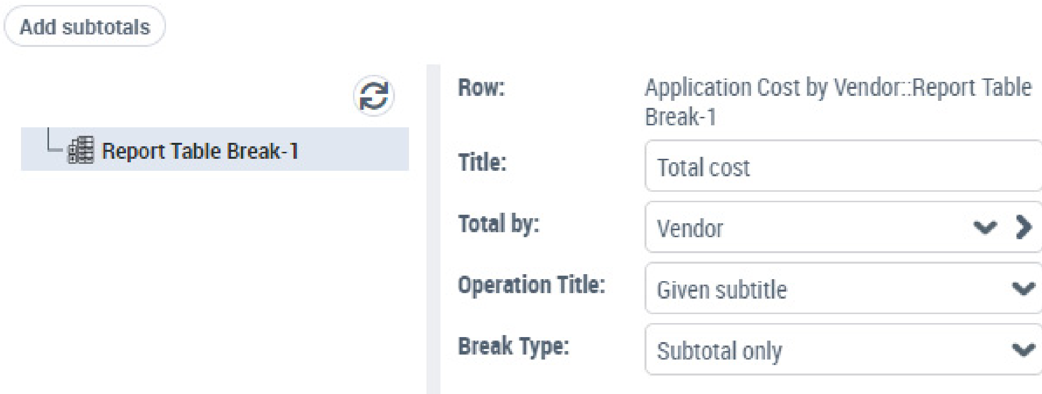
1. Access the **Report Table View** properties: **Table**.

   ☛ *See Accessing the Report Templates and their Constituents.*

2. Click **Add subtotals** to add a break.
   The break is created.

3. Select the break row, and define its characteristics:
   - (optional) in **Title** field, enter the title to be displayed in the break.
   - in **Total by**, select the grouping column.
   - in **Operation Title**, select the subtitle display (values are: Automatic/Given/No subtitle).
   - in **Break Type**, select the break types you want to add (sub-total only or sub-total and grand-total).

     ☛ *See Table (Report Table View).*



The break row is defined with a title.

4. Roll the mouse over the break and click **Add computation**:
   - in the **Column Concerned** field, select the column concerned by the computation
   - In the **Function** field, select the computation to be applied to the column.

   For example, you can group the table rows according to Vendor. You can add a break row after each group (Vendor), and for each vendor, you can display the number of Technologies and the sum of costs.

You get the following report:

⊞ Rating ⊞ Vendor
Detail

Vendor	Technology	Application	Cost
Debian	Debian Squeeze	Word Checker	€0.00
		Investors management	€12,000.00
		LogiKS	€36,000.00
		Communication Management	€36,000.00
	Debian Weezy	Trilix	€0.00
		Holidays/vacancies Management	€6,000.00
		Instant Messaging	€0.00
		Local Customer Management	€0.00
		CMDB Management	€0.00
		CRM US	€387,000.00
	2.0		€477,000.00
HP	Quality Center 11.5	Order Management	€0.00
		Compliance Management	€36,000.00
	1.0		€36,000.00
IBM	Cognos Business Intelligence 10.2	e-Strategy	€11,000.00
	DB2 v9.7	Account	€0.00
	2.0		€11,000.00
MEGA	HOPEX V1R1	MEGA Audit	€0.00

How to Add a Grand Total Break (Use case)

In a Table-type Instant Report generated from a Report DataSet, the end-user can add breaks to the table columns. This break type is "Subtotal only".

☛ *See Defining the columns in a table-type instant report.*

In HOPEX Studio, you can modify the break type as "Subtotal and Grand total", which adds subtotal breaks and a grand total break.

This customization is performed in the **Report Table View** properties.

To add a break with grand total in a table:

1. Generate a table-type Instant Report from a Report DataSet.

   ```
   For example use the : "Application Technonologies" Report DataSet
   Definition to generate a Table-type Instant Report from its Report
   DataSet.
   ```

2. In the **Report** section, click **More** ⋮ **> Save As Report Template** .

3. In the **Name** field, enter your Report Template name.
   A public Report Template is created, with its public **Report Table View**. They both have the same name.
4. Access the **Report Table View** properties.

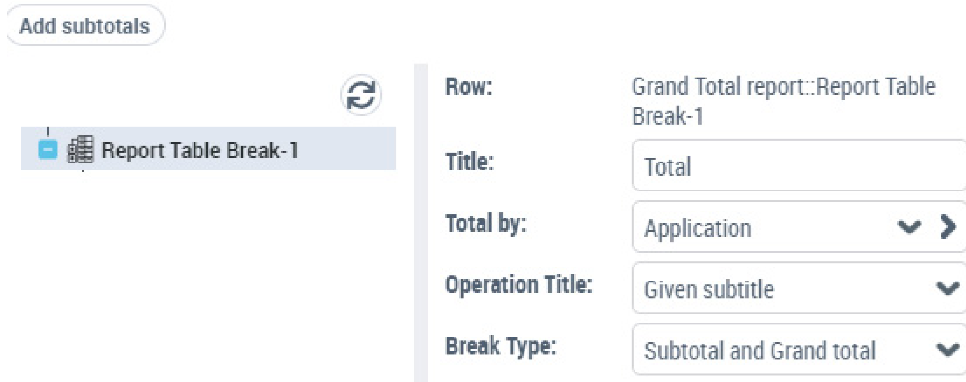
   ☛ *The Report Table View is directly accessible from the* ***Public Report Template*** *folder (****Report Studio > Report Template Definition > Report Table View****).*

   ☛ *The Report Template is accessible from the* ***Public Report Template*** *folder (****Report Studio > Report Template Definition > Report Templates****).*
5. Display its **Table** page.
6. Click **Add subtotals** to add a break.
7. Select the break row, and define its characteristics:
   - (optional) in the **Title** field, enter the title to be displayed in the break.
   - in the **Total by** field, select the grouping column.

     `Example: select Application to add a break on applications.`
   - (optional) in the **Operation Title** field, select the subtitle display (values are: Automatic/Given/No subtitle).
   - in the **Break Type** field, select the "Sub-total and Grand total" break type.

     ☛ *See Table (Report Table View).*

   The break row is defined.



8. Roll the mouse over the break and click **Add computation**.
9. In the **Function** field, select the computation to be applied to the column.

10. Test the report: in the Report Template tree, right-click the Report Template and select Test:
Both total breaks and grand total row are displayed.

z/OS	0 Star			€0.00	0
Total		0.0			
Zeus Purchasing v2.0	0 Star	.NET Framework 3.5 SP1	Microsoft	€740,000.00	0
		Internet information Services (IIS) 8	Microsoft	€740,000.00	0
		Oracle 10gR1	Oracle	€740,000.00	0
		SAP ERP CRM 7.1	SAP	€740,000.00	0
		SAP NetWeaver	SAP	€740,000.00	0
		SAP R/3 AM	SAP	€740,000.00	0
		SAP R/3 CO	SAP	€740,000.00	0
		SAP R/3 FI	SAP	€740,000.00	0
		SAP R/3 HR	SAP	€740,000.00	0
		SAP R/3 IS	SAP	€740,000.00	0
		SAP R/3 MM	SAP	€740,000.00	0
		SAP R/3 PM	SAP	€740,000.00	0
		SAP R/3 PP	SAP	€740,000.00	0
		SAP R/3 PS	SAP	€740,000.00	0
		SAP R/3 QM	SAP	€740,000.00	0
		Windows 2012 Server	Microsoft	€740,000.00	0
		Windows 7	Microsoft	€740,000.00	0
		Windows 8.1	Microsoft	€740,000.00	0
Total		18.0			
Grand Total		172.0			

How to Apply a Conditional Style on a Pie Chart

To add a conditional in a pie chart, you need to apply a conditional style on the Report Table Column property of the Report Table View.

To apply a conditional style in a pie chart:

1. Access the Report Table Column properties.

☛ *Access and expand the Report Table View concerned, see Accessing the Report Templates and their Constituents.*

2. In the Report Table Column properties, display its **Style** page.

☛ *See Report Table Column Properties.*

3. Click **Add** + .
4. Enter a **Name** to your conditional style.
5. In the **Style Parameters** list, select the style you want to add on condition.

   ☺ *Only the style parameters applying to the Report Data View renderer are listed. You can sort them according to a specific column (for example, according to the* ***Applies to*** *column) or use the list filtering tool to filter the styles.*

   ☛ *You can select several styles.*
6. Click **Next**.
7. In the **Style Parameters** list, select the style parameter and configure it in the right part.

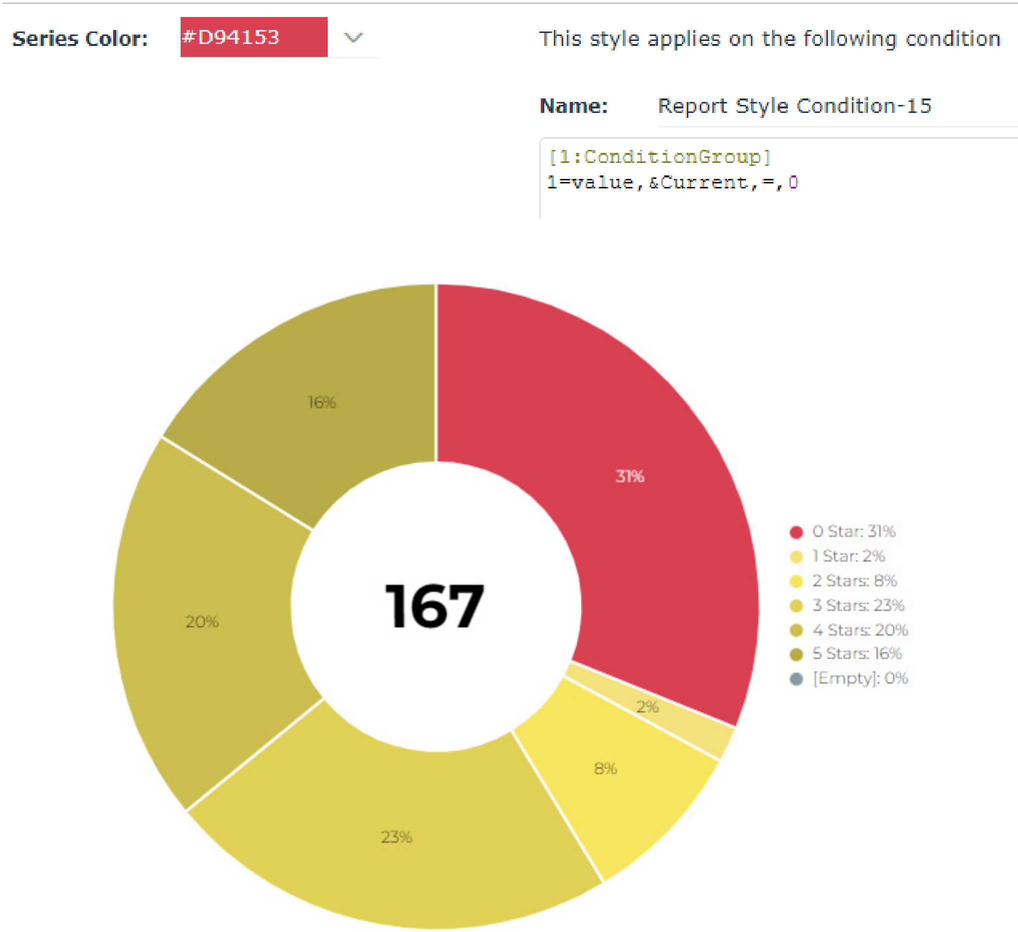
   Example: for **a Cell Background Color** or a **Series Color**, in **Value** keep "Fixed" and select a color.
8. Click **OK**.
9. In the **Style** page, select your conditional style row.

10. In **This style applies on the following condition**:

- in the **Name** field enter a name describing the condition
- in the pane enter the code for your condition.

  ☛ *For code syntax, see Report Style Condition Properties.*

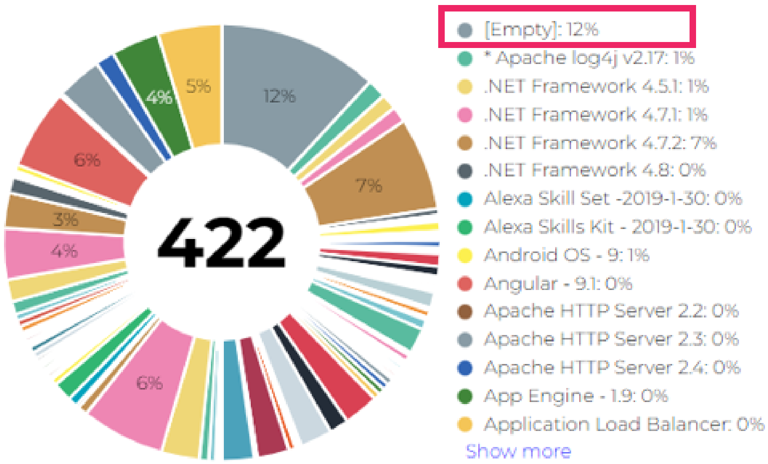
```
Example: for a column with ranking, to add a conditional style on "0
Star" ranking
[1:ConditionGroup]
1=value,&Current,=,0
```



How to Customize the Empty Label in a Pie Chart

Pie chart reports may include an empty series.

> `E.g.: the pie chart showing the breakdown of applications according to technologies, displays an [`**`Empty]`**` series corresponding to the Applications with no Technology.`



You can customize this [Empty] label.

This customization is performed, in HOPEX Studio, in the **Report Table View** properties.

To customize the label [Empty]:

1. Generate a Breakdown-type Instant Report from a Report DataSet.

   `For example use the : "Application Technonologies" Report DataSet Definition to generate a Breakdown-type Instant Report from its Report DataSet.`

2. In the **Report** section, click **More** **> Save As Report Template** .
3. In the **Name** field, enter your Report Template name.
   A public Report Template is created, with its public **Report Table View**. They both have the same name.
4. Access the **Report Table View** properties.

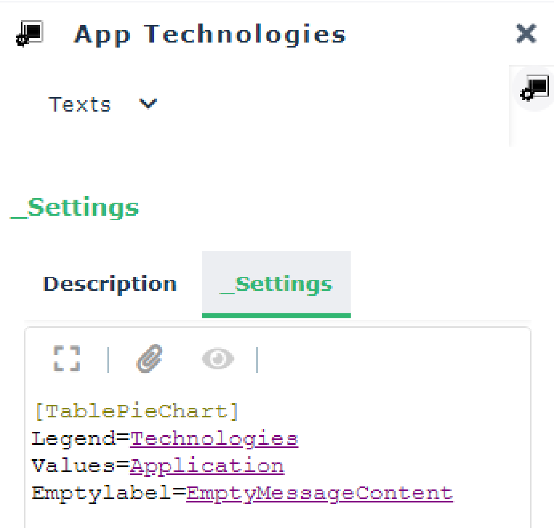
   ☛ *The Report Table View is directly accessible from the **Public Report Template** folder (**Report Studio > Report Template Definition > Report Table View**).*

   ☛ *The Report Template is accessible from the **Public Report Template** folder (**Report Studio > Report Template Definition > Report Templates**).*

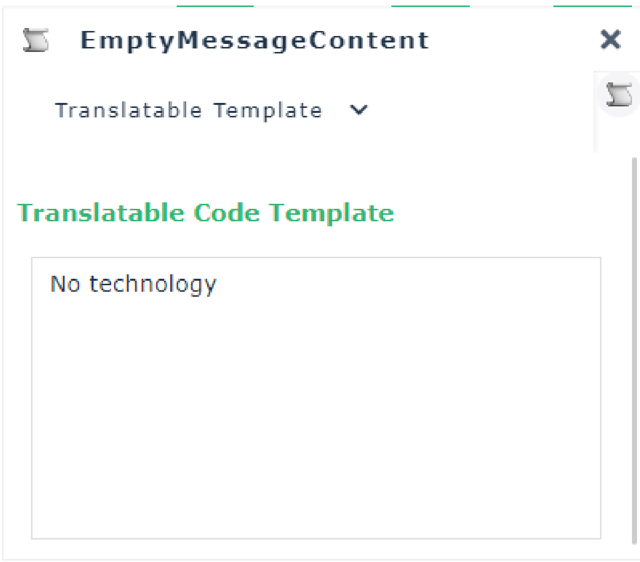
5. In its **Text** page, display **_Settings** tab.

6. Enter the following code:

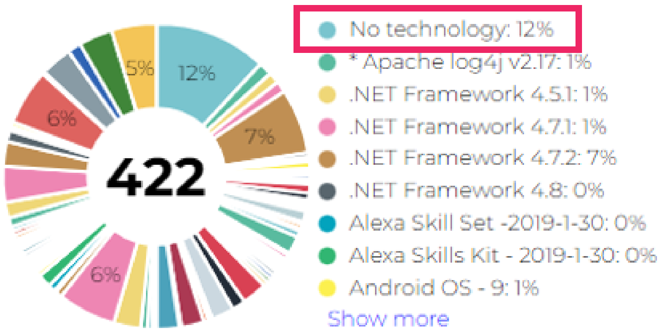
```
Emptylabel=EmptyMessageContent
```



Where "EmptyMessageContent" is a _Code Template, whose **Translatable _Code Template** value replaces the [Empty] label.




```
For example, the report shows "No Technology" instead of "Empty" for
applications that are not linked to any technology.
```


422
12%
5%
6%
7%
6%
No technology: 12%
* Apache log4j v2.17: 1%
.NET Framework 4.5.1: 1%
.NET Framework 4.7.1: 1%
.NET Framework 4.7.2: 7%
.NET Framework 4.8: 0%
Alexa Skill Set -2019-1-30: 0%
Alexa Skills Kit - 2019-1-30: 0%
Android OS - 9: 1%
Show more

CUSTOMIZING A GRAPH REPORT TEMPLATE

See:

- How to apply a Conditional Style to a Graph
- How to Modify the Arc Display in a Graph
- How to Replace Node Names by Property Values in a Graph
- How to Group Arcs in a Graph
- How to Hide the Path Intermediate Nodes

See also Customizing a Report Template Style.

