IMPORTING A MODULE INTO HOPEX

To work with certain HOPEX Solutions, you might need to import modules in HOPEX.

These modules can be uploaded from **HOPEX Application Server** (**HAS**) console.

☺ *If you do not have access to Internet, you can use an off-line mode to upload the module from a local file.*

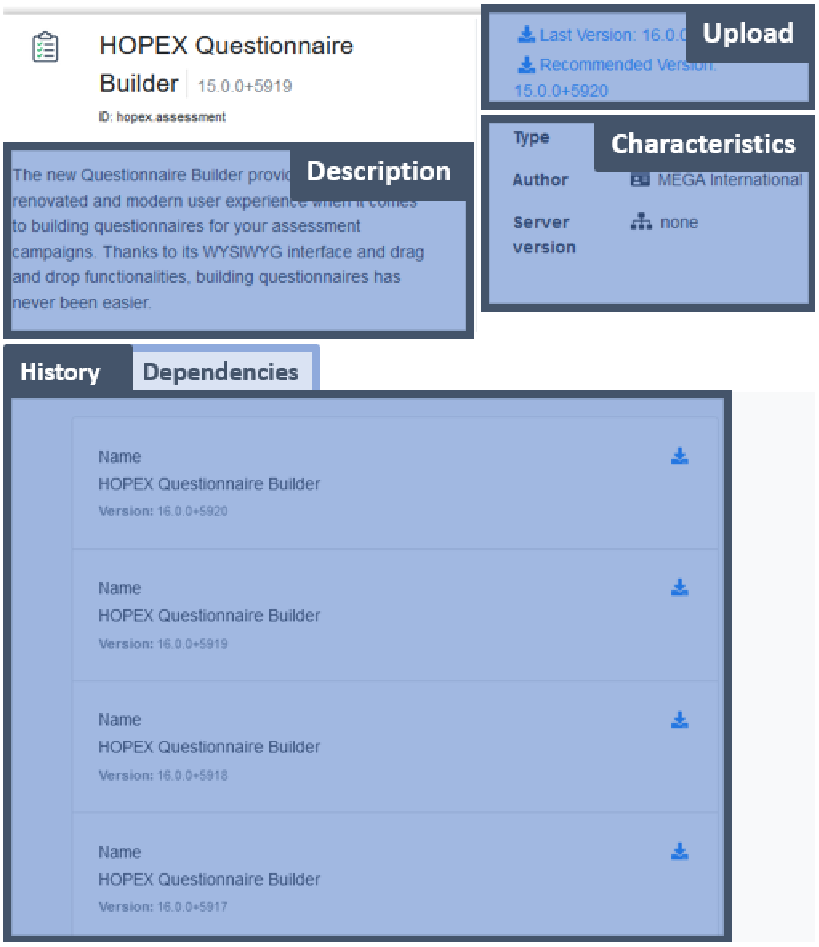
Certain modules modify HOPEX metamodel. You must stop HOPEX Core Back-End module before importing them in HOPEX. Make sure to perform this action when users are not connected.

Module

Module description

A module page shows:

- the module ***description***
- direct access to its ***upload*** (last version and recommended version)
- its characteristics: type, author, server version
- two tabs to access:
  - the module version ***history***
  - its ***dependencies***: some modules require prior installation of other modules


HOPEX Questionnaire Builder 15.0.0+5919

ID: hopex.assessment

The new Questionnaire Builder provi renovated and modern user experien to building questionnaires for your assessment campaigns. Thanks to its WYSIWYG interface and drag and drop functionalities, building questionnaires has never been easier.

Last Version: 16.0.0

Recommended Version: 15.0.0+5920

Type	
Author	MEGA International
Server version	none

History | Dependencies

Name
HOPEX Questionnaire Builder
Version: 16.0.0+5920

Name
HOPEX Questionnaire Builder
Version: 16.0.0+5919

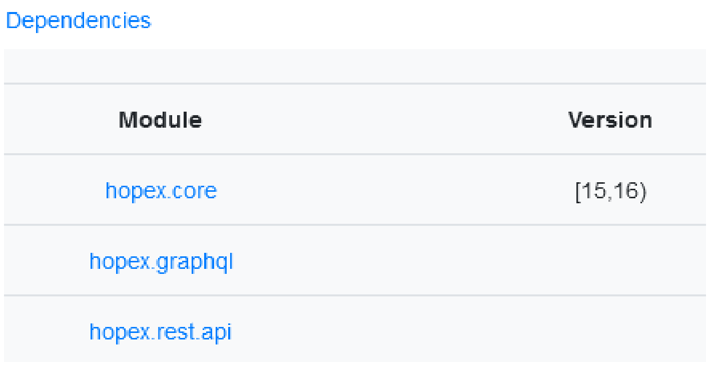
Name
HOPEX Questionnaire Builder
Version: 16.0.0+5918

Name
HOPEX Questionnaire Builder
Version: 16.0.0+5917

Required dependent modules

Certain modules require prior installation of other modules. These modules are listed in the **Dependencies** tab of the module page.

```
E.g. : hopex.core, hopex.graphql, hopex.rest.api.
```



Dependencies

Module	Version
hopex.core	[15,16)
hopex.graphql	
hopex.rest.api	

Version

When necessary, the required version of the dependent module is indicated.

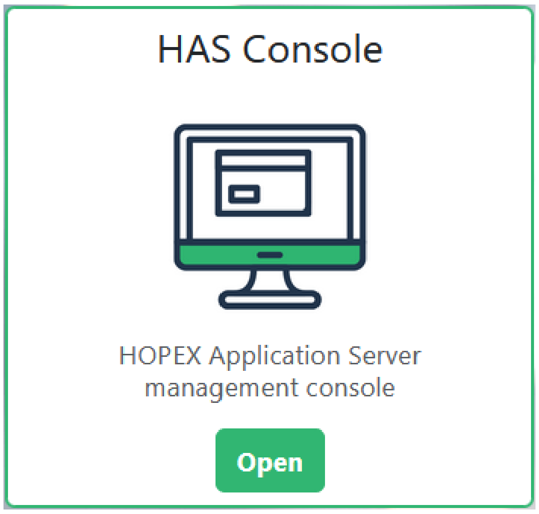
☛ *For details regarding the version range notation, see https://docs.microsoft.com/en-us/nuget/concepts/package-versioning#version-ranges Web site.*

```
E.g.: [15,16) means that all of the versions from 15.0.0 are
authorized. The version 16 and more recent versions are not
authorized.
```

Importing a Module into HOPEX

To import a module in HOPEX:

1. Access **HAS** console:
   - In your web browser enter the https address, then click **Open**.

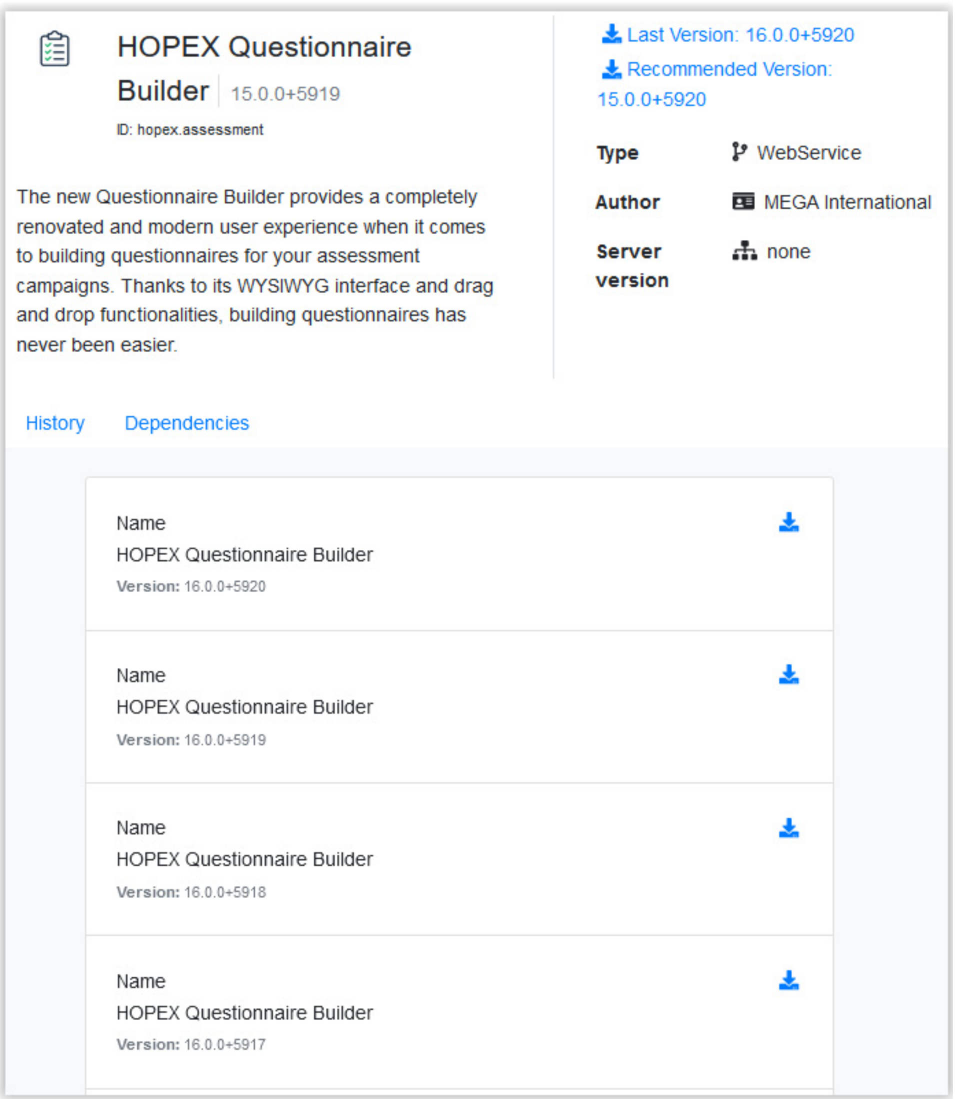


   - Enter the login and password of HAS administrator and click **Sign in**.
2. In the navigation menus of **HAS** console, select **Modules > Module List**
3. In the right pane, click the **Add new** tab.

4. Click the module you want to upload.

   ☺ *If needed, use the* ***Search*** *and filtering (****Type, Tag****) tools to help you find the module.*

   ☛ *If you work off-line, but you have access to the module in a local file (e.g.: via a USB key): click* Upload from file *to upload it.*



HOPEX Questionnaire Builder 15.0.0+5919

ID: hopex.assessment

The new Questionnaire Builder provides a completely renovated and modern user experience when it comes to building questionnaires for your assessment campaigns. Thanks to its WYSIWYG interface and drag and drop functionalities, building questionnaires has never been easier.

Last Version: 16.0.0+5920

Recommended Version: 15.0.0+5920

Type	WebService
Author	MEGA International
Server version	none

History Dependencies

Name
HOPEX Questionnaire Builder
Version: 16.0.0+5920

Name
HOPEX Questionnaire Builder
Version: 16.0.0+5919

Name
HOPEX Questionnaire Builder
Version: 16.0.0+5918

Name
HOPEX Questionnaire Builder
Version: 16.0.0+5917

5. In the module page, check that all the required modules are installed: click **Dependencies**.

   ☛ *See Required dependent modules.*

6. Upload the required version of the module: click .
   In the **Installed** tab, the module is added to the installed module list.

   ☛ *See Module description.*

7. (If needed) Upload other modules.
8. In the **Administration** application (Administration.exe):
   - Connect to the environment.

     ☛ *See Connecting to HOPEX Administration.*
   - Right-click the environment and select **Environment automatic update**.

     To install certain modules, HOPEX Core Back-End module must be stopped.

     *If the **System Repository Update** message is displayed, stop **HOPEX Core Back-End** before proceeding:*

     *In the **HAS console**, **Cluster** navigation menu, **Modules** tab: on **HOPEX Core Back-End** module, click **Plus** ⋮ **> Stop**.*
   - In the **Environment Update Center** window, validate the ***Warning*** and click **Next**.

   The environment is updated with the uploaded module(s).

   ☛ *If you stopped **HOPEX Core Back-End**, restart the module.*

   *In the **HAS console**, **Cluster** navigation menu, **Modules** tab: on **HOPEX Core Back-End** module, click **Plus** ⋮ **> Start**.*

Customization Lifecycle Management

1. Foreword

The modular approach of HOPEX V5 and onward (Vx or AQUILA…) and especially its **HOPEX Application Server Customization** module enable to improve HOPEX customization process:

- extracting/selecting customization performed in the SystemDB
- pushing the customization to production

You can download the modules from the HOPEX Store: https://store.mega.com/modules/details/has.custom

1.1. Vocabulary

1.1.1. Definitions

Terms	Definition
Platform	Is the reference to a server that represents a step in the process of delivery. There are often: DEV, STAGE, PROD as platform
Mode	Includes the following values: • Development (DEV) • Staging (STAGE) • Training • Production (PROD)
Instance	Represents an installation on a given port containing only one environment. Customer have often one instance by platform. An instance has a given mode such as DEV, STAGE, PROD.
Environment	Is the definition of the repositories contained in one HAS instance. This environment is composed of at least: • the SystemDB repository • one data repository
Repository	Represents a database that physically store the content: either technical data or functional data.

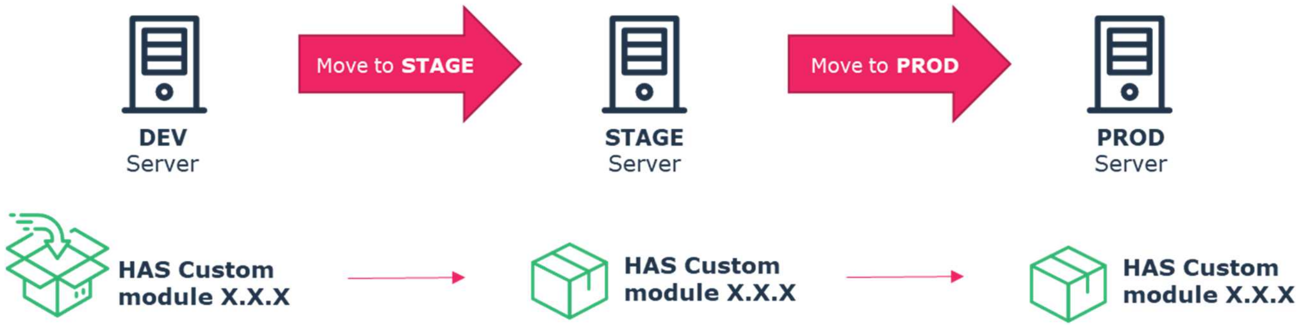
1.1.2. Synonym

Main Term	Synonym or alternative name
STAGE	STAGING, PRE-PROD, UAT, TESTING, INTEGRATION

1.2. From DEV to PROD process

When customizing, you must follow these rules:

- You have a DEV **platform** and **instance** where you can perform your customization.
- You have a STAGE **instance** where you can test the package containing all your customization.
- You have a PROD **instance** where you "ship" a fully tested customization package.



HOPEX is a highly customizable platform: data structure, business logic, UX behavior.

Each customization, at each level, can involve changes in:

- **data**
  - in the SystemDB, or
  - in the data repository

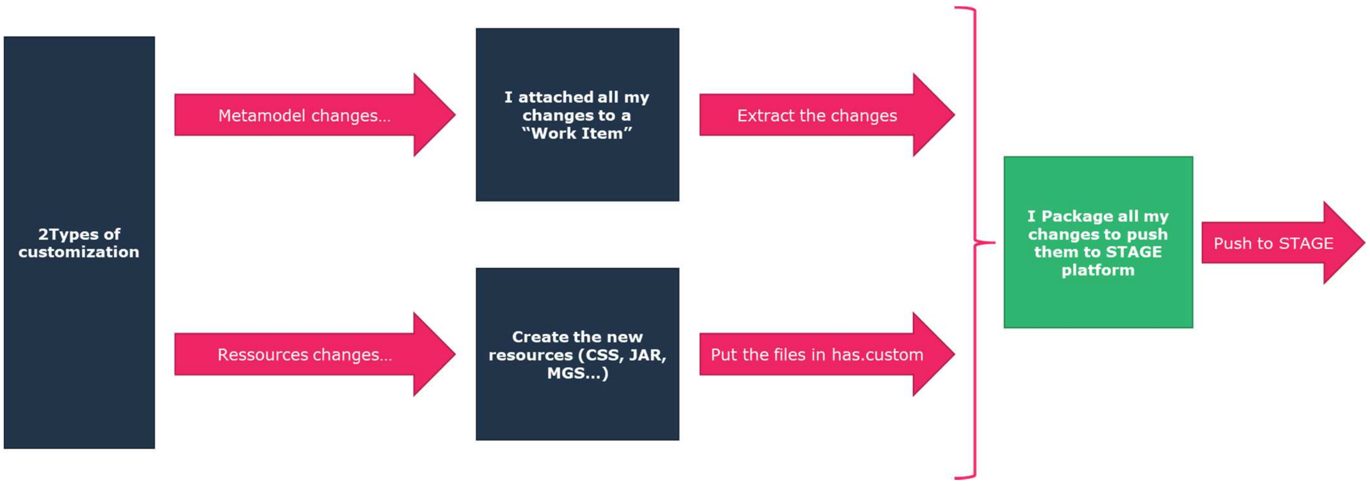
  This data is stored in the database and need to be extracted to be imported in the production platform.
- **files**
  - static resources files (CSS, JS, PNG…)
  - source/compiled code (JAR, DLL…)

  These files need to be stored in the right folders.

The process described here explains how to:

1. Create your customization in the SystemDb.
2. Extract the customization out of the SystemDb.
3. Package all the files (resource files and MGR) in the right location.
4. Push everything to production

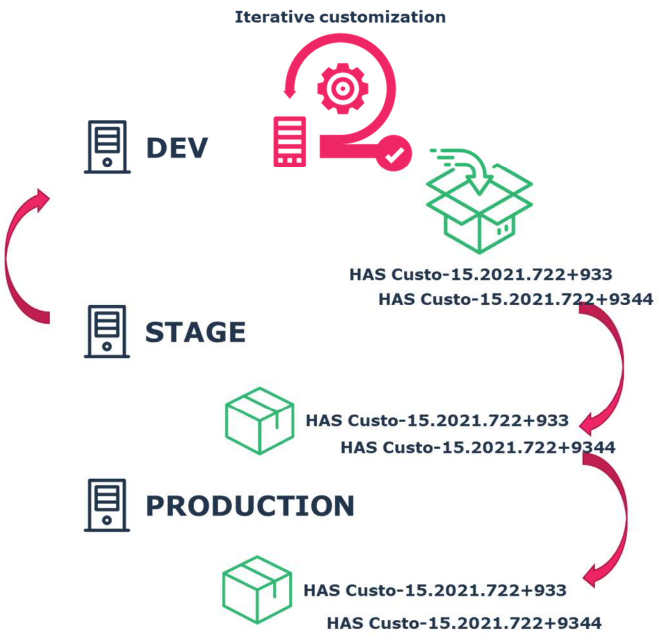
Once the package is created from the DEV instance, this package is used in STAGE (also called UAT / TESTING / PRE-PROD) and at the end in production "as is" without any changes. This allows you to confirm that, what was customized is valid and not altered in the move to PROD process.


2Types of customization
Metamodel changes...
Ressources changes...
I attached all my changes to a "Work Item"
Create the new resources (CSS, JAR, MGS...)
Extract the changes
Put the files in has.custom
I Package all my changes to push them to STAGE platform
Push to STAGE

The process is iterative: it allows you to perform it many times and keep track of the iteration.

From DEV to PROD

- **3 steps :**
  1. **Extract** the customization
     - Repo changes
       - ✓ SystemDB
       - ✓ Repository
     - Files: resources (png, ico, css ...)
     - Code: JAR / DLL
  2. **Package** the customization
     - Generate a unique package containing all the changes since the beginning.
  3. **Push** the customization in next instance
     - Import the module
     - Launch automatic update


Iterative customization
DEV
STAGE
PRODUCTION
HAS Custo-15.2021.722+933
HAS Custo-15.2021.722+9344
HAS Custo-15.2021.722+933
HAS Custo-15.2021.722+9344
HAS Custo-15.2021.722+933
HAS Custo-15.2021.722+9344

1.3. The big picture

This document includes the following chapters:

Customization Capabilities	**Introduction to customization** ➔ **Full customization capability is out of scope for this document.**
Getting Ready for Customization	Mandatory steps prior to customization
Creating Customization (SystemDB)	SystemDb repository customization following best practices
Extracting Customization	Extraction process of valid customization
New/Replace/Override Resource Files	Storing your resources in the appropriate location
Packaging customization	Packaging all your resources (e.g.: MGR, CSS, JAR)
Pushing Customization to STAGE/PROD	Understanding the process of pushing your customization in stage and production

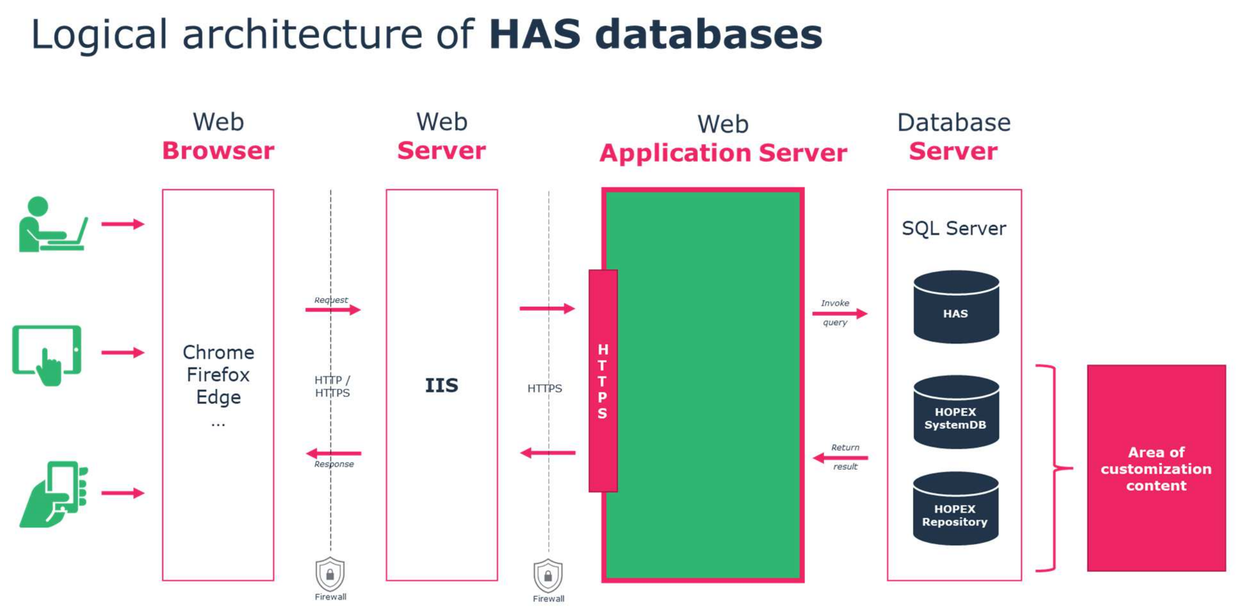
2. Customization Capabilities

As mentioned in the introduction a wide range of elements can be customized. Depending on the customization you need to perform the action may be different. In the end, what is **mandatory** is that, **including history**, all your customization is contained in the "has.custom" module.

2.1. Metamodel, desktops, profiles

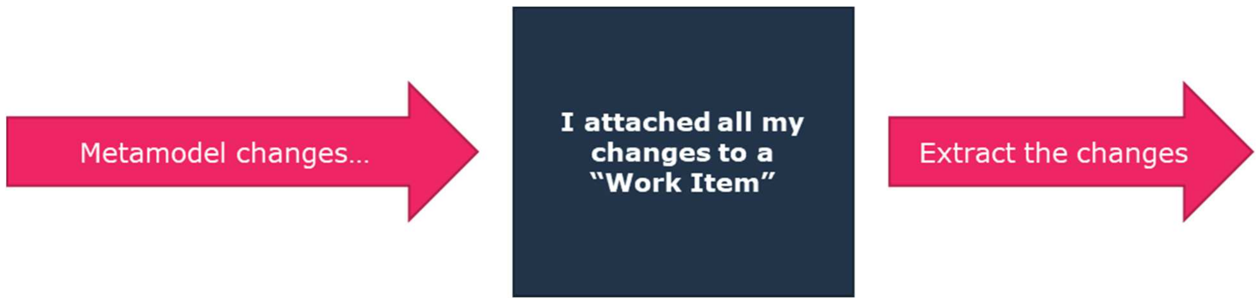
HOPEX contains a minimum of two repositories that can be impacted by your customizations:

- the SystemDB repository: it holds all metamodel, desktops… customized content.
- the Data repository: it contains objects that can be used sometime for customization such as "Keywords" or "Tags".



Each change performed within the HOPEX Studio desktop is stored in the mentioned repositories.

At some point in the lifecycle of your customization you will need to extract these changes to push them in STAGE and then in PRODUCTION.



➔ To generate this extract, see "Creating " chapter.

2.2. Other resources

This section is not exhaustive.

Customizations include:

- Changing shapes in diagrams (*.MGS)
- Changing icon of object (*.ICO)
- Changing default logo or background (*.PNG or *.JPG)
- Changing some style (*.CSS)
- Changing some script (*.JS)
- Changing or adding behaviour (JAR/DLL)

For these customizations most of the changes are not stored in any databases. These are changes made to files of different types.

The rules are as follows:

- You **cannot** edit or replace the **standard** files provided by HOPEX.
- You store all your **new** or **replacing**/**overriding** files into the "has.custom module".

If you respect the folder structure and hierarchy your custom file will be considered instead of the standard one.

Create your new resources and save them in the right location.

Ressources changes...
Create the new resources (CSS, JAR, MGS...)
Put the files in has.custom

3. Getting Ready for Customization

Before starting any customization, you must first:

1. Install the **HOPEX Application Server customization** module.
2. Launch the **Environment automatic update**.
3. Get **NuGet** packages.

3.1. Installing HAS Customization module

You can install the **HOPEX Application Server customization** module:

- Online
  In the HAS Console, add the **HOPEX Application Server customization** module.
- Offline
  Download the latest version of **HOPEX Application Server customization** module from the store https://store.mega.com/modules/details/has.custom.

3.1.1. Online mode

To install HAS Customization module online:

1. Connect to **HOPEX Application Server – Console**.
2. Select **Modules > Module List**.
3. In **Add new** tab, search for **HOPEX Application Server customization** module.
4. Download the required version.

   The **HOPEX Application Server customization** module (the **has.custom** package) is added to the module list.

3.1.2. Offline mode

To install HAS Customization module offline:

1. Access HOPEX Store: https://store.mega.com/modules/details/has.custom.
2. Download the required version of **HOPEX Application Server customization** module.
3. Store it in a location where the offline server will be able to access it.
4. Connect to **HOPEX Application Server – Console**.
5. Select **Modules > Module List**.
6. In **Add new** tab, click **Upload from file**.

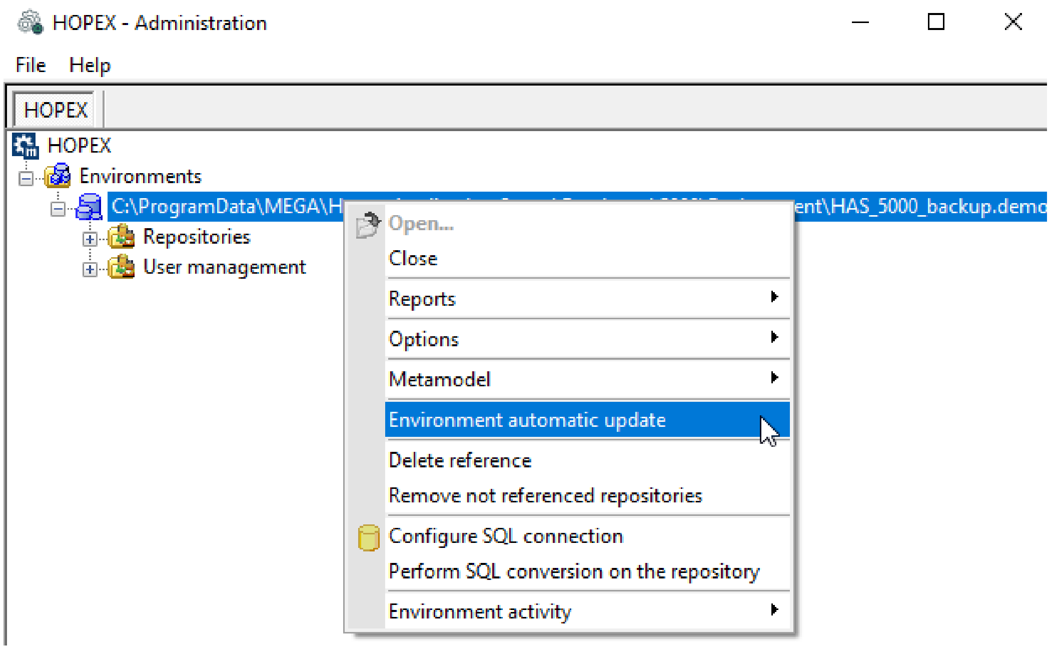
The **HOPEX Application Server customization** module (the **has.custom** package) is added to the module list.

3.2. Launching automatic update

Once the **HOPEX Application Server customization** module is deployed in HAS, you must launch the environment automatic update.

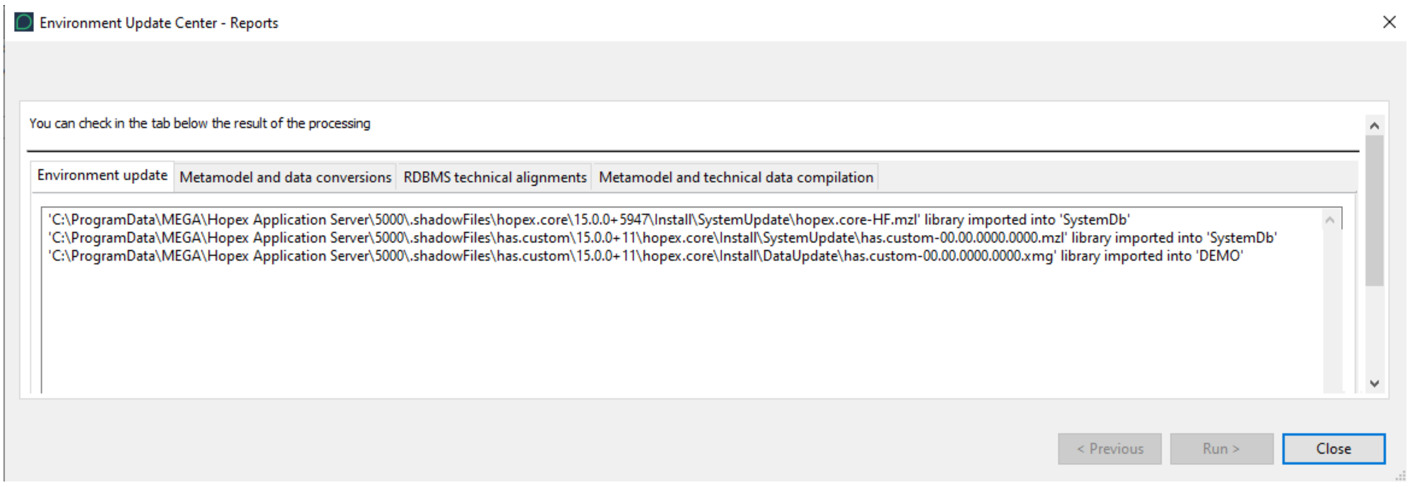
To launch the environment automatic update:

1. Make sure the **HOPEX Core Back-End** module is stopped.
2. In administration.exe, connect to the environment.
3. Right-click the environment and select **Environment automatic update**.



The following files with mandatory data are imported:

- hopex.core-HF-mzl to systemdb
- has.custom-00.00.0000.0000.mzl to systemdb
- has.custom-00.00.0000.0000.xmg to data repository


Environment Update Center - Reports
Environment update
Metamodel and data conversions
RDBMS technical alignments
Metamodel and technical data compilation
< Previous
Run >
Close

You can check in the tab below the result of the processing

```
'C:\ProgramData\MEGA\Hopex Application Server\5000\.shadowFiles\hopex.core\15.0.0+5947\Install\SystemUpdate\hopex.core-HF.mzl' library imported into 'SystemDb'
'C:\ProgramData\MEGA\Hopex Application Server\5000\.shadowFiles\has.custom\15.0.0+11\hopex.core\Install\SystemUpdate\has.custom-00.00.0000.0000.mzl' library imported into 'SystemDb'
'C:\ProgramData\MEGA\Hopex Application Server\5000\.shadowFiles\has.custom\15.0.0+11\hopex.core\Install\DataUpdate\has.custom-00.00.0000.0000.xmg' library imported into 'DEMO'
```

3.3. Getting NuGet package

Prerequisite:

Ensure you have the pre-requisite installed:

You are in a Development mode server. Download and install .NET Core SDK as described in the **HAS Installation guide > Installing the prerequisite software**.

To import the NuGet Package into the server:

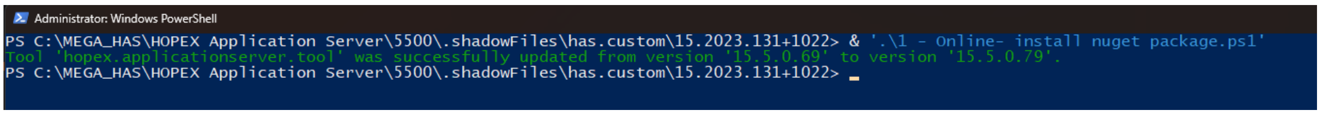
1. Connect to the server in RDP.
2. Go to the HAS installation folder.
3. Find the **has.custom** folder.

   E.g.: C:\ProgramData\HOPEX Application Server\5000\.shadowFiles\has.custom\15.2.0
4. Read the **HOW-TO-BUILD.md** file.

3.3.1. Online mode

With a user that have enough privilege:

1. Launch a PowerShell script windows.
2. Launch the "1 - Online- install nuget package.ps1" script.



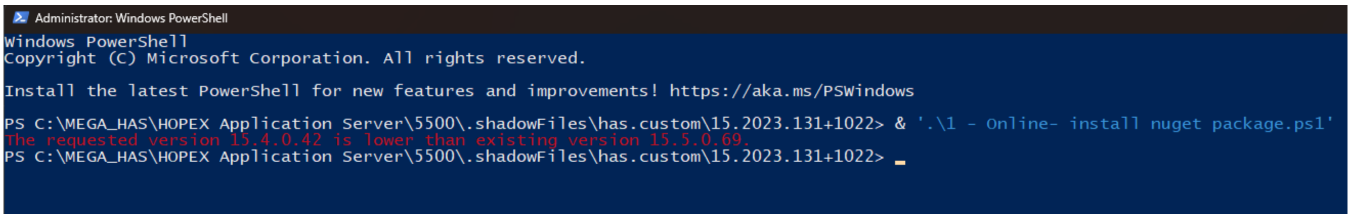
```
Administrator: Windows PowerShell
PS C:\MEGA_HAS\HOPEX Application Server\5500\.shadowFiles\has.custom\15.2023.131+1022> & '.\1 - Online- install nuget package.ps1'
Tool 'hopex.applicationserver.tool' was successfully updated from version '15.5.0.69' to version '15.5.0.79'.
PS C:\MEGA_HAS\HOPEX Application Server\5500\.shadowFiles\has.custom\15.2023.131+1022> _
```

When successful you will have a message such as:

Tool 'hopex.applicationserver.tool' was successfully updated from version '15.5.0.69' to version '15.5.0.79'

If this is not the first time you launch the command, existing version will be updated if needed.

In case a more recent version exists, you might get an error message. In that case edit the PowerShell script.



```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\MEGA_HAS\HOPEX Application Server\5500\.shadowFiles\has.custom\15.2023.131+1022> & '.\1 - Online- install nuget package.ps1'
The requested version 15.4.0.42 is lower than existing version 15.5.0.69.
PS C:\MEGA_HAS\HOPEX Application Server\5500\.shadowFiles\has.custom\15.2023.131+1022> _
```

You are now ready to start your customization and package them.

4. Creating Customization (SystemDB)

This chapter details the process you must follow when **you perform customization** within the SystemDB (and sometime in the Data repository).

Prerequisite:

- You must be in DEV instance.
- You must have properly imported "has.custom" module in the SystemDb.

4.1. Work Item concepts

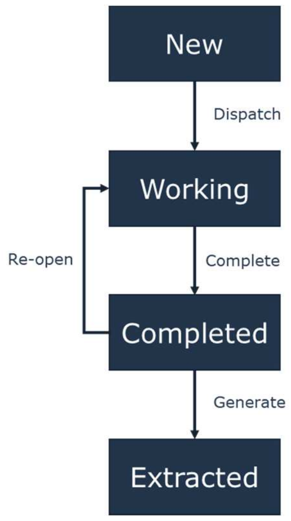
When performing customization, you may change a wide range of elements. You must group this changes into a unit of work that we call a "**Work Item**" or shortly named "**WI**".

A Work Item represents a set of changes; the nature of the grouping is **up to you** and can include a mix of:

- similar functional scope
- similar technical scope
- similar delivery constraint: project or timeline.

4.1.1. Work Item States

A Work Item has, by default, a linear lifecycle composed of 4 steps:



1. New

   This is the default state when a Work Item is created.

2. Working

   As soon as you dispatch a customization on a Work Item it goes into this state. This represents a work in progress. You can continue adding "dispatch" to this Work Item as long as it is in this state.

3. Completed

   The Work Item is closed and cannot be edited anymore.

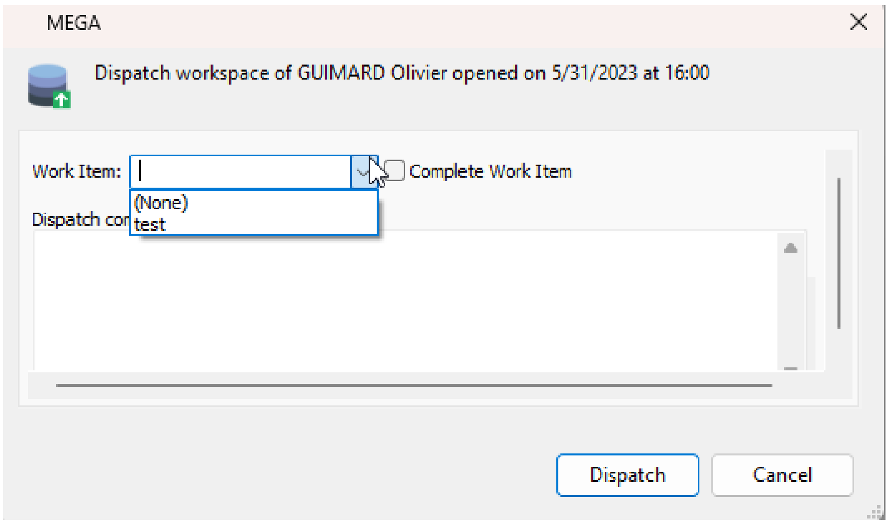
4. Extracted

   The Work Item has been extracted to an MGR/XMG files.

While in "Working" state the Work Item can be worked on for several days/weeks with several dispatches.

4.1.2. Creating/Using a Work Item

The Work Item can be created or used at the moment of the dispatch of your work.


MEGA

Dispatch workspace of GUIMARD Olivier opened on 5/31/2023 at 16:00

If you do not see the "Work Item" it is because you do not meet the prerequisite.

To create a Work Item:

1. Enter a name in the **Work Item** field.
2. Click **Dispatch** (or press Enter).

4.1.3. Recommendation when working with WI

When working on your customization project you might be facing Scenario where you want to have several Work Items. This can be due to grouping topics or delivery timing.

Naming convention

We recommend that you give explicit name to your Work Item.

- Name contains the functional scope.

  E.g.: "Metamodel customization", "Static website customization".
- Name contains a planned delivery version/time.

  E.g.: "Project Customization for V2", "Next release sprint July".
- Name contains a functional domain or project name.
  E.g.: "ITPM customization", "Project 1 customization".

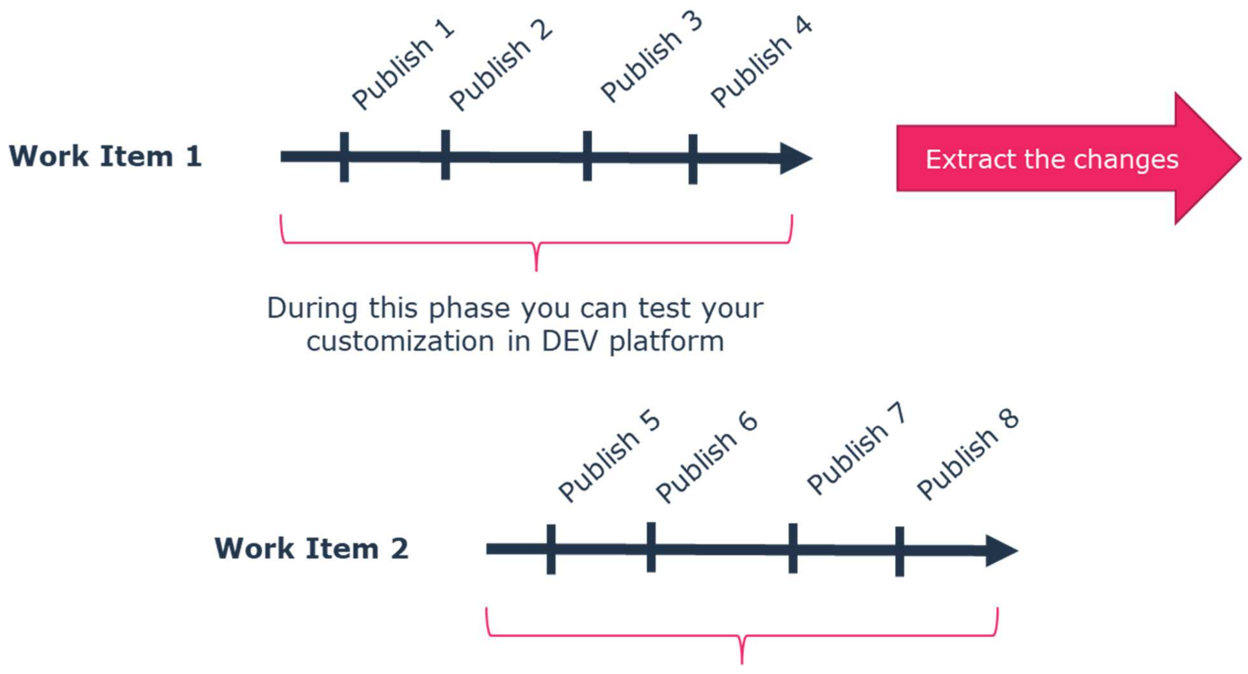
It can be a mix of these propositions.

What you should avoid in naming a Work Item:

- giving a random name.
  E.g.: "Work Item 1", "Work Item 2", "WI A", "WI B".
- giving a non-explicit name.
  E.g;: "Oliver WI", "Today Work Item".

Grouping Work Items

We recommend you limit the number of Work Items in "Working" stage, to limit confusion of where things can be contained and what needs to be grouped.



4.2. Dispatch customization

In a development platform, once you are done with a customization and you want to **dispatch** it into the repository, you **must** associate a Work Item to a dispatch.

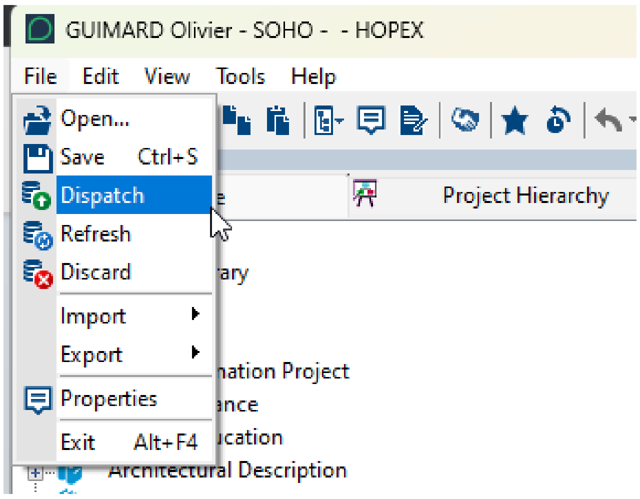
You can associate several dispatches to the same Work Item.

You can repeat the dispatch on Work Item as you are working on your customization. A Work Item can contain multiple dispatch.

4.2.1. Publication process

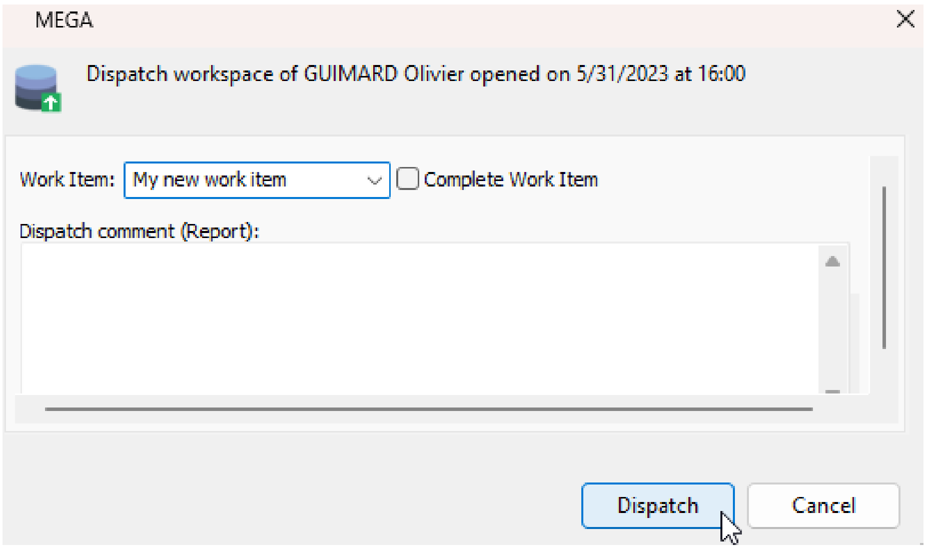
From HOPEX (Windows Front-End)

1. From **HOPEX** toolbar select **File > Dispatch**.



2. In the **Work Item** dropdown list, select your Work Item.

   If you do not have any Work Item or need to create a new one, enter a name in the **Work Item** field and press Enter.



MEGA

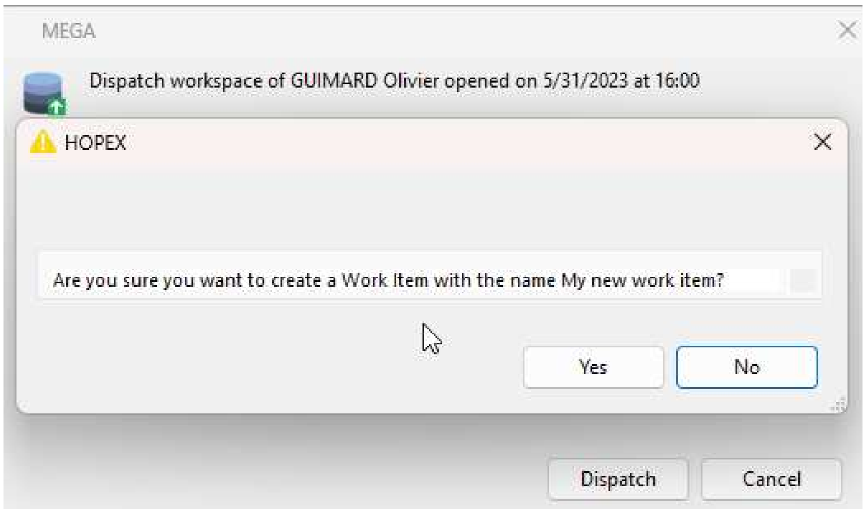
Dispatch workspace of GUIMARD Olivier opened on 5/31/2023 at 16:00

Work Item: My new work item ☐ Complete Work Item

Dispatch comment (Report):

3. Click **Dispatch**.

   If prompted click **Yes** to create the new Work Item.



MEGA

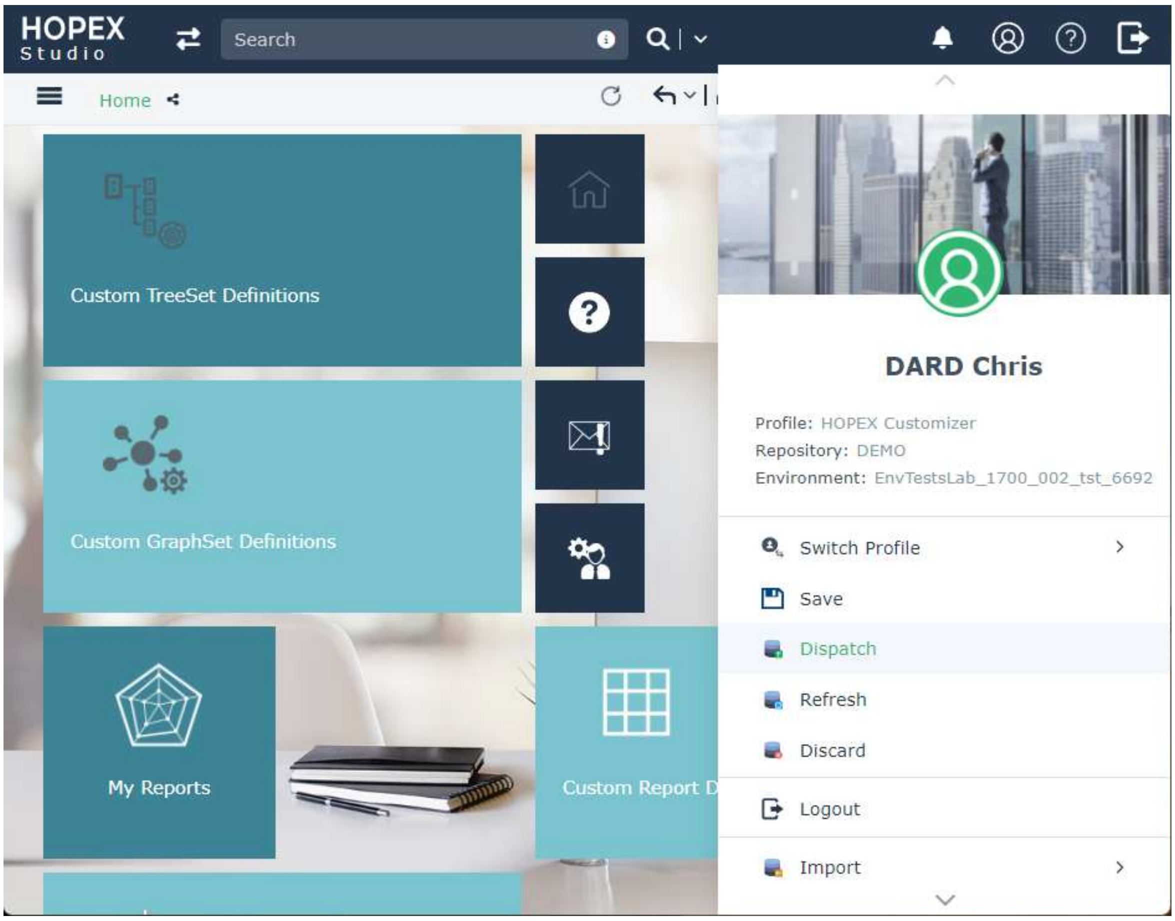
Dispatch workspace of GUIMARD Olivier opened on 5/31/2023 at 16:00

HOPEX

Are you sure you want to create a Work Item with the name My new work item?

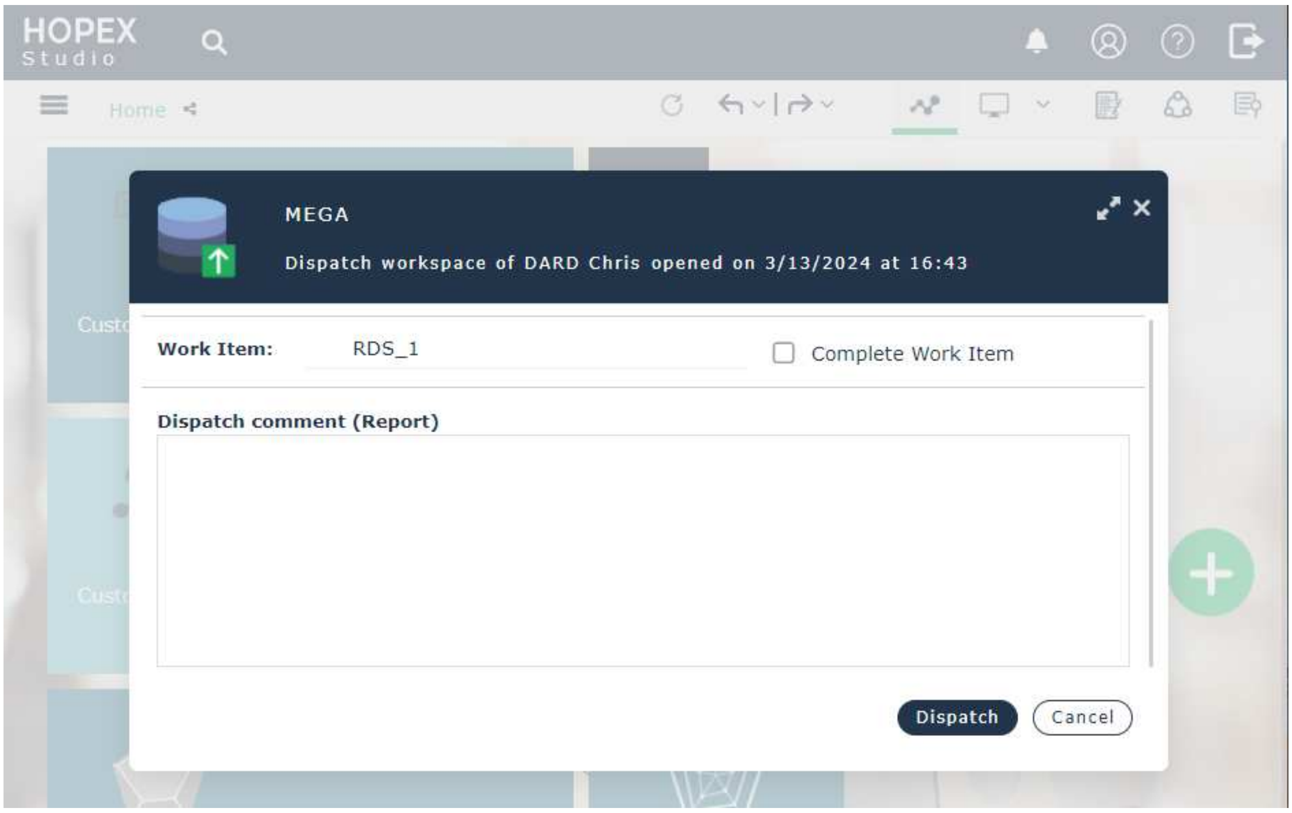
From HOPEX Studio desktop (Web Front-End)

1. From HOPEX main menu, select **Dispatch**.



2. In the **Work Item** dropdown list, select your Work Item.

   If you do not have any Work Item or need to create a new one, enter a name in the **Work Item** field.


MEGA

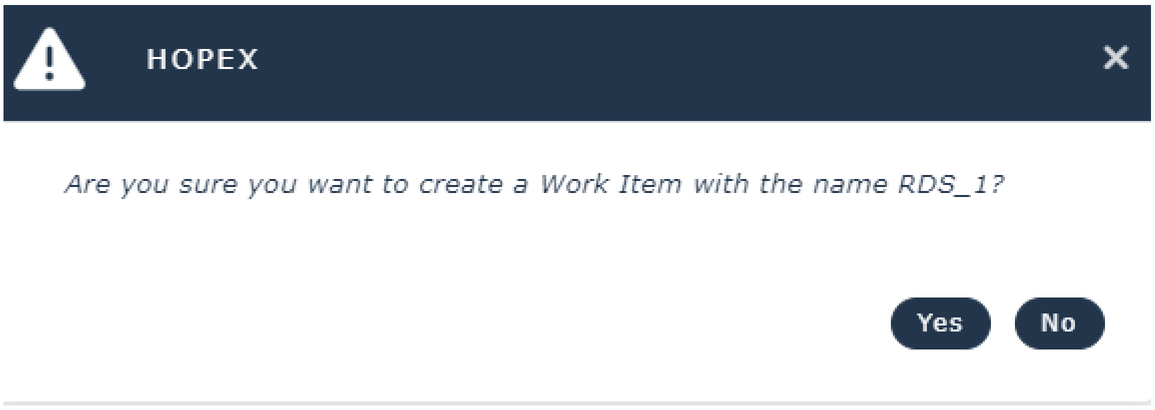
Dispatch workspace of DARD Chris opened on 3/13/2024 at 16:43

Work Item: RDS_1

☐ Complete Work Item

Dispatch comment (Report)

3. (Optional) In **Dispatch comment (Report)** enter a description of your Work Item.
4. Click **Dispatch**.


HOPEX

Are you sure you want to create a Work Item with the name RDS_1?

4.3. Done with your customization?

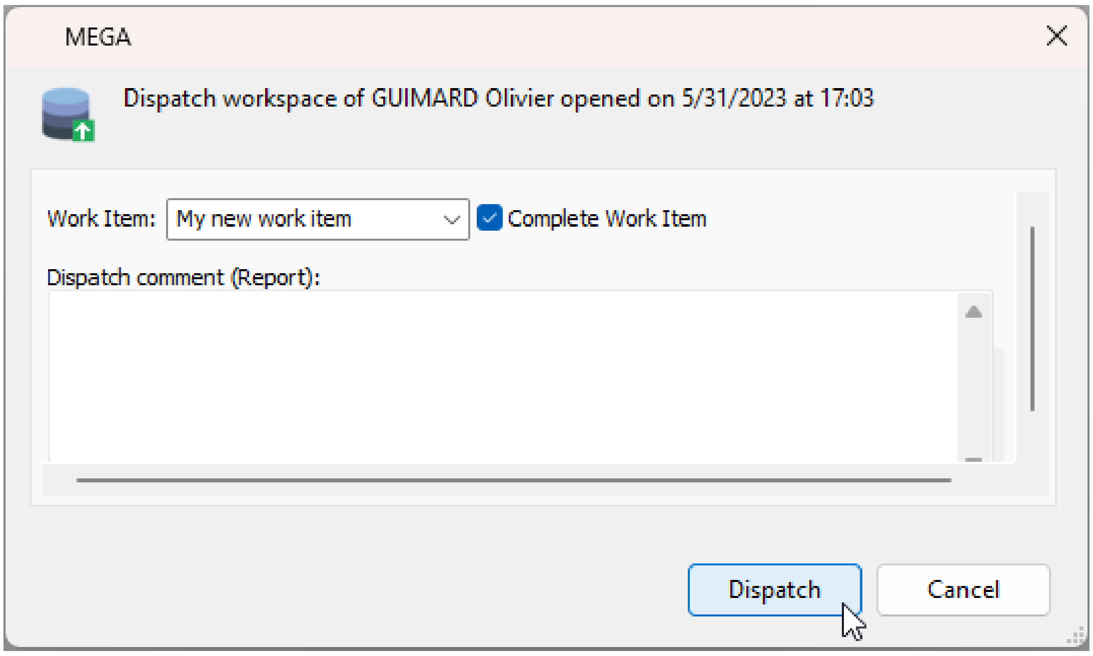
You have worked on several customizations and are now ready to push them to STAGE platform. It is important to do this only when you want to push your work to STAGE/PRODUCTION.

Before you can extract the customization into files, you must first set your Work Item as completed. If you have several Work Items that you want to extract, they all must be in the state "Completed".

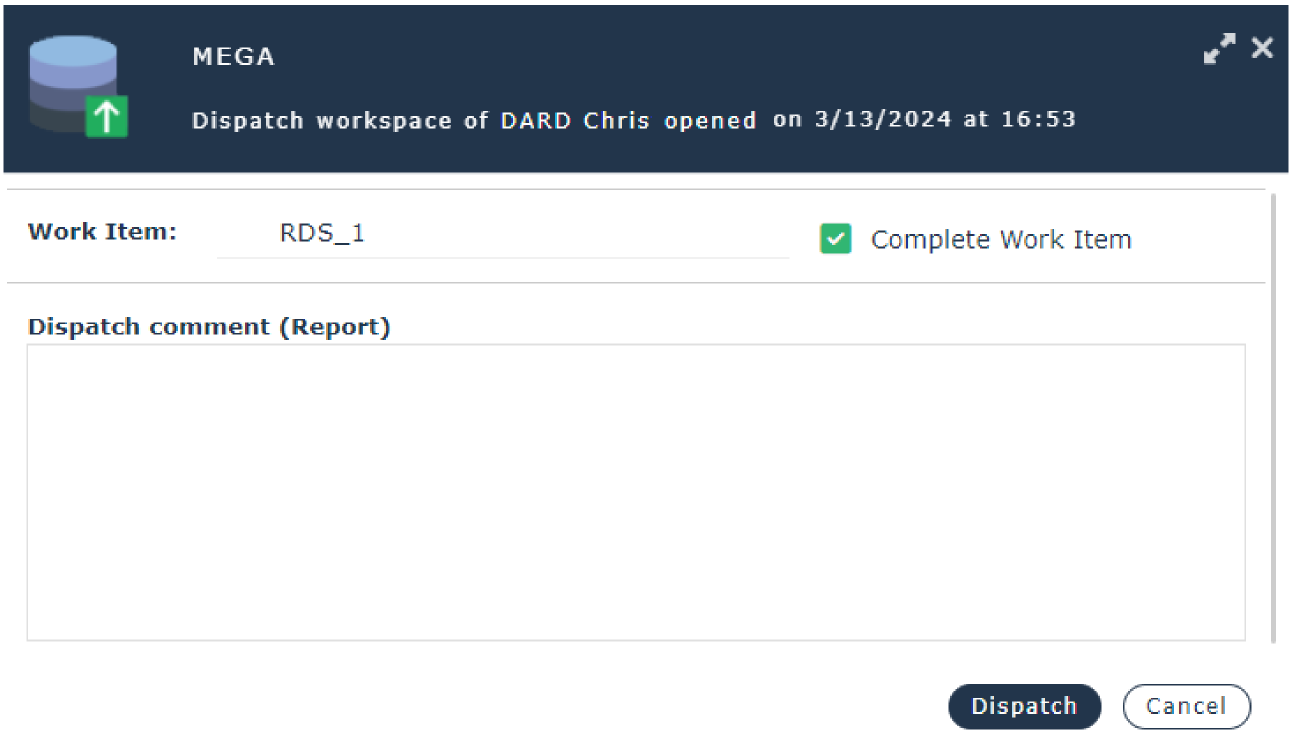
To complete a Work Item at Dispatch:

1. In the **Work Item** field select the Work Item concerned.
2. Select **Complete Work Item**:

- HOPEX (Windows Front-End)


MEGA
Dispatch workspace of GUIMARD Olivier opened on 5/31/2023 at 17:03
Work Item: My new work item
✓ Complete Work Item
Dispatch comment (Report):
Dispatch
Cancel

- HOPEX Studio desktop (Web Front-End)


MEGA
Dispatch workspace of DARD Chris opened on 3/13/2024 at 16:53
Work Item: RDS_1
✓ Complete Work Item
Dispatch comment (Report)
Dispatch
Cancel

3. Click **Dispatch**.

   You are now ready to extract the Work Item to files.

   ➔ See Extracting Customization.

5. Extracting Customization

You can generate a group of Work Items to extract your customization in mgr/xmg format.

Work Items are sorted in:

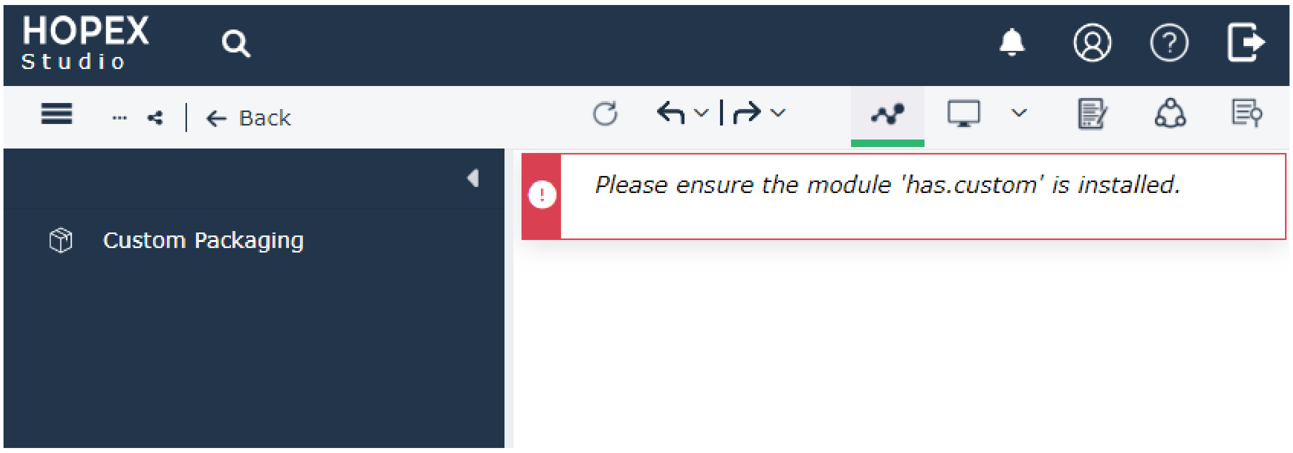
- the **Active Work Items** folder: all Work Items that are not completed yet.
- the **Completed Work Items** folder: all Work Items that are completed. Only this completed Work Items are taken into account for the extraction.

5.1. Prerequisites

To extract customizations to files:

1. Connect to HOPEX Studio (HOPEX Customizer profile).
2. From the Navigation menu select **Custom Packaging**.

   If you get the "Please ensure the module 'has.custom' is installed." error message, go to the Getting Ready for Customization chapter and launch the Environment Automatic Update.



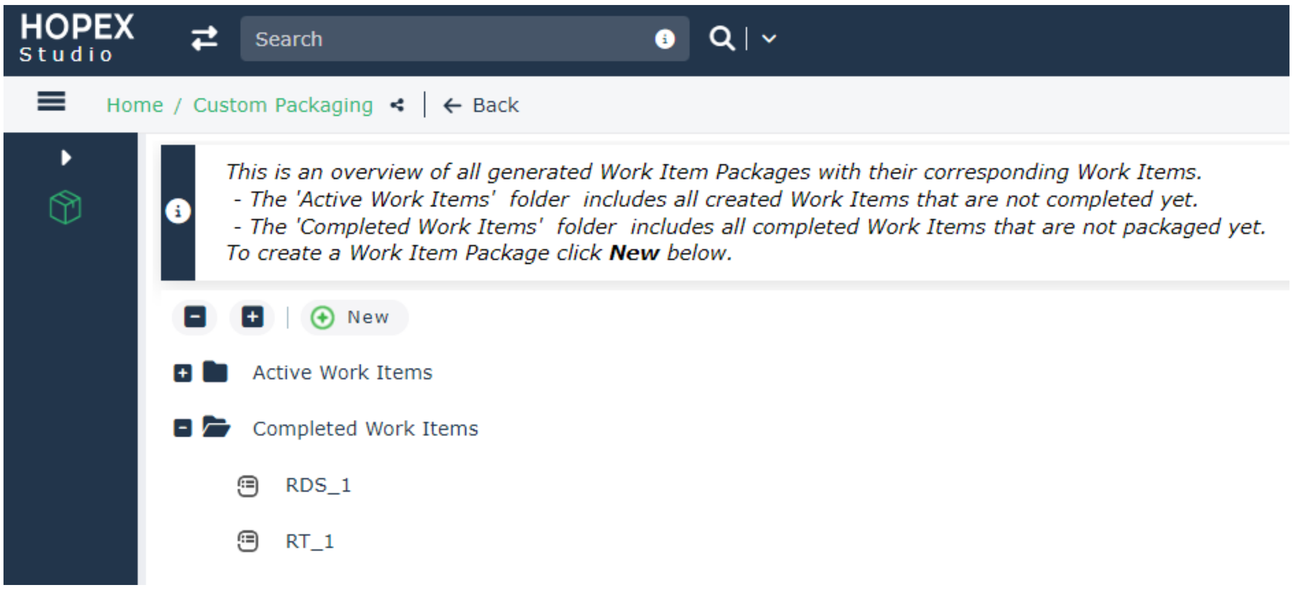
Please ensure the module 'has.custom' is installed.

5.2. Generating the MGR/XMG files

Customizations candidates to extraction are those in **Completed Work Items** folder.

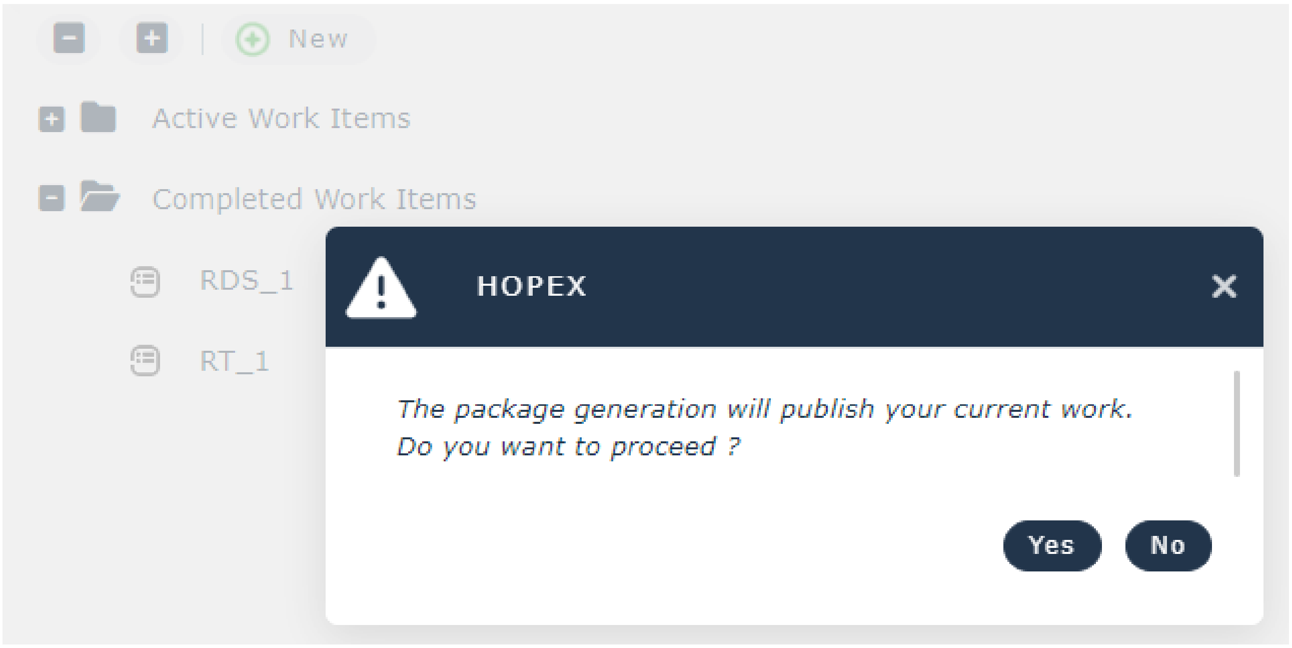
To extract customizations to files:

1. Connect to HOPEX Studio (HOPEX Customizer profile).
2. From the Navigation menu select **Custom Packaging**.



This is an overview of all generated Work Item Packages with their corresponding Work Items.
- The 'Active Work Items' folder includes all created Work Items that are not completed yet.
- The 'Completed Work Items' folder includes all completed Work Items that are not packaged yet.
*To create a Work Item Package click **New** below.*

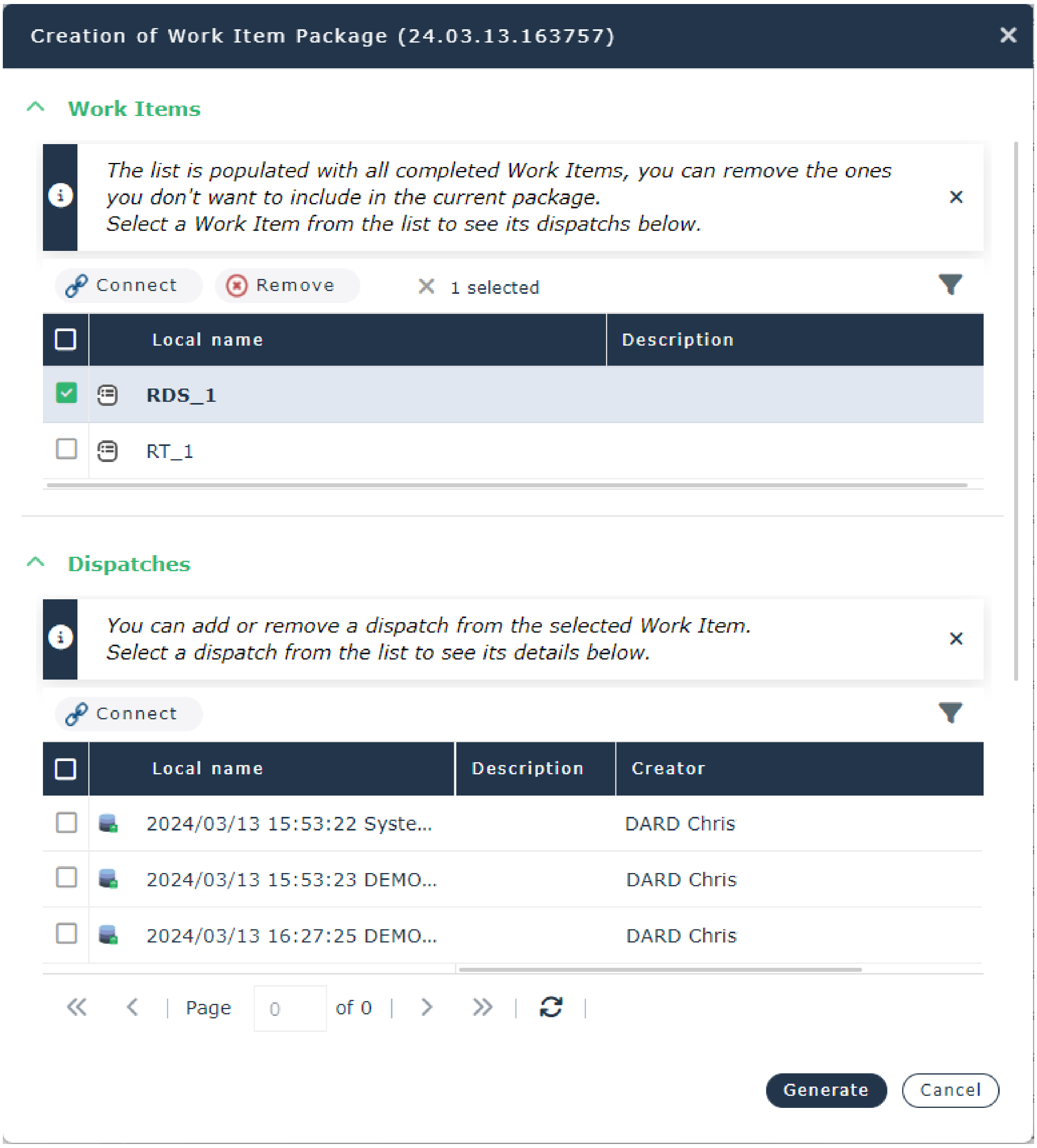
3. Click **New** to generate a Work Item Group.



The package generation will publish your current work.
Do you want to proceed ?

4. Click **Yes** to confirm the group generation.
5. All completed Work Items are included.

Although it is possible, we do not recommend you amend dispatches or work items selected by default at that stage.



Creation of Work Item Package (24.03.13.163757)

Work Items

The list is populated with all completed Work Items, you can remove the ones you don't want to include in the current package.
Select a Work Item from the list to see its dispatchs below.

Connect | Remove | 1 selected

	Local name	Description
✓	**RDS_1**	
	RT_1	

Dispatches

You can add or remove a dispatch from the selected Work Item.
Select a dispatch from the list to see its details below.