How to apply a Conditional Style to a Graph

You can apply a ***style*** to a graph:

- at Report Graph ***View*** level, so that the style applies to all of the arcs and nodes in the graph.

  ☛ *In that case, see How to Apply a General Style section.*

- at each Report Graph ***View element*** level, i.e. at each Arc level and each Node level.
  At View element level you can add a condition to the style, else the style is always applied.

The condition can apply to:

- node value
  The condition can concern:
  - the object associated with the node
  - any field value of the node
  - any property of the object field value.
- path value
  The condition can concern:
  - any field value of node
  - any field value of arc of the path
  - any node of the path.
- grouped arcs
  The condition is available on arc.

Configuring the Report Graph Layer

To customize the arc style and/or the node style, you must configure the **Report Graph Layer** to which they belong beforehand.

To configure the Report Graph Layer:

1. Access the **Report Graph View** properties.

   ☛ *See Accessing the Report Templates and their Constituents.*

2. Display the **Graph** page.
3. In the **Report Graph Layer** list, select the Report Graph Layer you want to configure.

4. In the **Nodes of the Graph Layer** list, click **New** + and configure each node as follows:
   - In the **Local name** field, enter its name.
     - ☛ *E.g.: Appli, Techno.*
   - With the **GraphSet Node Description** drop-down list, select its associated ***GraphSet Node***.
     - ☛ *E.g.: Application, Technology.*

   You can apply a conditional style to the node.
     - ☛ *See Applying a conditional style to a node.*
5. In the **Arcs of the Graph Layer** list, click **New** + and configure the arc as follows:
   - In the **Local name** field, enter its name.
     - ☛ *E.g.: Appli-Techo*
   - Keep the **Arrow Direction** and **Visibility** default values (you may modify them later if needed).
     - ☛ *To modify the arc display, see How to Modify the Arc Display in a Graph*
   - In the **GraphSet Arc** drop-down list, select its associated ***GraphSet Arc***.
     - ☛ *E.g.: Appli-Techno.*

   You can apply a conditional style to the arc.
     - ☛ *See Applying a style to an arc.*

Applying a style to an arc

To apply a style to an arc, you need to create a style and define it with as many style parameters as needed:

- arc color
- arc thickness
- arc line curvature

  `E.g.: curved, straight.`
- arc line style

  `E.g.: dotted, dashed, line.`
- arc start and/or arc end

  `E.g.: dot, empty arrow, filled diamond, half arrow.`

Each style parameter value can be:

- fixed
- dependent on a field, or
- dependent on the number of grouped arcs

If needed, you can apply the style on a condition based on a ***field of the arc***, else the style always applies.

Prerequisite: The Report Graph Arc is available, see Configuring the Report Graph Layer.

To apply a style to an arc:

1. Access the Report Graph Arc properties.

   ☛ *Access and expand the Report Graph View concerned, see Accessing the Report Templates and their Constituents.*

   `E.g.: "App-Techno" arc, representing the arc between Application and Technologie.`
2. In the Report Graph Arc properties, display its **Style** page.
3. Click **Add**.

   ☛ *Click **Connect** if the style you want to add already exists.*
4. Enter a **Name** to the style.

   `E.g.: "Dashed arc".`
5. In the **Style Parameters** list select a style.

   ☺ *Only the style parameters applying to a graph (renderer of a Report Graph View) are listed. You can sort them according to a specific column (for example, according to the **Applies to** column) or use the list filtering tool to filter the styles.*

   E.g.: select **Arc Line Style**.

   ☛ *You can select several styles.*
6. Click **Next**.
7. In the **Style Parameters** list select a style to configure it, and either:
   - in the **Value** field select "Fixed, and in the field that appears select the value.

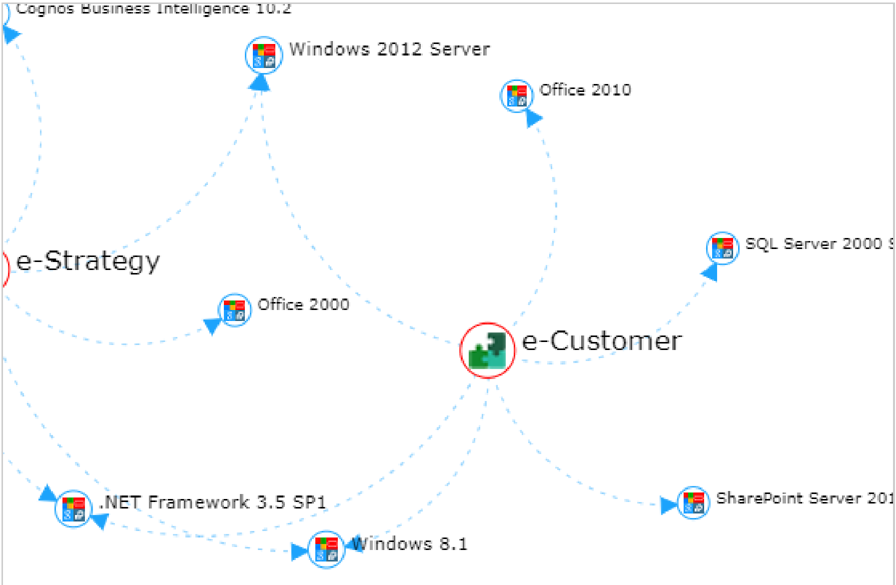
     E.g.: for **Arc Line Style**, select "Dashed".

   or
   - in the **Value** field select "Field, and in the **Field** field select the field on which is based the style.

   or
   - in the **Value** field select "Number of grouped arc".

8. Click **OK**.
   The style is applied to the arc.

   ```
   The graph displays dashed lines for all of its arcs.
   ```



There is no field on arcs between Applications and Technologies (i.e. no MetaAttribute available on the MetaAssociation Application-Technology), so that in this example, a conditional style cannot be applied.

Applying a conditional style to a node

To apply a style to a node, you need to create a style and define it with as many style parameters as needed:

- node size
- node border color
- node fill color

Each style parameter value can be:

- fixed, or
- dependent on a field

If needed, you can apply the style on a condition based on a field of the node, else the style always applies.

Prerequisite: The Report Graph Node is available, see Configuring the Report Graph Layer.

To apply a conditional style to a node:

1. Access the Report Graph Node properties.

   ```
   E.g.: "App" node, representing Applications.
   ```

   ☛ *Access and expand the Report Graph View concerned, see Accessing the Report Templates and their Constituents.*

2. In the Report Graph Node properties, display its **Style** page.
3. Click **Add** ✚.
4. Enter a **Name** to the style.

   `E.g.: "Red filled node" to create a style showing a red filled node when a condition is satisfied.`
5. In the **Style Parameters** list select a style.

   ☺ *Only the style parameters applying to a graph (renderer of a Report Graph View) are listed. You can sort them according to a specific column (for example, according to the* ***Applies to*** *column) or use the list filtering tool to filter the styles.*

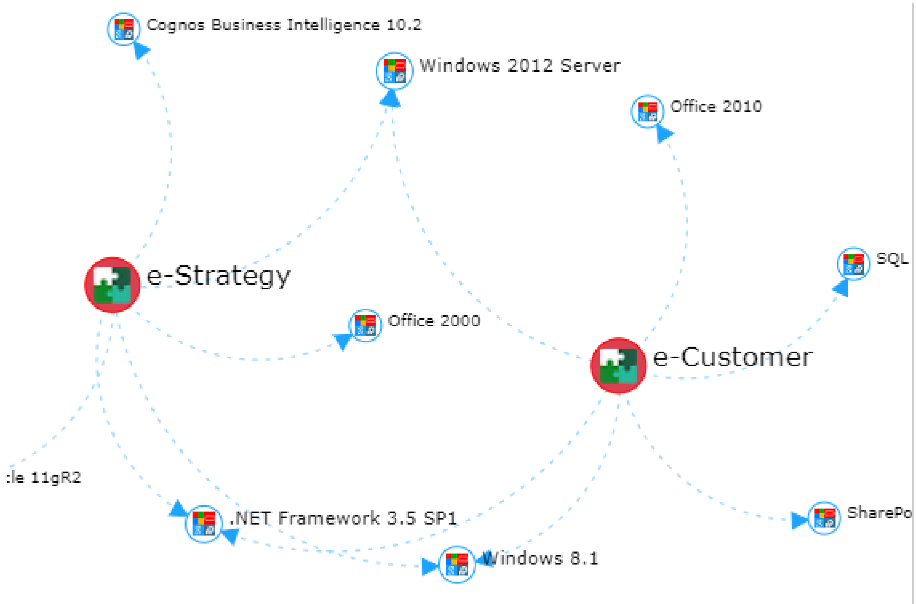
   `E.g.: select "Node Fill Color".`

   ☛ *You can select several styles.*
6. Click **Next**.
7. In the **Style Parameters** list select a style to configure it, and either:
   - in the **Value** field select "Fixed, and in the field that appears select the value.

     E.g.: for **Node Fill Color**, select a red color.

   or
   - in the **Value** field select "Field, and in the **Field** field select the field on which is based the style.
8. Click **OK**.
   The style is applied to the corresponding nodes.

   `All the "App" nodes of the graph are filled with red color.`



9. To apply the style (e.g.: "red filled node") to the node on condition: in the Report Graph Node properties > **Style** page, select the style row.
10. In the bottom page, in the **Name** field, enter a name for the condition.

    E.g.: "SLA critical" to apply a condition when **SLA level** value is "Critical".

11. In the pane enter the code for the condition:

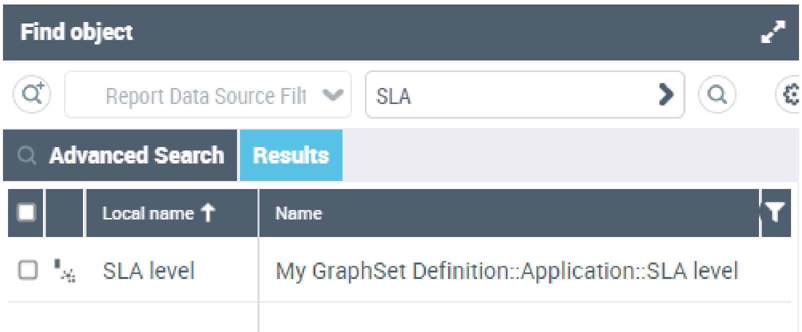
☛ *For code syntax, see Condition on value (Report Graph View element).*

For example, to emphasize "App" nodes with a red fill color when the "SLA level" of the corresponding Application is "Critical", enter:

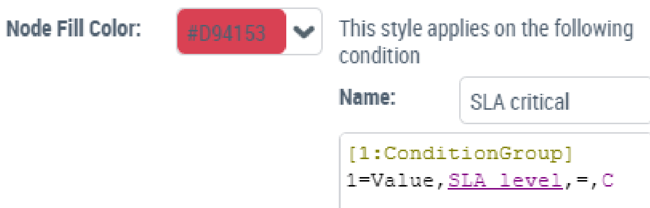
[1:ConditionGroup]

1=Value,<SLA level field>,=,C

☺ *To enter the "SLA level" field, use the Intellisense tool (press [Ctrl]+[Space] keys) and select **Insert a report Data Source Filterable Element** and click **OK**. In the **Find Objet** field, enter "SLA" and select the "SLA level" GraphSet Field Definition.*

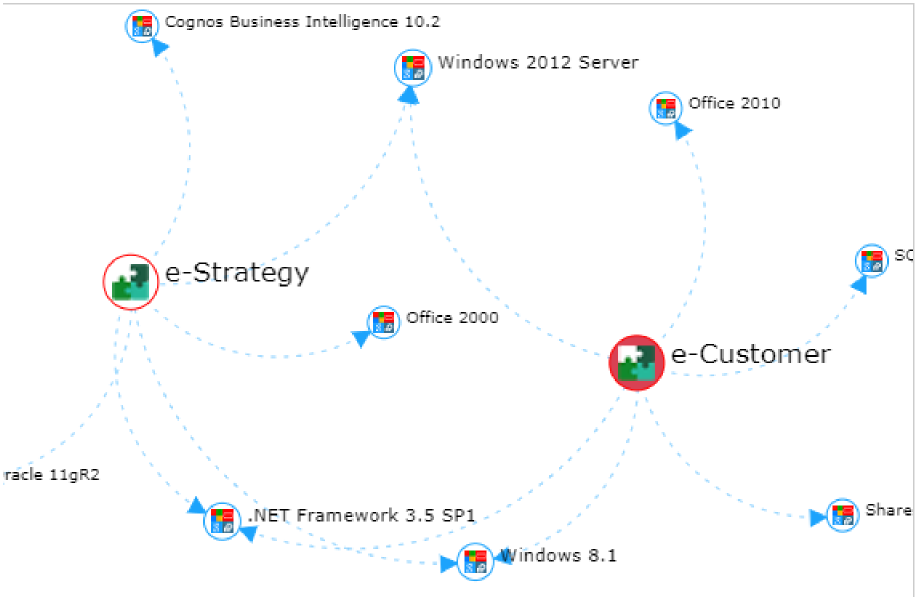


The conditional style is defined:



The conditional style is applied.

In the graph, only the "App" nodes of the graph, representing Applications that have their "SLA Level" to value "Critical" are filled with red color: here "e-customer" Application.



How to Modify the Arc Display in a Graph

A path is a kind or arc and is managed as an arc. The following procedures apply to arcs or path-based arcs.

You can modify the arc display as follows:

- arc direction
  - reverse the arc direction

    `Arc Direction` value: `Reverse`
  - do not display the direction

    `Arc Direction` value: `No Arrow`
- arc visibility

  `Visibility` value: `No` (to hide the arc)

Prerequisite: The Report Graph Arc is available, see Configuring the Report Graph Layer.

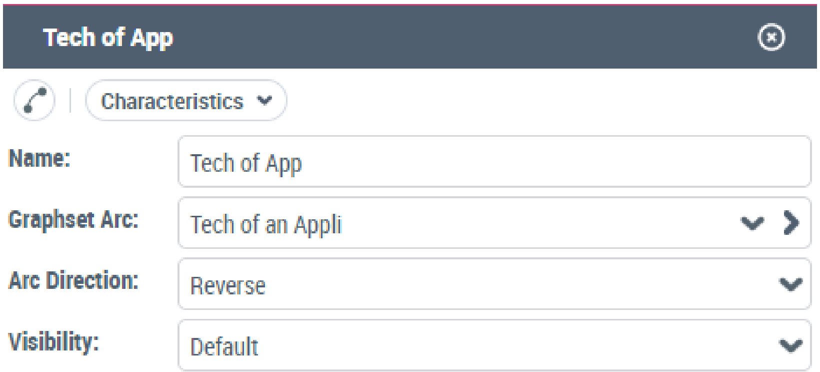
To modify the arc display in a graph:

1. Access the Report Graph Arc properties.

   ☛ *Access the Report Graph View and expand its Report Graph Layer concerned, see Accessing the Report Templates and their Constituents.*

2. In the **Properties** window, display its **Characteristics** page.

3. Modify the arc as required:
   - In the **Arc Direction** field, use the drop-down list to select the new value.
   - In the **Visibility** field, use the drop-down list to select the new value.



How to Replace Node Names by Property Values in a Graph

In the graph, you might want to identify the nodes through one of their properties instead of their names.

```
For example instead of displaying the application names, you can display
their codes.
```

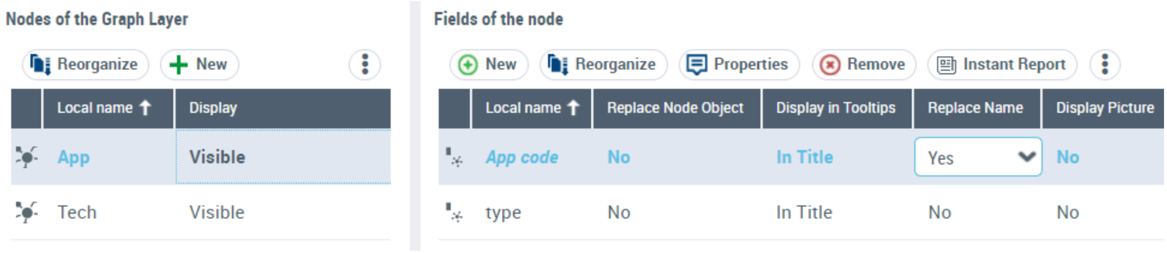
To display a property value instead of the node name:

1. Access the Report Graph View.

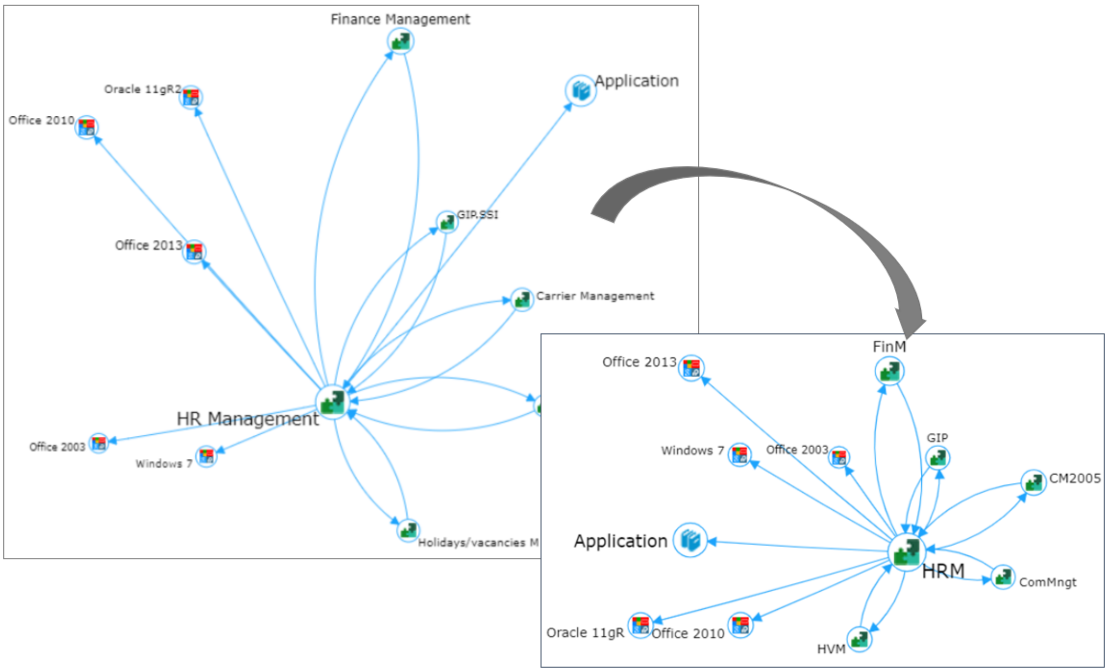
   ☛ *Access the Report Graph View concerned, see Accessing the Report Templates and their Constituents.*

2. In its **Graph** page, select the Report Graph Layer concerned.
3. In the **Nodes of the Graph Layer**, select the node concerned.

4. In the **Fields of the node** list, in the row of the field concerned, use the drop-down list to set **Replace Name** to "yes".



In the graph, applications are identified through their codes.



How to Group Arcs in a Graph

You can group arcs (or path) of the same type or arcs carrying the same field value or the same object value.

The **Grouping on** values are:

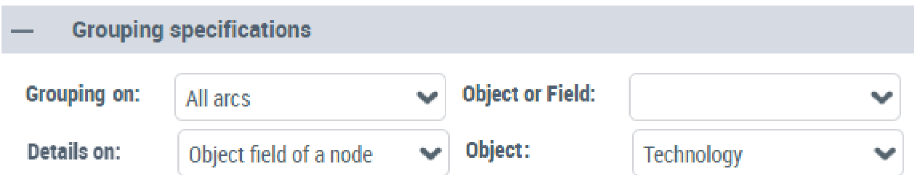
- (by default) "All arcs"
  Groups together all of the arcs matching the GraphSet Arc Definition between two node instances.
- "By field"
  Groups together all of the arcs carrying the same field value. The grouping field must be displayed either on the graph or in a tooltip.
  - ☛ *If the field is a multivalued attribute, its picture is shown in the tooltip.*
  - ☛ *Restricted to one object type.*
- (path-based arc only) "By object"
  Groups together all of the arcs carrying the same objects.

To group arcs of the same type:

1. Access the Report Graph Arc properties.

   ☛ *Access the Report Graph View and expand its Report Graph Layer concerned, see Accessing the Report Templates and their Constituents.*

2. In the **Properties** window, display its **Characteristics** page.
3. In the **Grouping specifications** section, in the **Grouping on** field, use the drop-down list to select the grouping criteria.
4. If you selected a grouping:
   - **By field**: in the **Object or Field** field, use the drop-down list to select one of the available fields of the nodes.
   - **By object**: in the **Object or Field** field, use the drop-down list to select the object.



☛ *You can set a display on details of your arc grouping*

Setting a display on details of an arc grouping

From an arc grouping tooltip, you can display its detailed list of objects, either:

- (path-based arc only) the list of objects of one of its intermediate nodes (**Nodes**), or
- the list of the objects represented by one of the fields of the arc (**Object field of a node**)

☛ *Note: in a path, you can only add fields on its intermediate nodes (not its intermediate arcs).*

When the end-user clicks **Click to see details** (displayed in the arc grouping tooltip), the list of objects is displayed in the **Search and Result** window.

To set a display on details on an arc grouping:

1. Access the Report Graph Arc properties. The arc is a path.

   ☛ *Access the Report Graph View and expand its Report Graph Layer concerned, see* *Accessing the Report Templates and their Constituents.*

2. In the **Properties** window , display its **Characteristics** page.
3. In the **Grouping specifications** section, in the **Details on** field, use the drop-down list to specify the object you want to display.
4. In the **Object** field, use the drop-down list to select the object.

How to Hide the Path Intermediate Nodes

By default the path intermediate nodes are displayed.

If needed, you can change this default behavior and not display the path intermediate nodes. In this case the path appears as an arc.

Hiding the path intermediate nodes is useful when you are only interested in the fields of these nodes. That way, intermediate nodes are not displayed but you can display their fields on the path (arc).

You can modify the **Display** value to:

- "Always detailed" (default value)
- "Never detailed": details are never displayed

To modify the path display:

1. Access the Report Graph Arc properties. The arc is a path.

   ☛ *Access the Report Graph View and expand its Report Graph Layer concerned, see* *Accessing the Report Templates and their Constituents.*

2. In the **Properties** window , display its **Characteristics** page.
3. In the **Path specifications** section, in the **Display** field, use the drop-down list to select the display type.

CUSTOMIZING A GAUGE REPORT TEMPLATE

See also Customizing a Report Template Style.

How to Create and Customize a Gauge Report Template

You can create a gauge Report Template, where the gauge displays:

- an occurrence count
  In this case occurrences are retrieved through a query, and the Report Template is based on a ***Report Value View***.

  ☛ *See Creating a gauge-type Report Template from a query.*

  ☺ *Using queries to create gauge Report Templates does not imply time consuming computation as with Report DataSets.*

- a computed numeric value
  In this case the computed numeric value is retrieved from a Report DataSet, and the Report Template is based on a ***Report Table View***.

  ☛ *See Creating a Report Data View from the Report DataSet, then access the Report Table View properties:*

  *- in its **Characteristics** page, set **Display**: "Graphic" and **Chart Type**: "Gauge Chart".*

  *- in its **Chart** page configure the gauge, see Report Data View Properties: Value.*

  ☛ *Note that, from a Report DataSet, end-users can create Instant Reports showing a set of gauges.*

You can customize the gauge Report Template via its ***Report Data View*** properties.

Example: applications in deployment

You can create a gauge showing the number of applications in deployment. At creation the gauge range is 0-<number of applications in the repository>.

You can still display the number of applications in deployment and customize the gauge, i.e. modifying its minimum and maximum values, for example:

- ***In the minimum value*** you can display the number of applications in production.
  So that you can see the number of applications that you can shortly put in deployment.
- ***In the maximum value*** you can display the number of applications with a cost line.
  So that you can check if you have applications costing money (i.e. applications with cost line but not in production).

To create and customize a gauge Report Template displaying applications in deployment:

1. Connect to **HOPEX Studio** desktop.

   ☛ *See Connecting to HOPEX Studio Desktop.*

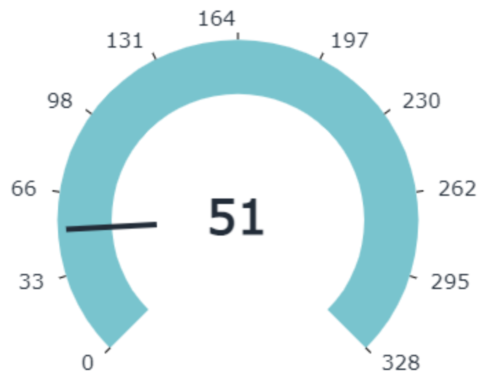
2. Click the Navigation menu and select **Report Studio > Queries** .
3. In the **Queries without mandatory parameters** list, right-click the "ITPM - Get Applications with Deployment" query and select **Documentation > New gauge report**.

4. In the **Name** field, enter a name for your report.

   ```
   E.g.: Use case 1.
   ```

5. Click **OK**.
   The gauge report is displayed.
   Its Report Template and associated Report Value View are also automatically created.

   In this example, ***51*** applications (out of 328) are in deployment.

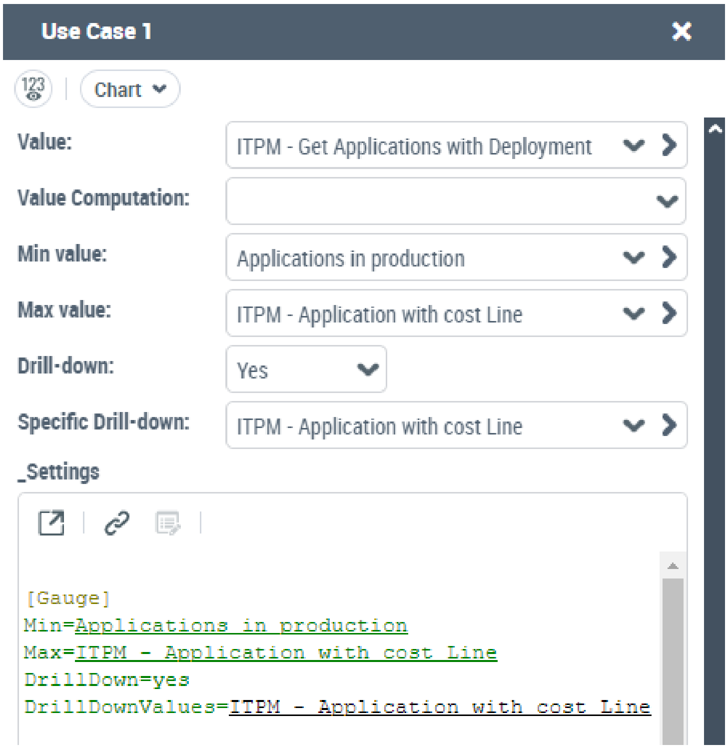


6. Access the Report Value View properties to customize the gauge: from its Report Template: **Report Studio > Report Template Definition**: **Report Templates > Private Report Template > My report Templates** folder.

   ☛ *Else, you can access the Report Value view from **Report Studio > Report Template Definition: Public Report Data Views**.*

7. In its **Chart** page, customize the gauge as follows:
   - in the **Min value** field, select the "Application in production" query.
   - in the **Max value** field, select the "ITPM - Application with cost Line" query.
   - in the **Drill-down** field, select "Yes".
   - in the **Specific Drill-down**, select the "ITPM - Application with cost Line" query.

   The **_Settings** pane automatically displays the customization.



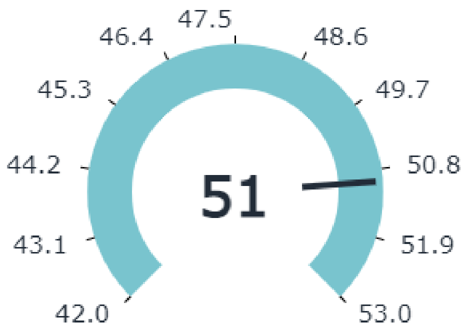
8. Refresh the report to see your modifications.

For example:

- the minimum is now 42 and represents the number of applications in production, which means that 9 applications are on their way to deployment.

- the maximum is now 53 and represents the number of applications with a cost line, which means that 2 applications are costing money.

In that case, when you click the gauge, the list of the 53 applications with cost line is detailed in the **Results** window.



CUSTOMIZING A TREE REPORT TEMPLATE

How to Replace Node Names by Property Values in a Dendrogram

> ☛ *See also How to Replace Node Icons by Property Values in a Dendrogram*

In a dendrogram, you might want to identify the nodes through one of their properties instead of their names.

```
For example, instead of displaying the Technology names, you can display
their codes.
```

To display a property value instead of the node name:

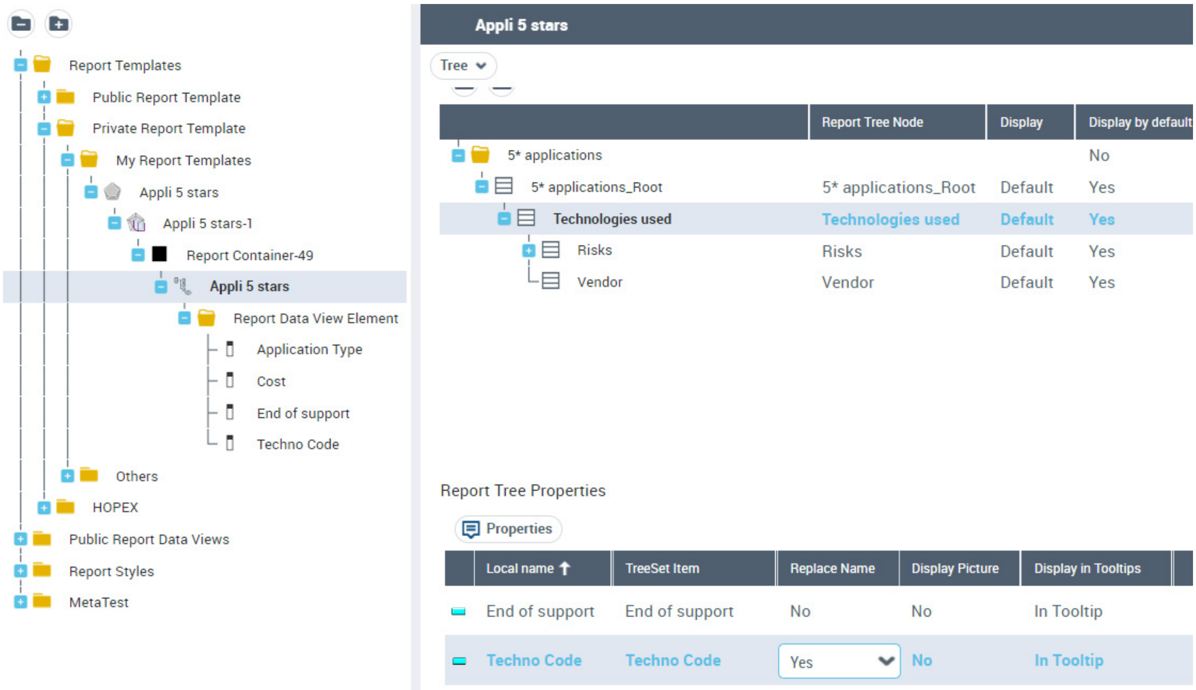
1. Access the Report Tree View.

> ☛ *Access the Report Tree View concerned, see Accessing the Report Templates and their Constituents.*

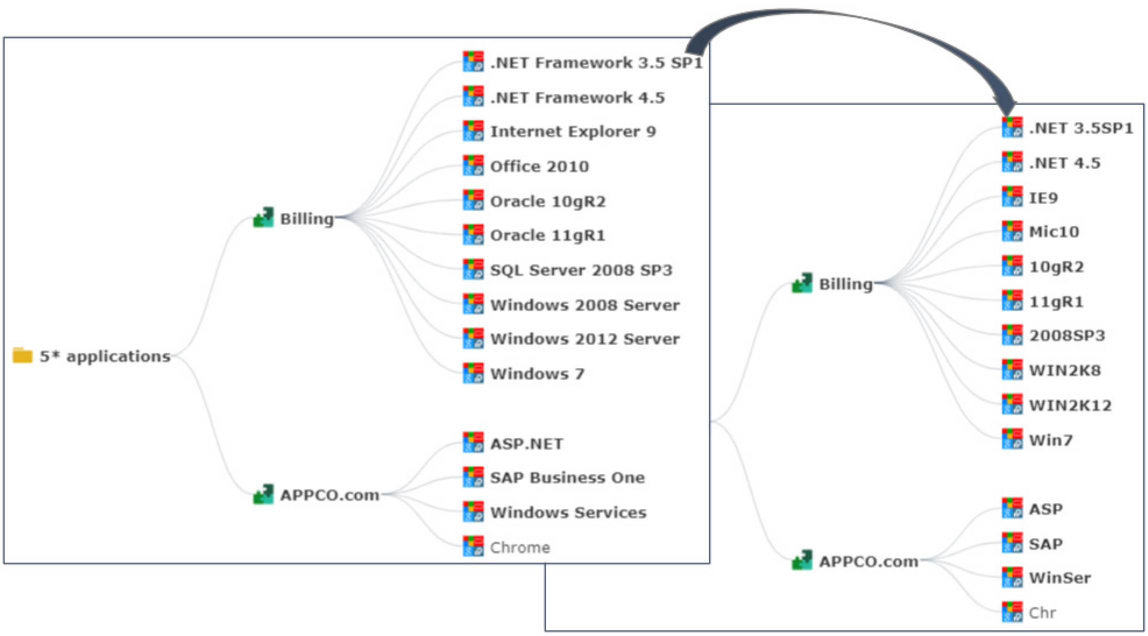
2. In its **Tree** page, select the row of the **Report Tree Node** concerned.

```
Example: Technologies used.
```

3. In the **Report Tree Properties** list, in the row of the property concerned, use the drop-down list to set **Replace Name** to "Yes".



In the dendrogram, the technologies are identified through their codes.