Connect

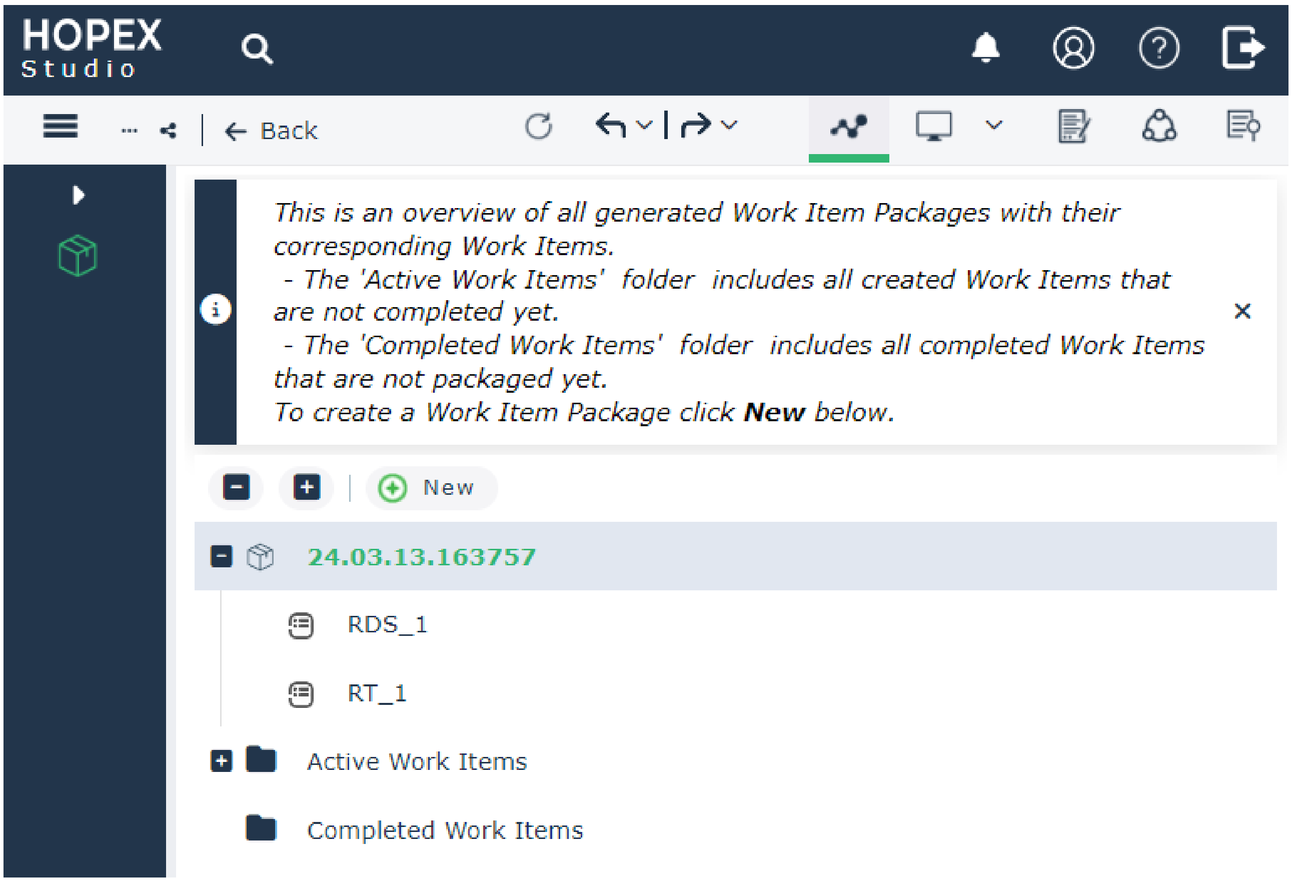
	Local name	Description	Creator
	2024/03/13 15:53:22 Syste...		DARD Chris
	2024/03/13 15:53:23 DEMO...		DARD Chris
	2024/03/13 16:27:25 DEMO...		DARD Chris

Page 0 of 0

Generate Cancel

6. Click **Generate**.

   A Work Item group is created as defined.


This is an overview of all generated Work Item Packages with their corresponding Work Items.
- The 'Active Work Items' folder includes all created Work Items that are not completed yet.
- The 'Completed Work Items' folder includes all completed Work Items that are not packaged yet.
*To create a Work Item Package click **New** below.*

The naming convention for this group is: YY.MM.DD.HHMMSS

- YY: Year on 2 digits
- MM: Month on 2 digits
- DD: Day on 2 digits
- HHMMSS: timestamp in UTC+0

5.3. Checking generated files

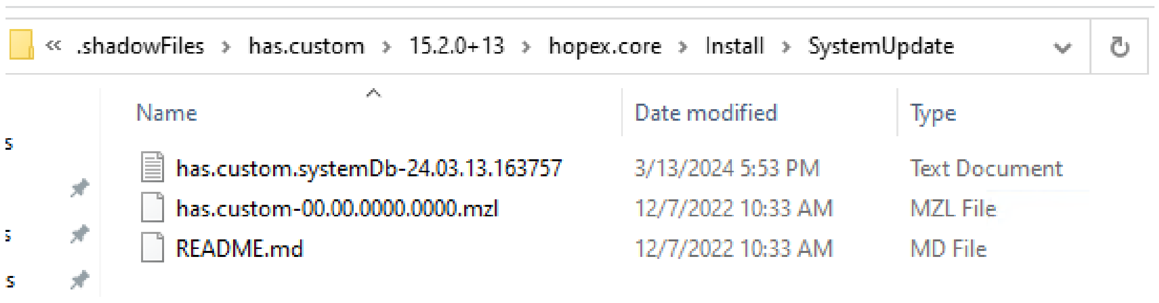
Once the generation of the file is successful, the file is available in the installation folder.

Location:

> ***C:\…\HOPEX Application Server\<HAS instance name>\.shadowFiles*** \ *has.custom\<Custom module version>* \ ***hopex.core\Install***

The folder includes the following sub folders:

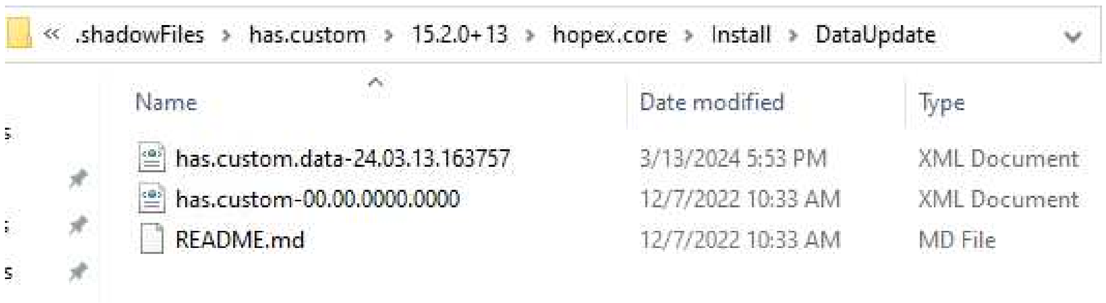
- **SystemUpdate:** This folder contains the modification you made that will be imported in the SystemDB.



« .shadowFiles › has.custom › 15.2.0+13 › hopex.core › Install › SystemUpdate

Name	Date modified	Type
has.custom.systemDb-24.03.13.163757	3/13/2024 5:53 PM	Text Document
has.custom-00.00.0000.0000.mzl	12/7/2022 10:33 AM	MZL File
README.md	12/7/2022 10:33 AM	MD File

- **DataUpdate:** This folder contains the modification you made that will be imported in ALL the repositories.



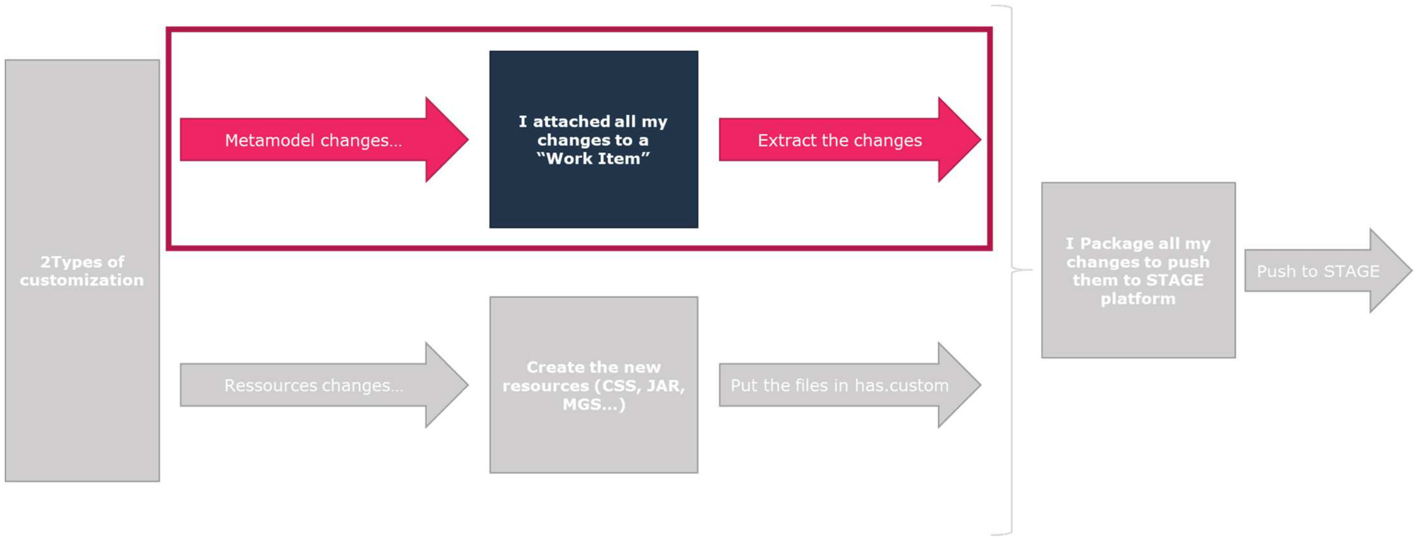
« .shadowFiles › has.custom › 15.2.0+13 › hopex.core › Install › DataUpdate

Name	Date modified	Type
has.custom.data-24.03.13.163757	3/13/2024 5:53 PM	XML Document
has.custom-00.00.0000.0000	12/7/2022 10:33 AM	XML Document
README.md	12/7/2022 10:33 AM	MD File

*CAUTION: Make sure in the **DataUpdate** folder you did not embedded **demo/sample** data by mistake.*

This timestamp is used to know which files have been already imported or not.

You have now successfully completed the creation and extraction of customization contained in the repositories (SystemDB and Data).


2Types of customization
Metamodel changes…
I attached all my changes to a "Work Item"
Extract the changes
Ressources changes…
Create the new resources (CSS, JAR, MGS…)
Put the files in has.custom
I Package all my changes to push them to STAGE platform
Push to STAGE

6. New/Replace/Override Resource Files

All the custom files you create, whatever their type, must be stored in the has.custom module. These files will be packaged to be later pushed to STAGE or PROD environment.

Ressources changes...
Create the new resources (CSS, JAR, MGS...)
Put the files in has.custom

Depending on the nature of the files the location might be different. The general rules are as follow:

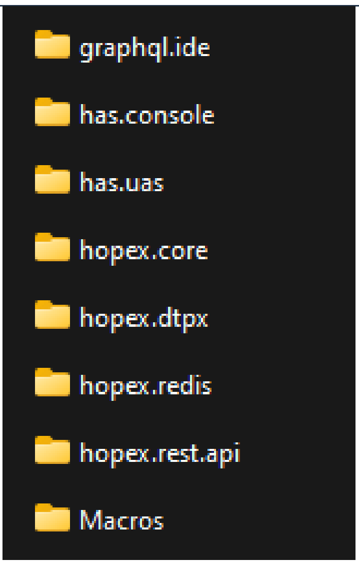
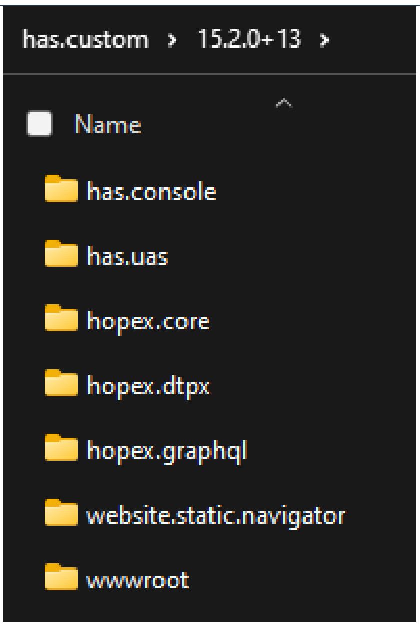
- Each module has a folder with sub-folders within the has.custom module.
- You must create a folder structure that reflects the folder structure of the standard module.

Note: you cannot replace/delete standard files. You must store your files in custom folder, even if they have the same name.

6.1. Folders and sub-folders structures

The standard folder structure located in ".shadowsFiles" contains one folder per module.

For each module you want to customize, create a new sub-folder with the same name (ID) in the custom module.

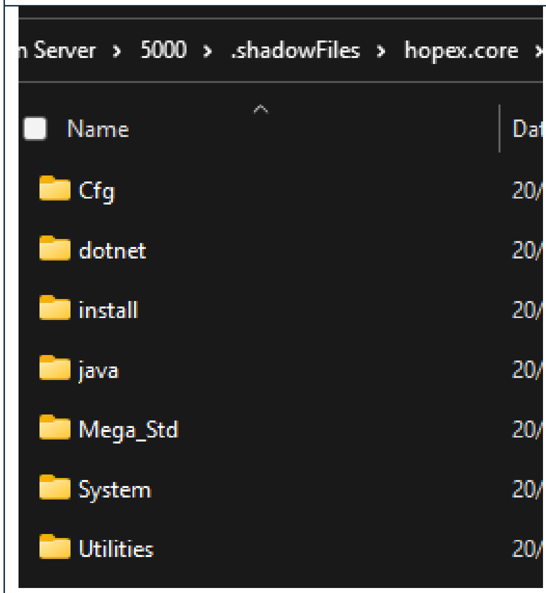
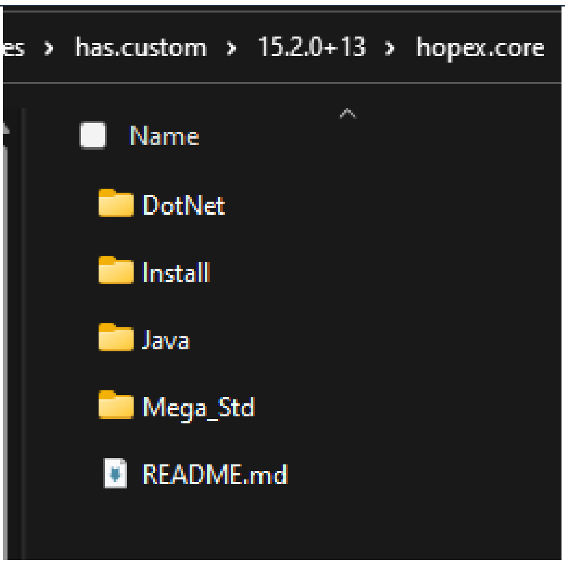
Standard modules folder name	Custom folder structure
 graphql.ide has.console has.uas hopex.core hopex.dtpx hopex.redis hopex.rest.api Macros	 has.custom › 15.2.0+13 › Name has.console has.uas hopex.core hopex.dtpx hopex.graphql website.static.navigator wwwroot
	In that example we want to customize login page, custom graphQL schema, custom scripts…

In the default custom module, you can find tree-structured folders (provided by default) corresponding to the most popular customizations.

In case the structure is not available, you must reproduce the tree-structure.

Once you have created your structure, you can bring in your custom files.

Example with the **hopex.core** module:

Standard folder structure	Custom folder structure to override
 n Server › 5000 › .shadowFiles › hopex.core › Name Dat Cfg 20/ dotnet 20/ install 20/ java 20/ Mega_Std 20/ System 20/ Utilities 20/	 es › has.custom › 15.2.0+13 › hopex.core Name DotNet Install Java Mega_Std README.md

*Note: **System** and **Utilities** folders cannot be overridden in custom module.*

Whether you create files for:

- having new behavior,
- replacing or overriding standard files,

you must follow exactly the folder structure.

Most commonly created or changed files are stored in the following folder locations:

File type	Location
DLL	C:\...\HOPEX Application Server\5000\.shadowFiles\has.custom\<Module version>\DotNet
JAR	C:\...\HOPEX Application Server\5000\.shadowFiles\has.custom\<Module version>\Java
ICO	C:\...\HOPEX Application Server\5000\.shadowFiles\has.custom\<Module version>\hopex.core\Mega_Std
MGS	C:\...\HOPEX Application Server\5000\.shadowFiles\has.custom\<Module version>\hopex.core\Mega_Std
PNG	C:\...\HOPEX Application Server\5000\.shadowFiles\has.custom\<Module version>\hopex.core\Mega_Std
DOCX For document template	C:\...\HOPEX Application Server\5000\.shadowFiles\has.custom\<Module version>\hopex.core\Mega_Std
CSS static website generation	C:\...\HOPEX Application Server\5000\.shadowFiles\has.custom\<Module version>\hopex.core\Mega_Std
CSS Hopex web CSS	C:\...\HOPEX Application Server\5000\.shadowFiles\has.custom\<Module version>\wwwroot

Where here 5000 is the HAS instance name.

➔ See common examples in the following section.

6.2. Examples of common files and folders customized

6.2.1. Diagram Shapes

To customize shapes in diagrams:

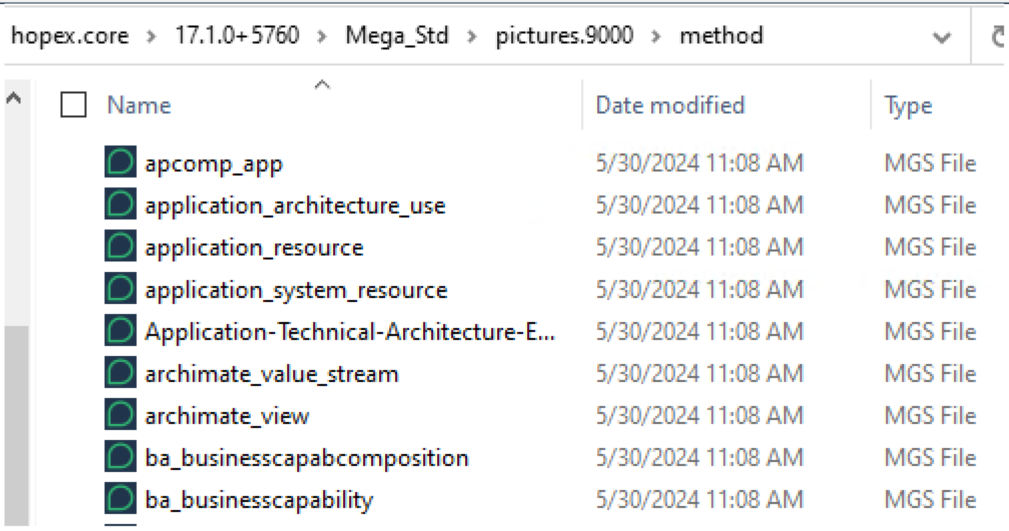
1. Edit the related files with extension "*.MGS".
2. Create your custom shapes.
3. Store the custom shapes in its dedicated folder.
   - The standard shapes are located here:

C:\...\HOPEX Application Server\<*HAS instance name*>\.shadowFiles\hopex.core\<Module version>**Mega_Std**

- The custum shapes are located here:

  C:\...\HOPEX Application Server\<HAS instance name>\.shadowFiles*\has.custom\<Module version>****hopex.core\Mega_Std***

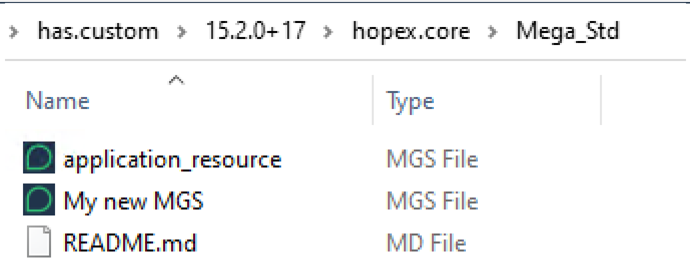
Standard folder structure



hopex.core > 17.1.0+5760 > Mega_Std > pictures.9000 > method

Name	Date modified	Type
apcomp_app	5/30/2024 11:08 AM	MGS File
application_architecture_use	5/30/2024 11:08 AM	MGS File
application_resource	5/30/2024 11:08 AM	MGS File
application_system_resource	5/30/2024 11:08 AM	MGS File
Application-Technical-Architecture-E...	5/30/2024 11:08 AM	MGS File
archimate_value_stream	5/30/2024 11:08 AM	MGS File
archimate_view	5/30/2024 11:08 AM	MGS File
ba_businesscapabcomposition	5/30/2024 11:08 AM	MGS File
ba_businesscapability	5/30/2024 11:08 AM	MGS File

Custom folder structure to override



> has.custom > 15.2.0+17 > hopex.core > Mega_Std

Name	Type
application_resource	MGS File
My new MGS	MGS File
README.md	MD File

All MGS must be stored directly in Mega_Std of the custom module without subfolder Pictures.XXXX.

6.2.2. Login page and Portal page

The portal page is contained in the "has.uas" module.

The login page is also contained in the "has.uas" module but has been overloaded by "hopex.specific.assets" module.

Should you want to change the background image of both the portal page and the login page, you must add your custom png/jpg images in a folder structure that matches the one from the standard, i.e. in:

C:\...\HOPEX Application Server\5000\.shadowFiles\has.custom\<module version>\has.uas**\wwwroot\images**.

Example: changing the login page background.

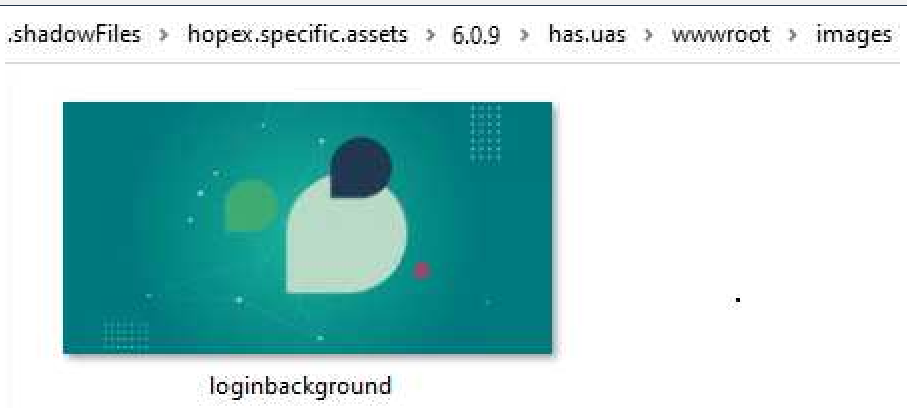
- The standard file “loginbackground.jpg” is located here:

  C:\...\HOPEX Application Server\5000\.shadowFiles**
  hopex.specific.assets*\6.0.9\has.uas\****wwwroot\images***

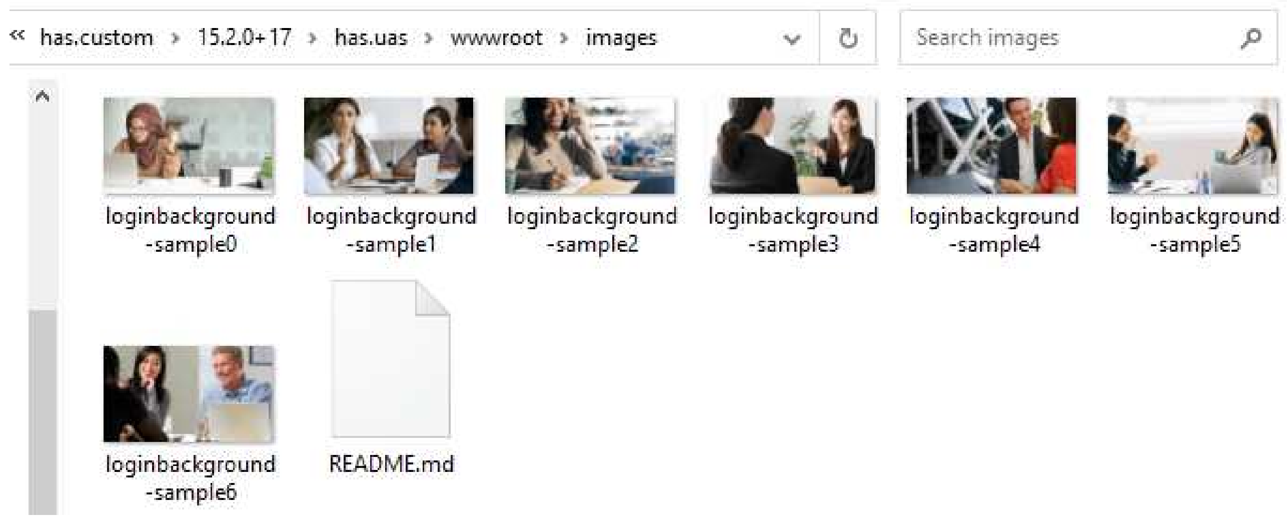
- Your custom file “loginbackground.jpg” is located here:

  C:\...\HOPEX Application Server\5000\.shadowFiles
  *has.custom\15.2.0+17\has.uas\****wwwroot\images***

Standard folder structure


.shadowFiles > hopex.specific.assets > 6.0.9 > has.uas > wwwroot > images
loginbackground

Custom folder structure to override



6.2.3. GraphQL custom schema

If you created custom schema for your GraphQL REST API, you must store them in the following folder:

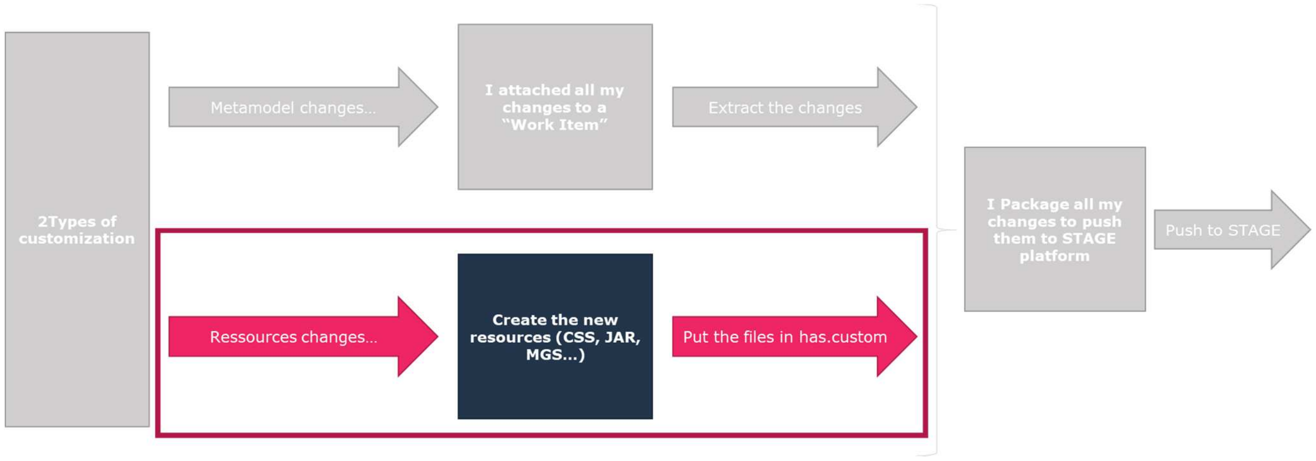
- If standard location is:

  C:\...\HOPEX Application Server\5000\.shadowFiles*\Macros\hopex.graphql\7.87.507+6676\****CONFIG\V3****\Standard*

- Custom location is:

C:\...\HOPEX Application Server\5000\.shadowFiles*\has.custom\15.2.0+17\hopex.graphql\****CONFIG\V3****\Custom*

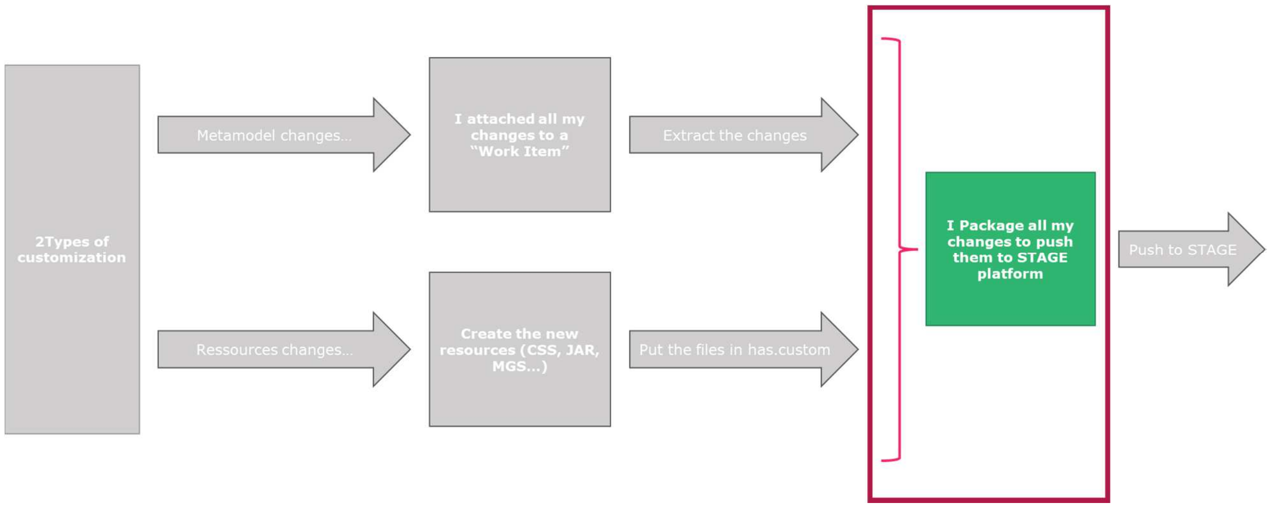
You have now successfully completed the tasks to put your files in the right folders.


2Types of customization
Metamodel changes…
I attached all my changes to a "Work Item"
Extract the changes
Ressources changes…
Create the new resources (CSS, JAR, MGS…)
Put the files in has.custom
I Package all my changes to push them to STAGE platform
Push to STAGE

7. Packaging customization

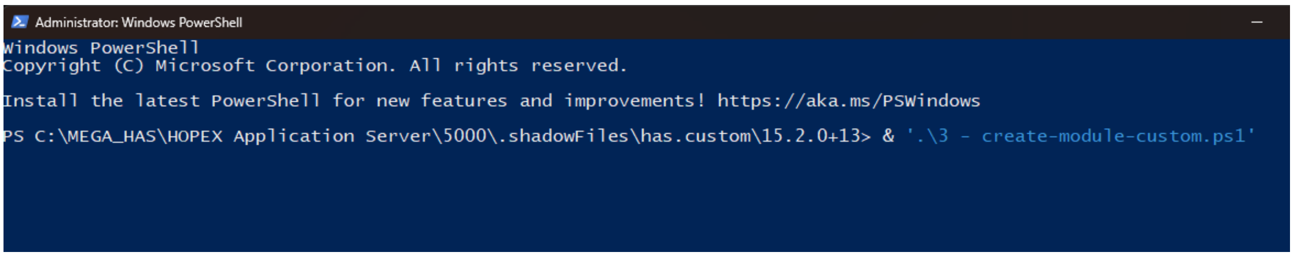
The module custom is what must be pushed to STAGE or PROD. As mentioned, this package contains all the resources and all MGR/XMG files.

This module is an "HASPKG", it is a smart "zip" of all the files required.


2Types of customization
Metamodel changes...
Ressources changes...
I attached all my changes to a "Work Item"
Create the new resources (CSS, JAR, MGS...)
Extract the changes
Put the files in has.custom
I Package all my changes to push them to STAGE platform
Push to STAGE

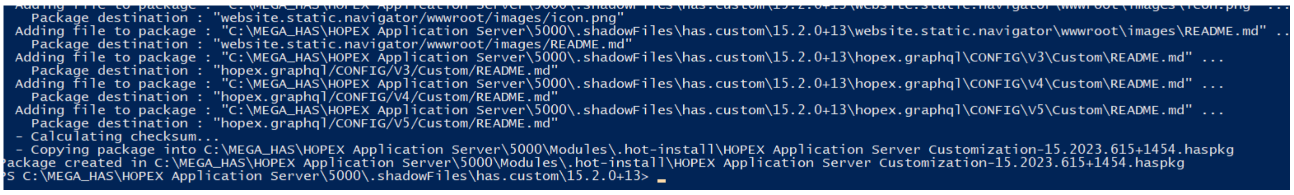
To create this package:

1. Connect to your DEV instance server in Remote Desktop.
2. Launch a provided PowerShell script called: "3 - create-module-custom.ps1".
   Default location: C:\...\HOPEX Application Server\5000\.shadowFiles\has.custom\...\



```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\MEGA_HAS\HOPEX Application Server\5000\.shadowFiles\has.custom\15.2.0+13> & '.\3 - create-module-custom.ps1'
```

   When successful you get a message "Packaged Created".



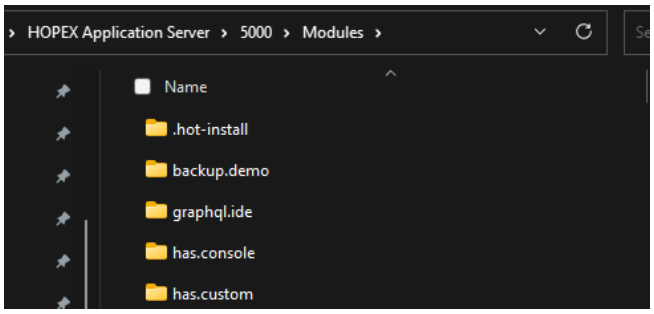
```
   Package destination : "website.static.navigator/wwwroot/images/icon.png"
 Adding file to package : "C:\MEGA_HAS\HOPEX Application Server\5000\.shadowFiles\has.custom\15.2.0+13\website.static.navigator\wwwroot\images\README.md" ..
   Package destination : "website.static.navigator/wwwroot/images/README.md"
 Adding file to package : "C:\MEGA_HAS\HOPEX Application Server\5000\.shadowFiles\has.custom\15.2.0+13\hopex.graphql\CONFIG\V3\Custom\README.md" ...
   Package destination : "hopex.graphql/CONFIG/V3/Custom/README.md"
 Adding file to package : "C:\MEGA_HAS\HOPEX Application Server\5000\.shadowFiles\has.custom\15.2.0+13\hopex.graphql\CONFIG\V4\Custom\README.md" ...
   Package destination : "hopex.graphql/CONFIG/V4/Custom/README.md"
 Adding file to package : "C:\MEGA_HAS\HOPEX Application Server\5000\.shadowFiles\has.custom\15.2.0+13\hopex.graphql\CONFIG\V5\Custom\README.md" ...
   Package destination : "hopex.graphql/CONFIG/V5/Custom/README.md"
 - Calculating checksum...
 - Copying package into C:\MEGA_HAS\HOPEX Application Server\5000\Modules\.hot-install\HOPEX Application Server Customization-15.2023.615+1454.haspkg
ackage created in C:\MEGA_HAS\HOPEX Application Server\5000\Modules\.hot-install\HOPEX Application Server Customization-15.2023.615+1454.haspkg
S C:\MEGA_HAS\HOPEX Application Server\5000\.shadowFiles\has.custom\15.2.0+13> _
```

   The newly created package is then copy in the ".hot-install" folder located here :
   C:\...\HOPEX Application Server\5000\Modules\.hot-install

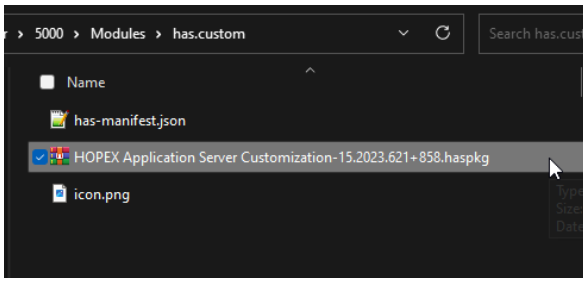
   - If the HAS instance is running, the file is automatically redeployed in the DEV instance.
   - If the HAS instance is not running, we recommend you start it for the module to be properly deployed.

3. Get the packaged that was generated for the next step.

The file is name "HOPEX Application Server Customization-15.2023.XXX+XXX.haspkg" and is located: C:\...\HOPEX Application Server\5000\Modules\has.custom



→


› 5000 › Modules › has.custom
Search has.cust
Name
has-manifest.json
HOPEX Application Server Customization-15.2023.621+858.haspkg
icon.png

You now have the file that you will push to the next platform.

8. Pushing Customization to STAGE/PROD

8.1. Pushing to STAGE or PROD principle

When pushing to STAGE or PROD you must follow these rules:

- **All changes** you want to perform, in STAGE or PROD, should be packaged in the custom module.
- **No manual** action should be performed on the STAGE or PROD concerning your customization.
- Should something be missing or badly customized you must update the custom module in DEV and generate a newer version for STAGE/PROD.

To ensure your customization is properly extracted and packaged make sure you **do not alter the haspkg** file generated from the previous step.

The push, of this new module custom, consists of the following steps:

1. Upload the new custom module on the server ➔ files will be unzipped and replicated in all nodes of the cluster.
2. Run the automatic update ➔ all the MGR/XMG will be imported in the appropriate repositories.

These actions must be done with **no users connected** to HOPEX as a restart will be required.

8.2. Uploading the custom module in HAS

To upload a module in HOPEX you have several possibilities. Choose one of the following ways to upload the new module custom:

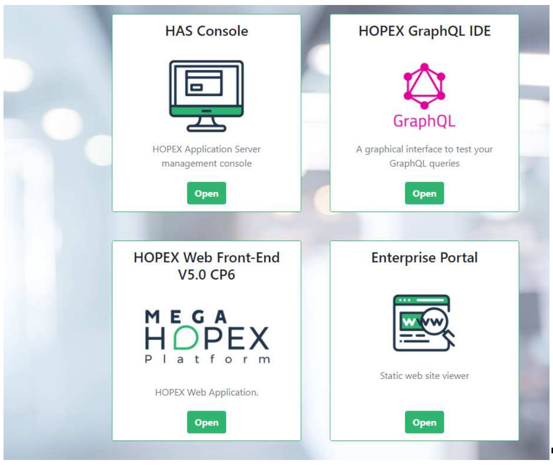
1. From the Web ➔ **preferred choice**.
2. From RDP access

When uploading the new module **some of the changes** may **appear immediately** such as CSS or PNG changes. Some changes **require to restart** some modules.