How to Replace Node Icons by Property Values in a Dendrogram

☛ *See also How to Replace Node Names by Property Values in a Dendrogram*

In a dendrogram, you might want to highlight a property of a node and replace its icons by the property values.

```
For example, for a Technology, instead of displaying the Technology icon,
you can display its "Company Standard" value (represented by a color).
```

To display a property value instead of the node icon:

1. Access the Report Tree View.

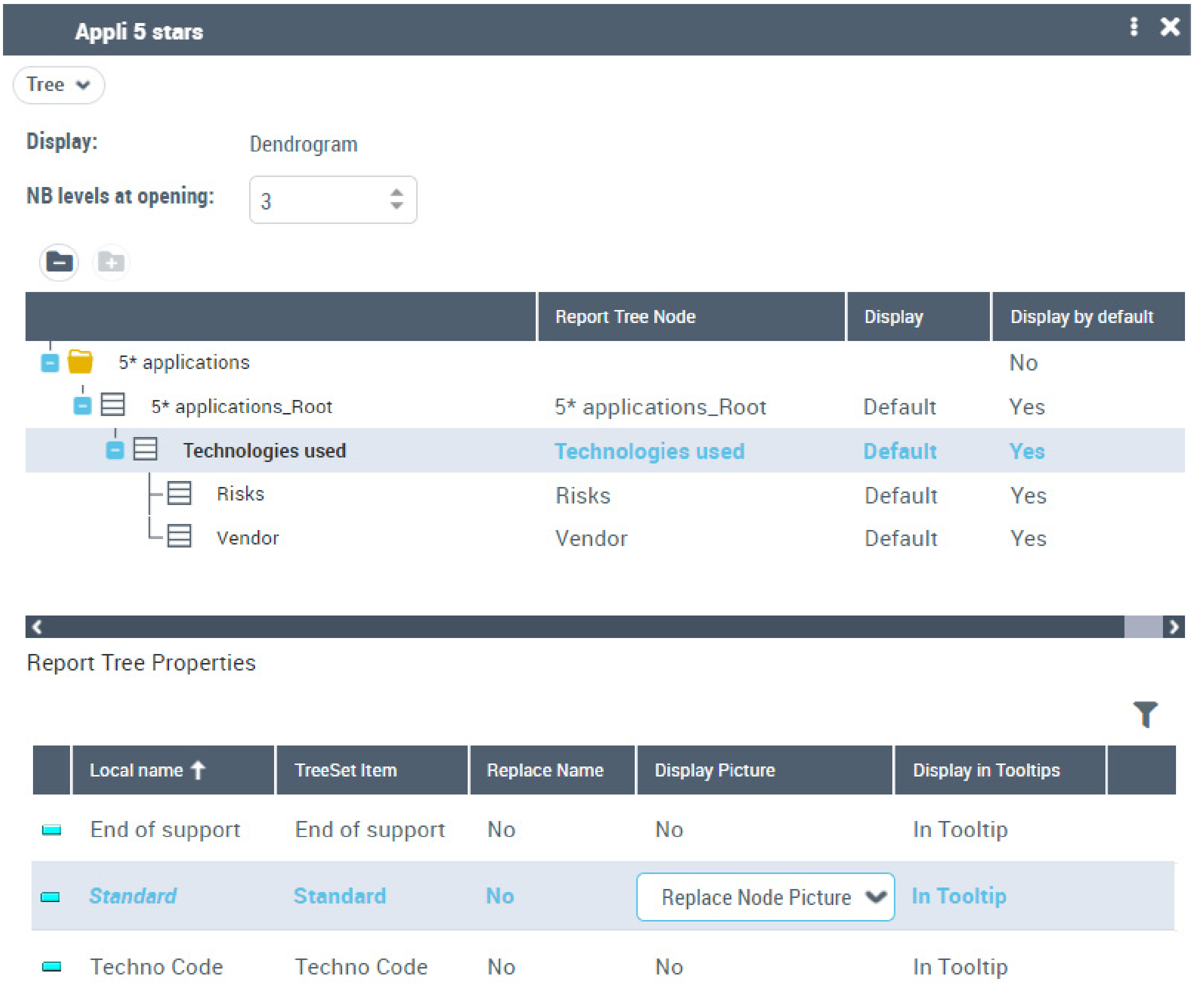
   ☛ *Access the Report Tree View concerned, see Accessing the Report Templates and their Constituents.*

2. In its **Tree** page, select the row of the **Report Tree Node** concerned.

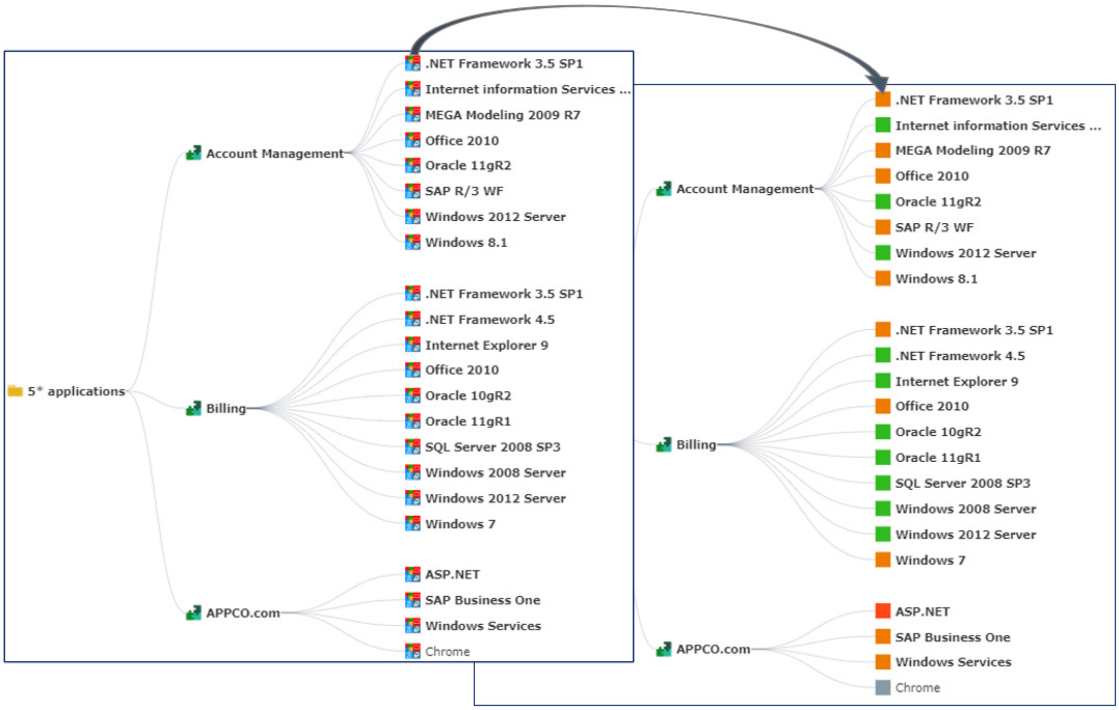
   ```
   Example: Technologies used.
   ```

3. In the **Report Tree Properties** list, in the row of the property concerned, use the drop-down list to set **Display Picture** to "Replace Node Picture".

```
Example: "Standard" property (which stands for "Company Standard"
MetaAttribute of "Software Technology" MetaClass).
```



In the dendrogram, colored squares corresponding to the values of the "Standard" property replace the Technology icons.



How to Apply a Conditional Style in a Treemap

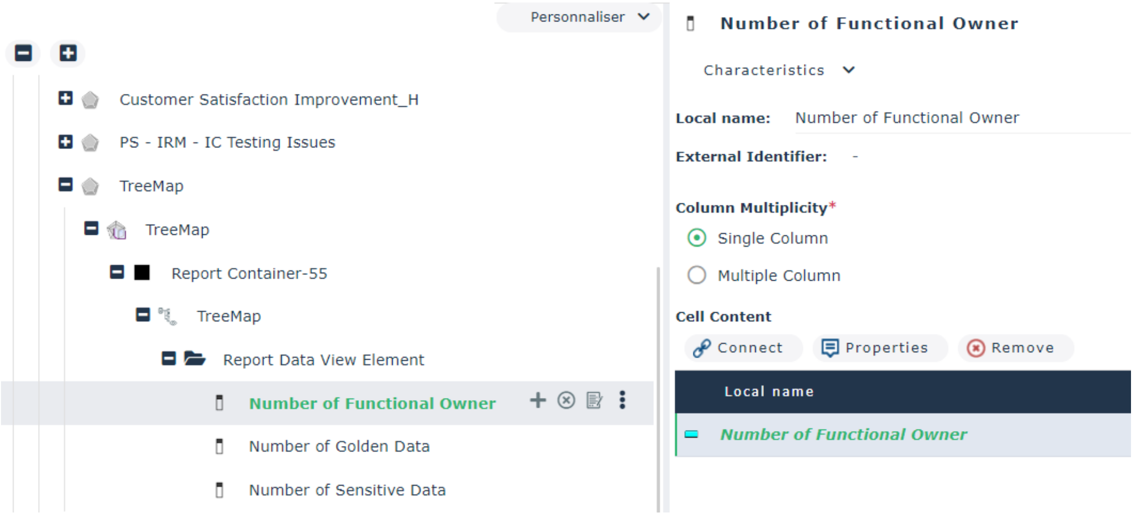
To apply a conditional style in a treemap, you need to apply a conditional style on the **Report Tree Property** associated with the **Tree Report Column** of a Report Tree View.

To apply a conditional style in a treemap:

1. Access the **Tree Report Column** properties.

   ☛ *Access the Report Tree View concerned, see Accessing the Report Templates and their Constituents.m*

2. Access the **Report Tree Property** associated with the **Tree Report Column** concerned:
   - Display the Tree Report Column **Characteristics** page.
   - In the **Cell Content** list, select the **Report Tree Property**.
   - Display its **Properties**.



3. Display its **Style** page.

   ☛ *For details regarding TreeSet properties, see [Adding TreeSet Properties to a TreeSet Collection](#).*

4. Click **Add** ➕.
5. Enter a **Name** to your conditional style.
6. In the **Style Parameters** list, select the style you want to add on condition.

   ☺ *Only the style parameters applying to the Report Tree View renderer are listed.*

   ☛ *You can select several styles.*

7. Click **Next**.
8. In the **Style Parameters** list, select the style parameter and configure it in the right part.

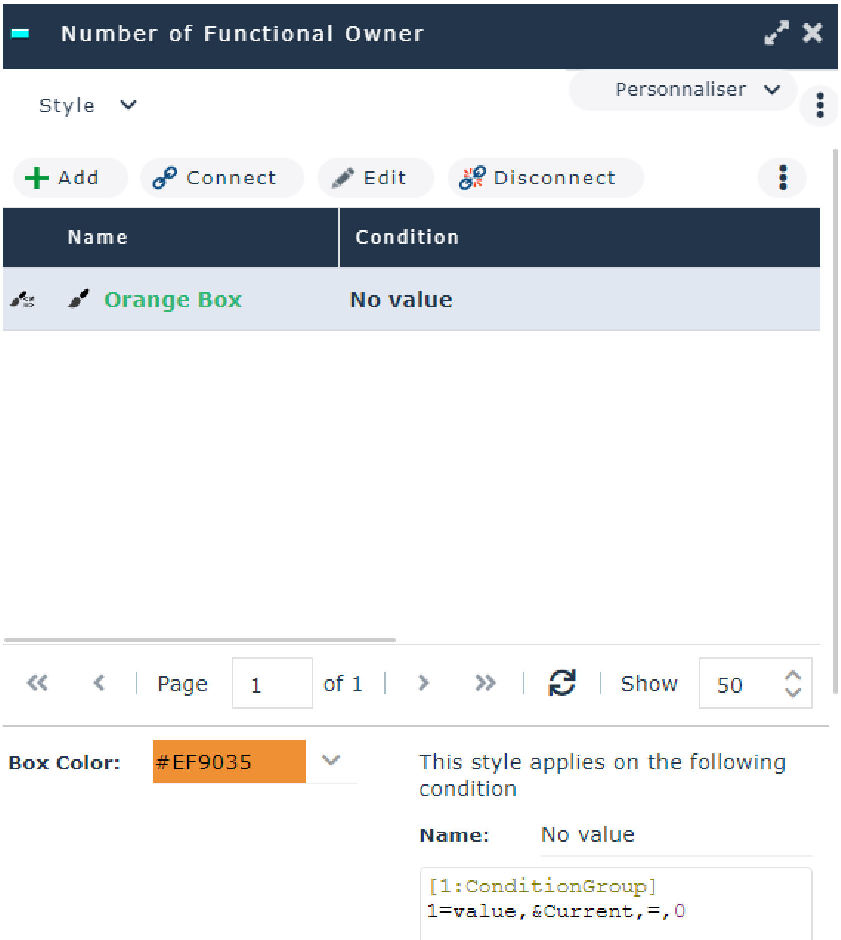
   `Example: for` **a Box Color**`, in` **Value** `keep "Fixed" and select a color.`

9. Click **OK**.
10. In the **Style** page, select your conditional style row.

11. In **This style applies on the following condition**:
- in the **Name** field enter a name describing the condition
- in the pane enter the code for your condition.

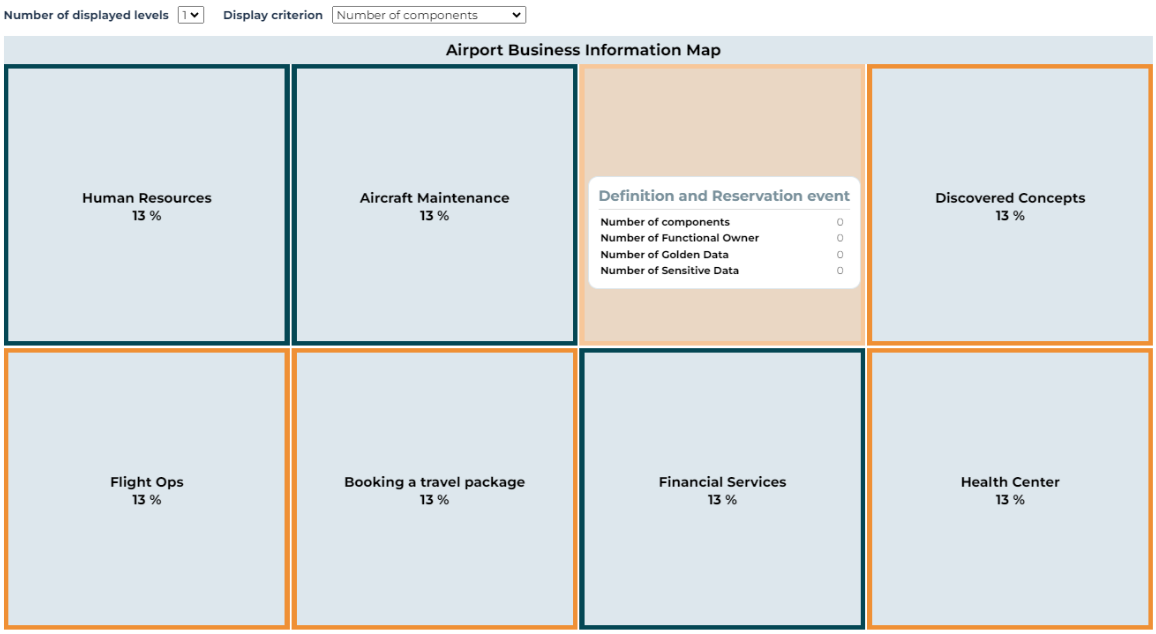
  ☛ *For code syntax, see Report Style Condition Properties.*

```
Example:

To apply a conditional style to "Number of Functional Owner" property
with no value.

[1:ConditionGroup]

1=value,&Current,=,0
```



In the treemap report, borders of boxes matching the condition are colored. The box background is colored when hovering the cursor over the box.

```
For example, the elements with no functional owner.
```


Number of displayed levels 1
Display criterion Number of components
Airport Business Information Map
Human Resources
13 %
Aircraft Maintenance
13 %
Definition and Reservation event
Number of components 0
Number of Functional Owner 0
Number of Golden Data 0
Number of Sensitive Data 0
Discovered Concepts
13 %
Flight Ops
13 %
Booking a travel package
13 %
Financial Services
13 %
Health Center
13 %

Writing Java report renderers

Java report renderers allow the extension of the report engine, offering more rendering possibilities to Java report chapters. This technical article presents how such renderers are modeled and written.

Introduction and Prerequisites

This technical article presents how renderers are modeled and implemented in Java.

It requires a configured Java development environment and that you be familiar with Java report chapter implementation and the data structure which reports generate.

If not, please refer to the following technical articles:

- MEGA Plug-ins with Java
- Writing Java report chapters
- Java report data structure

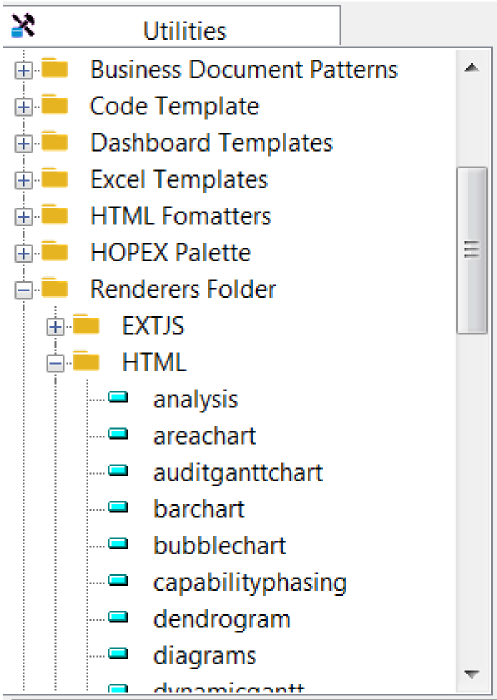
For each rendering format which should be handled (HTML, PDF, etc.), there is one renderer implementation.

> In Mega 2009 SP5, HTML and PDF renderers are taken into account. However, RTF is generated from HTML.
> In Mega 2009 SP4, only HTML renderers are taken into account.

Renderer modeling

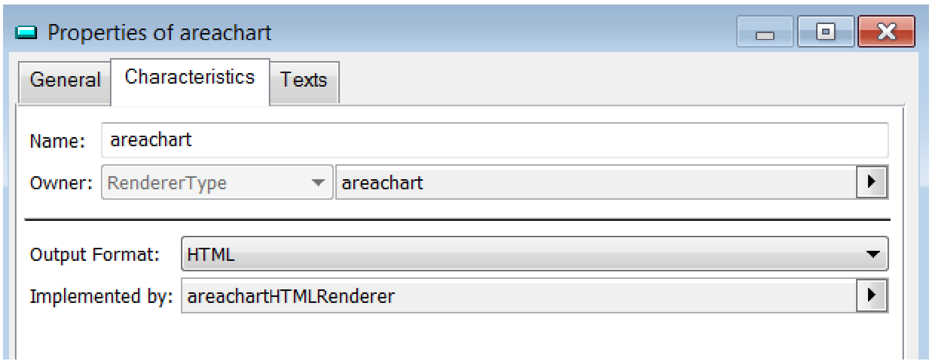
Renderers are modelled in HOPEX and called through an object identifier.

They are available in the Renderers Folder of the Utilities navigation window:



Each renderer is specific to a format and type e.g. HTML areachart renderer.

This is specified in renderer properties:



- Output Format: format handled by this renderer.

- Renderer Type: type of rendering produced. The renderer type is used in the analysis report to specify which rendering to apply to the dataset.

- Implemented by: the implementation macro, in Java (see next chapter for its implementation).

The call of the renderer is handled by the analysis engine depending on generation format (HTML, PDF, etc.).

RENDERER IMPLEMENTATION

Interface to implement

Renderers must implement the `com.mega.modeling.analysis.ViewRenderer` interface. This requires the implementation of a single Generate function.

This results in the following structure:

```
public class MyHTMLRenderer implements ViewRenderer {
  @Override
  public boolean Generate(final Object document, final ReportContent reportContent,
final Item i, final MegaCOMObject megaContext, final Object userData, final MegaRoot
root) {
    …
  }
}
```

The aim of the function is to append the document object with the rendering of Item i, using the data contained in the ReportContent object.

In this situation, the Item is usually the View object created by the analysis report to link a dataset to this renderer.

In the case of HTML, the document object is a StringBuffer.
In the case of PDF, the document object is an aspose.pdf.Pdf.

A Mega Context object, Mega Root and an optional userData are also made available.

This function produces a boolean that indicates:

- True: capable of rendering in this context with the data supplied, and has done so by appending the document object.

- False: not capable of rendering and has not appended the document object. The analysis engine should try to find another suitable renderer.

Implementation example

This example uses ChartDirector 4.1, a graph library provided with Mega capable of rendering many different types of charts.

Further documentation on ChartDirector can be obtained from ASE at:
http://download2.advsofteng.com/v4/cdjavadoc_html_v4.zip

HTML Renderer Code

```
import ChartDirector.BarLayer;
import ChartDirector.Chart;
import ChartDirector.LegendBox;
import ChartDirector.XYChart;
import com.mega.modeling.analysis.AnalysisRenderingToolbox;
import com.mega.modeling.analysis.ViewRenderer;
import com.mega.modeling.analysis.content.Dataset;
import com.mega.modeling.analysis.content.Dimension;
import com.mega.modeling.analysis.content.Item;
import com.mega.modeling.analysis.content.ReportContent;
import com.mega.modeling.analysis.content.Value;
import com.mega.modeling.analysis.content.View;
import com.mega.modeling.api.MegaCOMObject;
import com.mega.modeling.api.MegaRoot;

/**
 * Renderer for bar chart in HTML
 * Accepted dimensionalities: 2
 * @author NLE
 */
public class barchartHTMLRenderer implements ViewRenderer {
  @Override
  public boolean Generate(final Object document, final ReportContent reportContent,
final Item i, final MegaCOMObject megaContext, final Object userData, final MegaRoot
root) {
    if (i instanceof View) {
      final Dataset d = reportContent.getDataset(((View) i).getDatasetId());
      if (d.getDimensionCount() == 2) {
        AnalysisRenderingToolbox.setChartDirectorLicense(root);

        // The data for the chart
        final Dimension dim1 = d.getDimension(0);
        final Dimension dim2 = d.getDimension(1);

        // The labels & data for the chart
        final int labelCount = dim1.getItemCount();
        final int areaCount = dim2.getItemCount();

        final String[] labels = new String[labelCount];
```

```
        for (int ii = 0; ii < labelCount; ii++) {
          labels[ii] = AnalysisRenderingToolbox.getItemText(dim1.getItem(ii), root);
        }

        // Create a XYChart object of size 400 x 240 pixels
        final XYChart c = new XYChart(600, 260);

        // Add a title to the y-axis

c.yAxis().setTitle(AnalysisRenderingToolbox.getCodeTemplate(dim2.getCodeTemplateID(),
root));

        // Add a title to the x-axis

c.xAxis().setTitle(AnalysisRenderingToolbox.getCodeTemplate(dim1.getCodeTemplateID(),
root));

        // Set the x axis labels
        c.xAxis().setLabels(labels);

        // Set the plot area and size.
        c.setPlotArea(40, 30, 340, 190);

        // Add a legend box at top right corner using 10 pts Arial Bold
        // font. Set the background to silver, with 1 pixel 3D border effect.
        final LegendBox b = c.addLegend(570, 30, true, "Arial Bold", 10);
        b.setAlignment(Chart.TopRight);
        b.setBackground(Chart.silverColor(), Chart.Transparent, 1);

        //colors
        String sColors = ((View) i).getParameter("colors");
        String[] tColors = null;
        if ((sColors != null) && (sColors.length() > 1)) {
          tColors = sColors.split(",");
        }

        // Add a multi-bar layer with data sets and 3 pixels 3D depth
        final BarLayer layer = c.addBarLayer2(Chart.Side, 3);
        for (int j = 0; j < areaCount; j++) {
          final double[] data = new double[labelCount];
          for (int ii = 0; ii < labelCount; ii++) {
            final Item curItem = d.getItem((ii + 1) + "," + (j + 1));
            if (curItem instanceof Value) {
              data[ii] = (Double) ((Value) curItem).getValue();
            } else {
```

```
                data[ii] = 0.0;
              }
            }
            if (tColors != null) {
              layer.addDataSet(data, (int) Long.parseLong(tColors[j], 16),
AnalysisRenderingToolbox.getItemText(dim2.getItem(j), root));
            } else {
              layer.addDataSet(data, AnalysisRenderingToolbox.getHexaColor(j),
AnalysisRenderingToolbox.getItemText(dim2.getItem(j), root));
            }
          }


          // Get filename
          final String imgurl =
root.currentEnvironment().toolkit().getStringFromID(root.currentEnvironment().toolkit
().generateID()) + ".jpg";
          // Generate chart
          c.makeChart(AnalysisRenderingToolbox.getGenerationFolderWrite(megaContext,
root) + imgurl);
          // append html
          AnalysisRenderingToolbox.getShowHideStart(root, (StringBuffer) document,
(View) i, d, megaContext);
          ((StringBuffer) document).append("<p><img src=\"" +
AnalysisRenderingToolbox.getGenerationFolderRead(megaContext, root, imgurl) +
"\"/></p>");
          AnalysisRenderingToolbox.getShowHideEnd((StringBuffer) document, (View) i);
          return true;
        }
        return false;
      }
      return false;
    }
}
```

PDF Renderer Code

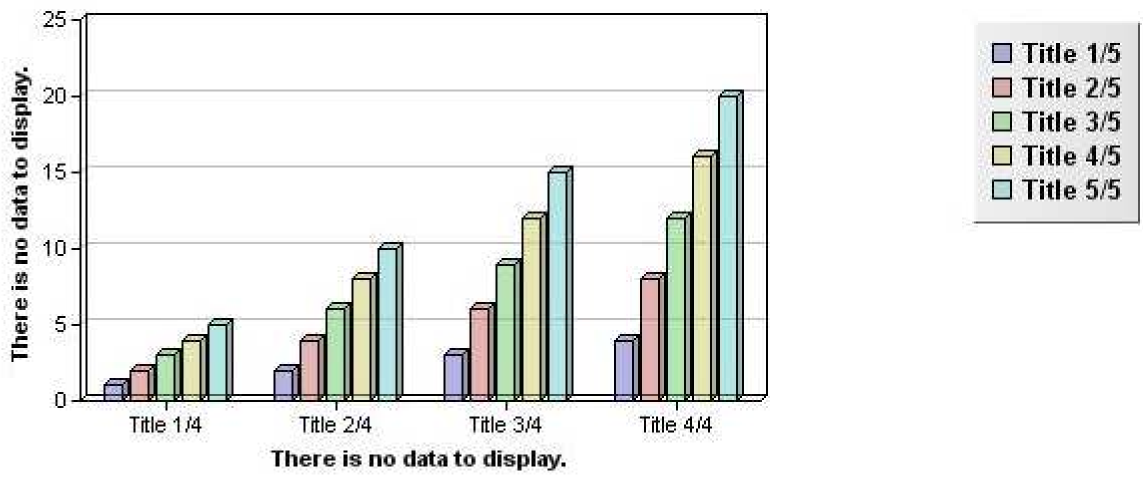
```
import java.io.File;
import ChartDirector.BarLayer;
import ChartDirector.Chart;
import ChartDirector.LegendBox;
import ChartDirector.XYChart;
import aspose.pdf.Image;
import aspose.pdf.Pdf;
import aspose.pdf.Section;
import com.mega.modeling.analysis.AnalysisRenderingToolbox;
import com.mega.modeling.analysis.ViewRenderer;
import com.mega.modeling.analysis.content.Dataset;
```

```java
import com.mega.modeling.analysis.content.Dimension;
import com.mega.modeling.analysis.content.Item;
import com.mega.modeling.analysis.content.ReportContent;
import com.mega.modeling.analysis.content.Value;
import com.mega.modeling.analysis.content.View;
import com.mega.modeling.api.MegaCOMObject;
import com.mega.modeling.api.MegaRoot;

/**
 * Renderer for bar chart in PDF
 * Accepted dimensionalities: 2
 * @author NLE
 */
public class barchartPDFRenderer implements ViewRenderer {
  @Override
  public boolean Generate(final Object document, final ReportContent reportContent,
final Item i, final MegaCOMObject megaContext, final Object userData, final MegaRoot
root) {
    if (i instanceof View) {
      final Dataset d = reportContent.getDataset(((View) i).getDatasetId());
      if (d.getDimensionCount() == 2) {
        AnalysisRenderingToolbox.setChartDirectorLicense(root);

        // The data for the chart
        final Dimension dim1 = d.getDimension(0);
        final Dimension dim2 = d.getDimension(1);

        // The labels & data for the chart
        final int labelCount = dim1.getItemCount();
        final int areaCount = dim2.getItemCount();

        final String[] labels = new String[labelCount];
        for (int ii = 0; ii < labelCount; ii++) {
          labels[ii] = AnalysisRenderingToolbox.getItemText(dim1.getItem(ii), root);
        }

        // Create a XYChart object of size 400 x 240 pixels
        final XYChart c = new XYChart(600, 260);

        // Add a title to the y-axis
c.yAxis().setTitle(AnalysisRenderingToolbox.getCodeTemplate(dim2.getCodeTemplateID(),
root));

        // Add a title to the x-axis
```

```java
c.xAxis().setTitle(AnalysisRenderingToolbox.getCodeTemplate(dim1.getCodeTemplateID(),
root));

        // Set the x axis labels
        c.xAxis().setLabels(labels);

        // Set the plot area and size.
        c.setPlotArea(40, 30, 340, 190);

        // Add a legend box at top right corner using 10 pts Arial Bold
        // font. Set the background to silver, with 1 pixel 3D border effect.
        final LegendBox b = c.addLegend(570, 30, true, "Arial Bold", 10);
        b.setAlignment(Chart.TopRight);
        b.setBackground(Chart.silverColor(), Chart.Transparent, 1);

        //colors
        String sColors = ((View) i).getParameter("colors");
        String[] tColors = null;
        if ((sColors != null) && (sColors.length() > 1)) {
          tColors = sColors.split(",");
        }

        // Add a multi-bar layer with data sets and 3 pixels 3D depth
        final BarLayer layer = c.addBarLayer2(Chart.Side, 3);
        for (int j = 0; j < areaCount; j++) {
          final double[] data = new double[labelCount];
          for (int ii = 0; ii < labelCount; ii++) {
            final Item curItem = d.getItem((ii + 1) + "," + (j + 1));
            if (curItem instanceof Value) {
              data[ii] = (Double) ((Value) curItem).getValue();
            } else {
              data[ii] = 0.0;
            }
          }
          if (tColors != null) {
            layer.addDataSet(data, (int) Long.parseLong(tColors[j], 16),
AnalysisRenderingToolbox.getItemText(dim2.getItem(j), root));
          } else {
            layer.addDataSet(data, AnalysisRenderingToolbox.getHexaColor(j),
AnalysisRenderingToolbox.getItemText(dim2.getItem(j), root));
          }
        }

        // Get filename
```

```
        final String imgurl =
root.currentEnvironment().toolkit().getStringFromID(root.currentEnvironment().toolkit
().generateID()) + ".jpg";
        // Generate chart
        File f = new
File(AnalysisRenderingToolbox.getGenerationFolderWrite(megaContext, root) + imgurl);
        f.deleteOnExit();
        c.makeChart(f.getAbsolutePath());
        // append PDF
        Section sec1 = ((Pdf) document).getSections().add();
        sec1.setIsNewPage(false);
        //Create an image object in the section
        Image img1 = new Image(sec1);

img1.getImageInfo().setTitle(AnalysisRenderingToolbox.getCodeTemplate(d.getCodeTempla
teID(), root));
        img1.getImageInfo().setFixWidth(400.0f);
        //Add image object into the Paragraphs collection of the section
        sec1.getParagraphs().add(img1);
        //Set the path of image file
        img1.getImageInfo().setFile(f.getAbsolutePath());
        return true;
      }
      return false;
    }
    return false;
  }
}
```

Example result

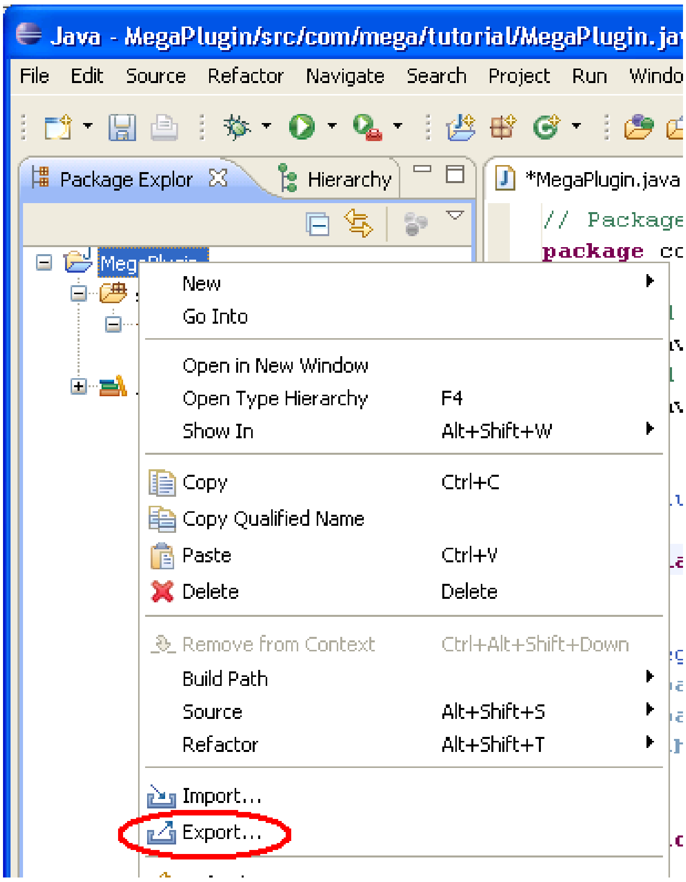


Using your implementation from the MEGA Renderer Macro

Compile

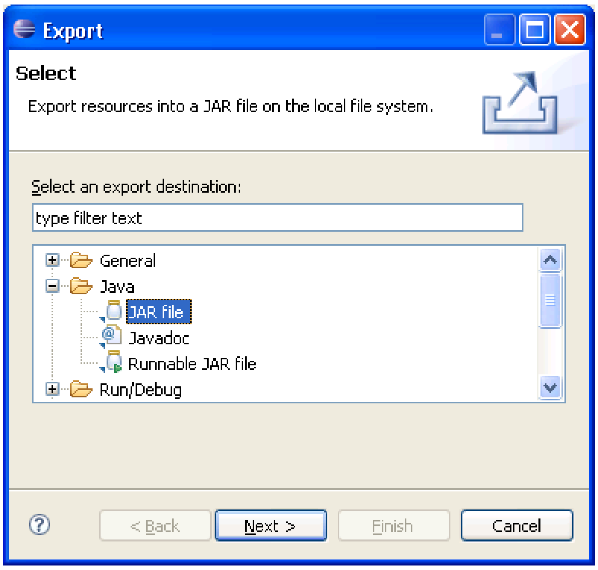
In order to use your Java Report Renderer it must be exported in a .jar in the java/lib directory of your MEGA installation.

Compilation of the Java component in the form of a JAR file is via the "Export" menu of the Java project:



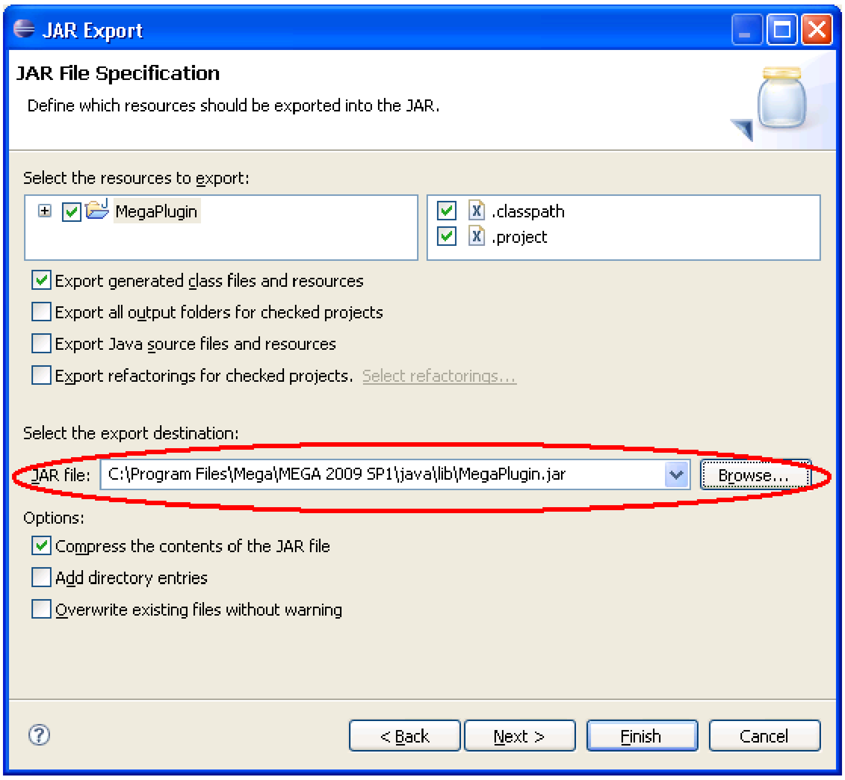
A JAR can contain as many Java Renderer implementing classes as you wish.