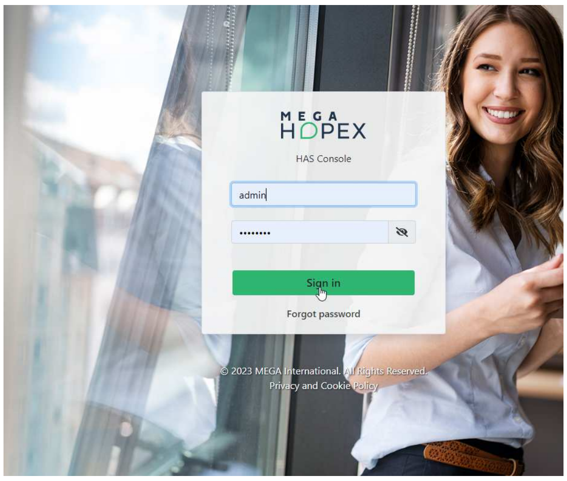
8.2.1. Uploading the module from the Web

To upload the HAS Cutomization module from the Web:

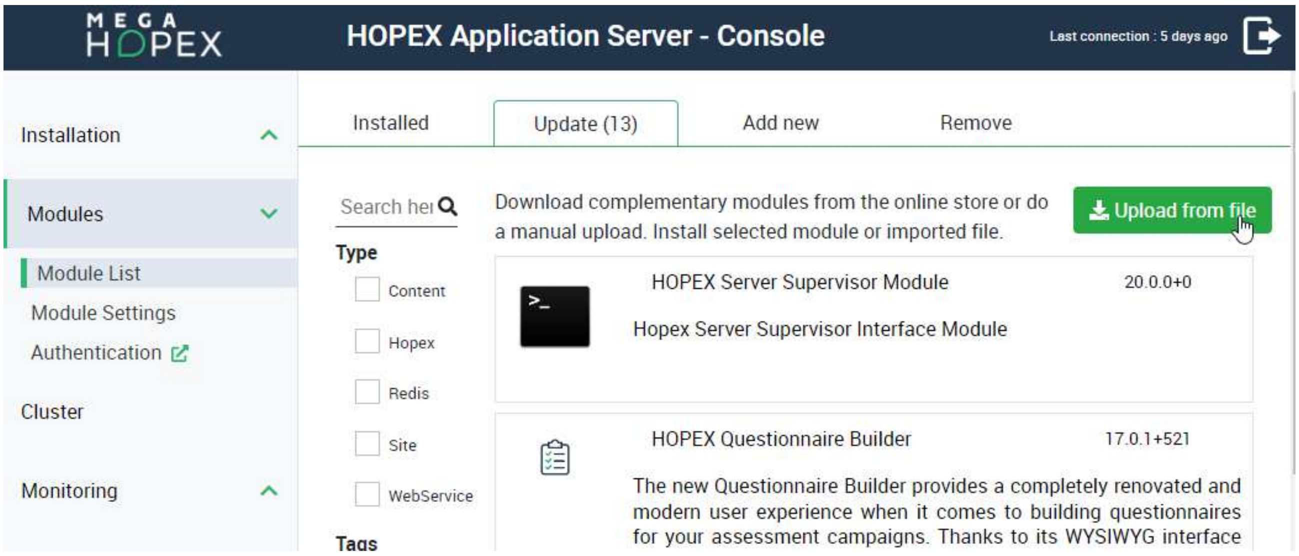
1. Connect to the HAS console.
2. Login with admin user.
3. Click **Modules > Modules List > Update**.



➔


MEGA HOPEX
HAS Console
admin
Sign in
Forgot password
© 2023 MEGA International. All Rights Reserved.
Privacy and Cookie Policy

4. Click **Upload from file**.
5. Browse and chose your new custom module "HOPEX Application Server Customization-15.2023.XXX+XXX.haspkg".



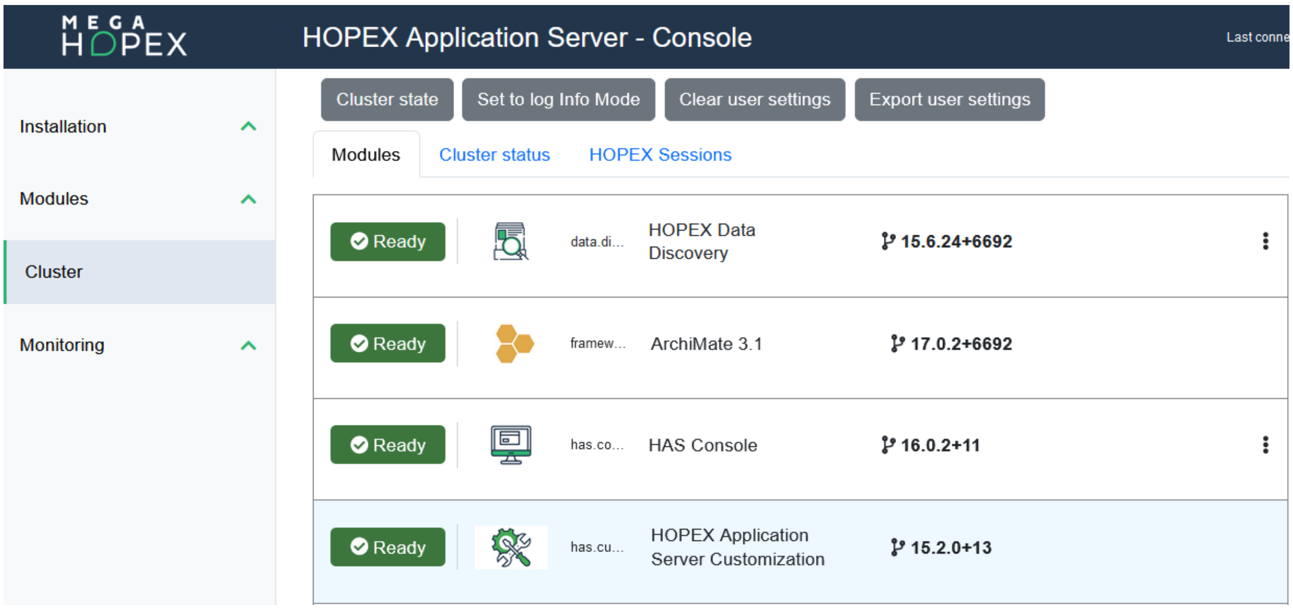
Download complementary modules from the online store or do a manual upload. Install selected module or imported file.

HOPEX Server Supervisor Module 20.0.0+0
Hopex Server Supervisor Interface Module

HOPEX Questionnaire Builder 17.0.1+521
The new Questionnaire Builder provides a completely renovated and modern user experience when it comes to building questionnaires for your assessment campaigns. Thanks to its WYSIWYG interface

Caution: the version number of this module must be higher than the previous one already deployed.

6. Click the **Cluster** menu.
7. Look for your customization module. Ensure it is ready with the proper version.
8. (If you are in cluster) Click **Cluster status** and ensure the module has the same version in all nodes.



Status	Id	Name	Version
Ready	data.di...	HOPEX Data Discovery	15.6.24+6692
Ready	framew...	ArchiMate 3.1	17.0.2+6692
Ready	has.co...	HAS Console	16.0.2+11
Ready	has.cu...	HOPEX Application Server Customization	15.2.0+13

You can now proceed to the next step: "Running Automatic update to apply your customization in repositories".

8.2.2. Uploading the module from the server (RDP)

To upload the HAS Cutomization module from the server:

1. Connect to HAS server (choose a server if you are in cluster).
2. Access the HOPEX installation folder.
3. If your instance:
   - is running, put your custom package in the following folder:

     *C:\...\HOPEX Application Server\5000\Modules**\.hot-install***

     The file will be immediately read and deployed in the right location.
   - is not running (or if you plan to restart it), put your custom package in the following folder:

     C:\...\HOPEX Application Server\5000\Modules

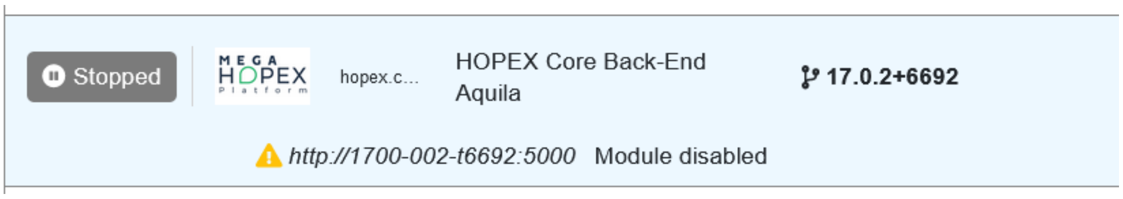
     A restart is required for the file to be properly deployed.

8.3. Running Automatic update to apply your customization in repositories

For the changes in the SystemDB or repository to be taken into account the file must be imported. The tool will take care to import the appropriate file based on folder location and tag timeline.

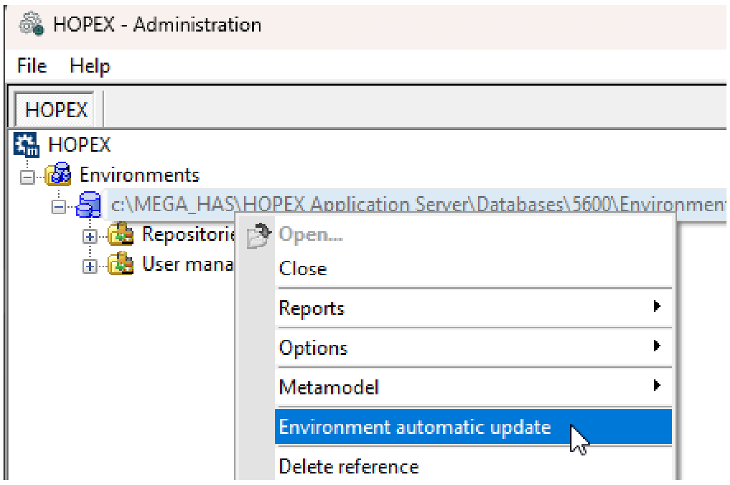
To run the Automatic update:

1. From **HAS Console**, stop the **HOPEX Core Back-End** module.

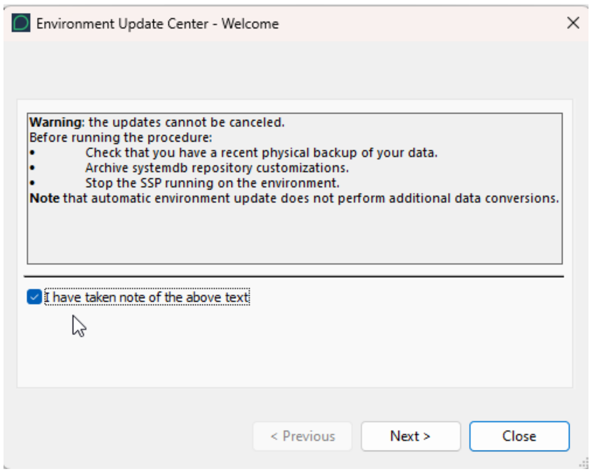


2. Launch **Administration** application (Administration.exe), from RDP access.
3. Connect to the environment.

   For example: use System user (Default password "Hopex").
4. Right-click the environment and select **Environment Automatic Update**.



5. Follow the wizard instruction. If unsure keep default values for the checkboxes.

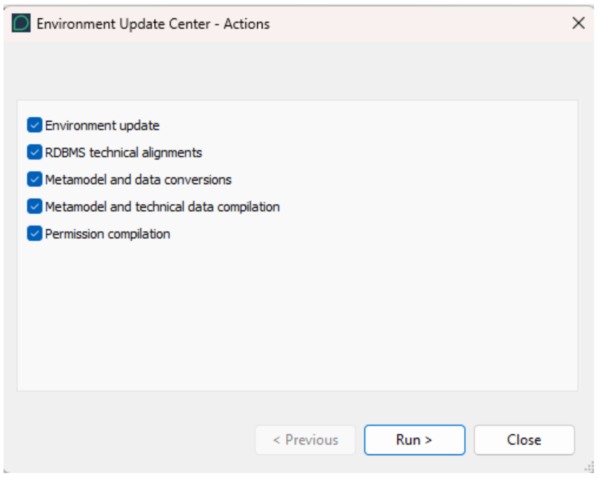

Environment Update Center - Welcome

Warning: the updates cannot be canceled.
Before running the procedure:
- Check that you have a recent physical backup of your data.
- Archive systemdb repository customizations.
- Stop the SSP running on the environment.

Note that automatic environment update does not perform additional data conversions.

☑ I have taken note of the above text

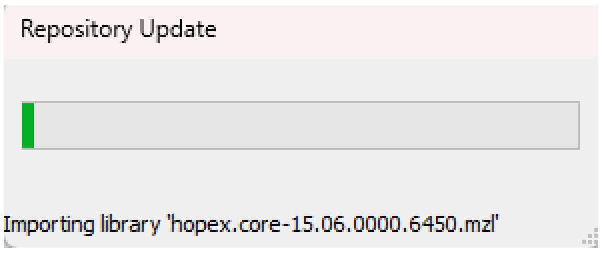
➔


Environment Update Center - Actions

☑ Environment update
☑ RDBMS technical alignments
☑ Metamodel and data conversions
☑ Metamodel and technical data compilation
☑ Permission compilation

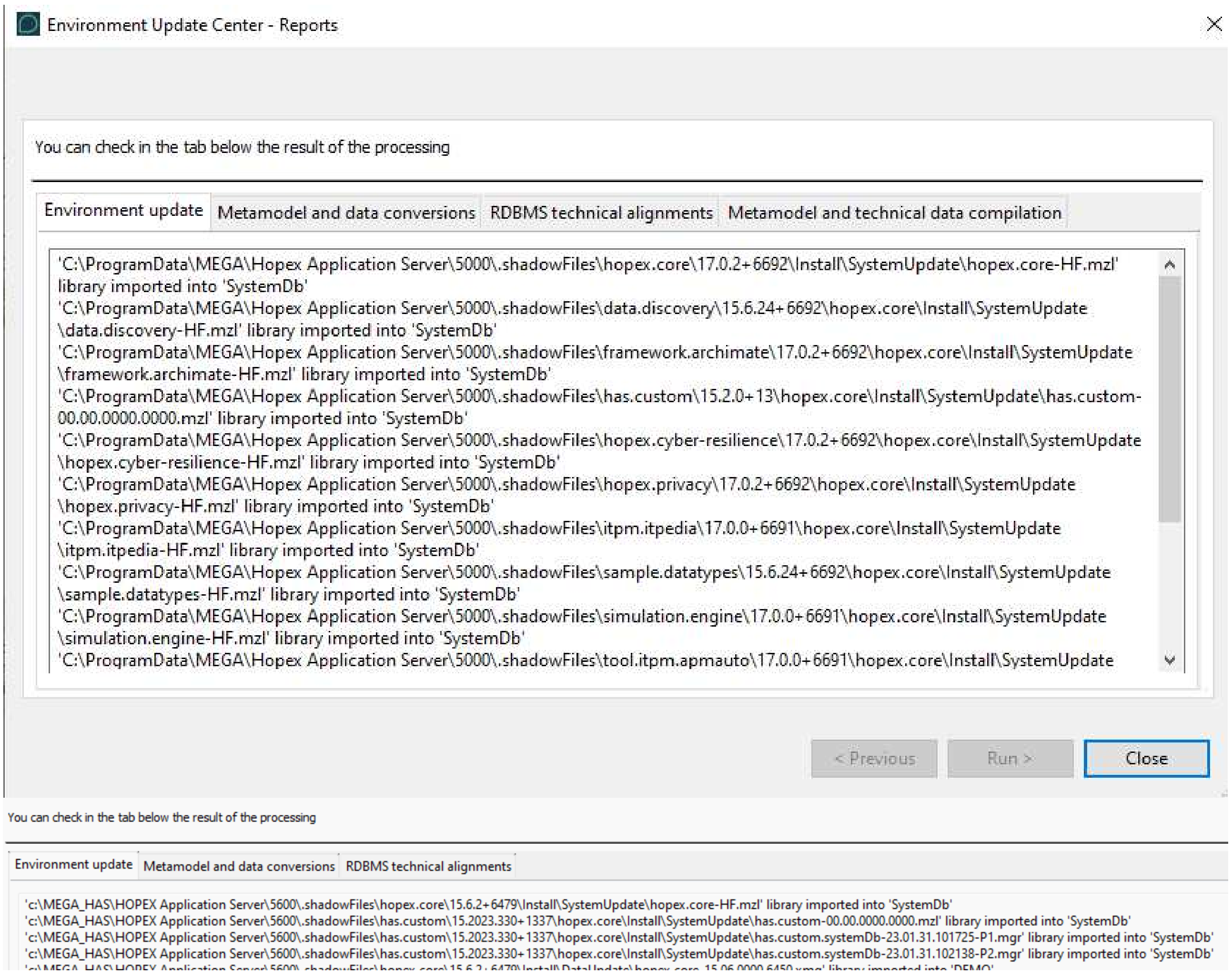
Caution: compilation of permission may take up to 1h

At that stage the system will compute all the files that need to be imported and will import them. You should see the import of your custom file.



Importing library 'hopex.core-15.06.0000.6450.mzl'

Once all the necessary steps have been performed the wizard shows you the result of the imported files. Make sure you see all the files you expect to be imported.


Environment Update Center - Reports
Environment update
Metamodel and data conversions
RDBMS technical alignments
Metamodel and technical data compilation
< Previous
Run >
Close

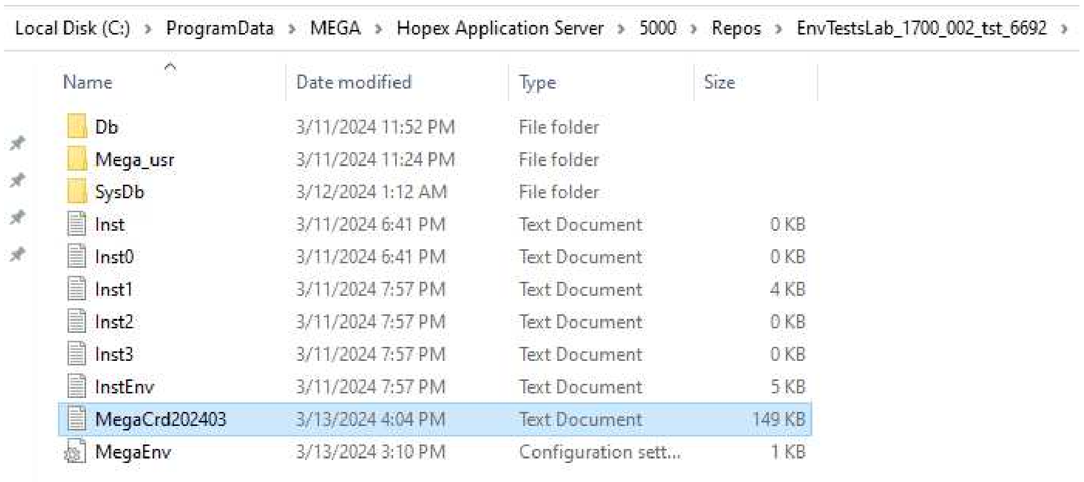
You can check in the tab below the result of the processing

```
'C:\ProgramData\MEGA\Hopex Application Server\5000\.shadowFiles\hopex.core\17.0.2+6692\Install\SystemUpdate\hopex.core-HF.mzl'
library imported into 'SystemDb'
'C:\ProgramData\MEGA\Hopex Application Server\5000\.shadowFiles\data.discovery\15.6.24+6692\hopex.core\Install\SystemUpdate
\data.discovery-HF.mzl' library imported into 'SystemDb'
'C:\ProgramData\MEGA\Hopex Application Server\5000\.shadowFiles\framework.archimate\17.0.2+6692\hopex.core\Install\SystemUpdate
\framework.archimate-HF.mzl' library imported into 'SystemDb'
'C:\ProgramData\MEGA\Hopex Application Server\5000\.shadowFiles\has.custom\15.2.0+13\hopex.core\Install\SystemUpdate\has.custom-
00.00.0000.0000.mzl' library imported into 'SystemDb'
'C:\ProgramData\MEGA\Hopex Application Server\5000\.shadowFiles\hopex.cyber-resilience\17.0.2+6692\hopex.core\Install\SystemUpdate
\hopex.cyber-resilience-HF.mzl' library imported into 'SystemDb'
'C:\ProgramData\MEGA\Hopex Application Server\5000\.shadowFiles\hopex.privacy\17.0.2+6692\hopex.core\Install\SystemUpdate
\hopex.privacy-HF.mzl' library imported into 'SystemDb'
'C:\ProgramData\MEGA\Hopex Application Server\5000\.shadowFiles\itpm.itpedia\17.0.0+6691\hopex.core\Install\SystemUpdate
\itpm.itpedia-HF.mzl' library imported into 'SystemDb'
'C:\ProgramData\MEGA\Hopex Application Server\5000\.shadowFiles\sample.datatypes\15.6.24+6692\hopex.core\Install\SystemUpdate
\sample.datatypes-HF.mzl' library imported into 'SystemDb'
'C:\ProgramData\MEGA\Hopex Application Server\5000\.shadowFiles\simulation.engine\17.0.0+6691\hopex.core\Install\SystemUpdate
\simulation.engine-HF.mzl' library imported into 'SystemDb'
'C:\ProgramData\MEGA\Hopex Application Server\5000\.shadowFiles\tool.itpm.apmauto\17.0.0+6691\hopex.core\Install\SystemUpdate
```

You can check in the tab below the result of the processing

```
'c:\MEGA_HAS\HOPEX Application Server\5600\.shadowFiles\hopex.core\15.6.2+6479\Install\SystemUpdate\hopex.core-HF.mzl' library imported into 'SystemDb'
'c:\MEGA_HAS\HOPEX Application Server\5600\.shadowFiles\has.custom\15.2023.330+1337\hopex.core\Install\SystemUpdate\has.custom-00.00.0000.0000.mzl' library imported into 'SystemDb'
'c:\MEGA_HAS\HOPEX Application Server\5600\.shadowFiles\has.custom\15.2023.330+1337\hopex.core\Install\SystemUpdate\has.custom.systemDb-23.01.31.101725-P1.mgr' library imported into 'SystemDb'
'c:\MEGA_HAS\HOPEX Application Server\5600\.shadowFiles\has.custom\15.2023.330+1337\hopex.core\Install\SystemUpdate\has.custom.systemDb-23.01.31.102138-P2.mgr' library imported into 'SystemDb'
```

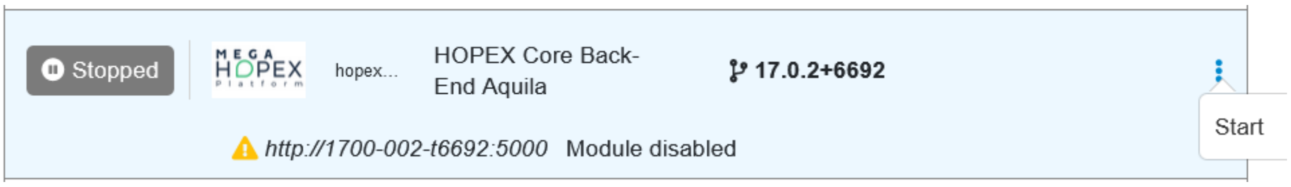
*The content of this wizard is stored in the **MegaCrd.txt** file located in the environment folder.*



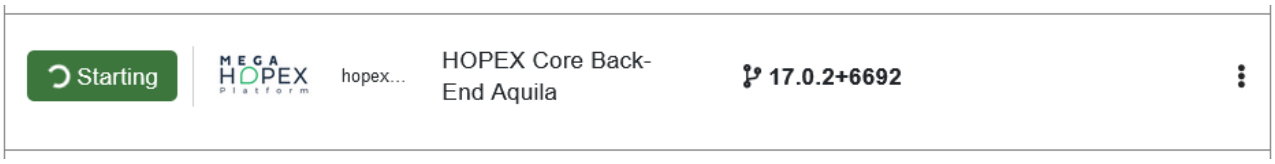
Local Disk (C:) › ProgramData › MEGA › Hopex Application Server › 5000 › Repos › EnvTestsLab_1700_002_tst_6692 ›

Name	Date modified	Type	Size
Db	3/11/2024 11:52 PM	File folder	
Mega_usr	3/11/2024 11:24 PM	File folder	
SysDb	3/12/2024 1:12 AM	File folder	
Inst	3/11/2024 6:41 PM	Text Document	0 KB
Inst0	3/11/2024 6:41 PM	Text Document	0 KB
Inst1	3/11/2024 7:57 PM	Text Document	4 KB
Inst2	3/11/2024 7:57 PM	Text Document	0 KB
Inst3	3/11/2024 7:57 PM	Text Document	0 KB
InstEnv	3/11/2024 7:57 PM	Text Document	5 KB
MegaCrd202403	3/13/2024 4:04 PM	Text Document	149 KB
MegaEnv	3/13/2024 3:10 PM	Configuration sett...	1 KB

6. Restart the **HOPEX Core Back-End** module.



Stopped
hopex...
HOPEX Core Back-End Aquila
17.0.2+6692
http://1700-002-t6692:5000 Module disabled
Start



You have now successfully completed the full workflow to push your customization to STAGE or PRODUCTION. It is now time to test that your changes are taken effectively.

9. Managing a Team to Move from DEV to PROD

Depending on the size of your company and the complexity of topics you are working on you might choose one way or the other of working.

9.1. Most common (recommended) working way

9.1.1. Principle

The most common way of working when dealing with customization is the following:

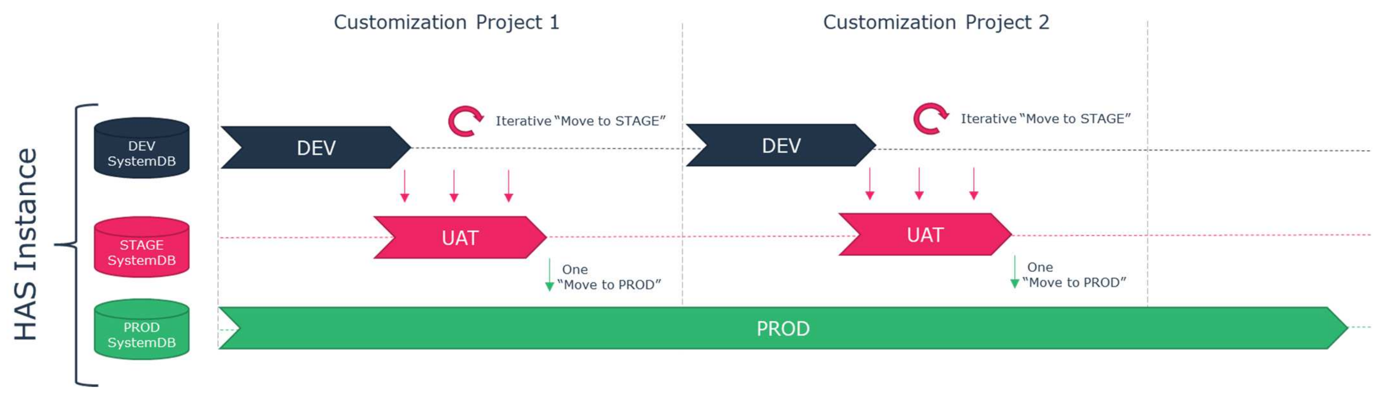
- You have one DEV platform with one DEV Instance.
- You have one team of people performing all of the customizations.
- All customizations are done with this unique SystemDb.
- All customizations are done using Work Item with a defined naming convention.
- If you have multiple projects, the delivery dates are sequential: one project after the other.

Benefit:

You have a unique source of customization, and all customizations are always coherent among them.

Drawback:

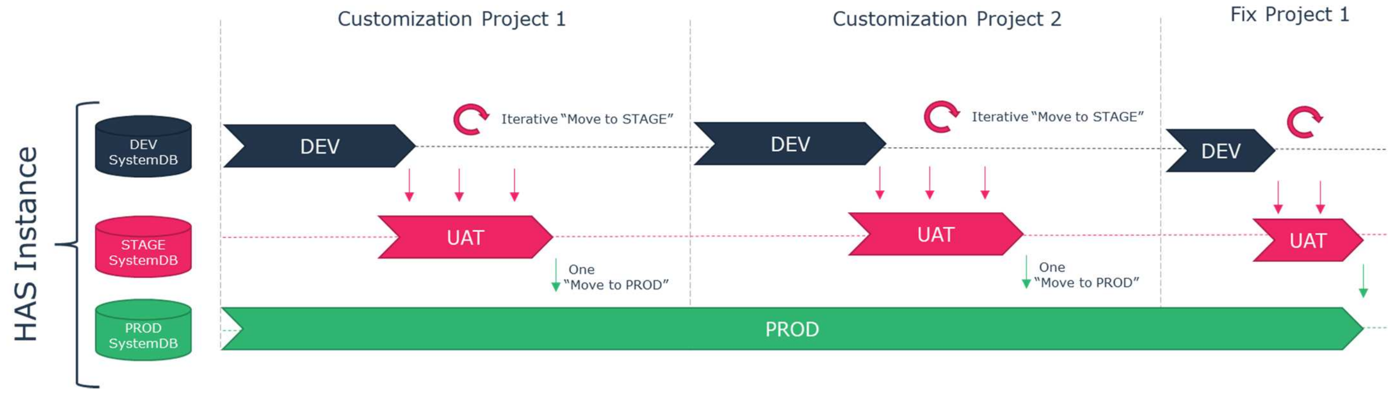
In case of multiple projects, it requires to align sequentially project timeline and delivery.



9.1.2. Dealing with iterative fix

If you have multiple projects, you need to fix customization from previous project. In that case two solutions are possible:

- The fixes of project 1 are included in project 2.
- The fixes of project 1 are done at the end of project 2.



9.2. Multiple team – Multiple Projects – Multiple DEV Instance

In some context you may have:

- Several teams that can perform customizations.
- Several projects with overlapping delivery timeline.

In that context you may not be able to manage your work with only one DEV instance. You need to have multiple DEV instances.

We recommend you try everything to have only one DEV instance

In that situation we recommend having a very rigorous organization and precise delivery time. The key point for such way of working is to ensure each project is totally independent from each other from a functional point of view: no overlaps in customization of objects.

The way of working when dealing with multiple parallel project:

- You have one DEV platform for each project with one DEV instance
- You have one centre DEV Instance to integrate each DEV from each project
- Each project works in their respective DEV instance.

Benefit:

You handle complex organization working on various projects and topics at the same time.

Drawback:

You need to be rigorous in the merging or overlapping customization to avoid regression.

9.2.2. Architecture of multiple project

In such architecture you have first a similar architecture as described in "Most common (recommended) working way" section.

- A DEV instance, a UAT instance, a PROD instance

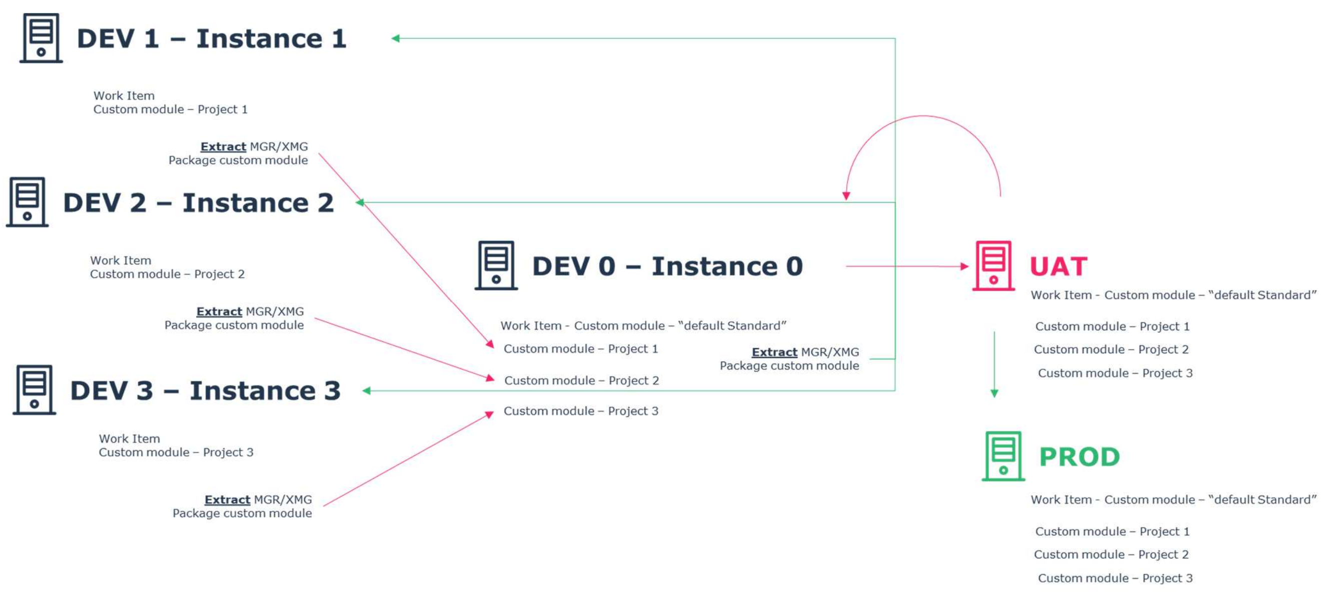
The DEV instance "zero" will be used to put all common customizations that apply to all projects (e.g.: login background, css stylesheet changes, common jar or DLL).

You have then a "DEV instance" for each project: DEV instance 1, DEV instance 2, DEV instance 3…

Caution: *project means functional scope such as Audit, ITPM, BPA. Technical split (e.g.: static website, metamodel, UX) is not suitable for this approach.*

In each project you must create a "project custom module". This module is unique to each project with a **unique module ID**.

The "default standard" custom module has to be replicated on each project DEV instance 1, 2 and 3.


DEV 1 – Instance 1
Work Item
Custom module – Project 1
Extract MGR/XMG
Package custom module
DEV 2 – Instance 2
Work Item
Custom module – Project 2
Extract MGR/XMG
Package custom module
DEV 3 – Instance 3
Work Item
Custom module – Project 3
Extract MGR/XMG
Package custom module
DEV 0 – Instance 0
Work Item - Custom module – "default Standard"
Custom module – Project 1
Custom module – Project 2
Custom module – Project 3
Extract MGR/XMG
Package custom module
UAT
Work Item - Custom module – "default Standard"
Custom module – Project 1
Custom module – Project 2
Custom module – Project 3
PROD
Work Item - Custom module – "default Standard"
Custom module – Project 1
Custom module – Project 2
Custom module – Project 3

The process to follow is:

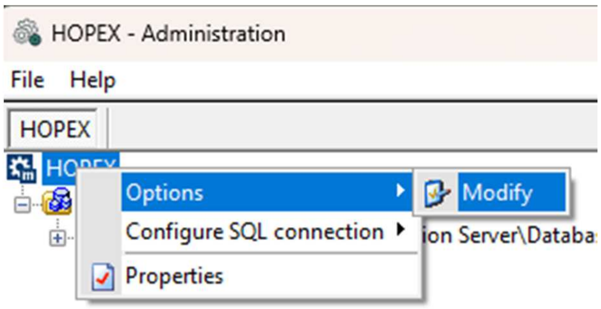
1. For project 1, perform all your customizations in "DEV instance 1".
2. Once ready, push your customizations to "DEV instance 0".
3. Perform a first test of validity of your customizations, to check particularly any side effect on other project.
4. If no impact, continue with the normal Move to PROD process.

9.2.3. How to create your custom project module

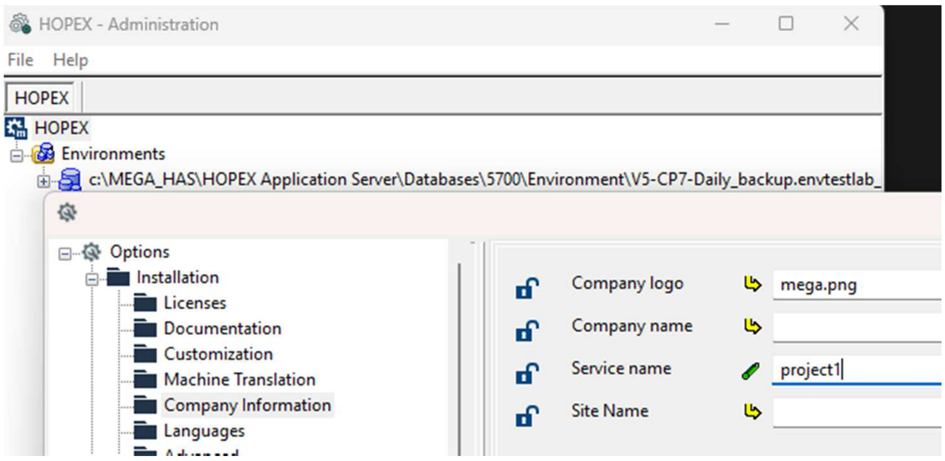
On each DEV instance 1, 2, and 3 you must give a name to your project.