"JAR File" export in the Java directory should be selected:



Indicate the location of the JAR file to be generated and click "Finish":

The JAR file **must** be generated (or copied after generation) in the "java\lib" directory of the MEGA installation site.

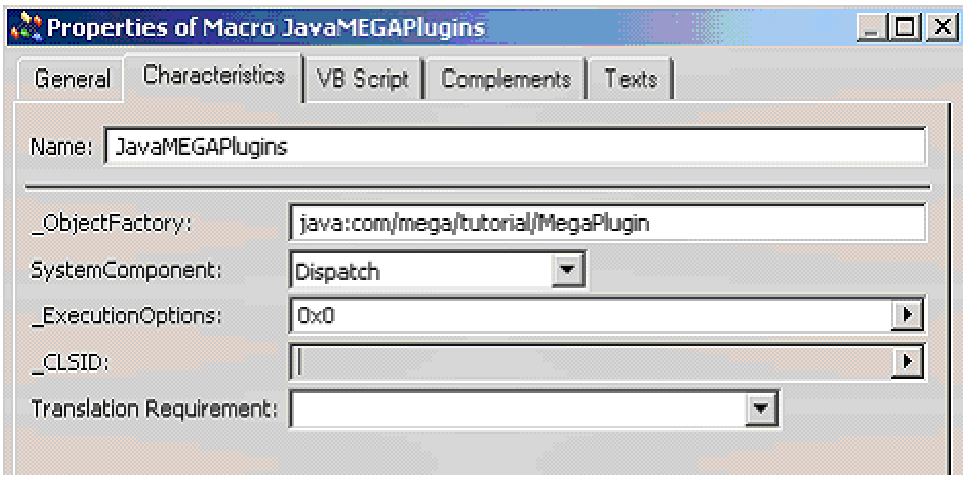


The name of the JAR file itself is not significant; you should use a name that makes sense in your project.

Configuring the macro

Once you have added the JAR file to the “java\lib” directory, restart Mega and edit the properties of your renderer macro in order to reference your Java class.

In the macro properties dialog box, assign the "Dispatch" value to the "SystemComponent" attribute, then specify the "_ObjectFactory" attribute using the renderer Java class. For exemple, the value "java:com/mega/tutorial/MegaPlugin" identifies the "MegaPlugin" class of the "com.mega.tutorial" package.



The VB Script of the macro should be kept empty.

Writing Java report chapters

This article explains how to write report chapters in Java, the new architecture and the differences with VBS analyses.

INTRODUCTION TO JAVA REPORT CHAPTERS

Generalities

A report chapter can be written in Java.

It is highly recommended to write new reports in this language in order to benefit from the platform improvements and also to better handle PDF/RTF document generation. The old report platform will not be improved whereas the Java platform will be.

Compared to VB Script reports, Java reports are written differently as described in this article.

Both use the same metamodel described in the product documentation.

Data and view separation

VB Script report chapter macros generate HTML.

Java report chapter macros generate an object structure describing datasets (report data) and the type of view to apply to these datasets.

For more details on this structure, see the "Java Report Data Structure". technical article.

Conversion from this object structure to a report output (HTML or PDF for example) is handled by the report engine using specific renderers. For more details on renderers see the "Writing Java report renderers" technical article.

This new architecture allows for better management of different output formats and more flexibility and modularity. It is therefore highly recommended to write new report chapter macros in Java, using the method described hereafter.

PREREQUISITES

Report modeling

This technical article does not cover modeling of reports, report templates, report parameters and report chapters.

Modeling is the same as for VB Script reports. You should refer to product documentation on this subject.

Setup a Java development environment

A Java report chapter macro is a Java Plug-in which uses Mega APIs.

You should first refer to the "*Create a Java component*" chapter (*All about starting with APIs* Technical Article) to set up a Java development environment adapted to HOPEX.

Referencing jars and javadoc

Following the method described in "Create a Java component" you should reference the following jars in your development environment:

- mj_toolkit.jar
- mj_api.jar
- mj_anls.jar

To allow for contextual help in Eclipse, you should also reference the corresponding javadocs:

- mj_api.doc.zip
- mj_anls.doc.zip

WRITING THE MACRO IMPLEMENTATION

Interfaces to implement

Report chapters

A Java report chapter must implement one of the following interfaces:

- `com.mega.modeling.analysis.AnalysisReport`
- `com.mega.modeling.analysis.AnalysisReportWithContext`

The `getReportContent` method is the only method required to implement these interfaces. Its parameters are:

- a Mega Root,
- a Map of parameter Lists referenced by the HexaIdAbs of the "Analysis Parameter" repository object,
- an optional userData object,
- and in the case of `AnalysisReportWithContext` an extra Analysis object which provides contextual information.

It produces the report content in the form of a ReportContent object. This is an instance of the ReportContent Java class, to which all data and view information is given in order to pass it to the analysis engine and its renderers.

More information on ReportContent objects is available in the "Java analysis data structure" technical article and in the engine Javadoc.

This typically results in either of the following structures:

```
public class MyReport implements AnalysisReport {
  @Override
  public ReportContent getReportContent(final MegaRoot root, final Map<String, List<AnalysisParameter>> parameters, final Object userData) {
    …
  }
}
```

```
public class MyReport implements AnalysisReportWithContext {
  @Override
```

```
  public ReportContent getReportContent(final MegaRoot root, final Map<String,
List<AnalysisParameter>> parameters, final Analysis analysis, final Object
userData) {
    …
  }
}
```

Callbacks

Callback calls allow for user interactivity in tables and trees, by allowing the execution of a Callback function when the user clicks on a cell. The callback function subsequently updates the cell content.

The Java implementation of the macro that handles callback calls must implement the `com.mega.modeling.analysis.AnalysisCallback` interface.

This can be the same macro and Java class as the report chapter itself or a different macro.

The `Callback` method is the only method required to implement the AnalysisCallback interface. Its parameters are:

- a Mega Root,
- the HTML content of the cell that was clicked,
- MegaCollections of the line and column objects of the cell,
- an optional userData.

It produces the new content to be rendered in the clicked cell, in the form of an HTML string.

For example:

```
public class MyReportCallback implements AnalysisCallback {
  @Override
  public String Callback(final MegaRoot root, final String HTMLCellContent,
final MegaCollection ColumnMegaObjects, final MegaCollection LineMegaObjects,
final Object userData) {
    …
  }
}
```

Implementing the macro

General steps

The implementation of the getReportContent function follows the following general steps:

- Create a new ReportContent object that will contain all report data and views:

```
final ReportContent reportContent = new ReportContent("");
```

- Create one or more Datasets (Java objects that contain all the data to be rendered) and add them to the ReportContent, getting the ID of the dataset for future use:

```
final Dataset d = new Dataset("");
final int datasetID = reportContent.addDataset(d);
```

- Create one or more Dimensions (the equivalent of x, y, etc. axis in a diagram) and add them to the Dataset(s):

```
final Dimension dim = new Dimension("");
d.addDimension(dim);
```

- Create one or more items and add them to the Dimension(s). This is the equivalent of row or column headers in a table.

```
dim.addItem(new Text("Some text…", false));
```

- Create one or more items and add them to the Dataset(s):

```
d.addItem(new Value((double) n), "1");
```

- Create one or more Views (or Texts or MegaObjectProperties), using the ID of the dataset they should represent, and add them to the ReportContent. A view links a dataset to a renderer in order to define where and how the dataset should be rendered.

```
final View v = new View(datasetID);
reportContent.addView(v);
```

- Add one or more renderers to the View(s) to specify how the view dataset should be shown (here, as a table):

```
v.addRenderer(AnalysisReportToolbox.rTable);
```

- Return the now complete ReportContent:

```
return reportContent;
```

More information on the data structure of ReportContent, Dataset, Dimension, View, Item, etc. is available in the "Java report data structure" technical article and in the Javadoc engine (HOPEX Customization (Windows) > Using APIs > JavaDoc).

Example

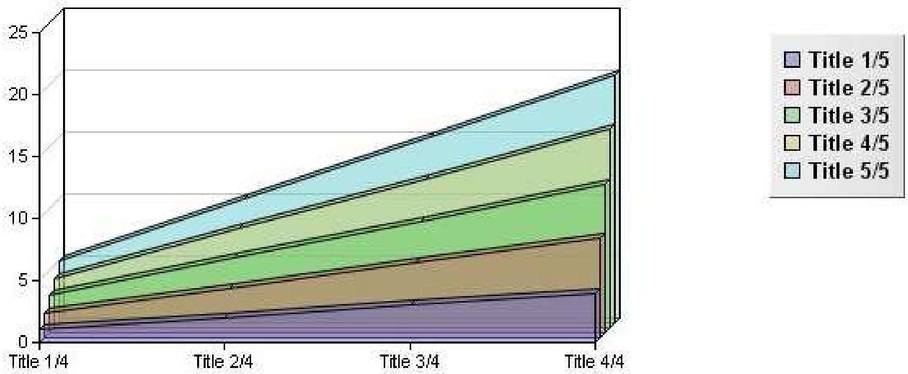
A basic working example is as follows:

```
import java.util.List;
import java.util.Map;

import com.mega.modeling.analysis.AnalysisParameter;
import com.mega.modeling.analysis.AnalysisReport;
import com.mega.modeling.analysis.AnalysisReportToolbox;
import com.mega.modeling.analysis.content.Dataset;
import com.mega.modeling.analysis.content.Dimension;
import com.mega.modeling.analysis.content.ReportContent;
import com.mega.modeling.analysis.content.Text;
import com.mega.modeling.analysis.content.Value;
import com.mega.modeling.analysis.content.View;
import com.mega.modeling.api.MegaRoot;

/**
 * This is a basic demonstration report.
 * @author NLE
 */

public class MyReport implements AnalysisReport {
  public ReportContent getReportContent(final MegaRoot root, final Map<String,
List<AnalysisParameter>> parameters, final Object userData) {
    final ReportContent reportContent = new ReportContent("");

    // Creating a 2D Dataset and its dimensions
    final Dataset d2 = new Dataset("");
    final Dimension dim21 = new Dimension("");
    final Dimension dim22 = new Dimension("");
    // Set the dimension sizes
    // (compulsory only when no Items are added to the dimension)
    dim21.setSize(4);
    dim22.setSize(5);
    // Add the dimensions to the Dataset
    d2.addDimension(dim21);
    d2.addDimension(dim22);

    // Filling in the dataset and its dimensions
    // Arbitrary data is used here
    for (int i = 1; i <= 4; i++) {
      dim21.addItem(new Text("Title " + i + "/4", false));
```

```
      for (int j = 1; j <= 5; j++) {
        d2.addItem(new Value((double) i * j), i + "," + j);
      }
    }
    for (int j = 1; j <= 5; j++) {
      dim22.addItem(new Text("Title " + j + "/5", false));
    }

    // Add the dataset to the report
    final int datasetID = reportContent.addDataset(d2);

    // Add a table and list view of the dataset
    // List will only be used if table cannot be used
    final View v1 = new View(datasetID);
    v1.addRenderer(AnalysisReportToolbox.rTable);
    v1.addRenderer(AnalysisReportToolbox.rList);
    reportContent.addView(v1);

    // Add an area chart view, with the same dataset (not duplicated)
    final View v2 = new View(datasetID);
    v2.addRenderer(AnalysisReportToolbox.rAreaChart);
    reportContent.addView(v2);

    return reportContent;
  }
}
```

This code typically results in the following rendering:

	Title 1/5	Title 2/5	Title 3/5	Title 4/5	Title 5/5
Title 1/4	1.0	2.0	3.0	4.0	5.0
Title 2/4	2.0	4.0	6.0	8.0	10.0
Title 3/4	3.0	6.0	9.0	12.0	15.0
Title 4/4	4.0	8.0	12.0	16.0	20.0



Going further

You should refer to the Javadoc for all possible options and possibilities.

The Javadoc is accessible:

- in HOPEX online documentation (**HOPEX Customization (Windows) > Using APIs > JavaDoc**):
    - Report API
    - MEGA API
    - Toolkit API
    - Workflow Engine API
- contextually in Eclipse if you configured it as suggested in the Prerequisites section
- in your HOPEX installation directory ("java\doc"), in HTML format in a zip file "mj_anls.doc.zip".

You can make use of all MEGA Java APIs (also available in the corresponding "mj_api.doc.zip" javadoc) to handle MegaObjects and MegaCollections.

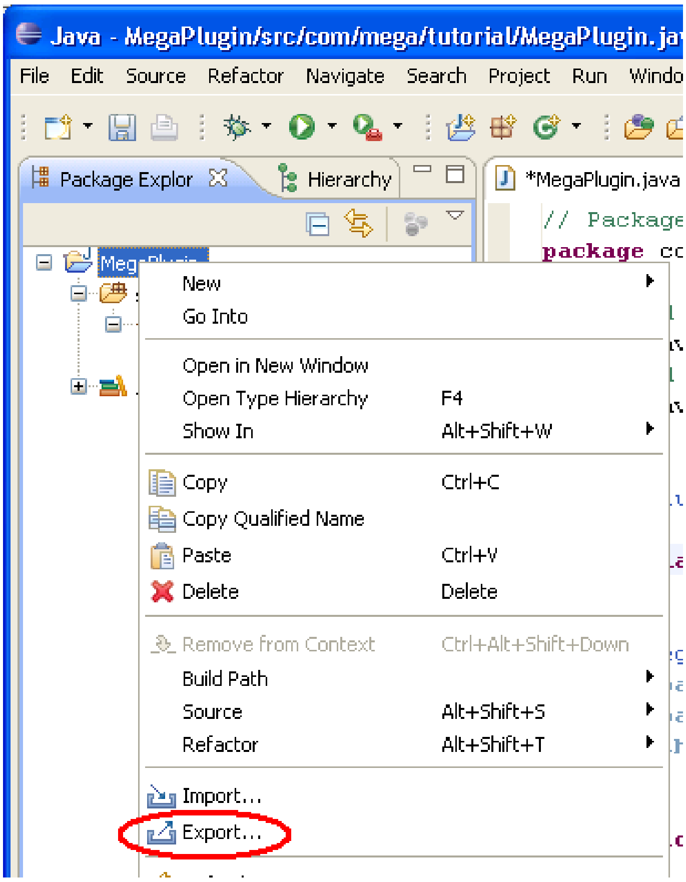
A more complex example is provided at the end of this document.

USING YOUR IMPLEMENTATION FROM THE MEGA REPORT MACRO

Compile

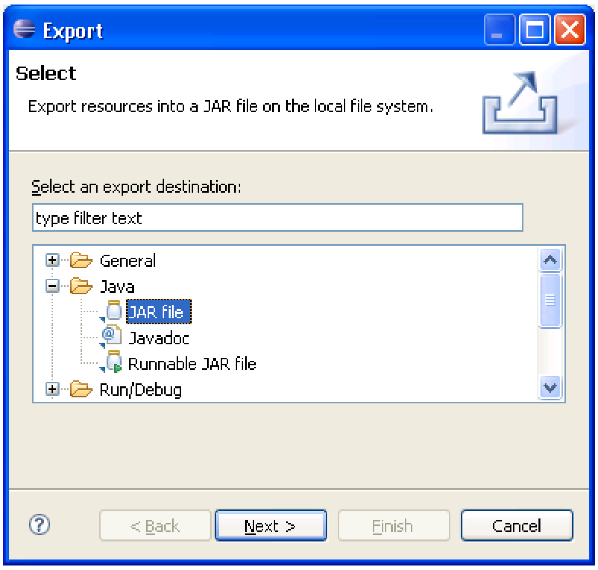
In order to use your Java report chapter, it must be exported in a .jar in the java/lib directory of your HOPEX installation.

Compilation of the Java component in the form of a JAR file is via the "Export" menu of the Java project:



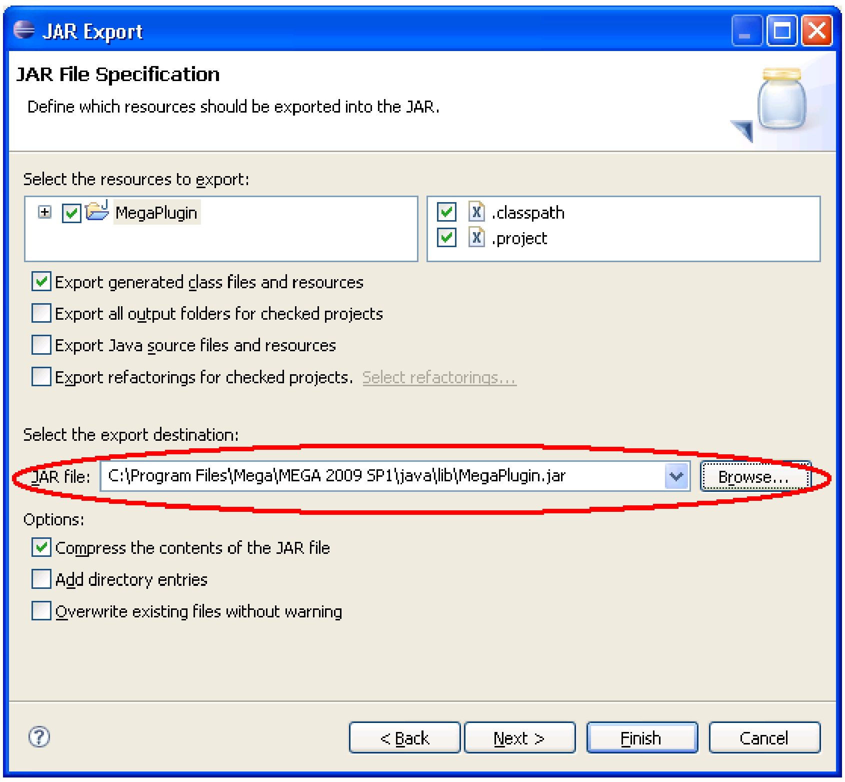
A JAR can contain as many Java report chapter implementing classes as you want.