To give a name to your project:

1. Launch Administration.exe.
2. Access the Site options.



3. Access **Installation > Company Information** options.



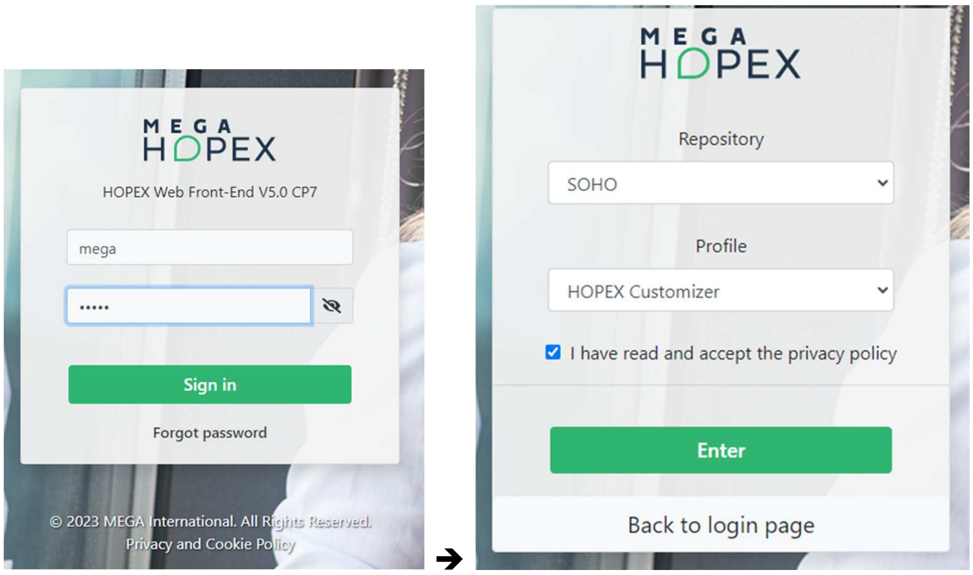
Company logo	mega.png
Company name	
Service name	project1
Site Name	

4. In **Service Name** enter your project name.
5. Cick **OK**.
6. Close Administration.exe.

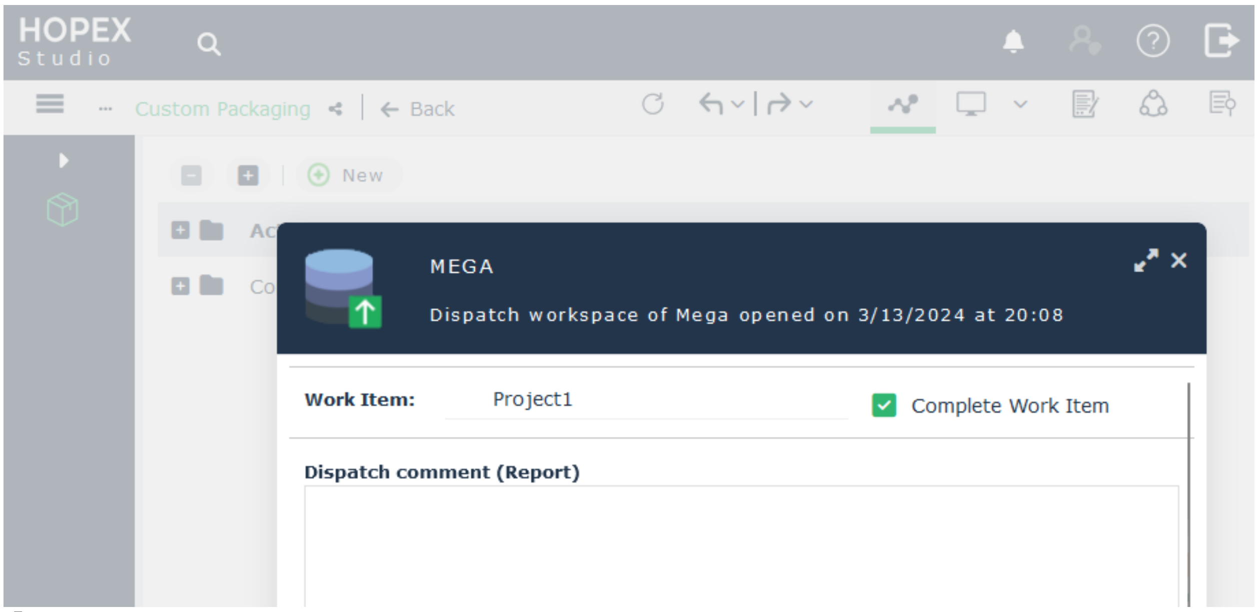
Avoid specific charset and space

Generate your first extract to create the module:

1. Connect to **HOPEX Studio** (e.g.: with user MEGA, profile “HOPEX Customizer”).


MEGA HOPEX
HOPEX Web Front-End V5.0 CP7
mega
•••••
Sign in
Forgot password
© 2023 MEGA International. All Rights Reserved.
Privacy and Cookie Policy
MEGA HOPEX
Repository
SOHO
Profile
HOPEX Customizer
I have read and accept the privacy policy
Enter
Back to login page

2. Perform the task as described in chapter "Extracting Customization".
3. Create a "Project1" Work Item and set it as complete and extract it



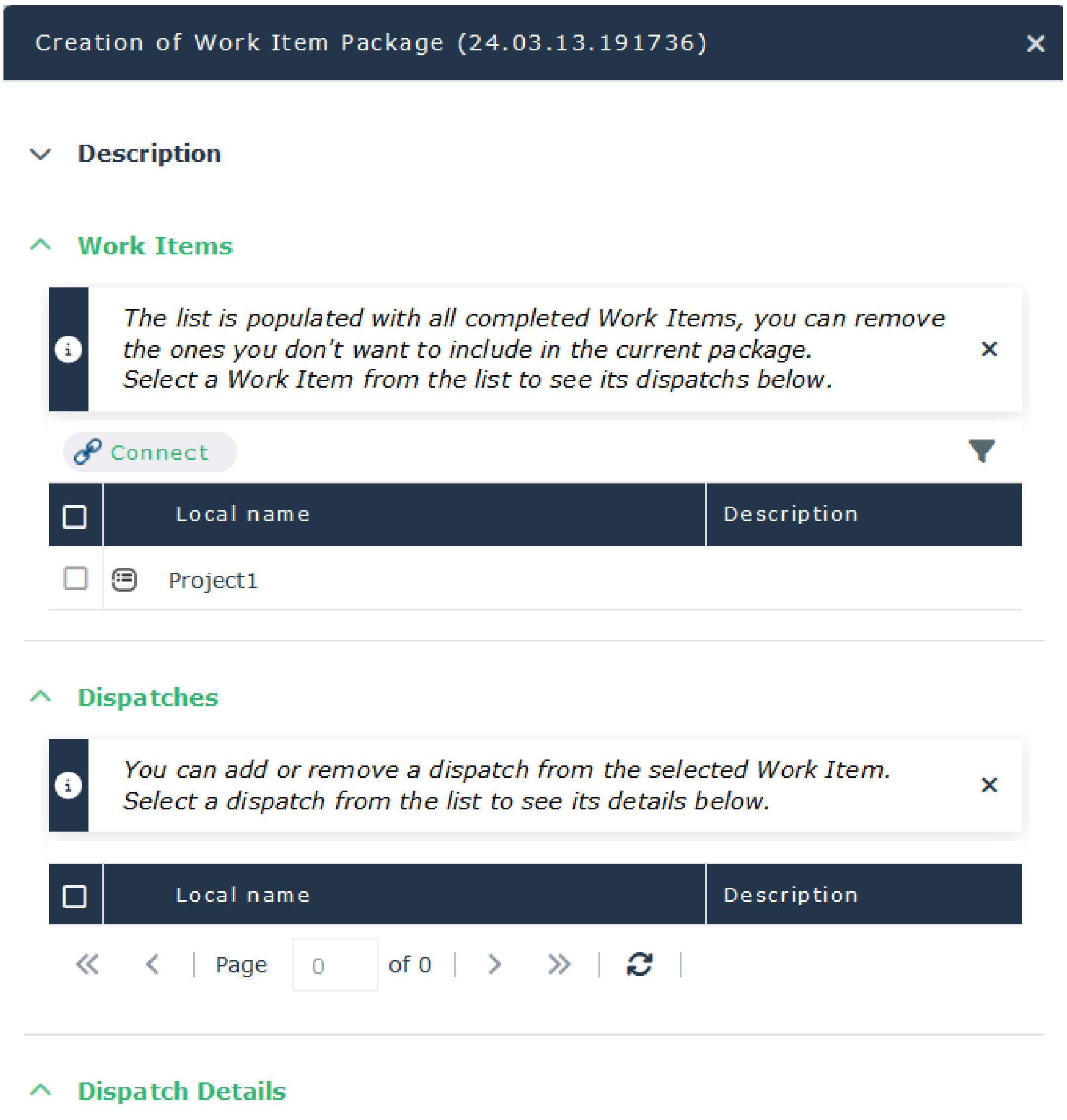
MEGA

Dispatch workspace of Mega opened on 3/13/2024 at 20:08

Work Item: Project1 ✓ Complete Work Item

Dispatch comment (Report)

➔



Creation of Work Item Package (24.03.13.191736)

Description

Work Items

The list is populated with all completed Work Items, you can remove the ones you don't want to include in the current package.
Select a Work Item from the list to see its dispatchs below.

☐	Local name	Description
☐	Project1	

Dispatches

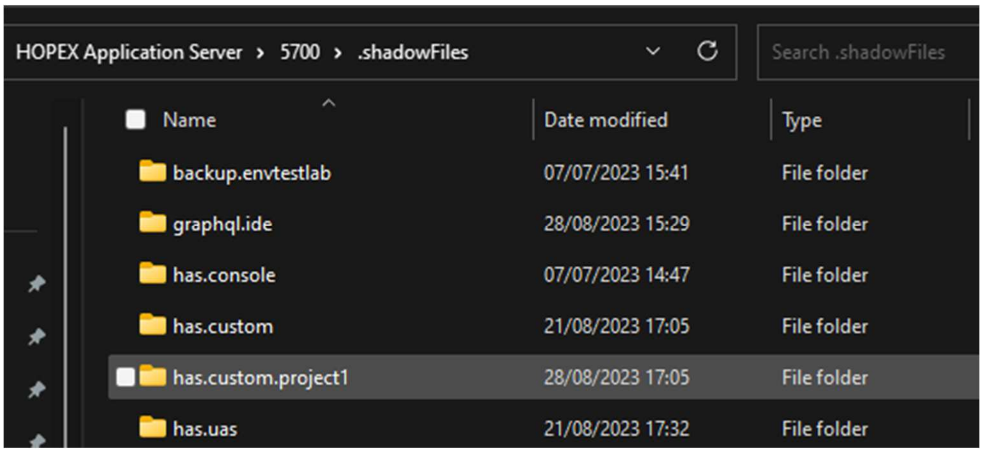
You can add or remove a dispatch from the selected Work Item.
Select a dispatch from the list to see its details below.

☐	Local name	Description

Page 0 of 0

Dispatch Details

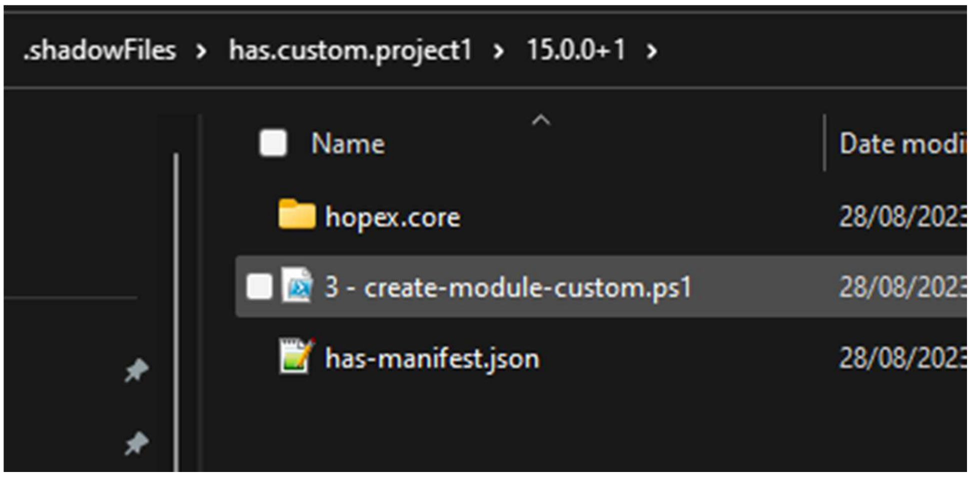
A new folder appears in shadowFiles: "**Has.custom.project1**"



HOPEX Application Server › 5700 › .shadowFiles

Name	Date modified	Type
backup.envtestlab	07/07/2023 15:41	File folder
graphql.ide	28/08/2023 15:29	File folder
has.console	07/07/2023 14:47	File folder
has.custom	21/08/2023 17:05	File folder
has.custom.project1	28/08/2023 17:05	File folder
has.uas	21/08/2023 17:32	File folder

4. Perform the task as described in chapter "Packaging customization".



.shadowFiles › has.custom.project1 › 15.0.0+1 ›

Name	Date modi
hopex.core	28/08/2023
3 - create-module-custom.ps1	28/08/2023
has-manifest.json	28/08/2023

A new custom module project 1 is now available for you to put all the customizations for this specific project.

9.2.4. Workflow to manage the project

You must follow these rules:

- On "DEV instance 0" the custom module is the reference.

  This module must always be up-to-date and replicated in all "DEV instance 1, 2, and 3.
- For a project push the custom module to "DEV instance 0" first.
- If you want to share element across project, push the project other project instance.

10. Migrating to HOPEX V5?

If you are an existing customer of HOPEX, you will be migrating your environment to HOPEX V5.

You must follow the following rules:

- The custom package is now mandatory starting from HOPEX V5 onward.
- All resources previously contained in **Mega_Usr** folder must now be stored in the custom module.

For the customization already imported within the SystemDb. You have 2 possibilities:

Option 1: Use the existing customization

This means:

- Let the customizations within the SystemDB repository.
- Add all of the new customizations (even modifications of existing customization) following the process described in this document.

Benefit:

Safe to avoid regression on customization.

Drawback:

The custom module does not contain "all" the customizations since "Day-0".

Option 2: Rebuilt the environment

This means:

- Start from a totally new environment (with no customization).
- Recreate previous customization and package them following the process described in this document.
- Any new customizations must follow the process described in this document.

Benefit:

Safe to start from a "clean" ground and get rid of the past.

Drawback:

Takes more time and a is bit riskier in terms of regression.

11. Frequently Asked Questions

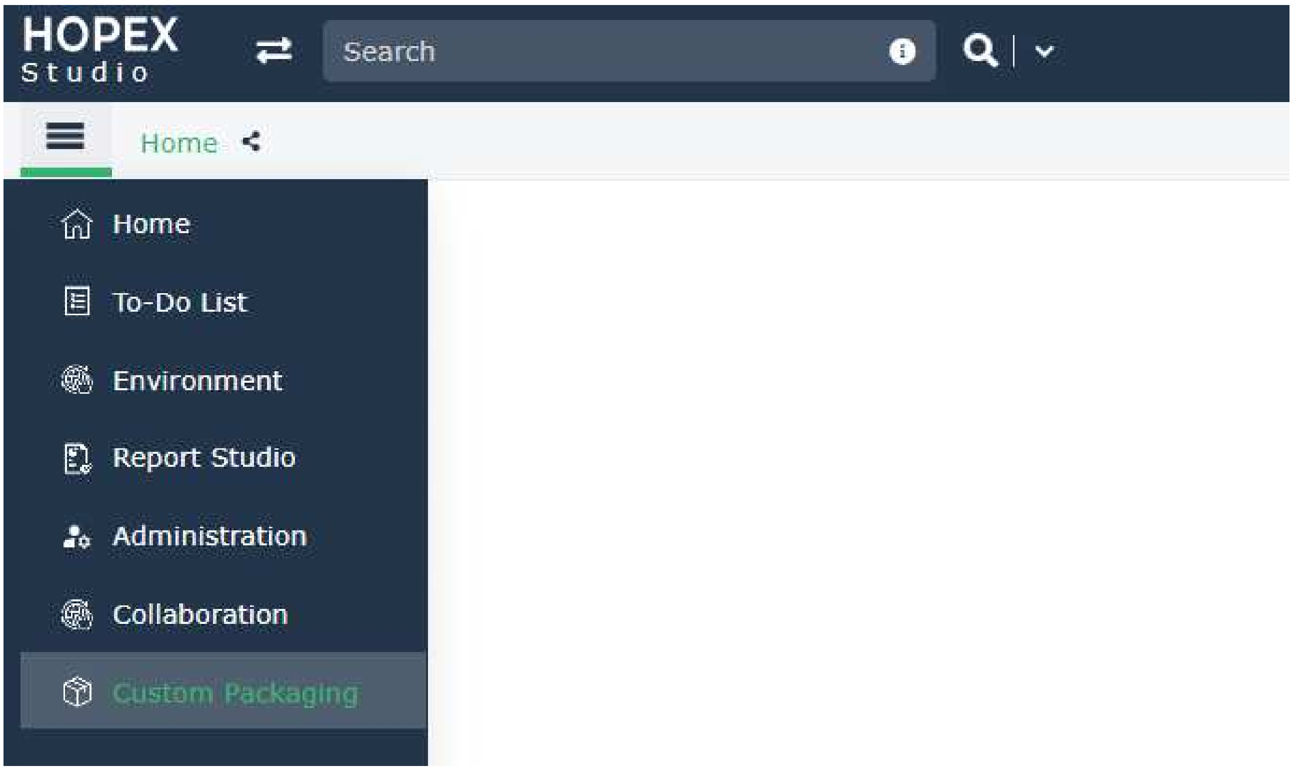
11.1.1. The Work Item dropdown is not available when dispatching

Your HAS instance is not in DEV mode. Customization cannot be performed in STAGE or PROD.

11.1.2. The Custom Packaging menu content is not available to extract my customization

Either:

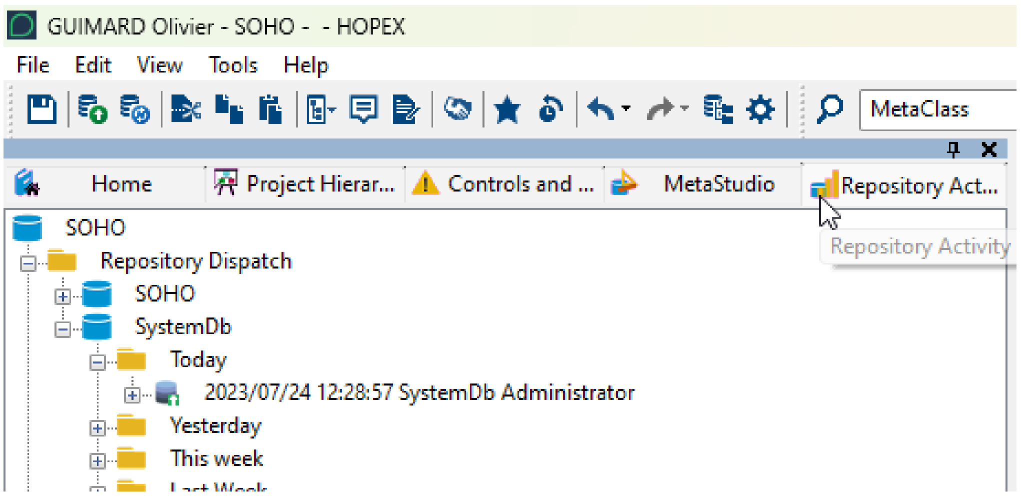
- your HAS instance is not in DEV mode.
- your HAS instance is in DEV mode, but you forgot to launch the automatic update after adding the **has.custom** module.



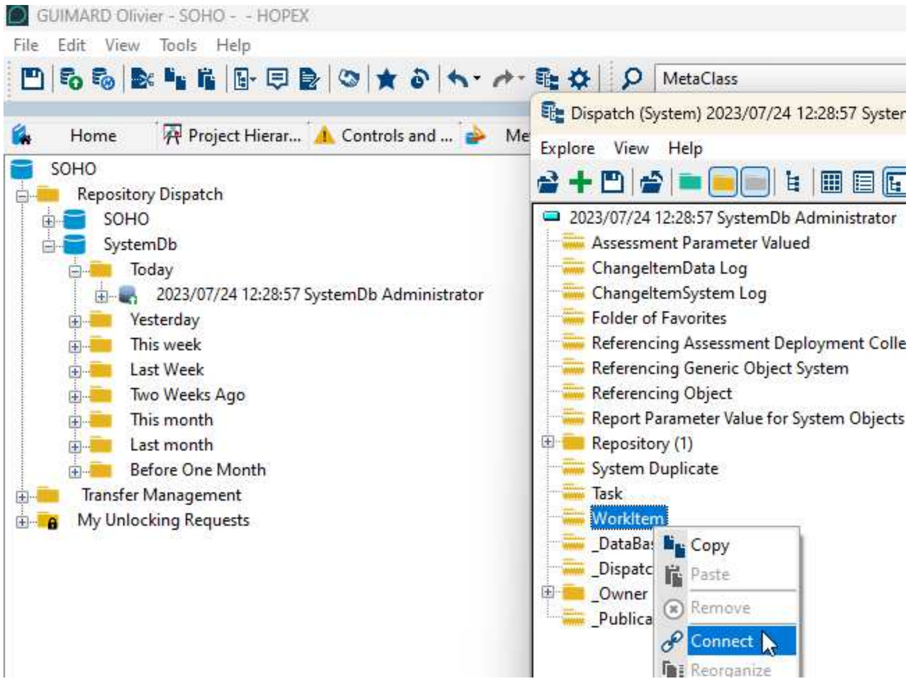
11.1.3. I forgot to attach my dispatches to a Work Item, what can I do?

You can connect to HOPEX.exe and manually connect the required dispatch to the Work Item you want.

1. In the **Repository Activity** tab, search for your dispatch.



2. Right-click the dispatch and select **Explore**.
3. In the yellow folders, right-click the **Work Item** folder and select **Connect**.



4. Select the work item you want to attached it to.

Select only active work item.

11.1.4. I have embedded demo data in my dispatch, how can I remove them?

You can remove this unwanted data either:

- Case 1: by removing the dispatch from the Work Item.
- Case 2: by removing the dispatch at extraction phase.

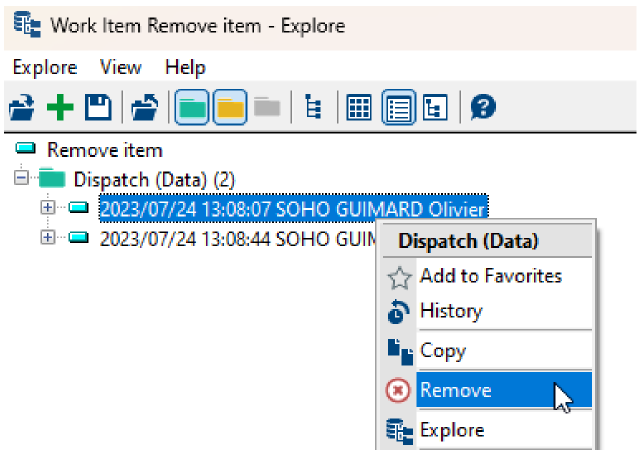
  See “Extracting Customization” chapter.

- Case 3: by removing the files generated in the .shadowfiles folder.

Note: this removal action can only be done prior to import in STAGE or PROD.

Case 1: Removing the dispatch from the work item

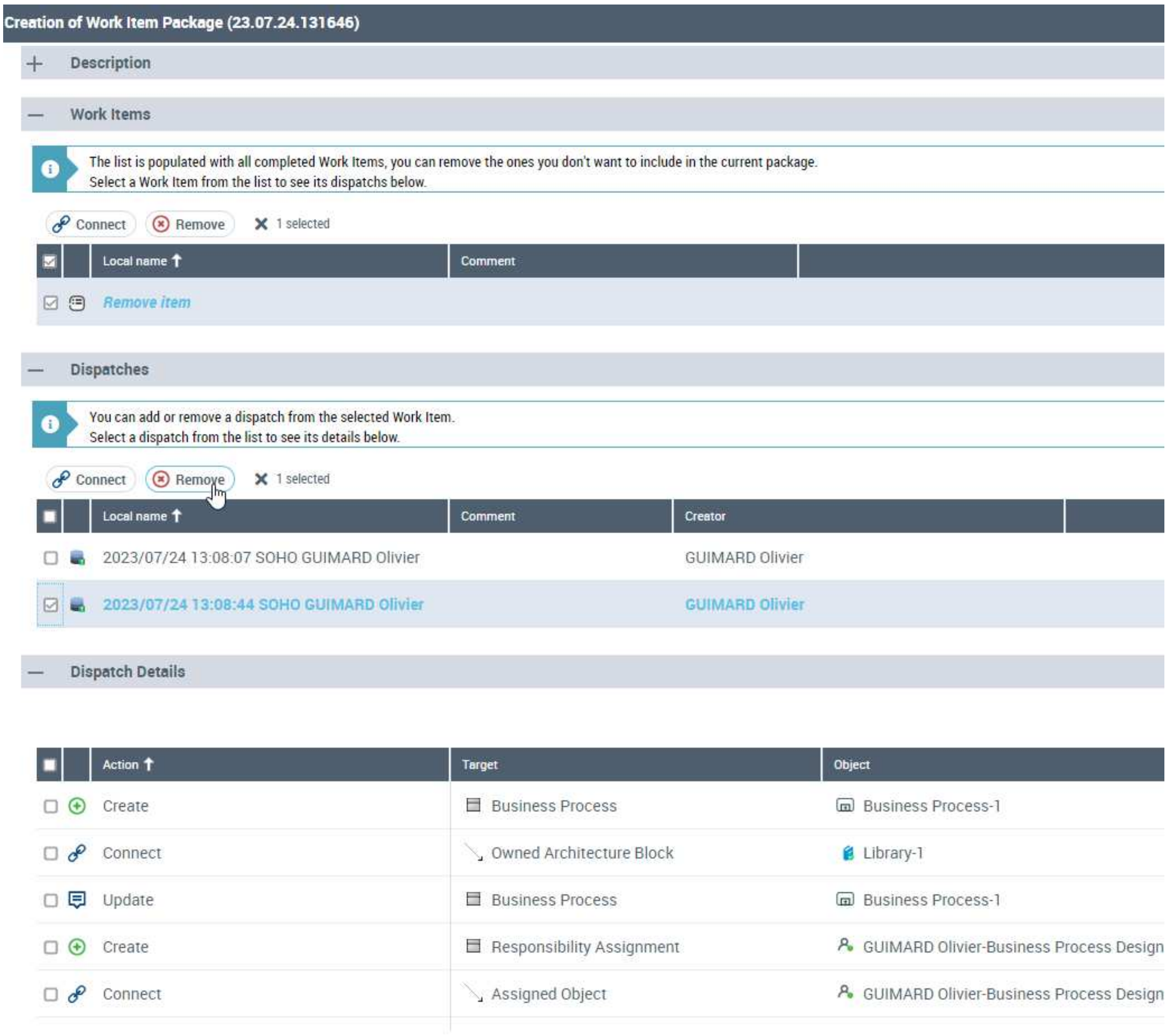
1. Search for your Work Item in the repository.
2. Right-click your Work Item and select **Explore**.
3. Search for **Dispatch (Data)** folder (Green folder).
4. Right-click the Dispatch that contains the data you want to remove and select **Remove**.


Work Item Remove item - Explore
Explore
View
Help
Remove item
Dispatch (Data) (2)
2023/07/24 13:08:07 SOHO GUIMARD Olivier
2023/07/24 13:08:44 SOHO GUIM
Dispatch (Data)
Add to Favorites
History
Copy
Remove
Explore

You can only remove the full dispatch with all its content. If you want to delete only one item of the dispatch, you must delete the object and dispatch this deletion on a new dispatch.

Case 2: Removing the dispatch at extraction.

1. From HOPEX Administration desktop, in the Customization extraction wizard.
2. In the **Dispatches** section, select the dispatch you do not want.
3. Click **Remove**.
4. Click **Generate**.



Creation of Work Item Package (23.07.24.131646)

+ Description

— Work Items

The list is populated with all completed Work Items, you can remove the ones you don't want to include in the current package.
Select a Work Item from the list to see its dispatchs below.

Connect | Remove | ✕ 1 selected

	Local name	Comment
✓	Remove item	

— Dispatches

You can add or remove a dispatch from the selected Work Item.
Select a dispatch from the list to see its details below.

Connect | Remove | ✕ 1 selected

	Local name	Comment	Creator
☐	2023/07/24 13:08:07 SOHO GUIMARD Olivier		GUIMARD Olivier
✓	2023/07/24 13:08:44 SOHO GUIMARD Olivier		GUIMARD Olivier

— Dispatch Details

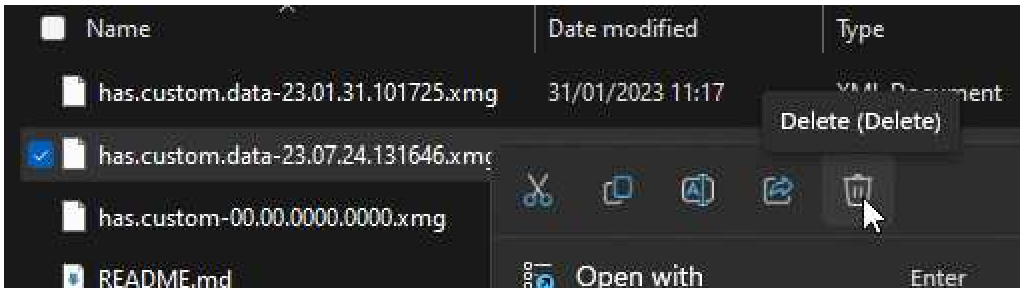
Action	Target	Object
Create	Business Process	Business Process-1
Connect	Owned Architecture Block	Library-1
Update	Business Process	Business Process-1
Create	Responsibility Assignment	GUIMARD Olivier-Business Process Design
Connect	Assigned Object	GUIMARD Olivier-Business Process Design

Case 3: Removing the dispatch at extraction.

1. In Windows file explorer go to the folder:

   *C:\...\HOPEX Application Server\5000\.shadowFiles\has.custom\<module version>\hopex.core\Install\**DataUpdate***

2. Select the file that contains the data you do not need and **Delete** it.



Name	Date modified	Type
has.custom.data-23.01.31.101725.xmg	31/01/2023 11:17	XML Document
✓ has.custom.data-23.07.24.131646.xmg		
has.custom-00.00.0000.0000.xmg		
README.md		

11.1.5. How can I remove imported XMG/MGR in stage or prod?

Once the automatic update has run and imported element in the SystemDb or repository you cannot delete them with an automated process.

You can either:

- Do a new custom module that contains the delete instructions ➔ **preferred choice.**
- Do a manually removal of the item directly in the repository

11.1.6. How to force re-import of MGR/XMG files?

Once a file has been imported, the automatic update wizard cannot re-import it again. This is due to a marker contained in the file and within the repository that says the file has been already imported.

The recommended solution is to manually re-import the file you want, by following the proper order from Administration.exe.

11.1.7. My custom files got imported with errors, What should I do?

You should create a newer version of the custom module with all the missing items (creation, connection…).

11.1.8. I did customization but I don't have the initial custom module, what should I do?

The HAS Custom module should contains all the customization from "Day-0". Before performing any additional customization you must start from the last published custom module.

11.1.9. Can I launch the automatic update without service interruption?

No, for the moment the automatic update must be performed with the **HOPEX Core Back-End** module stopped.

11.1.10. Is delta custom module package possible ?

No, the custom module must contain all files and XMG/MGR from all time (since Day-0 of customization).

The End

HOPEX diagrams in MS Teams

1. Teams Module Description

The **Teams** module enables to share MEGA HOPEX diagrams in Microsoft Teams.

Add a MEGA HOPEX diagram as a tab in your Teams channel, with an embedded editor, anyone in the channel can view and comment on your diagram without ever leaving Microsoft Teams. You can also edit your HOPEX diagram from a tab if you have a MEGA HOPEX license.

Sharing diagrams in Teams will help you to onboard your team members as subject matter expert in your transformation project.

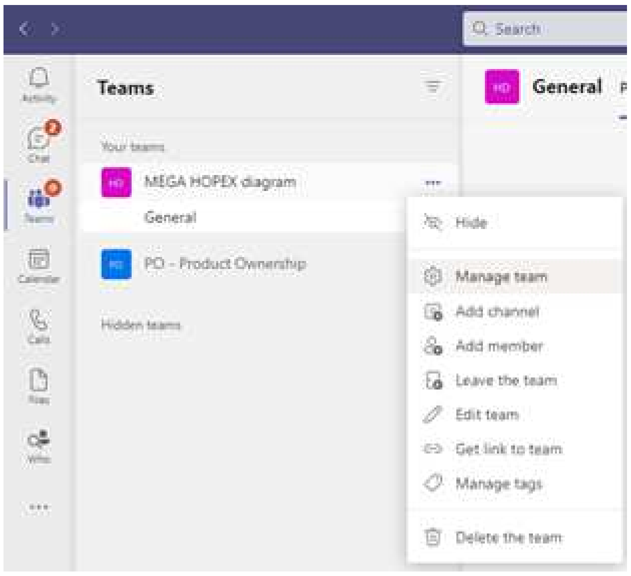
2. HOPEX in Microsoft Teams

If this is the first time you use the app to embed HOPEX in Microsoft Teams, go to HOPEX Configuration section and follow the step-by-step instructions to configure the module.

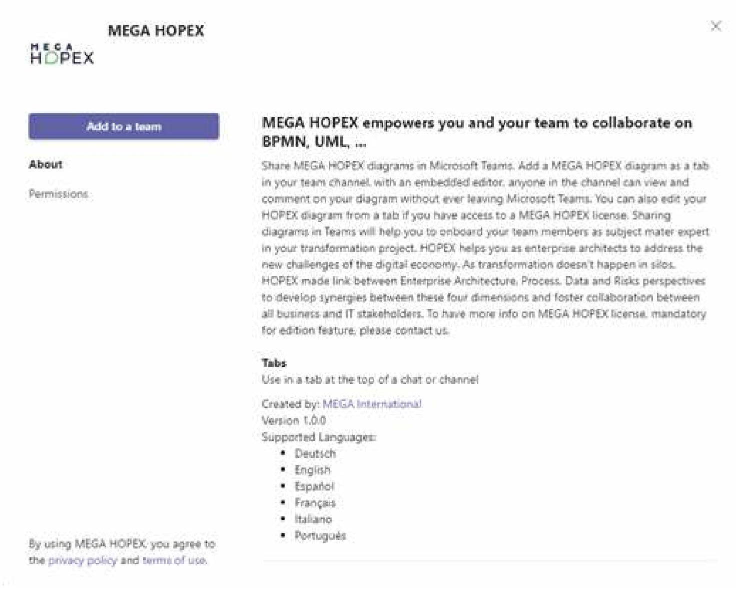
2.1. Adding the App to your Teams (optional)

In your Teams:

1. Open the menu to manage your team.



2. Select the **Apps** tab.
3. Search for **MEGA HOPEX** app.


MEGA HOPEX

Add to a team

About

Permissions

MEGA HOPEX empowers you and your team to collaborate on BPMN, UML, ...

Share MEGA HOPEX diagrams in Microsoft Teams. Add a MEGA HOPEX diagram as a tab in your team channel, with an embedded editor, anyone in the channel can view and comment on your diagram without ever leaving Microsoft Teams. You can also edit your HOPEX diagram from a tab if you have access to a MEGA HOPEX license. Sharing diagrams in Teams will help you to onboard your team members as subject mater expert in your transformation project. HOPEX helps you as enterprise architects to address the new challenges of the digital economy. As transformation doesn't happen in silos, HOPEX made link between Enterprise Architecture, Process, Data and Risks perspectives to develop synergies between these four dimensions and foster collaboration between all business and IT stakeholders. To have more info on MEGA HOPEX license, mandatory for edition feature, please contact us.

Tabs
Use in a tab at the top of a chat or channel

Created by: MEGA International
Version 1.0.0
Supported Languages:
- Deutsch
- English
- Español
- Français
- Italiano
- Português

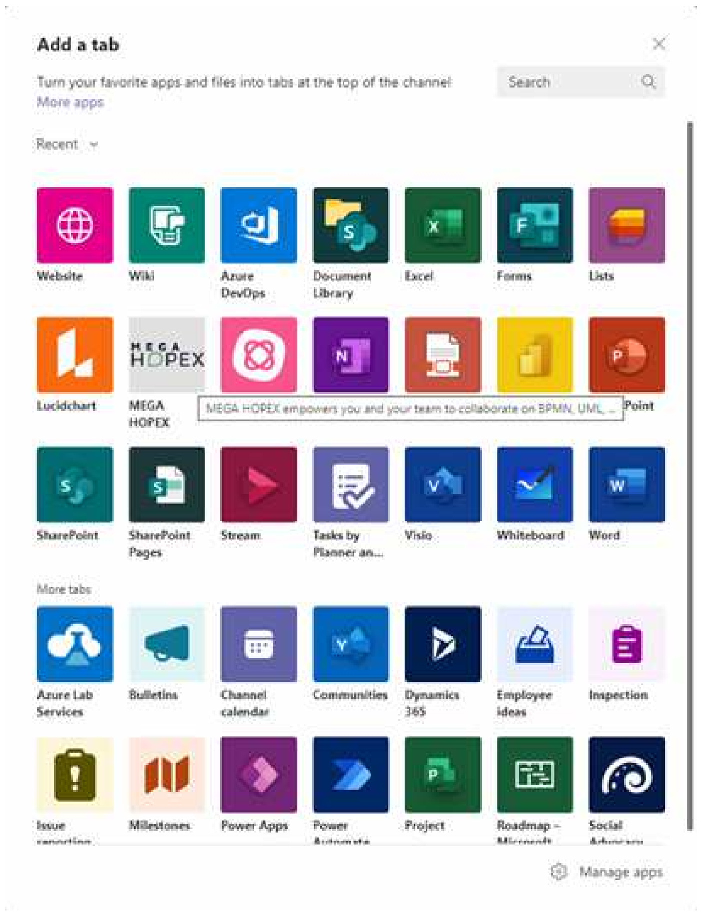
By using MEGA HOPEX, you agree to the privacy policy and terms of use.

4. Click **Add**.

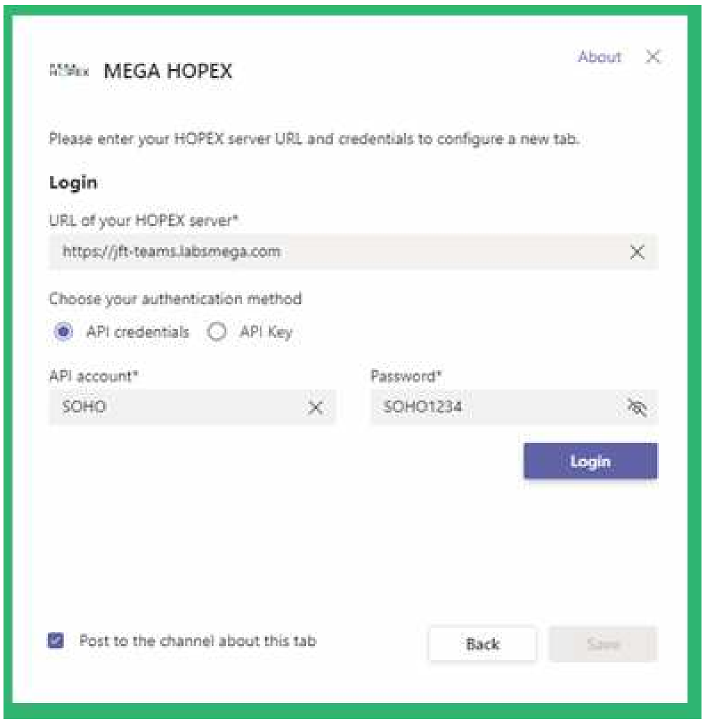
2.2. Adding a diagram tab in your Team

To add a tab on a channel:

1. Click the plus sign.
2. Select **MEGA HOPEX** app.



3. Fill in the **Login** information given by your HOPEX administrator (see Authentication):
   - URL of your HOPEX server
   - API credentials/API Key
4. Click **Login**.


MEGA HOPEX
About

Please enter your HOPEX server URL and credentials to configure a new tab.