"JAR File" export in the Java directory should be selected:



Indicate the location of the JAR file to be generated and click **Finish**:

The JAR file **must** be generated (or copied after generation) in the "java\lib" directory of the HOPEX installation site.



The name of the JAR file itself is not significant; you should use a name that makes sense in your project.

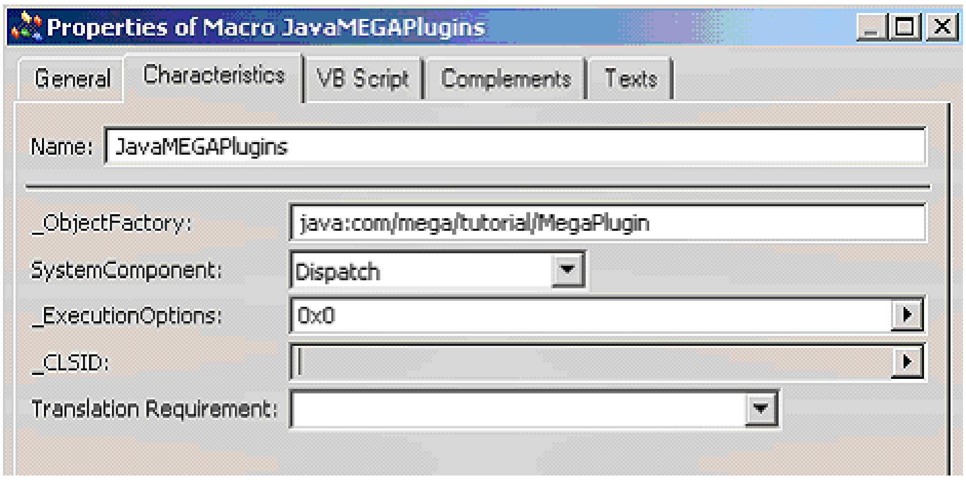
Configure the Macro

Once you have added the JAR file to the “java\lib” directory, restart HOPEX and edit the properties of your report macro in order to reference your Java class.

In the macro properties dialog box:

1. Assign the "Dispatch" value to the "**SystemComponent**" attribute
2. Specify the **_ObjectFactory** attribute using the report Java class.

   For example: the value "java:com/mega/tutorial/MegaPlugin" identifies the "MegaPlugin" class of the "com.mega.tutorial" package.



The VB Script of the macro should be left empty.

CODE EXAMPLE

```java
import java.util.Date;
import java.util.List;
import java.util.Map;

import com.mega.modeling.analysis.AnalysisCallback;
import com.mega.modeling.analysis.AnalysisParameter;
import com.mega.modeling.analysis.AnalysisReport;
import com.mega.modeling.analysis.AnalysisReportToolbox;
import com.mega.modeling.analysis.content.Dataset;
import com.mega.modeling.analysis.content.Dimension;
import com.mega.modeling.analysis.content.Item;
import com.mega.modeling.analysis.content.MegaObjectProperty;
import com.mega.modeling.analysis.content.ReportContent;
import com.mega.modeling.analysis.content.Text;
import com.mega.modeling.analysis.content.Value;
import com.mega.modeling.analysis.content.View;
import com.mega.modeling.api.MegaCollection;
import com.mega.modeling.api.MegaObject;
import com.mega.modeling.api.MegaRoot;
import com.mega.modeling.api.MegaAttribute.OutputFormat;
import com.mega.toolkit.errmngt.ErrorLogFormater;

/**
 * This is a demonstration report.
 * @author NLE
 */

public class MyReport implements AnalysisReport, AnalysisCallback {
  @Override
  public ReportContent getReportContent(final MegaRoot root, final Map<String,
List<AnalysisParameter>> parameters, final Object userData) {
    // Error Management exemple
    final ErrorLogFormater err = new ErrorLogFormater();
    err.openSession(root);
    // Do not forget to update this line
    err.addSessionInfo("Component", "(Java) New Analysis Engine: TestReport:
getReportContent");

    // Initialize the report content
    final ReportContent reportContent = new ReportContent("");
```

```java
try {
```

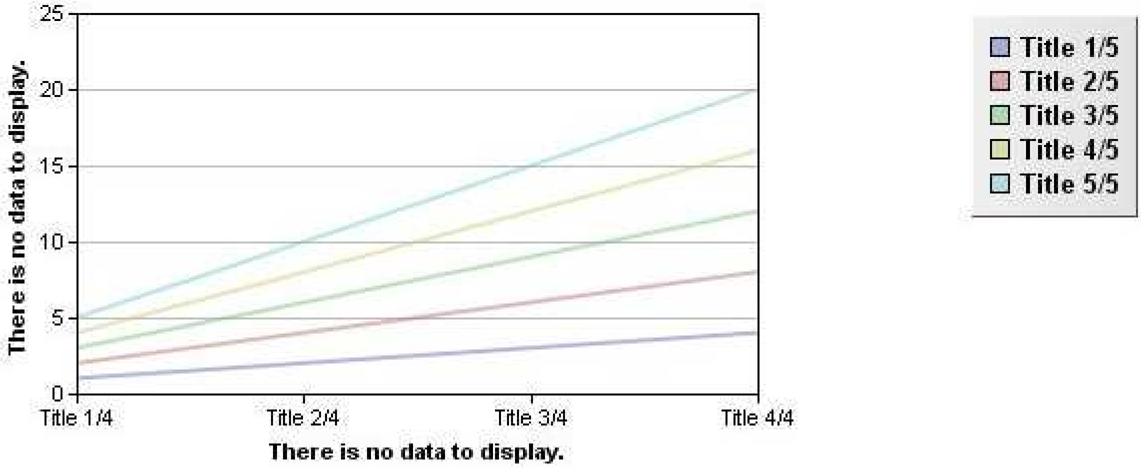
Demo 1: Charts using 2D datasets

```java
final Dataset d2 = new Dataset("~LshNx7Mw6zE0[No Data To Display]");
final Dimension dim21 = new Dimension("~LshNx7Mw6zE0[No Data To Display]");
final Dimension dim22 = new Dimension("~LshNx7Mw6zE0[No Data To Display]");
dim21.setSize(4);
dim22.setSize(5);
d2.addDimension(dim21);
d2.addDimension(dim22);

// Filling in the dataset (here with arbitrary data)
for (int i = 1; i <= 4; i++) {
  dim21.addItem(new Text("Title " + i + "/4", false));
  for (int j = 1; j <= 5; j++) {
    d2.addItem(new Value((double) i * j), i + "," + j);
  }
}
for (int j = 1; j <= 5; j++) {
  dim22.addItem(new Text("Title " + j + "/5", false));
}

// Add the Dataset to the report
final int datasetID = reportContent.addDataset(d2);

// Add the Dataset to many different views
final View v21 = new View(datasetID, true, false);
v21.addRenderer(AnalysisReportToolbox.rAreaChart);
reportContent.addView(v21);

final View v22 = new View(datasetID);
v22.addRenderer(AnalysisReportToolbox.rLineChart);
reportContent.addView(v22);

final View v23 = new View(datasetID);
v23.addRenderer(AnalysisReportToolbox.rBarChart);
reportContent.addView(v23);

final View v24 = new View(datasetID);
```

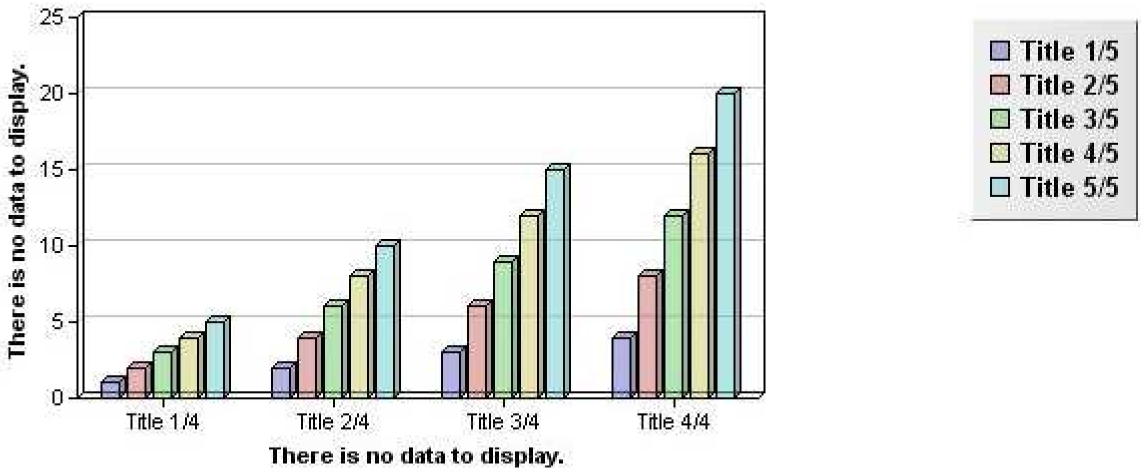
```
v24.addRenderer(AnalysisReportToolbox.rRadarChart);
reportContent.addView(v24);
```

⊞ There is no data to display.

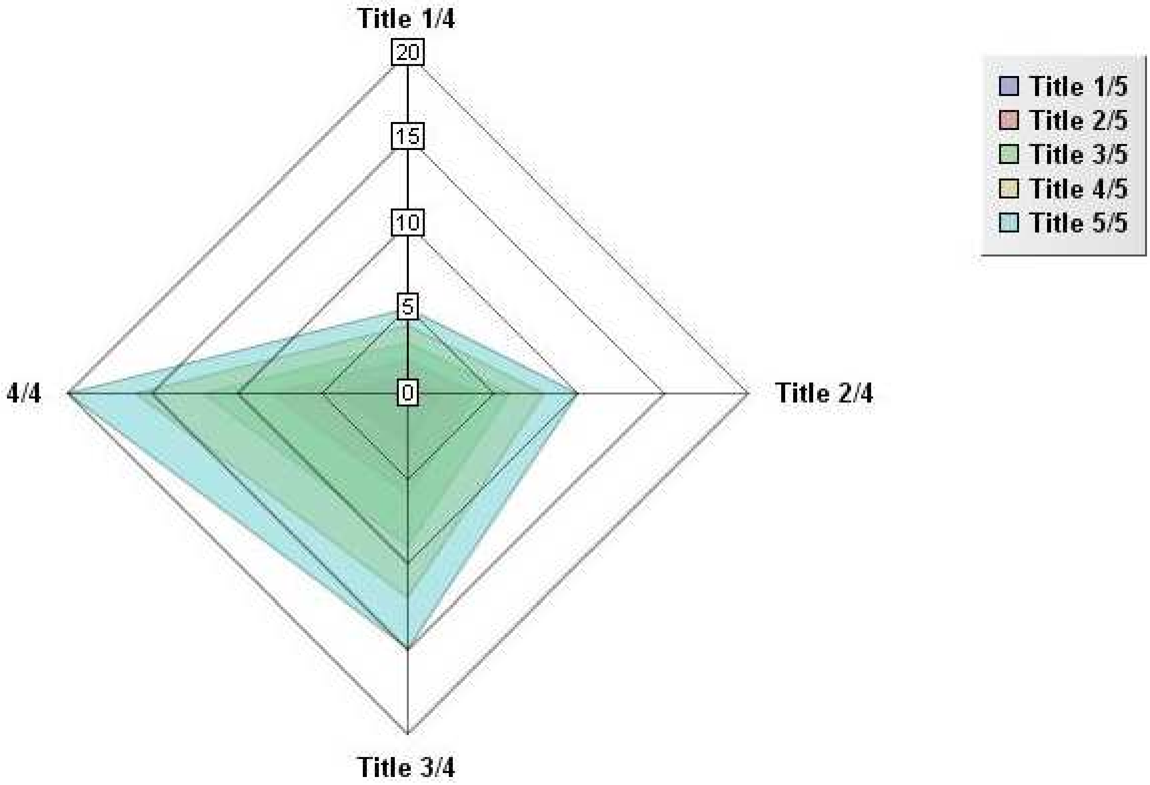
There is no data to display.



There is no data to display.



There is no data to display.



Demo 2: Parameters in tables with vertical headers

```
// Going through parameters
for (final String paramType : parameters.keySet()) {
  reportContent.addText(new Text("" +
root.getObjectFromID(paramType).getAttribute("~Z20000000D60[Short
Name]").getFormated(OutputFormat.html, ""), false, 3));

  // Going through its values
  for (final AnalysisParameter paramValue : parameters.get(paramType)) {
    reportContent.addText(new
Text(paramValue.getParameterObject().getAttribute("~Z20000000D60[Short
Name]").getFormated(OutputFormat.html, ""), false, 4));

    // and the actual individual values
    final Dataset paramDataset = new Dataset("");
    final Dimension dim1 = new Dimension("");
    final Dimension dim2 = new Dimension("");
    dim2.setSize(1);
    final Item title = new Text("Column Title", false);
    // Set the title column header to allow ordering of column
```

```
            title.setOrderable(true);
            dim2.addItem(title);
            int i = 1;
            for (final MegaObject value : paramValue.getValues()) {
              paramDataset.addItem(new MegaObjectProperty(value.megaField(),
"~Z20000000D60[Short Name]"), i + ",1");
              i++;
            }
            dim1.setSize(i - 1);
            paramDataset.addDimension(dim1);
            paramDataset.addDimension(dim2);

            final int paramDatasetID = reportContent.addDataset(paramDataset);

            final View paramView1 = new View(paramDatasetID, true, false);
            paramView1.addRenderer(AnalysisReportToolbox.rVerticalTextTable);
            reportContent.addView(paramView1);

          }
        }
```

Prohibited Technology

Prohibited Technology

Accepted Technology

Accepted Technology

Expected Technology

Expected Technology

Information System

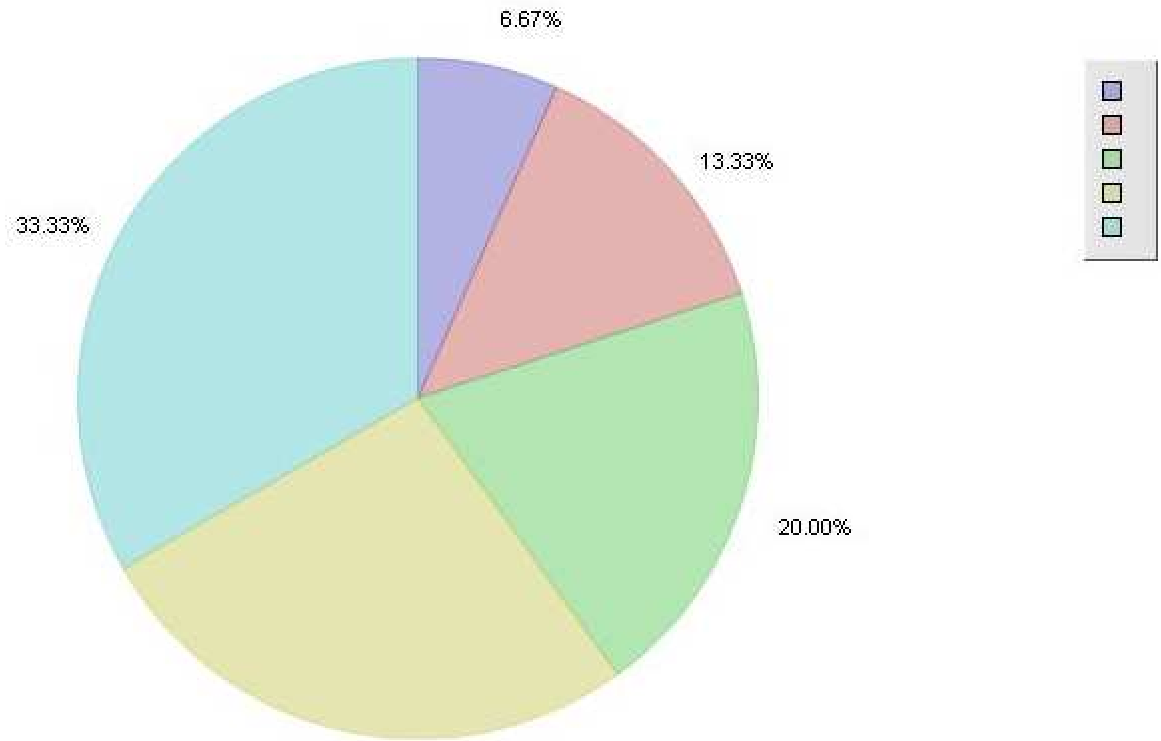
American States

Column Title
Missouri
Alabama
Kansas
Virginia
Oregon
Texas
New Jersey
California

Demo 3: Pie chart

```
final Dataset d1 = new Dataset("");
final Dimension dim11 = new Dimension("");
```