Login

URL of your HOPEX server*: https://jft-teams.labsmega.com

Choose your authentication method: [✓ API credentials] API Key

API account*: SOHO

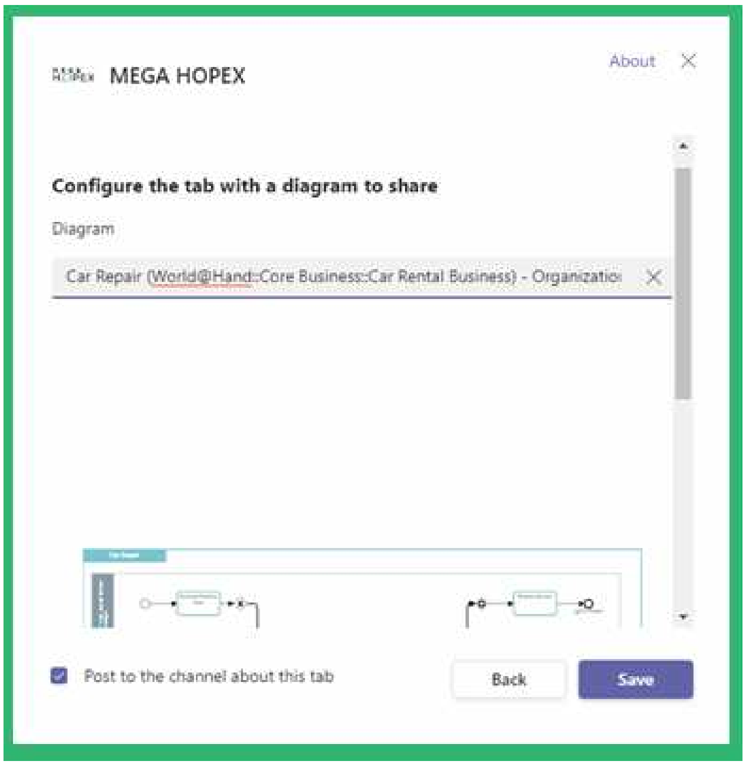
Password*: SOHO1234

Login

[✓] Post to the channel about this tab

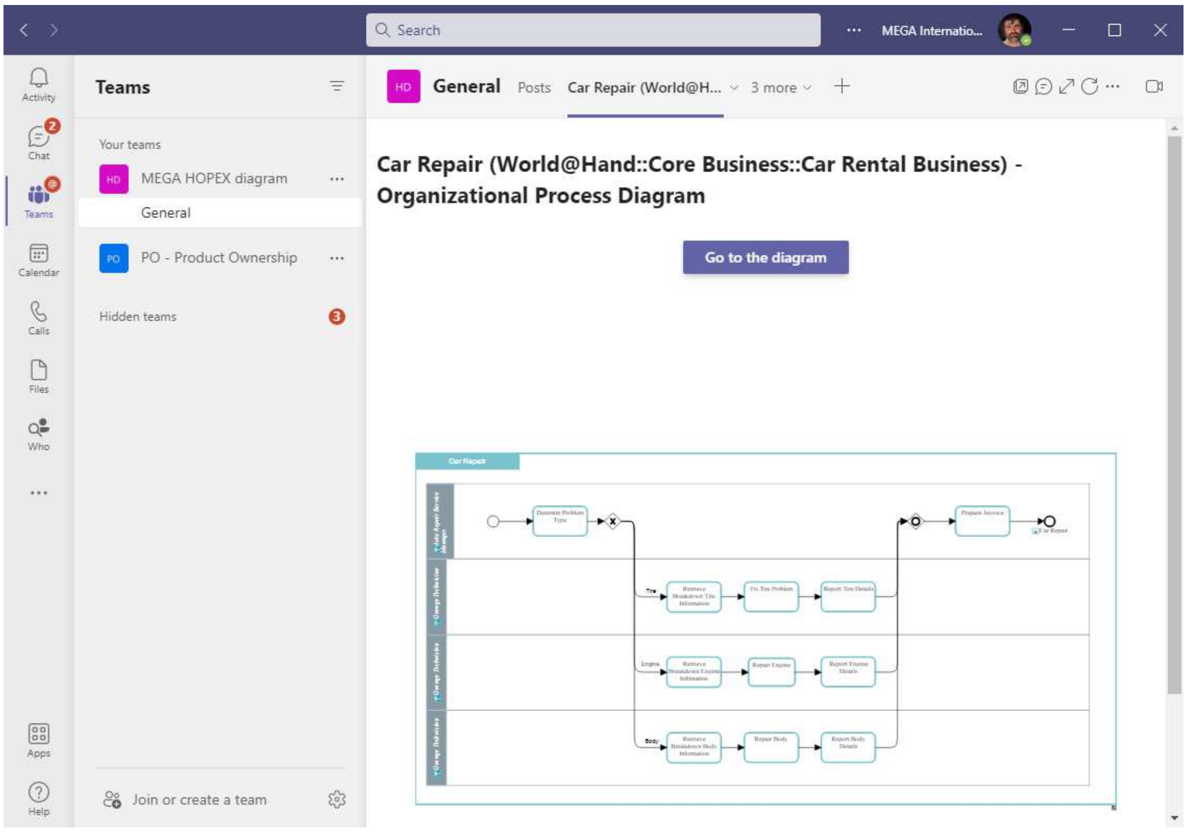
Back Save

5. Select a Diagram:
   a) Search for the diagram in the list or enter its name (e.g. Car Repair)
   b) Select the diagram.
   c) Use the preview to check if it is the diagram you want to share.
   d) Click **Save** to create the tab corresponding to the diagram.


MEGA HOPEX
About
Configure the tab with a diagram to share
Diagram
Car Repair (World@Hand::Core Business::Car Rental Business) - Organizatio
Post to the channel about this tab
Back
Save

2.3. Viewing a HOPEX diagram

On the channel select the diagram you want to see. Every team member can view the shared diagram.

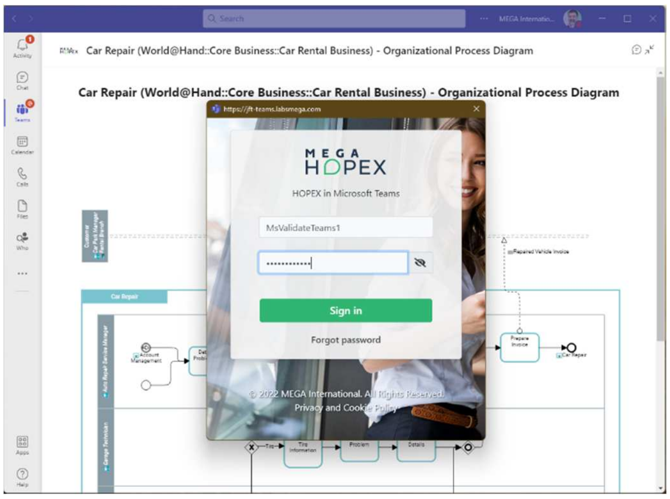


2.4. Editing the HOPEX diagram

To edit the HOPEX diagram:

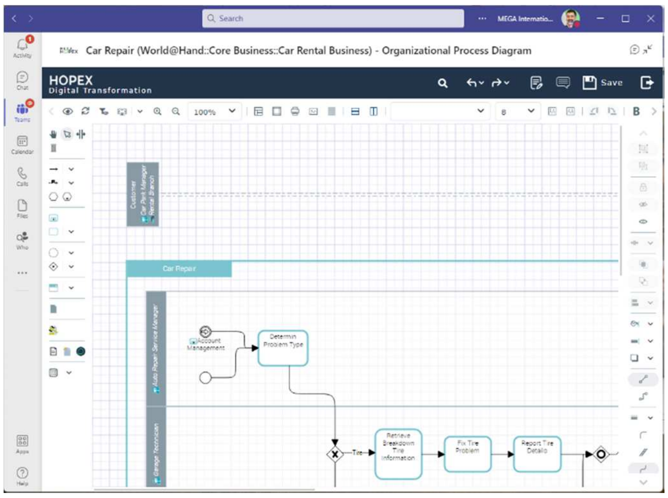
1. In the channel, select the diagram you want to edit.
2. Click **Go to the diagram**.

   A popup appears.


Car Repair (World@Hand::Core Business::Car Rental Business) - Organizational Process Diagram
HOPEX in Microsoft Teams
MsValidateTeams1
Sign in
Forgot password

3. Enter your HOPEX credential.
   According to your organization, it can be a simple authentication, OpenID Connect, or SAML2.
   HOPEX can leverage your company authentication system.
4. Click **Sign in**.

   HOPEX loads into a frame embedded into Microsoft Teams.


Car Repair (World@Hand::Core Business::Car Rental Business) - Organizational Process Diagram
HOPEX
Digital Transformation
Save
Car Repair

5. Click **Logout** to logout and return to the view mode.

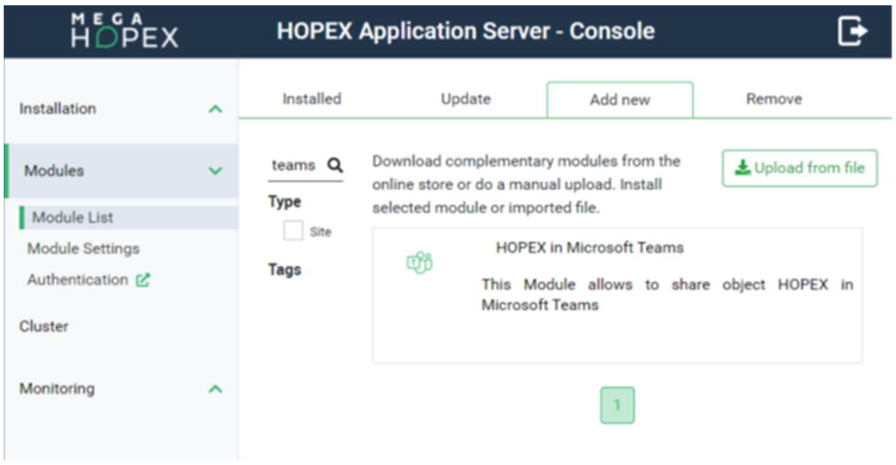
3. HOPEX Configuration

To be able to use HOPEX in Microsoft Teams you need to install the module and configure HOPEX as described below.

3.1. Installing Teams Module

To install Teams module:

1. Connect to **HOPEX Application Server – Console**.
2. Select **Modules** > **Module List**.
3. In **Add new** tab, search for “Teams”.

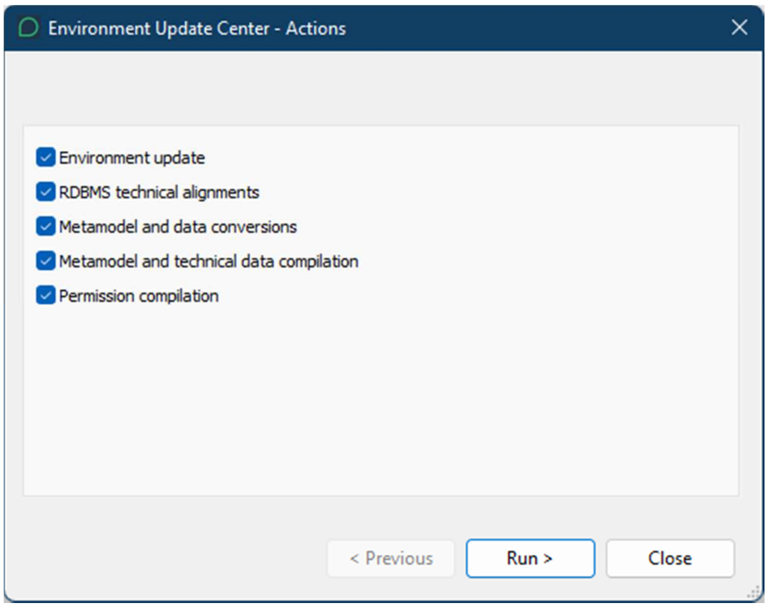


Download complementary modules from the online store or do a manual upload. Install selected module or imported file.

HOPEX in Microsoft Teams

This Module allows to share object HOPEX in Microsoft Teams

4. Install **Teams** Module.
5. In the **Administration** application (administration.exe), run the “Environment Automatic Update” (right-click the environment and select Environment automatic update).



- [✓] Environment update
- [✓] RDBMS technical alignments
- [✓] Metamodel and data conversions
- [✓] Metamodel and technical data compilation
- [✓] Permission compilation

3.2. Configuring Http Security

To configure http security:

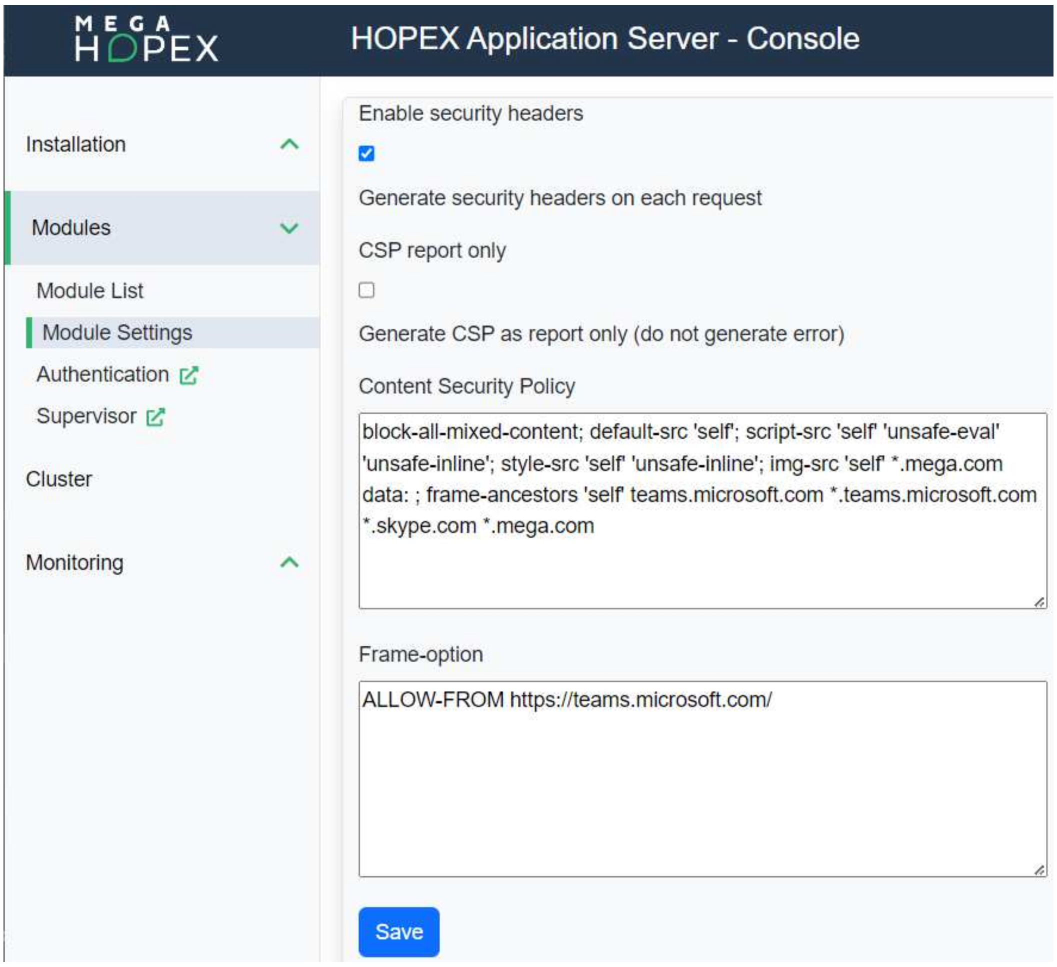
1. Connect to **HOPEX Application Server – Console**.
2. Select **Modules** > **Module Settings**.
3. Edit **Http Security Headers** and ensure **Enable Security Header** is selected.
4. In the **Content Security Policy** field, copy-paste the following content:

   block-all-mixed-content; default-src 'self'; script-src 'self' 'unsafe-eval' 'unsafe-inline'; style-src 'self' 'unsafe-inline'; img-src 'self' *.mega.com data: ; frame-ancestors 'self' teams.microsoft.com *.teams.microsoft.com *.skype.com *.mega.com

5. Set the **Frame-option** as displayed below:

   ALLOW-FROM https://teams.microsoft.com/

   Your settings should look like the following:



Enable security headers

☑

Generate security headers on each request

CSP report only

☐

Generate CSP as report only (do not generate error)

Content Security Policy

block-all-mixed-content; default-src 'self'; script-src 'self' 'unsafe-eval' 'unsafe-inline'; style-src 'self' 'unsafe-inline'; img-src 'self' *.mega.com data: ; frame-ancestors 'self' teams.microsoft.com *.teams.microsoft.com *.skype.com *.mega.com

Frame-option

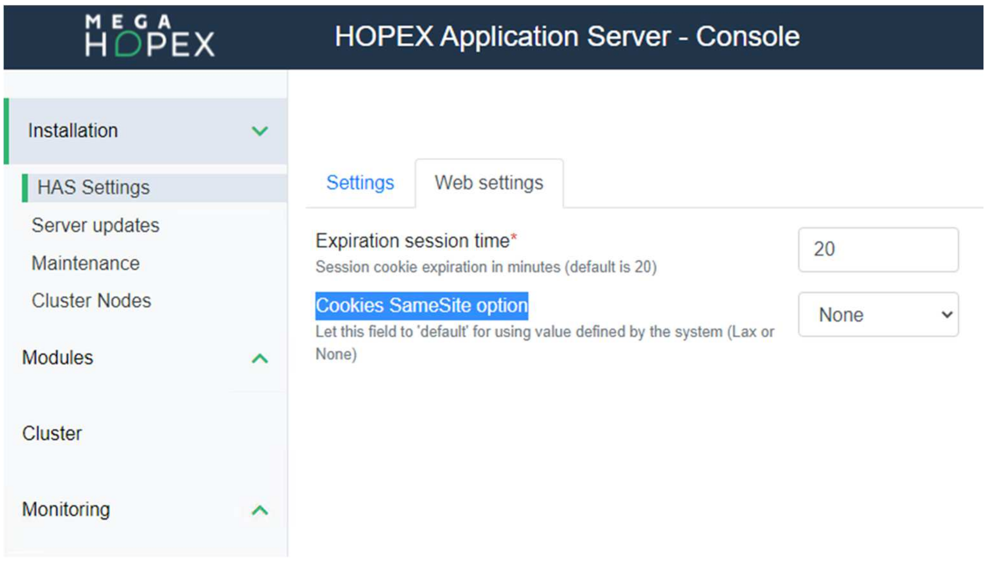
ALLOW-FROM https://teams.microsoft.com/

3.3. Setting SameSite cookie

You must set SameSite cookie attribute to none, secure (SameSite cookie attribute - Teams | Microsoft Docs).

To set SameSite cookie:

1. Connect to **HOPEX Application Server - Console**.
2. Select **Installation** > **HAS Settings**.
3. In the **Web settings** tab, set **Cookies SameSite option** to "None".



Expiration session time* 20
Session cookie expiration in minutes (default is 20)
Cookies SameSite option None
Let this field to 'default' for using value defined by the system (Lax or None)

3.4. Configuring the authentication

Create an API Key or, alternatively, a user Account for sharing diagrams in read only mode.

This will be used to open a session in read only mode on the repository that contains the diagrams to share.

3.4.1. User accounts

Create a user account and communicate its username and password to all of the users who need to share diagrams in Teams.

1. Connect to **HOPEX Application Server - Console**.
2. Select **Modules** > **Authentication**.
3. Select **User accounts** menu and click **Create**.
4. Set the **User Name** and **Password**.
5. Configure **Hopex session**:
   - Select **Open session**
   - Enter a valid **HOPEX login**
   - Select the **EnvironmentId**
   - Select the **Repository**

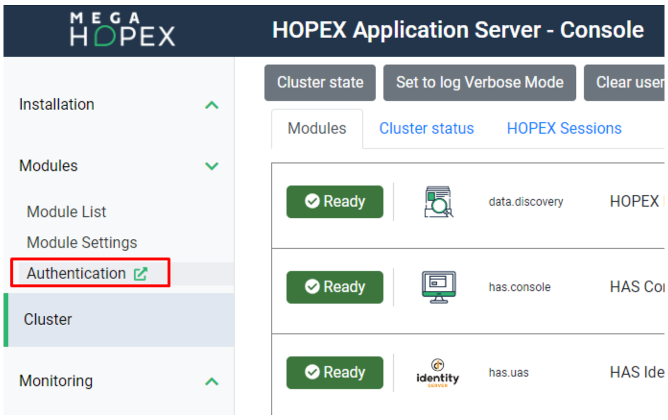
- Select the **Profile** with which you want to connect

6. Click **Submit**.

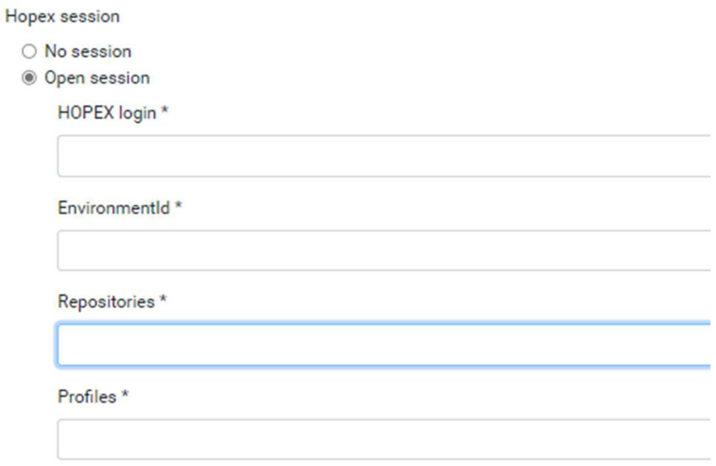
3.4.2. API Key

Create the API Key and communicate it to all of the users who need to share diagrams in Teams:

1. Connect to **HOPEX Application Server – Console**.
2. Select **Modules** > **Authentication**.

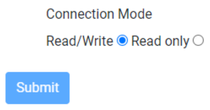


3. Select **API keys** menu and click **Create**.
   - Enter a key a **Name**.
4. Configure Hopex session:
   - Select **Open session**
   - Enter a valid **HOPEX login**
   - Select the **EnvironmentId**
   - Select the **Repository**
   - Select the **Profile** with which you want to connect



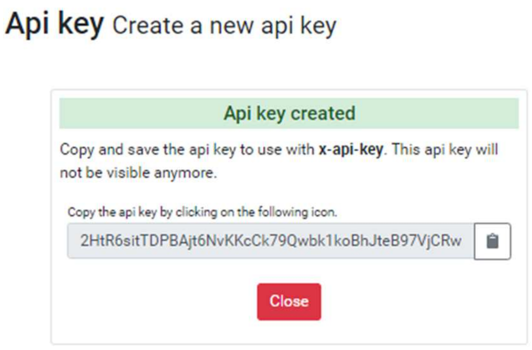
Hopex session
No session [✓ Open session]
HOPEX login *
EnvironmentId *
Repositories *
Profiles *

5. In **Connection Mode**, keep "Read/Write".
6. Click **Submit**.



Connection Mode
[✓ Read/Write] Read only

7. Copy the Key.



Api key Create a new api key

Api key created

Copy and save the api key to use with **x-api-key**. This api key will not be visible anymore.

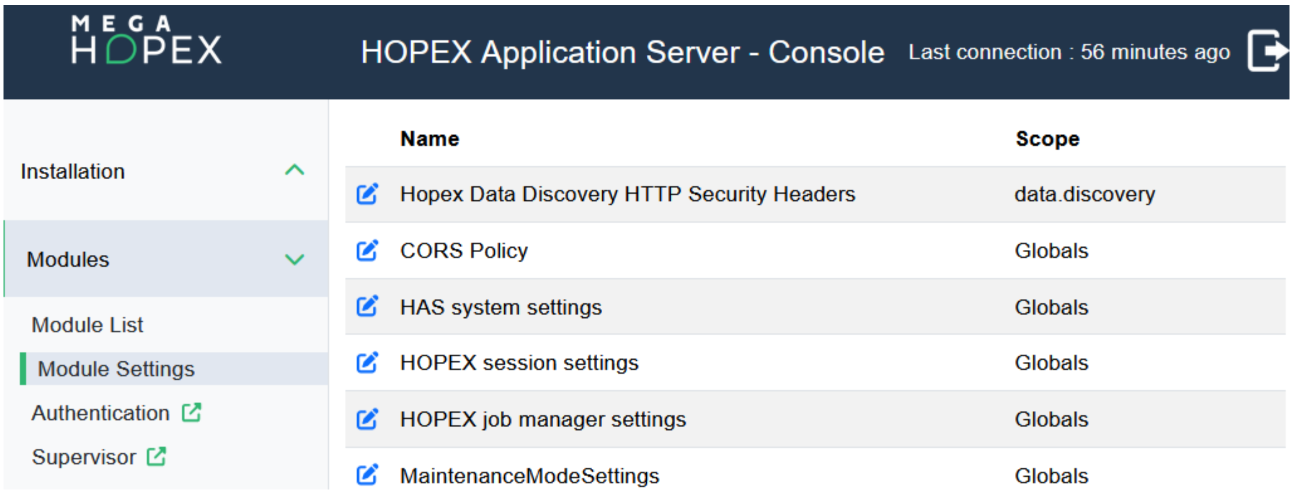
Copy the api key by clicking on the following icon.

2HtR6sitTDPBAjt6NvKKcCk79Qwbk1koBhJteB97VjCRw

3.4.3. Authorizing HOPEX connection from Teams

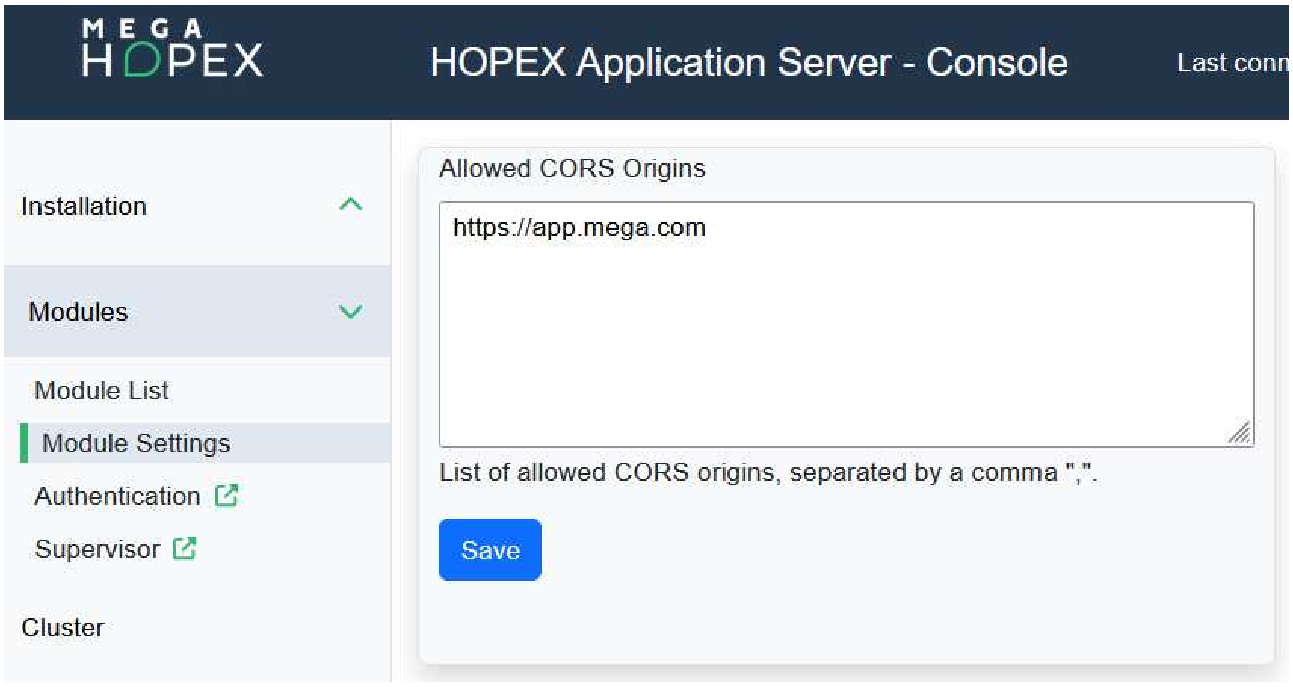
To allow to share HOPEX objects in Microsoft Teams:

1. Connect to **HOPEX Application Server – Console**.
2. Select **Modules** > **Modules Settings**.



Name	Scope
Hopex Data Discovery HTTP Security Headers	data.discovery
CORS Policy	Globals
HAS system settings	Globals
HOPEX session settings	Globals
HOPEX job manager settings	Globals
MaintenanceModeSettings	Globals

3. In **CORS Policy** row, click **Edit** .
4. In the **Allowed CORS origins** pane, enter: https://app.mega.com



Allowed CORS Origins

https://app.mega.com

List of allowed CORS origins, separated by a comma ",".

5. Click **Save**.

REST API and GraphQL

MEGA

1. REST API Documentation

HOPEX Platform comes with REST API that can be used in a wide range of use cases.

The documentation of the endpoints of the HOPEX REST API is available as catalog as well as classical textual documentation.

To access the documentation, click https://www.postman.com/mega-international

A Postman collection is published online on the MEGA profile for you to:

- get a description of what the API is providing and how it works
- get the list of all the available endpoints: GET / POST
- see examples of call request and result: status code, json response format...
- download ready to use in postman with parameters to adapt to your environment.

2. GraphQL endpoints

The REST API heavily leverages the GraphQL framework. The GraphQL framework enables to have only **one** endpoint for all REST API call.

To ease navigation in the HOPEX platform repository each Solution has its endpoint. The endpoint corresponds to the published schema:

- https://<<server url>>/HOPEXGraphQL/api/**ITPM**
- https://<<server url>>/HOPEXGraphQL/api/**BPA**
- https://<<server url>>/HOPEXGraphQL/api/**GDPR**
- https://<<server url>>/HOPEXGraphQL/api/**Audit**

Other endpoints are available for focused topics like:

- upload/download of documents
- download of diagrams

2.1. Synchronous versus Asynchronous

The endpoint can be called in synchronous or asynchronous way:

- https://<<server url>>/HOPEXGraphQL/api**/async/**ITPM
- https://<<server url>>/HOPEXGraphQL/api**/async/**BPA
- https://<<server url>>/HOPEXGraphQL/api**/async/**GDPR
- https://<<server url>>/HOPEXGraphQL/api**/async/**Audit

2.2. Version of the endpoint

The endpoint can also have version if the schema evolves overtime. In that case the particular version of the endpoint can be called by adding the version number in the URL:

- https://<<server url>>/HOPEXGraphQL/api**/v5/**ITPM
- https://<<server url>>/HOPEXGraphQL/api**/v5/**BPA
- https://<<server url>>/HOPEXGraphQL/api**/v5/**GDPR
- https://<<server url>>/HOPEXGraphQL/api**/v5/**Audit

3. GraphQL and Data Confidentiality (CRUD)

When making query or mutation to GraphQL REST API all access rights are checked based on the profile used.

The access rights are defined, in this order, at several level:

1. License level
2. Option level
3. Profile level
4. Workflow level
5. Data Reading or Writing access rules (graph or macro)

Each time you make a query or a mutation, HOPEX checks that you are allowed to perform this action:

- For a query: it checks the "Read" access rights (R).
- For a mutation: it checks the "Write" access rights (CRUD).
    - Create when creating an object
    - Update when trying to update an existing object
    - Delete when trying to delete an existing object

3.1. Query

In a query, if you are not allowed to view the requested information you will get:

- a null value for a field (MetaAttribute)
- an empty array for a relationship (MetaAssociation)

3.2. Mutation

In a mutation, if you are not allowed to create/update/delete the requested object or its fields you will get:

- an error on each field you are not allowed to edit, with a message:
  "You are not allowed to perform this action…"

3.3. Managing permission

You should ensure that the profile you use when querying the application is properly configured with the CRUD.

4. Selecting the data language with the REST API

In HOPEX the data can be entered in different languages (English, French, Spanish, Italian, German...). When connected by the GraphQL REST API it is possible to select the data language.

4.1. Querying data in the current data language

By default all the queries are performed in the current data language.

- The following query returns the name of the application in English:

Query	Result

query app {
  application {
  name
  }
}
	{
	  "data": {
	    "application": [
	      { "name": "AA" },
	      { "name": "Account Management" }
	    ]
	  }
	}

- The following query returns the current language:

Query	Result

query currentLanguage {
  _currentContext {
    language
    languageId
    languageName
  }
}
	{
	  "data": {
	    "_currentContext": {
	      "language": "EN",
	      "languageId": "00(6wlHmk400",
	      "languageName": "English"
	    }
	  }
	}

4.2. Querying data in a selected data language

When requesting the data you can force the language of the return data for all translatable fields (e.g.: name, comment). The selection of the language is done by its code (e.g.:EN, DE, JA, FR, ES, NL, IT).

In a query you can get the data in one or multiple languages.

- The following query returns the name of the application in English, French, and Italian. As there can be only one field with name, alias of the name are created. As seen in the example the translation may not be available and an empty string is returned.

Query	Result

query app {
  application {
  name(language:EN)
  nameFR:name(language:FR)
  nameIT:name(language:IT)
  }
}
	{
	  "data": {
	    "application": [
	      { "name": "Management", "nameFR": "Management", "nameIT": "" },
	      { "name": "Account Payable", "nameFR": "", "nameIT": "" },
	      { "name": "Account", "nameFR": "Comptabilité", "nameIT": "Conto" }
	    ]
	  }
	}

These requests do not change the default language of the user. Any next query performed without defining the language is returned in the default language.

4.3. Changing the current data language

To change the current data language you need to update the user context: execute a graphQL mutation on the context.

- In the following example the context will be changed to French.

Query	Result

mutation updateLanguage {
  _updateCurrentContext(
    currentContext:{
          language:FR
                }
)
  { language }
}
	{
	  "data": {
	    "_updateCurrentContext": {
	      "language": "FR"
	    }
	  }
	}

4.4. Changing the data language for a given user

Aside from the current user it is also possible to update the data language of a group of users.

4.4.1. Query to know the language of a given user

You can query the user to know his/her default language. If the returned value is blank the default language is the same as the installation one.

Query	Result

query user {
  personSystem(filter:{name:"Thomas"}) {
    id
    name
    dataLanguage {
      ...on Language {
        language:languageCode
      }
      languageId:id
      languageName:name
    }
  }
}
	{
	  "data": {
	    "personSystem": [
	      {
	        "id": "qqcxS(UhHzHC",
	        "name": "Thomas",
	        "dataLanguage": {
	          "language": "EN",
	          "languageId": "00(6wlHmk400",
	          "languageName": "English"
	        }
	      }
	    ]
	  }
	}

4.4.2. Mutation to update the language of a given user

You can set the language of a given user by its ID.

Query	Result

mutation updateUser {
updatePersonSystem(
  id:"qqcxS(UhHzHC"
  idType:INTERNAL
  personSystem:{
    dataLanguageCode:FR
  }
 	
) {	

dataLanguage {
  ...on Language {
   language:languageCode
  }
  languageId:id
  languageName:name
 } } }	

4.5. Getting the list of available data languages

To know the possible value of the language code available you can query graphQL.

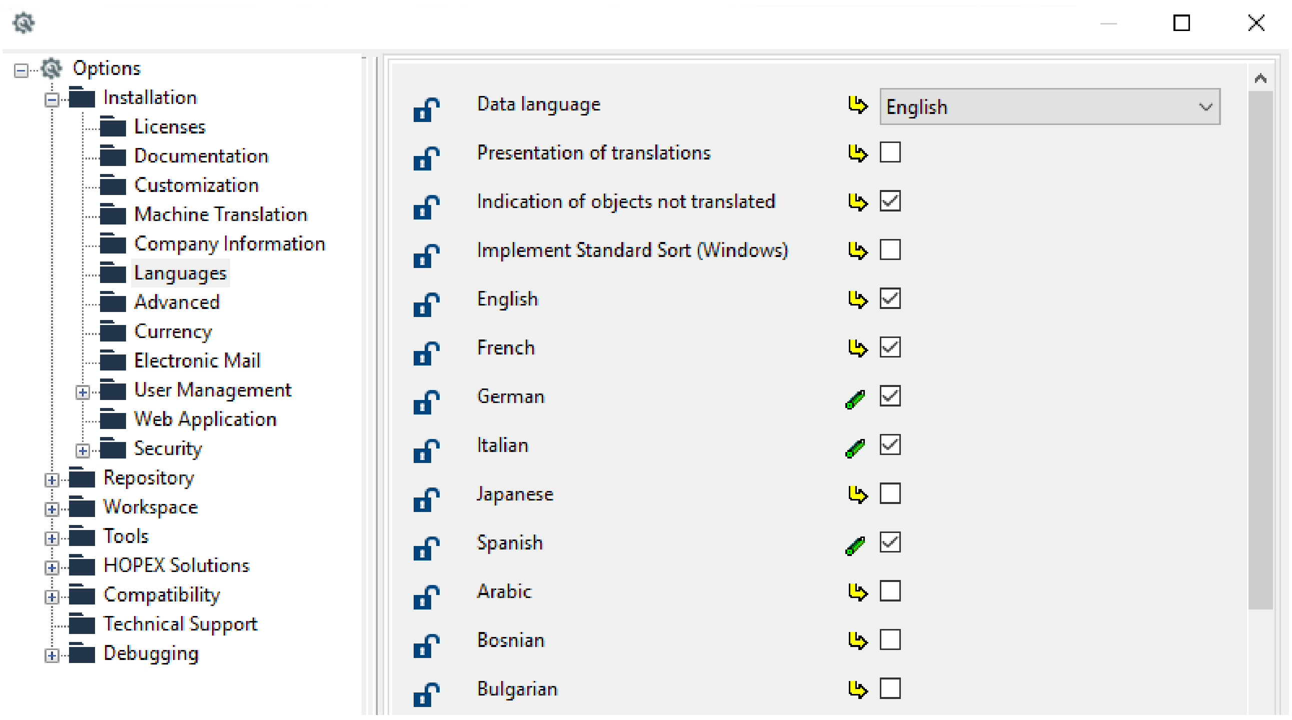
```
query availableLanguage {
  language(filter:{id:"I9o3by0knG00"}) {
    language_SpecializedLanguage {
      language:languageCode
      languageId:id
      languageName:name
    }
  }
}
```

You may have to do it on the MetaModel schema so that the query can work.

4.6. Adding data languages and seeing them in the API

The list of available data languages depends on the options defined in HOPEX.

In the Environment options, check the available languages in the **Installation** > **languages** folder:



Options
- Installation
  - Licenses
  - Documentation
  - Customization
  - Machine Translation
  - Company Information
  - Languages
  - Advanced
  - Currency
  - Electronic Mail
  - User Management
  - Web Application
  - Security
- Repository
- Workspace
- Tools
- HOPEX Solutions
- Compatibility
- Technical Support
- Debugging

Data language	English
Presentation of translations	☐
Indication of objects not translated	☑
Implement Standard Sort (Windows)	☐
English	☑
French	☑
German	☑
Italian	☑
Japanese	☐
Spanish	☑
Arabic	☐
Bosnian	☐
Bulgarian	☐

5. Basic Auth vs API Key

From HOPEX V5 and onward, the method for authentication for API has evolved.

1. With a Basic Auth.
2. With an API Key (preferred choice)

Former Bearer Token is not available in V5. Oauth2 Authentication is not supported for the moment for API calls.

Depending on the use case you want to use the API you may use one or the other authentication method. Regardless of the chosen authentication methods the other headers and body information remain the same.

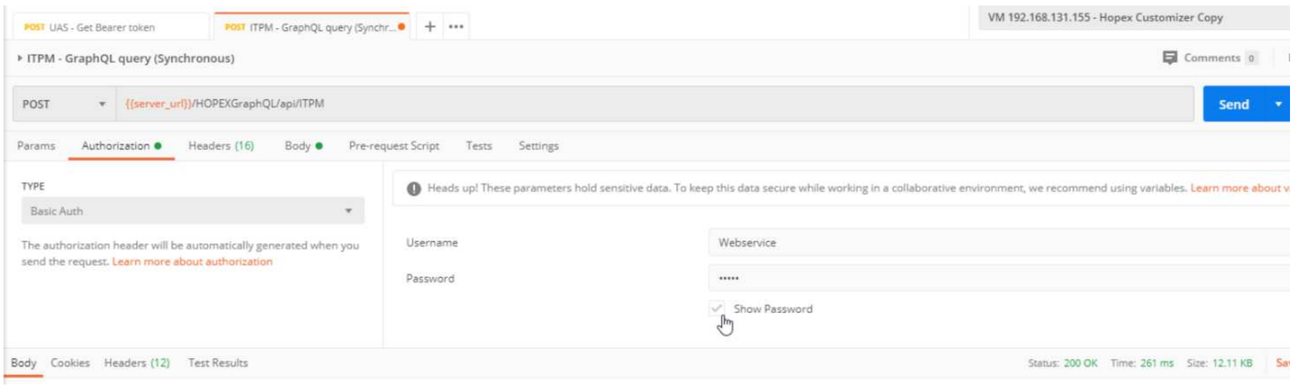
5.1. Basic Auth

The basic Auth allows you to access the API directly with credentials: login/password.

5.1.1. How to use it?

For example:

- In Postman when calling the API choose "Basic Auth" and fill-in the user password. The information will be encoded with Base64 to avoid to be readable when sent.


POST UAS - Get Bearer token
POST ITPM - GraphQL query (Synchr...
VM 192.168.131.155 - Hopex Customizer Copy
ITPM - GraphQL query (Synchronous)
Comments 0
Send
Params
Authorization
Headers (16)
Body
Pre-request Script
Tests
Settings
Body
Cookies
Headers (12)
Test Results

POST {{server_url}}/HOPEXGraphQL/api/ITPM

TYPE: Basic Auth

The authorization header will be automatically generated when you send the request. Learn more about authorization

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about v

Username: Webservice

Password: •••••

Show Password

Status: 200 OK Time: 261 ms Size: 12.11 KB

- In a script in curl add the header Authorization: Basic and pass the encoded value of the login and password.

```
curl --location --request POST 'httpx://www.myserver.com/HOPEXGraphQL/api/ITPM' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic V2Vic2VydmljZTpIb3BleA==' \
--data-raw '{"query":"query {\n    application {\n        id\n        name\n        cloudComputing\n
}\n}","variables":{}}'
```

This authentication method is useful when you need to check identity and get the data with a login/password logic. It is nonetheless less secure than an API Key.

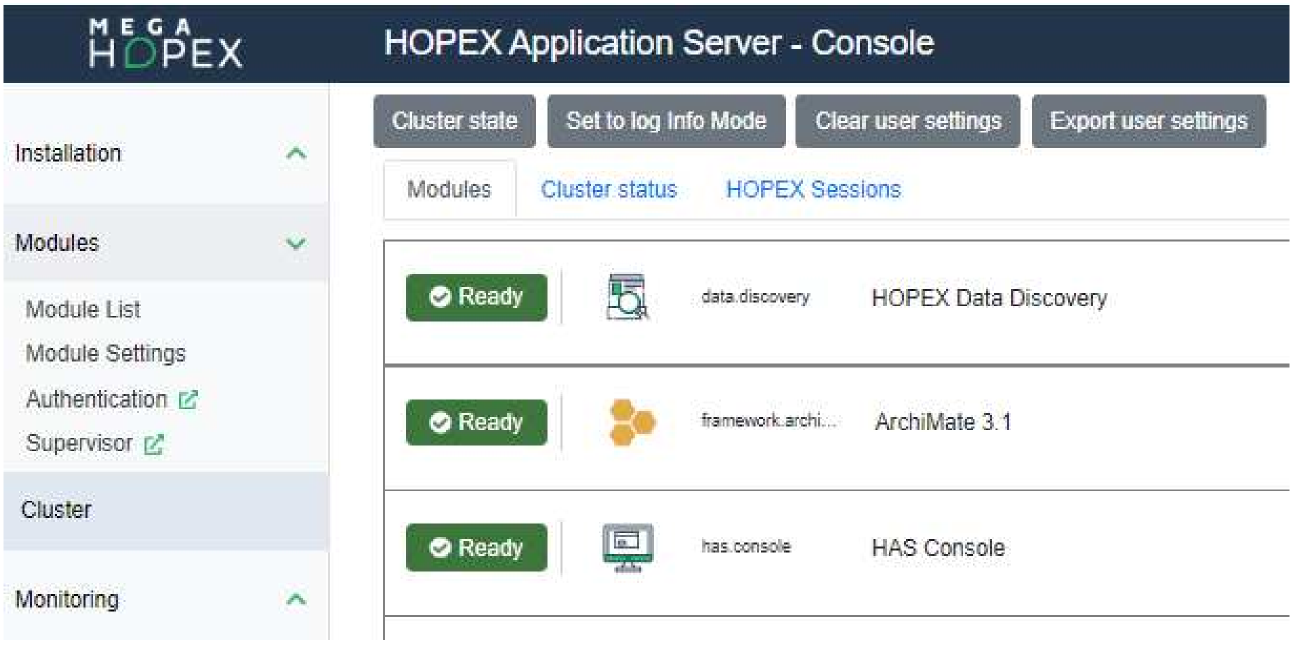
5.1.2. How to enable it?

You need to create a dedicated User/Password within the HAS console to be able to use it in API Call. This user can be:

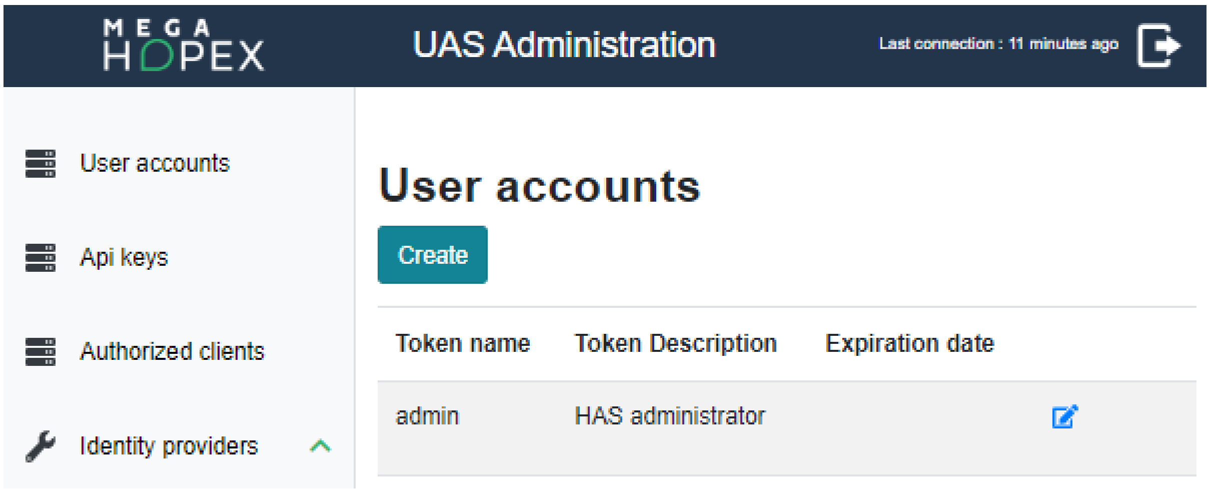
- Admin user.
- HOPEX user that connects with a profile.

Process step:

1. Connect to **HAS Console**.



2. Select **Modules > Authentication**.
3. Select **User accounts**.
4. Click **Create**.

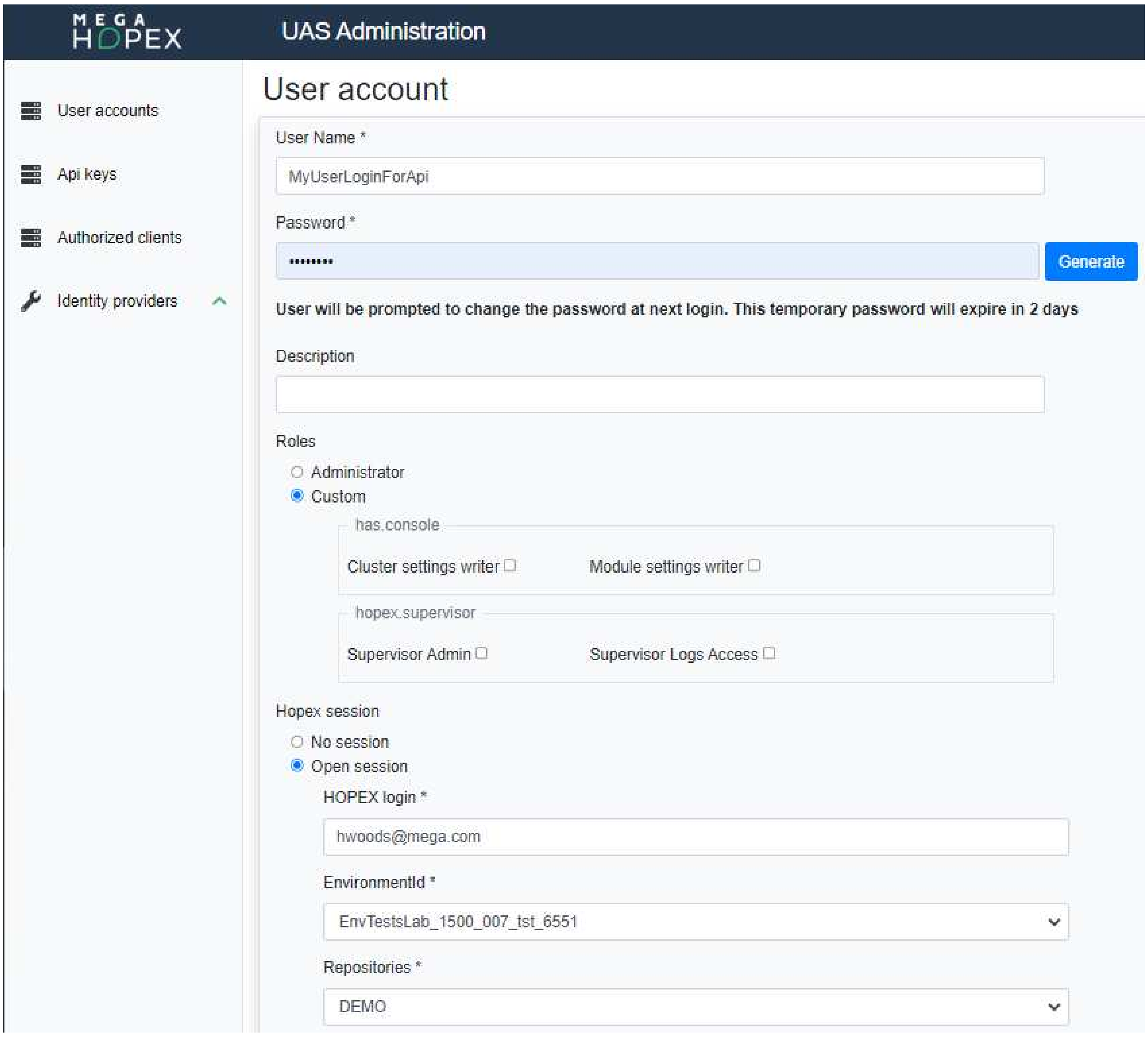


Token name	Token Description	Expiration date
admin	HAS administrator	

5. Fill in the form:
   a) Enter a login (**User Name**) to your user.
   b) Enter a **Password** or generate one.
   c) Select the Role: **Administrator** or **Custom.**
   d) Select if you allow to open a session on a specific **Repository** and **Profile.**
   e) Enter the login of the HOPEX user.

f) Select the Environment (there should be only one).
g) Select the **Repository** (if more than one).
h) Select the **Profile** (if more than one)
i) Selection the **Session mode**: multi or single (see below for more details on what to chose).
j) Select the **Connection mode**: read/write or read only.
k) Click Submit.

You can now use this login/password for API call.



User account

User Name *
MyUserLoginForApi

Password *
••••••••

User will be prompted to change the password at next login. This temporary password will expire in 2 days

Description

Roles
○ Administrator
● Custom

has.console
Cluster settings writer ☐ Module settings writer ☐

hopex.supervisor
Supervisor Admin ☐ Supervisor Logs Access ☐

Hopex session
○ No session
● Open session

HOPEX login *
hwoods@mega.com

EnvironmentId *
EnvTestsLab_1500_007_tst_6551

Repositories *
DEMO

5.2. API Key

To access the API with an API Key you need to create it and define all the technical information:

- admin or user api key
- repository and profile to connect to.

Once done, the system gives you the API key. This API Key can be valid for all time or have a validity period.

5.2.1. Security

The API Key generated does not contain any information that can be decrypted or decoded.

5.2.2. Use case

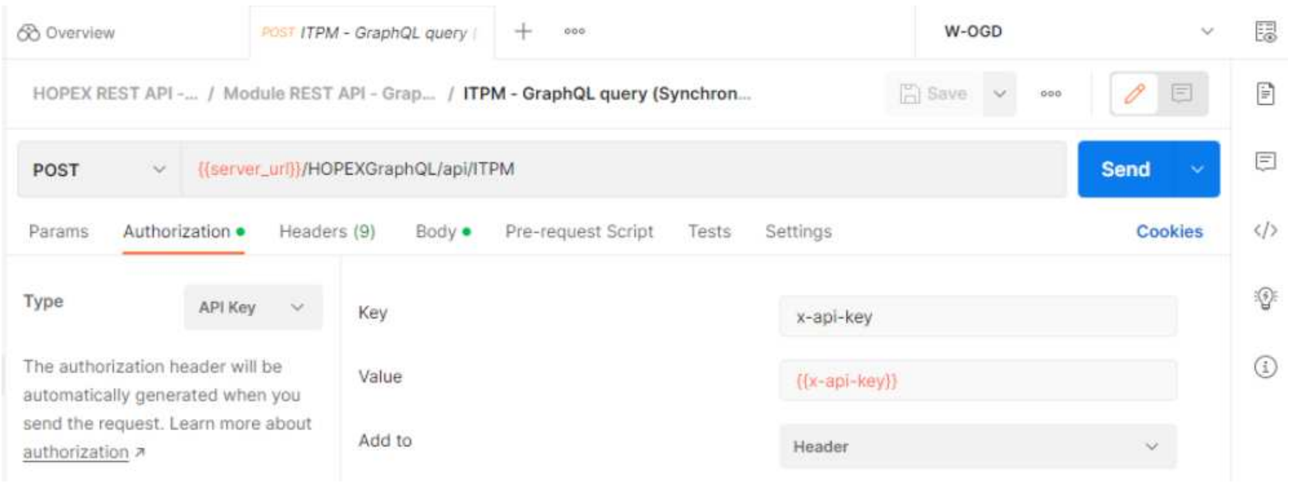
It is recommended Authentication methods whenever possible. It is ideal when scripting, when developing external app, or when doing integration with external tools.

5.2.3. How to use it?

For example:

- In Postman when calling the API choose "API Key" and fill-in the API Key value.
  - Key: x-api-key
  - Value: xxxxxxxxx

  Now you can make call to any endpoint.


Overview
POST ITPM - GraphQL query
W-OGD
HOPEX REST API -... / Module REST API - Grap... / ITPM - GraphQL query (Synchron...
Save
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies

POST {{server_url}}/HOPEXGraphQL/api/ITPM

Type	API Key
Key	x-api-key
Value	{{x-api-key}}
Add to	Header

The authorization header will be automatically generated when you send the request. Learn more about authorization ↗

- In a script in curl add the header x-api-key and pass the value of the API Key.

```
curl --location 'https://w-ogd/HOPEXGraphQL/api/ITPM' \
--header 'x-api-key:
5snybEHxGR8uTRAks2ySEgYs8t82rQ6KqkrcEsp9srw737WmPZcJvpk1gNctBCjVQZvBwrryaFzJkHk61
Q1eFJex' \
--header 'Content-Type: application/json' \
        --data '{"query":"query\n{\n  application\n  {\n    id\n    name\n  }\n}","variables":{}}'
```

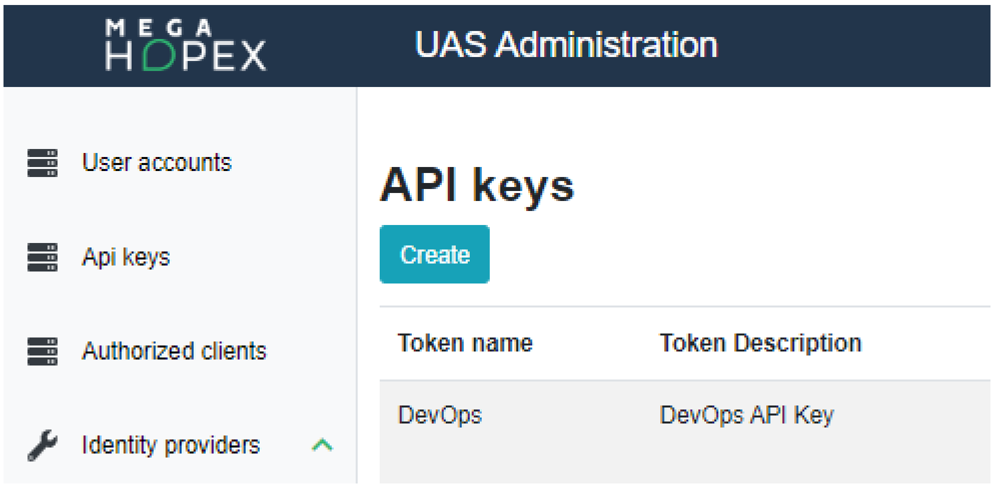
5.2.4. How to enable it?

You need to create a dedicated API Key within the HAS console to be able to use it in API Call. This API Key can be:

- Admin API Key.
- HOPEX user that connects with a profile.

Process step:

1. Connect to the **HAS Console**.
2. Select **Modules > Authentication**.
3. Click API Keys.



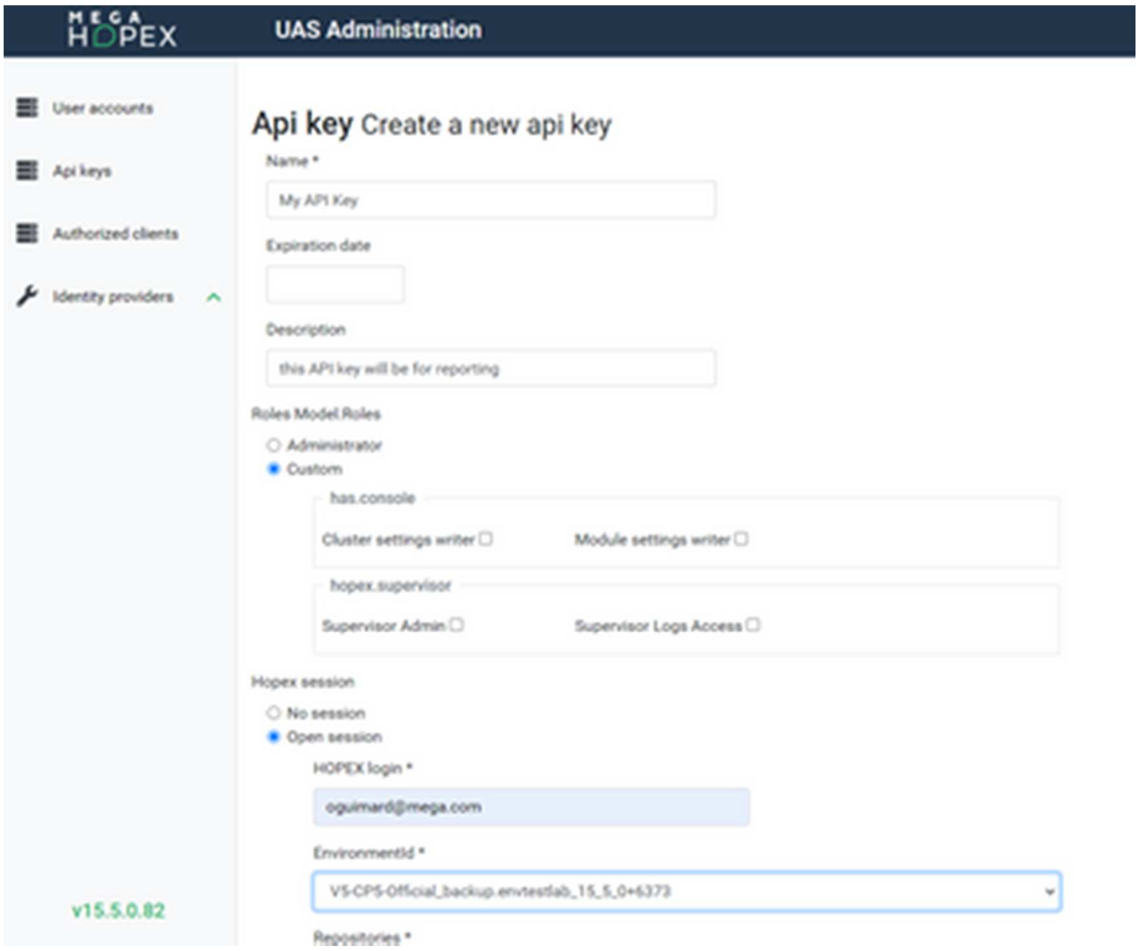
Token name	Token Description
DevOps	DevOps API Key

4. Click **Create**.
5. Fill in the information:
   a. Name: for you to remember what this API key will be used for
   b. Expiration date: if you want to limit the validity period of this API Key
   c. Description: for you to remember what this API key will be used for
   d. Select the Role: Administrator or Custom
   e. Select if you allow to open a session on a specific repository and profile
   f. Give the login of the HOPEX user
   g. Select the environment (there should be only one)
   h. Select the repository (if more than one)
   i. Select the profile (if more than one)
   j. Select the session mode: multi or single
   k. Select the connection mode: read/write or read only.
   l. Click **Submit**
   m. Copy/paste the API Key and save it for later use

You can now use this login/password for API call.

Caution:

- the API key is displayed **only once**, so keep it!
- there is no mean to regenerate the API key. You will have to reset all the parameters to create a new one.
- the API cannot be displayed later to get it again.



Api key Create a new api key

Name *
My API Key

Expiration date

Description
this API key will be for reporting

Roles Model Roles
○ Administrator
● Custom

has.console
Cluster settings writer ☐ Module settings writer ☐

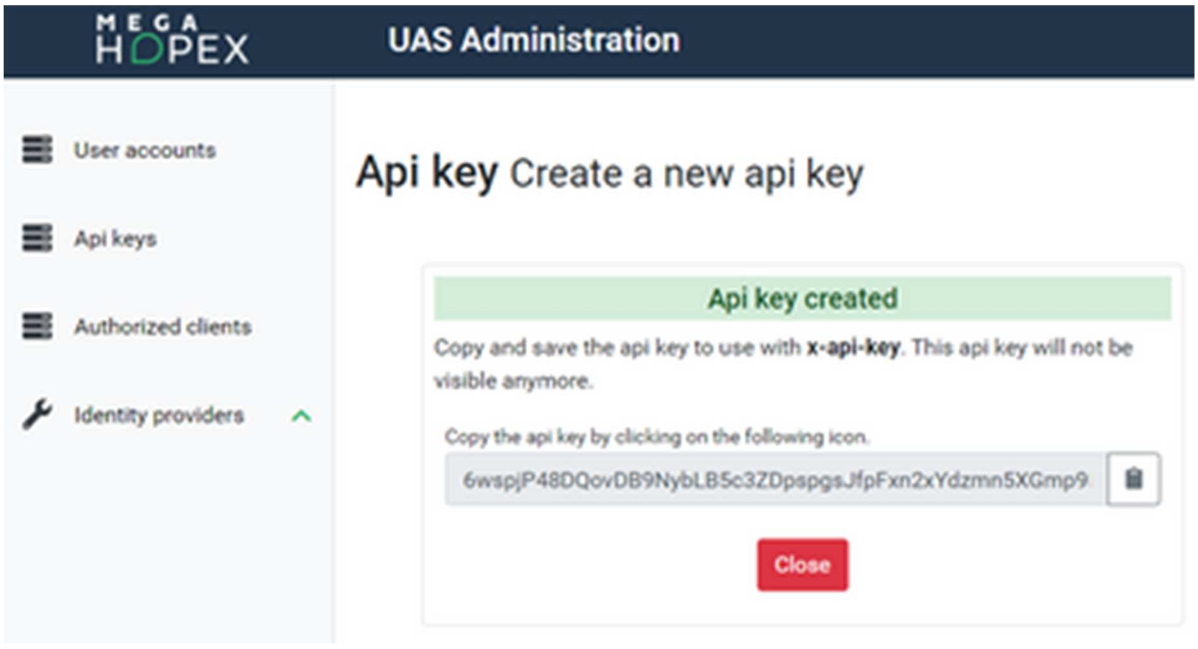
hopex.supervisor
Supervisor Admin ☐ Supervisor Logs Access ☐

Hopex session
○ No session
● Open session

HOPEX login *
oguimard@mega.com

EnvironmentId *
V5-CPS-Official_backup.envtestlab_15_5_0+6373

Repositories *



Api key Create a new api key

Api key created

Copy and save the api key to use with **x-api-key**. This api key will not be visible anymore.

Copy the api key by clicking on the following icon.

6wspjP48DQovDB9NybLB5c3ZDpspgsJfpFxn2xYdzmn5XGmp9

5.3. Multi or Single Mode

The mode choice changes the behavior in the back-end to process the request.

Choose:

- **Multi** for all purposes where you need responsiveness in the API calls.
  - Advantage: you benefit from cache, ready to use process to respond to your query.
  - Drawback: not adapted to static website generation
- Single for heavy computing treatment. Ideal for heavy batch or static website generation.

6. Querying/Creating/Updating/Deleting with GraphQL

The REST API based on GraphQL allows to perform all actions of the CRUD:

- Create
- Read
- Update
- Delete

This mechanism allows to perform a various kind of action into the repository. Read actions are performed with the keyword "**query**" in GraphQL whereas Read/Update/Delete actions are performed with the "**mutation**" keyword.

All the examples below are performed with the ITPM schema.

6.1. Basic queries

6.1.1. Getting an object with its attributes

The following query returns the name of the applications:

Query	Result

query app {
  application {
  name
  }
}
	{
	  "data": {
	    "application": [
	      { "name": "AA" },
	      { "name": "Account Management" }
	    ]
	  }
	}

6.1.2. Getting an object with its relations

The following query returns the applications and their related business process:

Query	Result

query {
  application {
    id
    name
    businessProcess {
      id
	{
	  "data": {
	    "application": [
	      {
	        "id": "snf5hRn0U91K",
	        "name": "AA",

```
    name
  }
 }
}
```

```
    "businessProcess": []
   },
   {
    "id": "IubjeRlyFfT1",
    "name": "Account Management",
    "businessProcess": [
     {
      "id": "12qog2pV99fG",
      "name": "Financial Reporting"
     }
    ]
   },
```

6.2. Basic mutations

6.2.1. Creating an object

The following mutation creates an application. If creation is successful the result returnes the created object.

Query

```
mutation {
 createApplication(application:{
  name:"new application"
 }) {
  id
  name
 }
}
```

Result

```
{
 "data": {
  "createApplication": {
   "id": "x2wrAYDkWP8H",
   "name": "new application"
  }
 }
}
```

6.2.2. Creating an object with a relationship

The following mutation creates an application and links it to an existing business process. If creation is successful the result returnes the created object.

Query

```
mutation {
 createApplication(application : {
  name: "new app 3"
```

Result

```
{
 "data": {
  "createApplication": {
```

businessProcess: {	"id": "Q2wradDkWbGH",
action:ADD	"name": "new app 3",
list:[{id:"39cXIxu2HHrI"}]	"businessProcess": [
}	{
}) {	"id": "39cXIxu2HHrI",
id	"name": "Accounting"
name	}
businessProcess {	]
id	}
name	}
}	}
}	
}	

6.2.3. Updating an object

The following mutation updates an existing application. If update is successful the result returnes the updated object.

Query	Result
mutation {	{
updateApplication(id:"IubjeRlyFfT1" application:{	"data": {
cloudComputing:Cloud_IaaS	"updateApplication": {
}) {	"id": "IubjeRlyFfT1",
id	"cloudComputing": "Cloud_IaaS"
cloudComputing	}
}	}
}	}

6.2.4. Deleting an object

The following mutation deletes an existing application. If delete is successful the result returnes the number of deleted objects.

Query	Result
mutation {	{
deleteApplication (id:"snf5hRn0U91K" cascade:false) {	"data": {
	"deleteApplication": {

```
    deletedCount
  }
}
```
	```
	      "deletedCount": 1
	    }
	  }
	}
	```

7. Creating custom schema (SDL/JSON) / custom endpoint

The GraphQL REST API exposes in standard a subset of the full HOPEX Metamodel. This subset is organized into several endpoint represented by a schema (SDL). The split is mainly done by solution scope or technical grouping (e.g.: ITPM, BPA, Audit)

It is possible to extend the delivered schema or to create new schema. The result of this customization will end up with a new endpoint to call by HTTP request.

7.1. How it works ?

7.1.1. General principle

To create a schema you need to convert the HOPEX Metamodel into a GraphQL syntax compatible. This mapping is done through a JSON file that maps the HOPEX ID with a naming convention compatible with GraphQL. From this file the system generates a GraphQL syntax. The default JSON mapping schema are installed in the **CONFIG** folder of the HOPEX installation:

C:\ProgramData\MEGA\Hopex Application Server\<Has instance name>\Modules\hopex.graphql\<GraphQL module version>\CONFIG\V3\Standard

Example:

C:\ProgramData\MEGA\Hopex Application Server\5000\Modules\hopex.graphql\HOPEX GraphQL-7.87.507+6623.zip\CONFIG\V3\Standard

Moreover, in the web part a key in the web.config exposes the list of available schema.

```
<add key="GraphQLSchemas" value="ITPM, Assessment, Audit, BPA, Data, DataPrivacy, MetaModel, Reporting, Risk, Workflow"/>
<!--      GraphQLSchema-->
```

In this example there are several schema about: ITPM, Assessment, Audit…

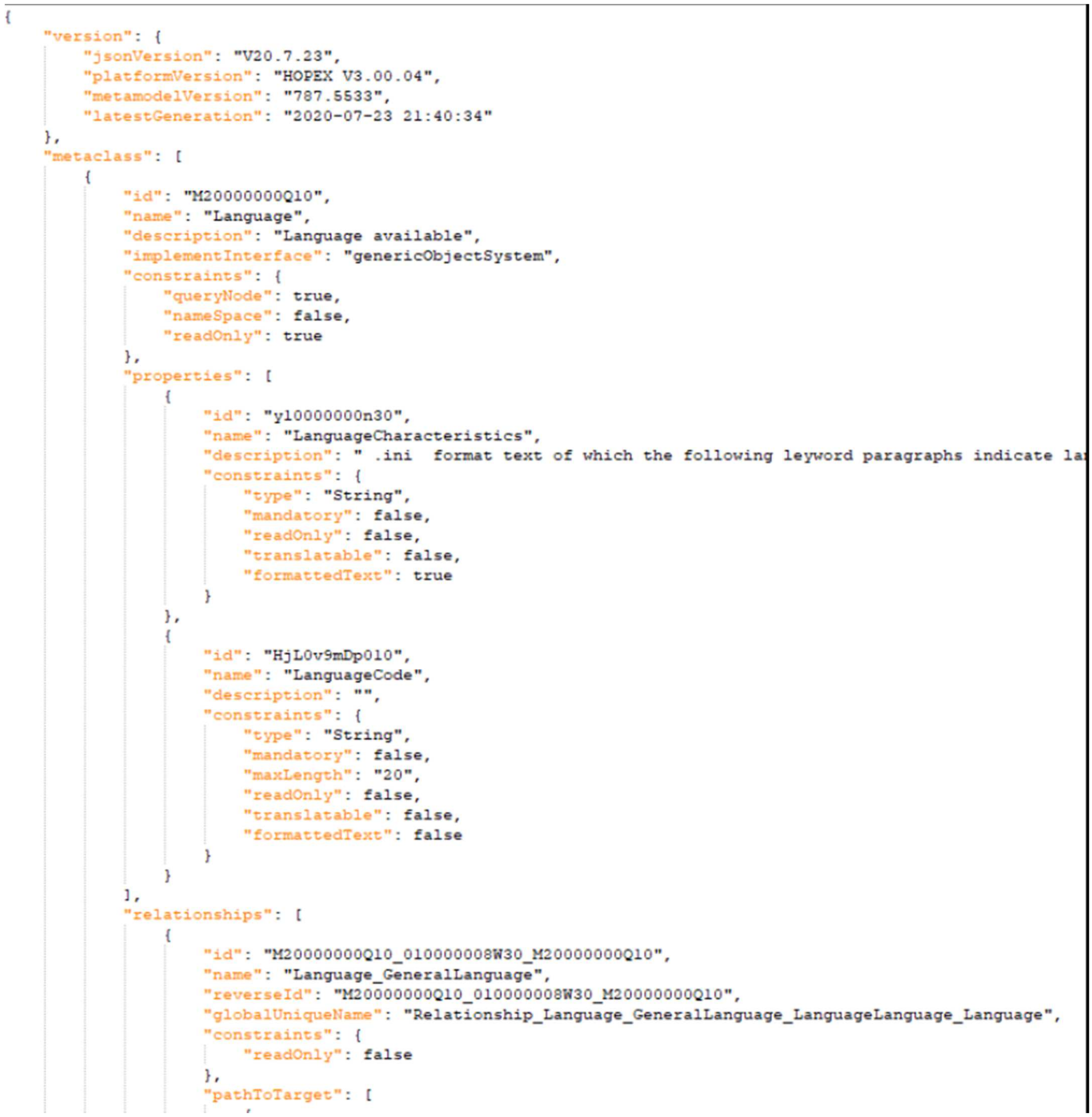
Within this JSON file the metamodel in terms of MetaClass, MetaAttribute, MetaAttributeValue, MetaAssociation.

Here is an extract of a JSON mapping schema for ITPM:

```
...
"metaclass": [{
"id": "MrUiM9B5iyM0",
"name": "Application",
"description": "An application is a software component that can be deployed and provides users with a set of functionalities.",
...
"properties": [{
"id": "Z20000000D60",
"name": "Name",
"description": "Short Name of the object ...",
"constraints": {
```

```
"type": "String",
"mandatory": true,
"maxLength": "1024",
"readOnly": false,
"translatable": true,
"formattedText": false
}}],
"relationships": [
{
"id": "7ChrtiDo4vb0_YChrYpDo4ri0_MrUiM9B5iyM0",
"name": "TimePeriod",
...
"pathToTarget": [
{
          "id": "VChrYpDo4fi0",
          "name": "TimeDependentObjectTimePeriod",
          "maeName": "PeriodOfValidity",
          "maeID": "YChrYpDo4ri0",
          "metaClassName": "TimePeriod",
          "metaClassID": "7ChrtiDo4vb0",
          "multiplicity": "*"
}]}]
...
```

See ITPM file for full JSON:



```
{
    "version": {
        "jsonVersion": "V20.7.23",
        "platformVersion": "HOPEX V3.00.04",
        "metamodelVersion": "787.5533",
        "latestGeneration": "2020-07-23 21:40:34"
    },
    "metaclass": [
        {
            "id": "M20000000Q10",
            "name": "Language",
            "description": "Language available",
            "implementInterface": "genericObjectSystem",
            "constraints": {
                "queryNode": true,
                "nameSpace": false,
                "readOnly": true
            },
            "properties": [
                {
                    "id": "y10000000n30",
                    "name": "LanguageCharacteristics",
                    "description": " .ini  format text of which the following leyword paragraphs indicate la
                    "constraints": {
                        "type": "String",
                        "mandatory": false,
                        "readOnly": false,
                        "translatable": false,
                        "formattedText": true
                    }
                },
                {
                    "id": "HjL0v9mDp010",
                    "name": "LanguageCode",
                    "description": "",
                    "constraints": {
                        "type": "String",
                        "mandatory": false,
                        "maxLength": "20",
                        "readOnly": false,
                        "translatable": false,
                        "formattedText": false
                    }
                }
            ],
            "relationships": [
                {
                    "id": "M20000000Q10_010000008W30_M20000000Q10",
                    "name": "Language_GeneralLanguage",
                    "reverseId": "M20000000Q10_010000008W30_M20000000Q10",
                    "globalUniqueName": "Relationship_Language_GeneralLanguage_LanguageLanguage_Language",
                    "constraints": {
                        "readOnly": false
                    },
                    "pathToTarget": [
```

7.1.2. Creating/Updating a schema

To create your own schema you can:

- Create a **totally new** schema

  In this case you are free to define what is exposed and you will not be impacted by future updates of the API.

- **Extend** an **existing** schema.

  This limits the customization to be performed but you will benefit from future updates of the default delivered schema..