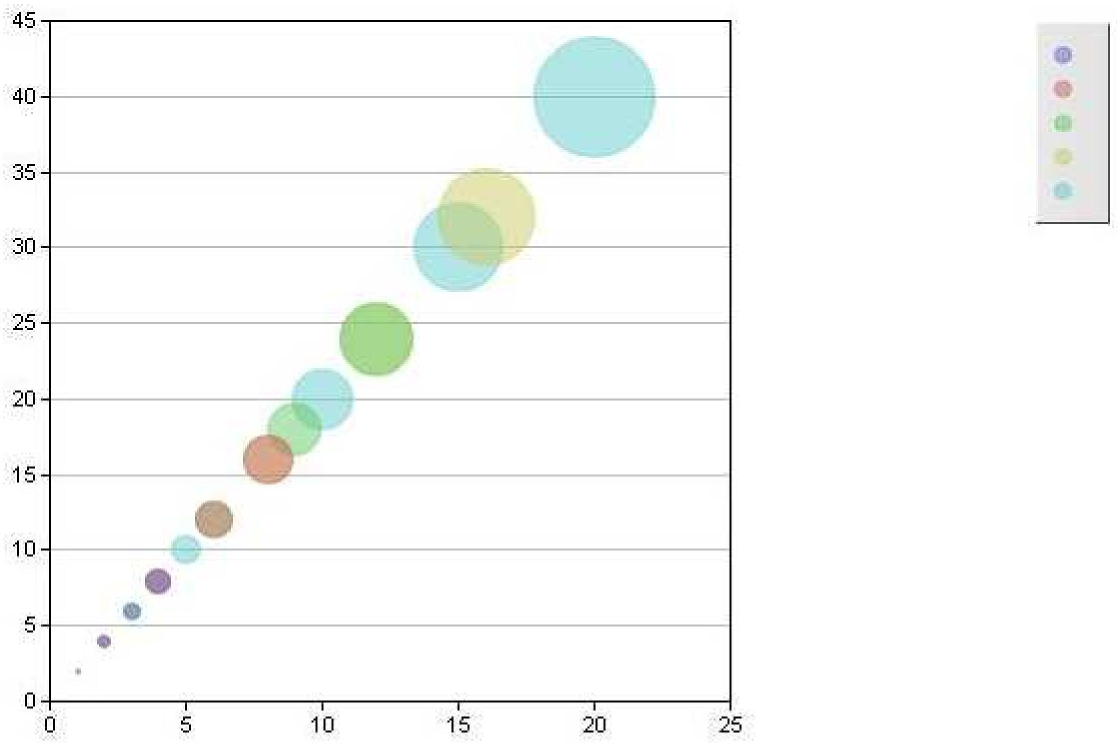
```
dim11.setSize(5);
d1.addDimension(dim11);
for (int i = 1; i <= 5; i++) {
  d1.addItem(new Value((double) i), i + "");
}
final View v1 = new View(reportContent.addDataset(d1));
v1.addRenderer(AnalysisReportToolbox.rPieChart);
reportContent.addView(v1);
```



Demo 4: Bubble chart

```
final Dataset d3 = new Dataset("");
final Dimension dim31 = new Dimension("");
final Dimension dim32 = new Dimension("");
final Dimension dim33 = new Dimension("");
dim31.setSize(5);
dim32.setSize(4);
dim33.setSize(3);
d3.addDimension(dim31);
d3.addDimension(dim32);
d3.addDimension(dim33);
for (int i = 1; i <= 5; i++) {
```

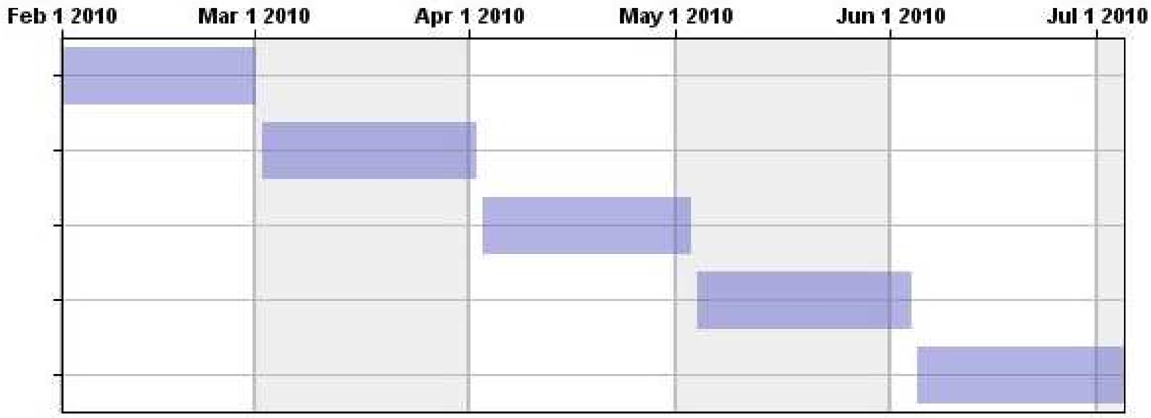
```
for (int j = 1; j <= 4; j++) {
  for (int k = 1; k <= 3; k++) {
    d3.addItem(new Value((double) i * j * k), i + "," + j + "," + k);
  }
}
}
final View v3 = new View(reportContent.addDataset(d3));
v3.addRenderer(AnalysisReportToolbox.rBubbleChart);
reportContent.addView(v3);
```



Demo 5: Simple Gantt chart

```
final Dataset dGantt = new Dataset("");
final Dimension dimGantt = new Dimension("");
dimGantt.setSize(5);
dGantt.addDimension(dimGantt);
for (int i = 1; i <= 5; i++) {
  dGantt.addItem(new Value(new Date(2010 - 1900, i, i), new Date(2010 - 1900, i
+ 1, i)), i + "");
}
final View vGantt = new View(reportContent.addDataset(dGantt));
```

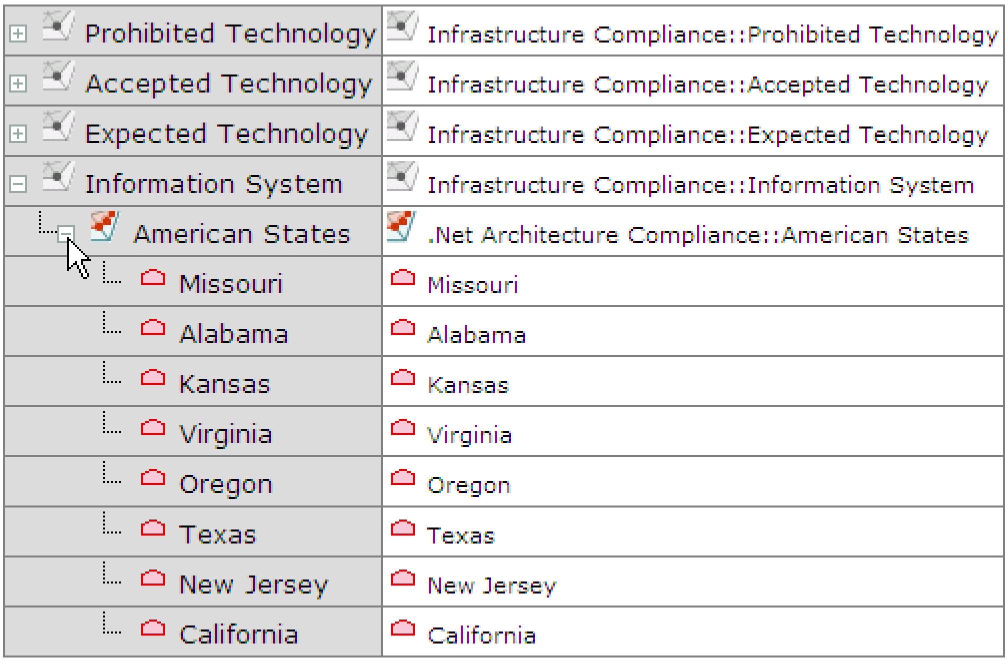
```
vGantt.addRenderer(AnalysisReportToolbox.rGanttChart);
reportContent.addView(vGantt);
```



Demo 6: Tree of parameters

```
final Dataset paramDataset = new Dataset("");
// Callback: set the Macro to be called back, in this exemple you should
// reference the macro referencing this Java class
paramDataset.setCallback("~Jq(Ipv4W4P50[Analysis - Set of Parameters]");
final Dimension dim1 = new Dimension("");
final Dimension dim2 = new Dimension("");
dim2.setSize(1);
int i = 0;
for (final String paramType : parameters.keySet()) {
  dim1.addItem(new
MegaObjectProperty(root.getObjectFromID(paramType).megaField(), "~Z20000000D60[Short
Name]"), 1);
  i++;
  paramDataset.addItem(new
MegaObjectProperty(root.getObjectFromID(paramType).megaField(),
"~210000000900[Name]"), i + ",1");
  // Going through its values
  for (final AnalysisParameter paramValue : parameters.get(paramType)) {
    dim1.addItem(new
MegaObjectProperty(paramValue.getParameterObject().megaField(), "~Z20000000D60[Short
Name]"), 2);
    i++;
    paramDataset.addItem(new
MegaObjectProperty(paramValue.getParameterObject().megaField(),
"~210000000900[Name]"), i + ",1");
    // and the actual individual values!!
```

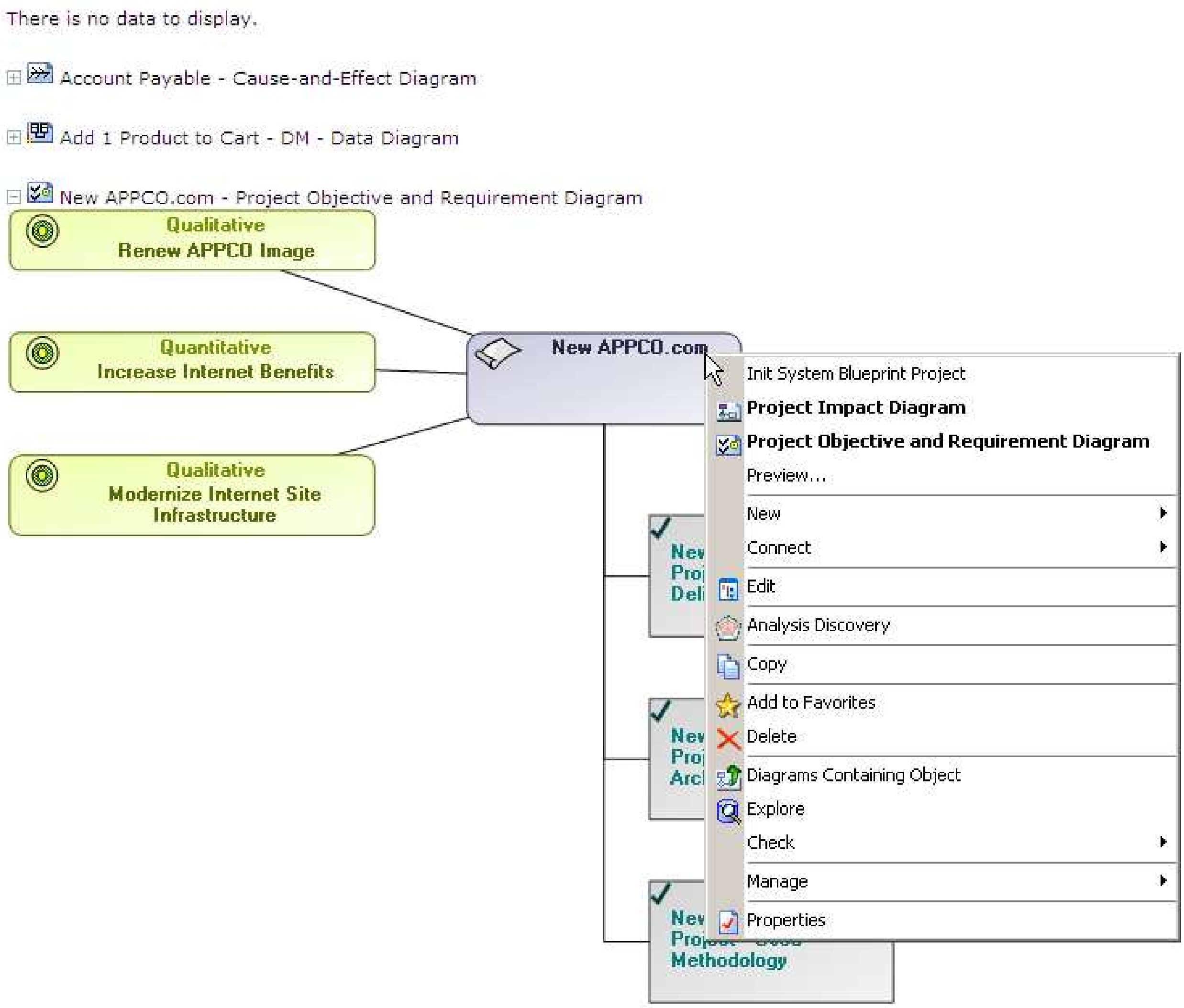
```
            for (final MegaObject value : paramValue.getValues()) {
              dim1.addItem(new MegaObjectProperty(value.megaField(), "~Z20000000D60[Short
Name]"), 3);
              i++;
              paramDataset.addItem(new MegaObjectProperty(value.megaField(),
"~210000000900[Name]"), i + ",1");
            }


          }
        }
        paramDataset.addDimension(dim1);
        paramDataset.addDimension(dim2);
        final View treeView = new View(reportContent.addDataset(paramDataset));
        treeView.addRenderer(AnalysisReportToolbox.rTree);
        treeView.addRenderer(AnalysisReportToolbox.rTable);
        reportContent.addView(treeView);
```



Demo 7: Clickable diagrams

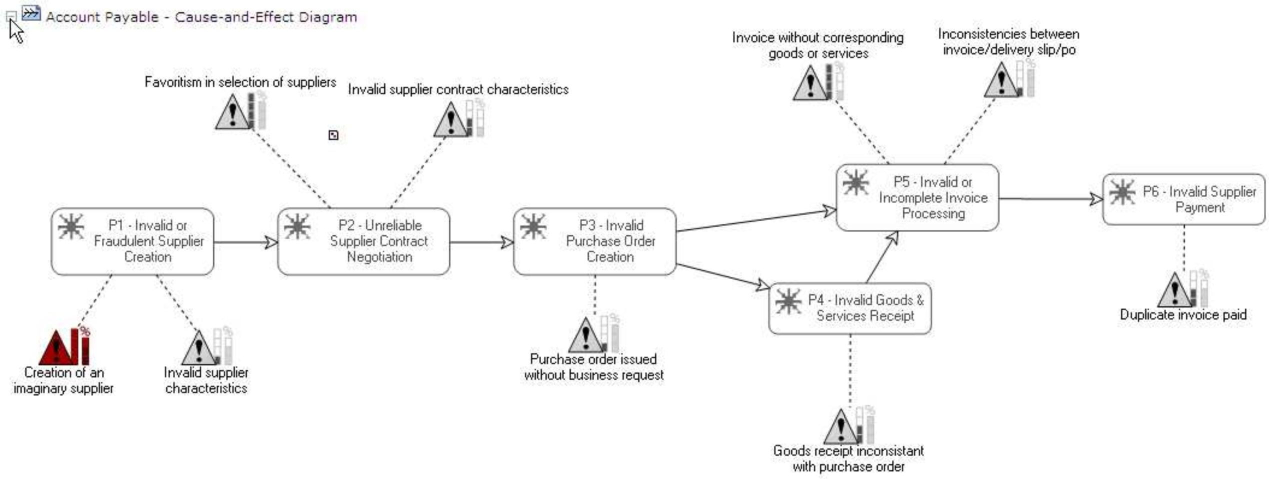
```
        final Dataset dDiags = new Dataset("~LshNx7Mw6zE0[No Data To Display]");
        final Dimension dimD1 = new Dimension("");
        dDiags.addDimension(dimD1);

        // filling in the dataset
```

```java
        dimD1.addItem(new MegaObjectProperty("~uGEJcPMZ4fM0[Account Payable - Cause-and-Effect Diagram]", ""));
        dimD1.addItem(new MegaObjectProperty("~B9QnnNye9940[Add 1 Product to Cart - DM - Data Diagram]", ""));
        dimD1.addItem(new MegaObjectProperty("~vU3v8M(a9L50[New APPCO.com - Project Objective and Requirement Diagram]", ""));
        final int datasetDiagID = reportContent.addDataset(dDiags);

        final View vDiag = new View(datasetDiagID);
        vDiag.addRenderer(AnalysisReportToolbox.rDiagrams);
        reportContent.addView(vDiag);
```



Demo 8: Clickable illustrating diagrams

```java
        final Dataset diDiags = new Dataset("");
        final Dimension dimiD1 = new Dimension("");
        diDiags.addDimension(dimiD1);

        // filling in the dataset with the objects we want to find in the diagram
        // and the color to apply to them
        dimiD1.addItem(new MegaObjectProperty("~4YRLwW5V49o0[Creation of an imaginary
supplier]", ""));
        final int datasetiDiagID = reportContent.addDataset(diDiags);
        diDiags.addItem(new Value((short) 200, (short) 0, (short) 0), "1");

        final View viDiag = new View(datasetiDiagID);
        viDiag.addRenderer(AnalysisReportToolbox.rIllustratingDiagrams);
        reportContent.addView(viDiag);

        // End of error management
      } catch (final Exception e) {
        err.logMessage("Report generation failed:");
        err.logError(e);
        err.closeSession();
      }
      return reportContent;
    }

    @Override
    public String Callback(final MegaRoot root, final String HTMLCellContent, final
MegaCollection ColumnMegaObjects, final MegaCollection LineMegaObjects, final Object
UserData) {
      // Exemple of callback in a table or tree
      return "[TEST] Was called back successfully, was[" + HTMLCellContent + "]";
    }
}
```


Account Payable - Cause-and-Effect Diagram
Favoritism in selection of suppliers
Invalid supplier contract characteristics
Invoice without corresponding goods or services
Inconsistencies between invoice/delivery slip/po
P1 - Invalid or Fraudulent Supplier Creation
P2 - Unreliable Supplier Contract Negotiation
P3 - Invalid Purchase Order Creation
P5 - Invalid or Incomplete Invoice Processing
P6 - Invalid Supplier Payment
P4 - Invalid Goods & Services Receipt
Creation of an imaginary supplier
Invalid supplier characteristics
Purchase order issued without business request
Duplicate invoice paid
Goods receipt inconsistant with purchase order