In both cases the principle is the same. You need to create a custom JSON file that you will put in the custom folder:

```
C:\...\HOPEX Application Server\<HAS instance
name>\.shadowFiles\has.custom\<Module version>\hopex.graphql\<GraphQL module
version>\CONFIG\V3\Standard
```

```
e.g.: C:\ProgramData\MEGA\HOPEX Application
Server\5000\.shadowFiles\has.custom\<Module version>\hopex.graphql\HOPEX
GraphQL-7.87.507+6623\CONFIG\V3\Standard
```

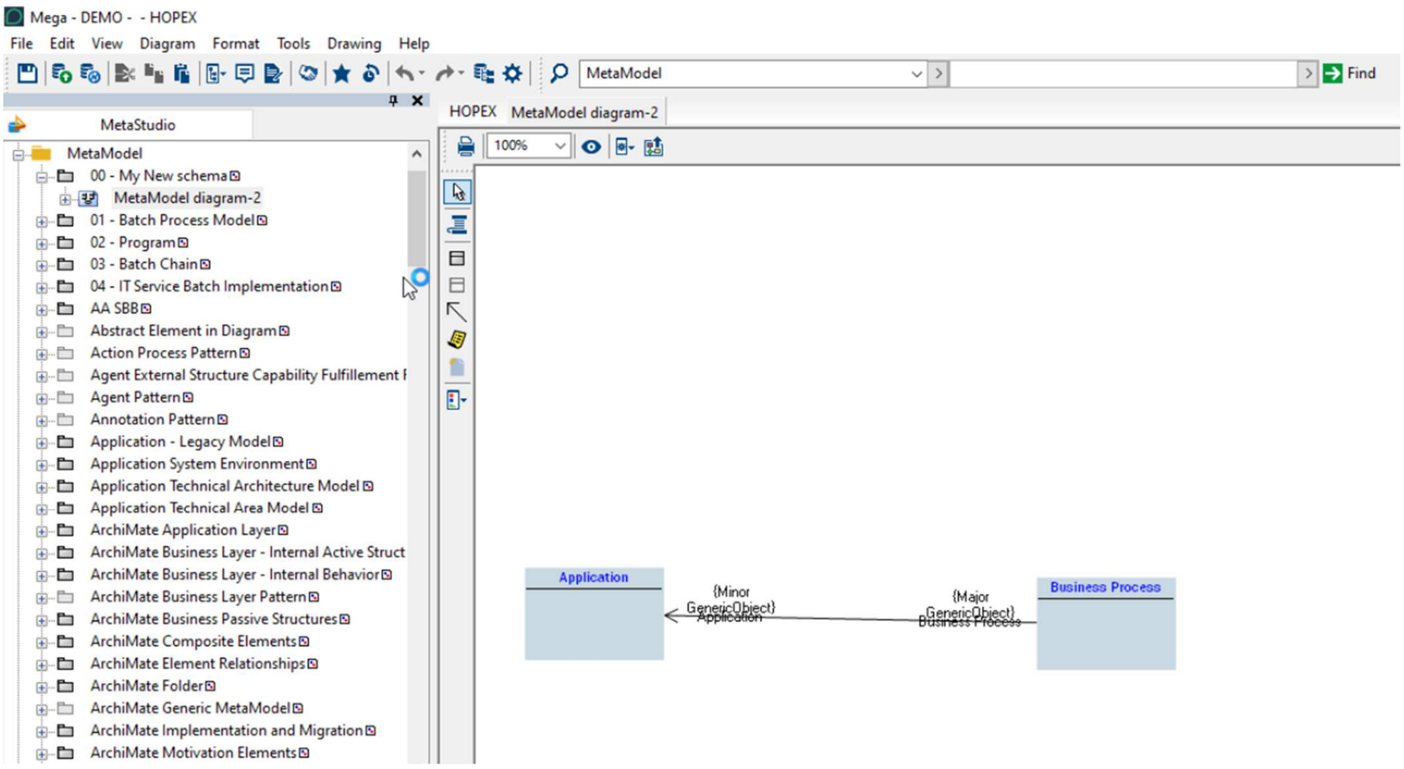
In case of a new JSON the name of the schema must be added in the web.config.

7.2. Step 1: Create your metamodel to expose in the REST API

The JSON delivered out of the box have been generated by a java program that reads a diagram of metamodel. To generate a schema JSON you can use the same tool on existing metamodel diagram or with your new metamodel diagram.

7.2.1. Creating Custom Metamodel

1. Connect to HOPEX (Windows Front-End).
2. In the **MetaStudio** tab create a new Metamodel.
3. Add a new diagram to this metamodel.
4. Add the elements of metamodel you want.
5. Copy and keep for later the absolute identifier of the metamodel object.


Mega - DEMO - - HOPEX
File Edit View Diagram Format Tools Drawing Help
MetaModel
Find
MetaStudio
HOPEX MetaModel diagram-2
100%
Application
{Minor GenericObject} Application
{Major GenericObject} Business Process
Business Process

- MetaModel
  - 00 - My New schema
    - MetaModel diagram-2
  - 01 - Batch Process Model
  - 02 - Program
  - 03 - Batch Chain
  - 04 - IT Service Batch Implementation
  - AA SBB
  - Abstract Element in Diagram
  - Action Process Pattern
  - Agent External Structure Capability Fulfillement
  - Agent Pattern
  - Annotation Pattern
  - Application - Legacy Model
  - Application System Environment
  - Application Technical Architecture Model
  - Application Technical Area Model
  - ArchiMate Application Layer
  - ArchiMate Business Layer - Internal Active Struct
  - ArchiMate Business Layer - Internal Behavior
  - ArchiMate Business Layer Pattern
  - ArchiMate Business Passive Structures
  - ArchiMate Composite Elements
  - ArchiMate Element Relationships
  - ArchiMate Folder
  - ArchiMate Generic MetaModel
  - ArchiMate Implementation and Migration
  - ArchiMate Motivation Elements

7.2.2. Completing default metamodel

Should you want to complete the default schema with additional MetaClass, MetaAttribute:

1. Import the MGL file located in "MGL" folder.
2. Duplicate the standard Metamodel and diagram you want to complete.
3. Add the elements of metamodel you want.
4. Copy and keep for later the absolute identifier of the metamodel object.

7.2.3. Important rules

This generator applies the following rules:

- Only concrete metamodel is generated
- MetaAssociation must be in the diagram to be generated, including the abstract version of the MetaAssociation.

7.3. Step 2: Configure the generator

From HOPEX store (https://store.mega.com/modules/) download **GraphQL Mapping Json Generator** modulehttps://community.mega.com/t5/HOPEX-Store/GraphQL-REST-API/td-p/21381.

Unzip the **GraphQL Mapping Json Generator.haspkg** in "java" folder (you may need to create it):

C:\temp\java

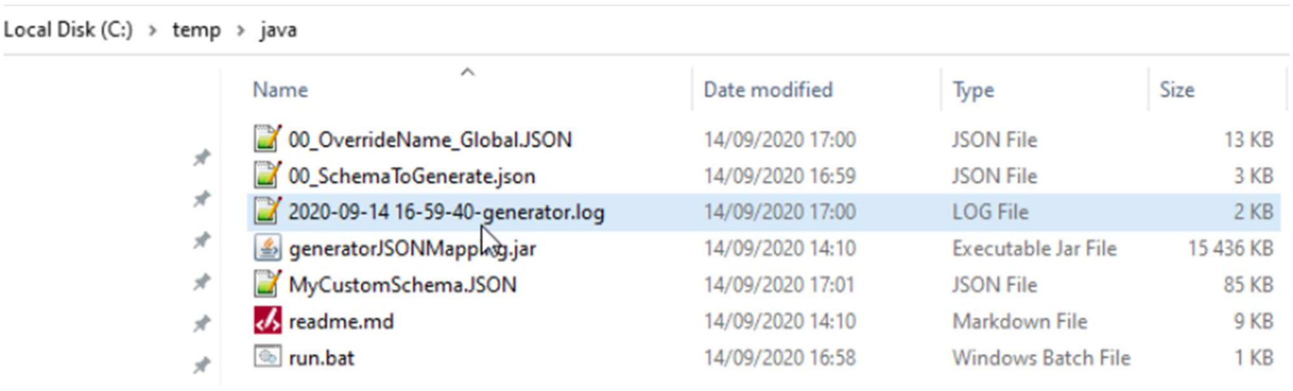
You can adjust according to your case:

- Open the "**00_SchemaToGenerate.json**" to add your schema. Past the absolute identifier you have saved from previous step.
- Edit the included section to false to avoid generating the other schema.

Example of "00_SchemaToGenerate.json":

```
{
"included": "true",
"schemaName": "ITPM",
"metaModelAbsoluteIdentifier": "TeEKeRMmSPYK",
"login": "Tibere",
"password": "Hopex",
"profile": "ITPM Functional Administrator"
},
```

- Login, password, and profile should reflect the restriction of metamodel you want to apply to the API. If you do not know which profile put 'HOPEX Customizer'. Access right will be read at runtime.


Local Disk (C:) › temp › java

Name	Date modified	Type	Size
00_OverrideName_Global.JSON	14/09/2020 17:00	JSON File	13 KB
00_SchemaToGenerate.json	14/09/2020 16:59	JSON File	3 KB
2020-09-14 16-59-40-generator.log	14/09/2020 17:00	LOG File	2 KB
generatorJSONMapping.jar	14/09/2020 14:10	Executable Jar File	15 436 KB
MyCustomSchema.JSON	14/09/2020 17:01	JSON File	85 KB
readme.md	14/09/2020 14:10	Markdown File	9 KB
run.bat	14/09/2020 16:58	Windows Batch File	1 KB

7.4. Step 3: Run the generator

Before running the generator edit the file '**run.bat**' to adjust:

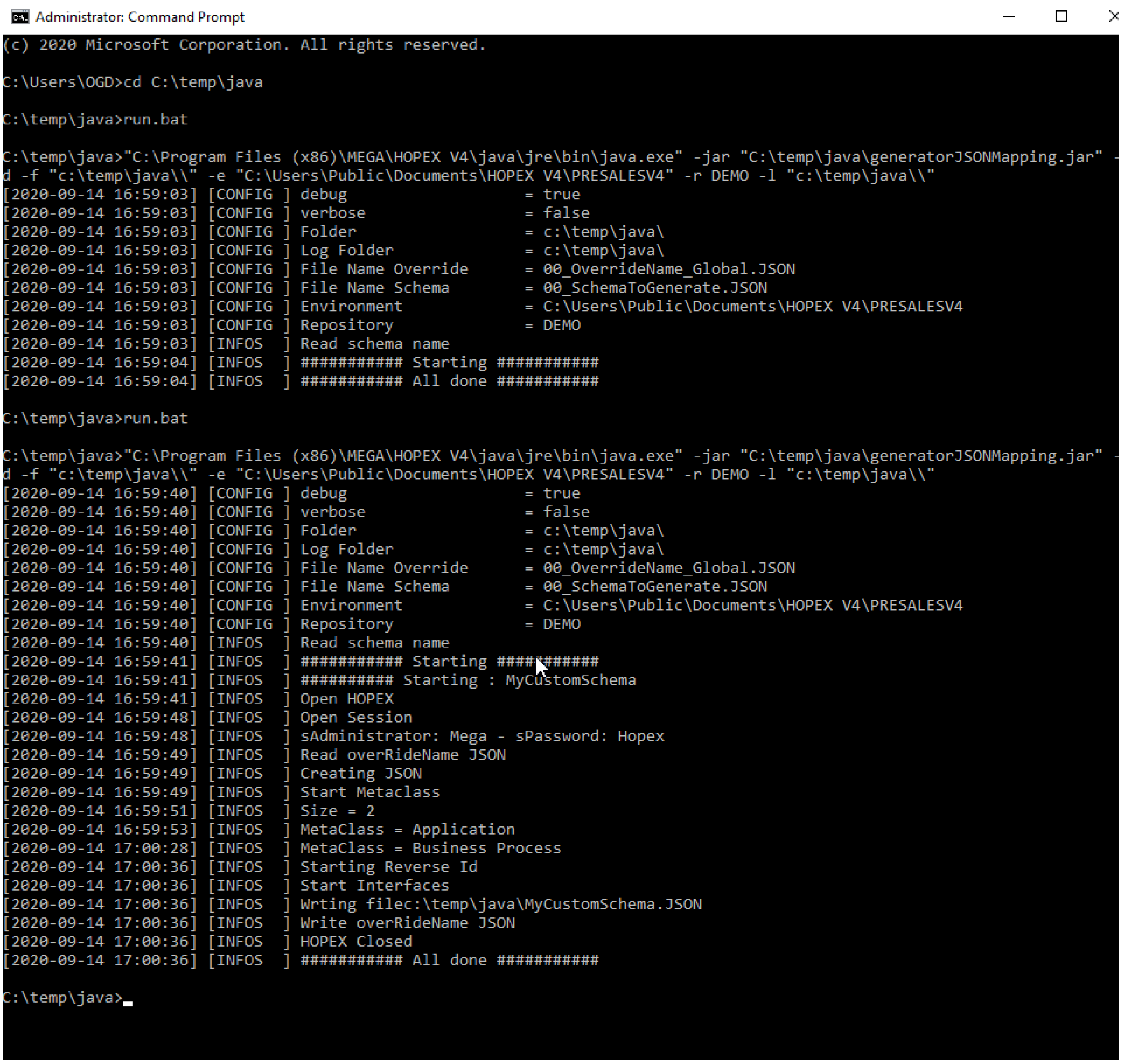
- folder location: HOPEX, log, config file
- environment path
- repository name
- debug option (-d)

To run the program, execute the run.bat file. In case of success the message appears in the console.

Processing the request may take **between 30 to 45 min... Be patient!** depending on the metamodel complexity.

Example of console message in debug:

```
[2020-09-14 08:35:21] [CONFIG ] debug                = true
[2020-09-14 08:35:21] [CONFIG ] verbose              = false
[2020-09-14 08:35:21] [CONFIG ] Folder               = c:\temp\java\
[2020-09-14 08:35:21] [CONFIG ] Log Folder           = c:\temp\java\
[2020-09-14 08:35:21] [CONFIG ] File Name Override   = 00_OverrideName_Global.JSON
[2020-09-14 08:35:21] [CONFIG ] File Name Schema     = 00_SchemaToGenerate.JSON
[2020-09-14 08:35:21] [CONFIG ] Environment          = C:\Users\Public\Documents\HOPEX
V4\PRESALESV4
[2020-09-14 08:35:21] [CONFIG ] Repository           = SOHO
[2020-09-14 08:35:21] [INFOS  ] Read schema name
[2020-09-14 08:35:22] [INFOS  ] ########### Starting ###########
[2020-09-14 08:35:22] [INFOS  ] ########## Starting : Custom
[2020-09-14 08:35:22] [INFOS  ] Open HOPEX
[2020-09-14 08:35:29] [INFOS  ] Open Session
[2020-09-14 08:35:29] [INFOS  ] sAdministrator: Mega - sPassword: *****
[2020-09-14 08:35:30] [INFOS  ] Read overRideName JSON
[2020-09-14 08:35:30] [INFOS  ] Creating JSON
[2020-09-14 08:35:30] [INFOS  ] Start Metaclass
[2020-09-14 08:35:32] [INFOS  ] Size = 1
[2020-09-14 08:35:33] [INFOS  ] MetaClass = Application
[2020-09-14 08:36:01] [INFOS  ] Starting Reverse Id
[2020-09-14 08:36:01] [INFOS  ] Start Interfaces
[2020-09-14 08:36:01] [INFOS  ] Wrting filec:\temp\java\SBB.JSON
[2020-09-14 08:36:01] [INFOS  ] Write overRideName JSON
[2020-09-14 08:36:01] [INFOS  ] HOPEX Closed
[2020-09-14 08:36:01] [INFOS  ] ########### All done ###########
```



```
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\OGD>cd C:\temp\java

C:\temp\java>run.bat

C:\temp\java>"C:\Program Files (x86)\MEGA\HOPEX V4\java\jre\bin\java.exe" -jar "C:\temp\java\generatorJSONMapping.jar" -
d -f "c:\temp\java\\" -e "C:\Users\Public\Documents\HOPEX V4\PRESALESV4" -r DEMO -l "c:\temp\java\\"
[2020-09-14 16:59:03] [CONFIG ] debug                  = true
[2020-09-14 16:59:03] [CONFIG ] verbose                = false
[2020-09-14 16:59:03] [CONFIG ] Folder                 = c:\temp\java\
[2020-09-14 16:59:03] [CONFIG ] Log Folder             = c:\temp\java\
[2020-09-14 16:59:03] [CONFIG ] File Name Override     = 00_OverrideName_Global.JSON
[2020-09-14 16:59:03] [CONFIG ] File Name Schema       = 00_SchemaToGenerate.JSON
[2020-09-14 16:59:03] [CONFIG ] Environment            = C:\Users\Public\Documents\HOPEX V4\PRESALESV4
[2020-09-14 16:59:03] [CONFIG ] Repository             = DEMO
[2020-09-14 16:59:03] [INFOS  ] Read schema name
[2020-09-14 16:59:04] [INFOS  ] ########### Starting ###########
[2020-09-14 16:59:04] [INFOS  ] ########### All done ###########

C:\temp\java>run.bat

C:\temp\java>"C:\Program Files (x86)\MEGA\HOPEX V4\java\jre\bin\java.exe" -jar "C:\temp\java\generatorJSONMapping.jar" -
d -f "c:\temp\java\\" -e "C:\Users\Public\Documents\HOPEX V4\PRESALESV4" -r DEMO -l "c:\temp\java\\"
[2020-09-14 16:59:40] [CONFIG ] debug                  = true
[2020-09-14 16:59:40] [CONFIG ] verbose                = false
[2020-09-14 16:59:40] [CONFIG ] Folder                 = c:\temp\java\
[2020-09-14 16:59:40] [CONFIG ] Log Folder             = c:\temp\java\
[2020-09-14 16:59:40] [CONFIG ] File Name Override     = 00_OverrideName_Global.JSON
[2020-09-14 16:59:40] [CONFIG ] File Name Schema       = 00_SchemaToGenerate.JSON
[2020-09-14 16:59:40] [CONFIG ] Environment            = C:\Users\Public\Documents\HOPEX V4\PRESALESV4
[2020-09-14 16:59:40] [CONFIG ] Repository             = DEMO
[2020-09-14 16:59:40] [INFOS  ] Read schema name
[2020-09-14 16:59:41] [INFOS  ] ########### Starting ###########
[2020-09-14 16:59:41] [INFOS  ] ########## Starting : MyCustomSchema
[2020-09-14 16:59:41] [INFOS  ] Open HOPEX
[2020-09-14 16:59:48] [INFOS  ] Open Session
[2020-09-14 16:59:48] [INFOS  ] sAdministrator: Mega - sPassword: Hopex
[2020-09-14 16:59:49] [INFOS  ] Read overRideName JSON
[2020-09-14 16:59:49] [INFOS  ] Creating JSON
[2020-09-14 16:59:49] [INFOS  ] Start Metaclass
[2020-09-14 16:59:51] [INFOS  ] Size = 2
[2020-09-14 16:59:53] [INFOS  ] MetaClass = Application
[2020-09-14 17:00:28] [INFOS  ] MetaClass = Business Process
[2020-09-14 17:00:36] [INFOS  ] Starting Reverse Id
[2020-09-14 17:00:36] [INFOS  ] Start Interfaces
[2020-09-14 17:00:36] [INFOS  ] Wrting filec:\temp\java\MyCustomSchema.JSON
[2020-09-14 17:00:36] [INFOS  ] Write overRideName JSON
[2020-09-14 17:00:36] [INFOS  ] HOPEX Closed
[2020-09-14 17:00:36] [INFOS  ] ########### All done ###########

C:\temp\java>
```

- **Advantage**: you benefit from dedicated process. Adapted to heavy computation that will need several minutes/hours to respond.
- **Drawback**: takes time to respond.

8. Pagination in REST API with GraphQL

When querying elements through the REST API a lot of items can be returned. To limit the size of items returned it can be relevant to paginate the data.

Pagination allows you to request a certain chunk of objects. You can seek forward or backward through the objects and supply an optional starting object:

- To seek **forward**, use **first**; specify a starting object with **after**.
- To seek **backward**, use **last**; specify a starting object with **before**.

To describ the pagination here is an example of a list of 30 objects.

This sample query is on the Application object. With no pagination:

```
query application {
  application {
    id
    name
  }
}
```

Caution

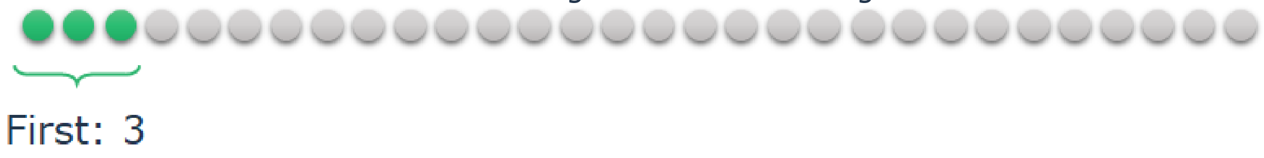
As soon as a pagination is applied an implicit sort by "ID" is executed, therefore a query without pagination and order will not be sorted as a query with pagination.

8.1. First elements with Skip or After

You can get the first element and skip an arbitrary amount of objects in forward direction you are seeking by supplying the first, skip and after.

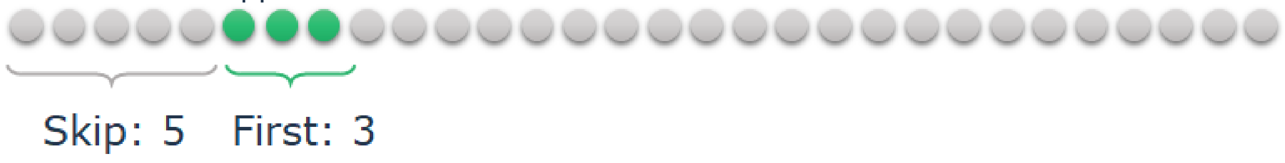
Five possibility of queries:

1. First: will take the number of elements given in first starting form the first item



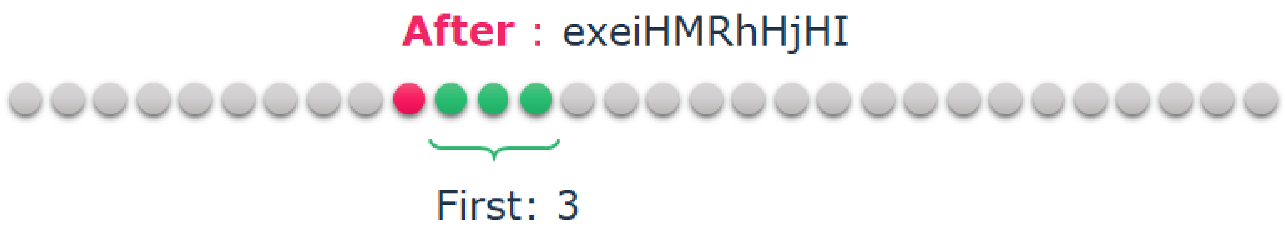
```
query application {
  application(first:3) {
    id
    name
  }
}
```

2. First with Skip: will take the number of elements given in first skipping the numbered of skipped elements



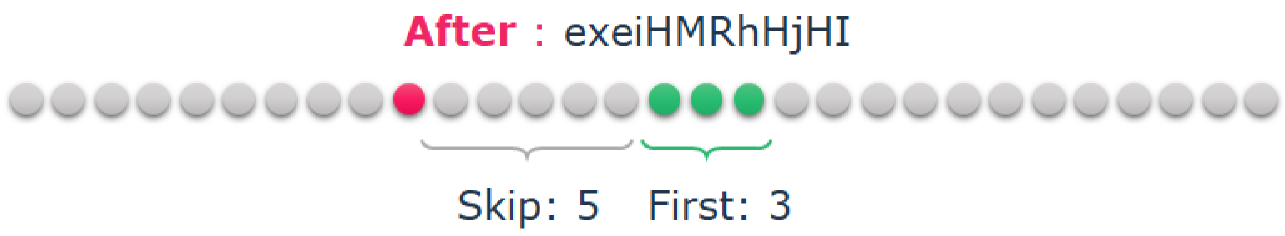
```
query application {
  application(first:3 skip:5) {
    id
    name
  }
}
```

3. First with After: will take the number of elements given in first starting after the given object ID



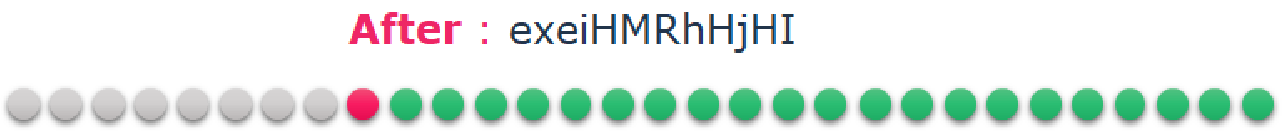
```
query application {
  application(first:3  after:"exeiHMRhHjHI" ) {
    id
    name
  }
}
```

4. Fist with After and Skip: will take the number of elements given in first starting after the given object ID and skipping the numbered of skipped elements



```
query application {
  application(first:3 skip:5 after:"exeiHMRhHjHI" ) {
    id
    name
  }
}
```

5. After alone: will start after the given object ID and take all the remaining objects



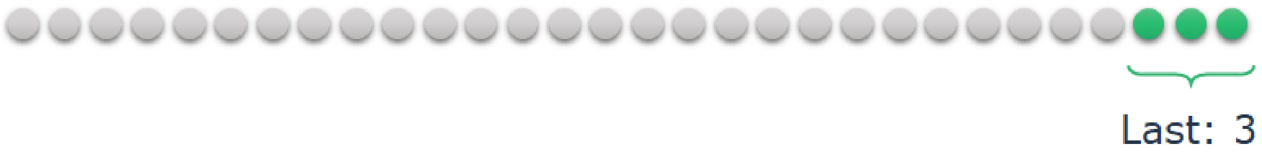
```
query application {
  application( after:"exeiHMRhHjHI" ) {
    id
    name
  }
}
```

8.2. Last elements with Skip and Before

You can get the last element and skip an arbitrary amount of objects in **backward** direction you are seeking by supplying the first, skip and before.

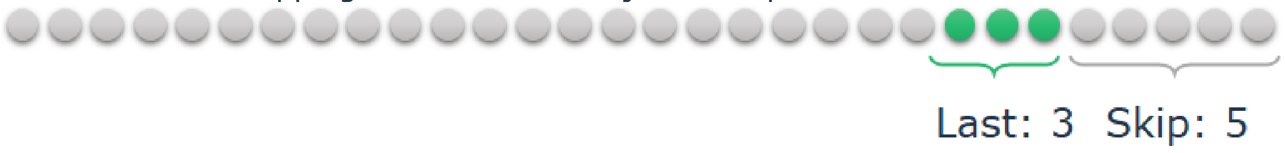
Five possibility of queries:

1. Last: will take the number of element given in last starting form the end of the item



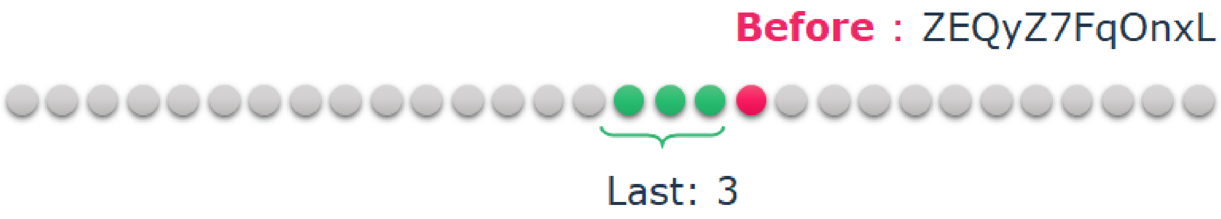
```
query application {
  application(last:3 ) {
    id
    name
  }
}
```

2. Last and Skip: will take the number of element given in last starting form the end of the item skipping the number of object in skip



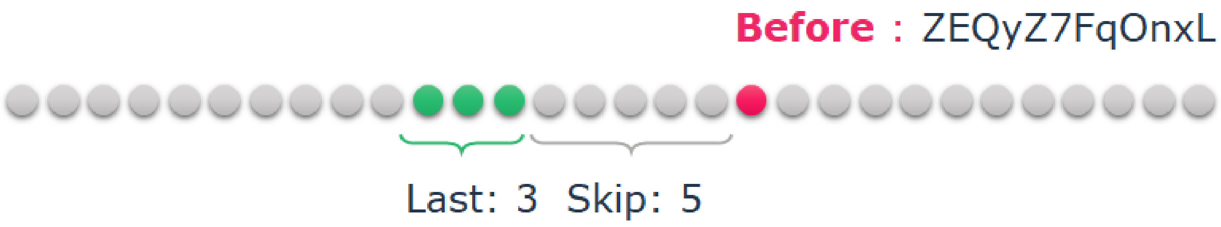
```
query application {
  application(last:3 skip:5 ) {
    id
    name
  }
}
```

3. Last and Before: will take the number of element given in last starting form the given id object



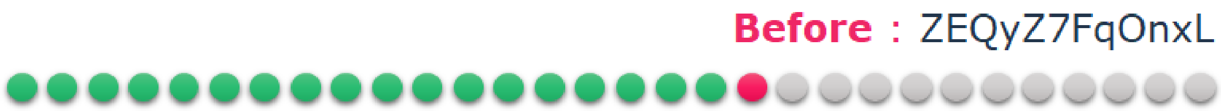
```
query application {
  application(last:3 before:"ZEQyZ7FqOnxL") {
    id
    name
  }
}
```

4. Last, Skip and Before: will take the number of element given in last starting form the given id object skipping the number of object in skip



```
query application {
  application(last:3 skip:5 before:"ZEQyZ7FqOnxL") {
    id
    name
  }
}
```

5. Before Only: will take all the object before the given object



```
query application {
  application(before:"ZEQyZ7FqOnxL") {
    id
    name
  }
}
```

Important

You cannot combine first with before or last with after. If you do so in a query, before or after will simply be ignored and only first or last is applied (at the very beginning or end of the list, depending on which you're using).

Note that you can query for more nodes than exist without an error message.

8.3. Pagination and Sort

If you combine pagination option and order capabilities the pagination will be done after the sort is executed.

Example of query:

```
query application {
  application(orderBy:[cloudComputing_ASC] first:3) {
    id
    name
    cloudComputing
  }
}
```

9. Filtering data (where condition) in REST API with GraphQL

When building graphQL to query element, it is sometime interesting to filter the data by different criteria: filtering by name, by Id, by relationships... Filters represent a "where" condition in queries.

Filtering can:

- filter by fields (MetaAttributes)
- filter by relationships (MetaAssociation)
- combine criteria: and, or
- adapt the criteria to the type: string, date, numbers...

9.1. How to make a filter?

The filter is an argument of the query with the keyword "filter". When using the auto-completion (with GraphiQl or Postman) the system propose the possible filter criteria. Technically the filter represents a "where" in an query (ERQL / SQL).

- Filtering an application by its name

Query	Result

```
query {
  application(filter:{name:"Account Management"})
  {
    id
    name
  }
}
```
	```
	{
	  "data": {
	    "application": [
	      {
	        "id": "IubjeRlyFfT1",
	        "name": "Account Management"
	      }
	    ]
	  }
	}
	```

The current filter will limit the applications that have the name strictly equals to the element given in the double quotes (see String filters for all string filters options).

- Filtering an application by its status

Query	Result

```
query {

application(filter:{statusReview_in:[Validated,UpdateInProgress]})
  {
    id
```
	```
	{
	  "data": {
	    "application": [
	      {
	        "id": "IubjeRlyFfT1",
	```

```
    name
  }
}
```

```
        "name": "Account
Management"
      },
      {
        "id": "pGCe26yn8zc3",
        "name": "Booking
Management"
      }
    ]
  }
}
```

The current filter will limit the applications that have their status review MetaAttribute strictly equals to Validated or UpdateInProgress . See below for all enumeration filters options.

- Filtering an application by its owner

Query

```
query {
  application(filter:{iTOwner_PersonSystem_some:{name:"Eric"}})
  {
    name
    iTOwner_PersonSystem {
      name
    }
  }
}
```

Result

```
{
  "data": {
    "application": [
      {
        "name": "MyCompany.com",
        "iTOwner_PersonSystem": [
          {
            "name": "Eric"
          }
        ]
      },
      {
        "name": "Splio EmailForge",
        "iTOwner_PersonSystem": [
          {
            "name": "Eric"
          }
        ]
      },
...
```

The current filter will limit the application that have **at least one** IT owner strictly named "Eric". See below for all relationships filters options.

9.2. How to combine filters?

By default combining filtering criteria will be an "**and**" operator. You can explicitly select the "**and**" or "**or**" operator.

1. Default "**and**" behavior

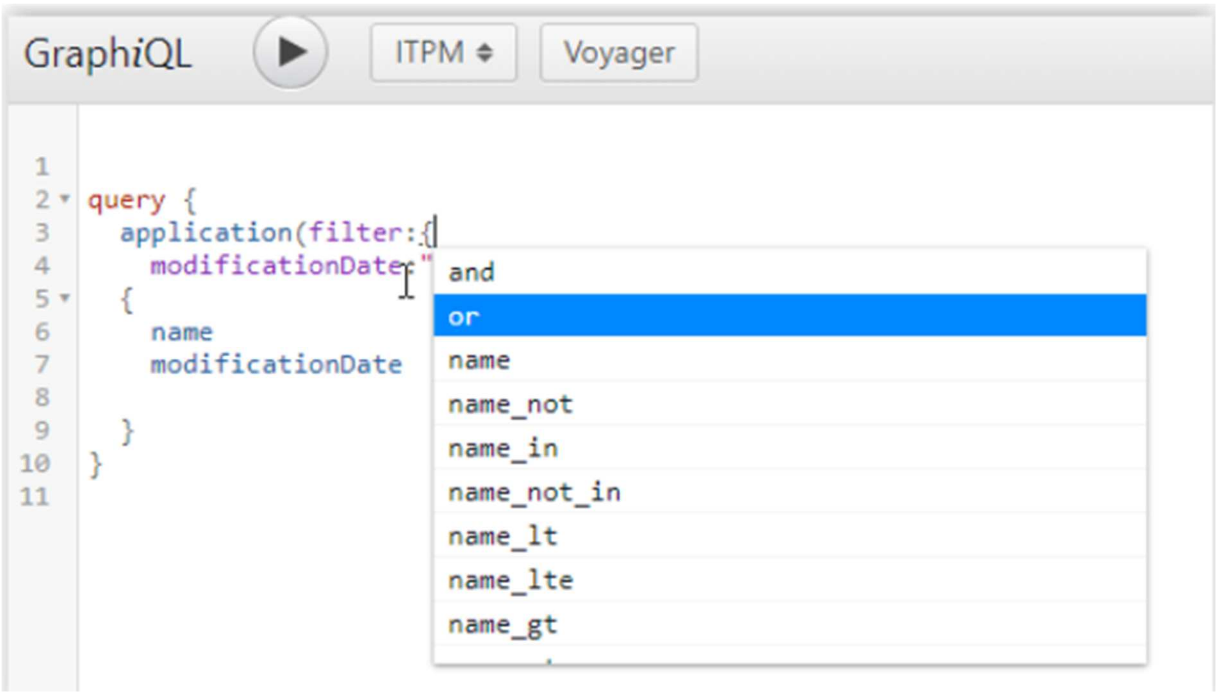
Query	Result

```
query {
  application(filter:{
modificationDate:"2020-04-07"
name_contains:"CRM"})
  {
    name
    modificationDate
  }
}
```
	```
	{
	  "data": {
	    "application": [
	      {
	        "name": "CRM US",
	        "modificationDate": "2020-04-07"
	      },
	      {
	        "name": "CRM Europe",
	        "modificationDate": "2020-04-07"
	      }
	    ]
	  }
	}
	```

In the filter curly bracket just add the filtering criteria and they will need to be **all** valid to return a result.

- Explicit "**and**" or "**or**"

In the filter you can select as a filter "**and**" or "or" as shown in the screenshot:



```
query {
  application(filter:{
    modificationDate:"
  {
    name
    modificationDate

  }
}
```

```
and
or
name
name_not
name_in
name_not_in
name_lt
name_lte
name_gt
```

Query	Result

query {
 	
application(filter:{or:{modificationDate:"2020-04-07" name_contains:"CRM"}})
  {
    name
    modificationDate
  }
}
	{
	  "data": {
	    "application": [
	      {
	        "name": "MyCompany.com",
	        "modificationDate": "2020-04-07"
	      },
	      {
	        "name": "CRM Management",
	        "modificationDate": "2017-09-28"
	      },
	  …

Because this filter contains an "or" **one of the criteria** must be valid to return a result. In this example either the date or the name.

9.3. Type of filters by types of fields

There are several types of fields:

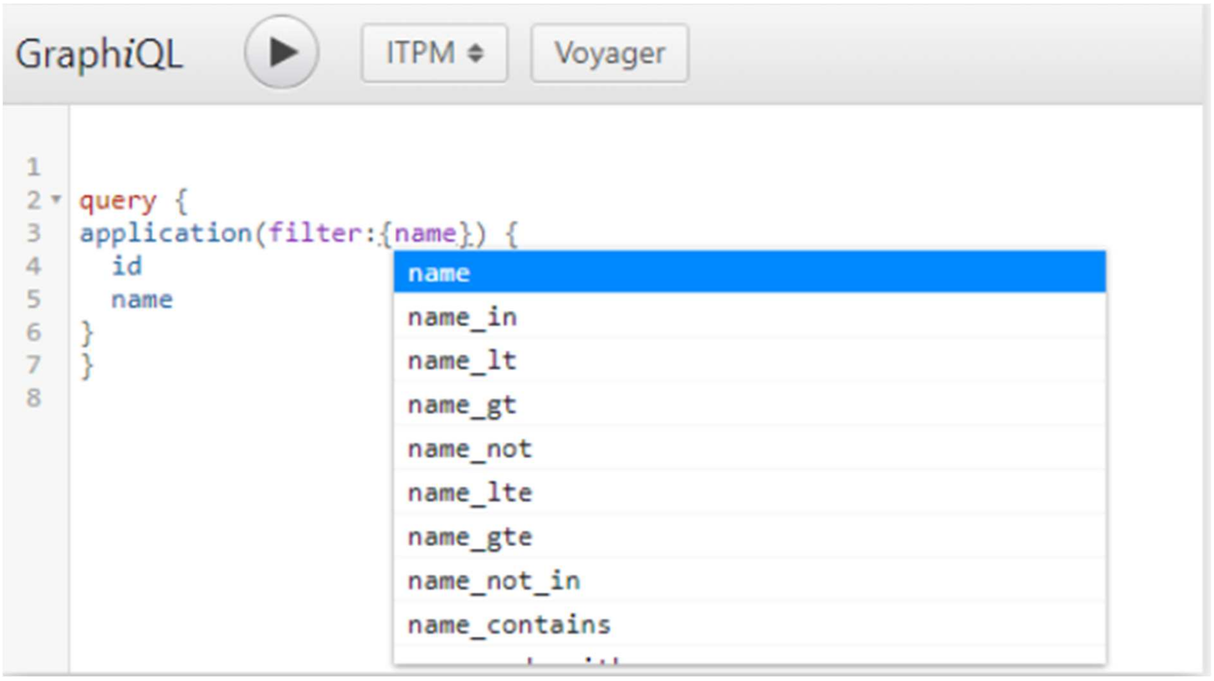
- String
- Date and DateTime
- Numbers: float, int, double, currency
- Enumeration
- Boolean

For each field the available filter allows various combination of criteria: equals, contains, greater than, in, not in.

9.3.1. String filters

All string fields propose the following filter criteria. For instance for **Name**:

name: String # matches all fields with exact value
name_not: String # matches all fields with different value
name_in: [String!] # matches all fields with value in the passed list
name_not_in: [String!] # matches all fields with value not in the passed list
name_lt: String # matches all fields with lesser value
name_lte: String # matches all fields with lesser or equal value
name_gt: String # matches all fields with greater value
name_gte: String # matches all fields with greater or equal value
name_contains: String # matches all fields with a value that contains given substring
name_not_contains: String # matches all fields with a value that does not contain given substring
name_starts_with: String # matches all fields with a value that starts with given substring
name_not_starts_with: String # matches all fields with a value that does not start with given substring
name_ends_with: String # matches all fields with a value that ends with given substring
name_not_ends_with: String # matches all fields with a value that does not end with given substring

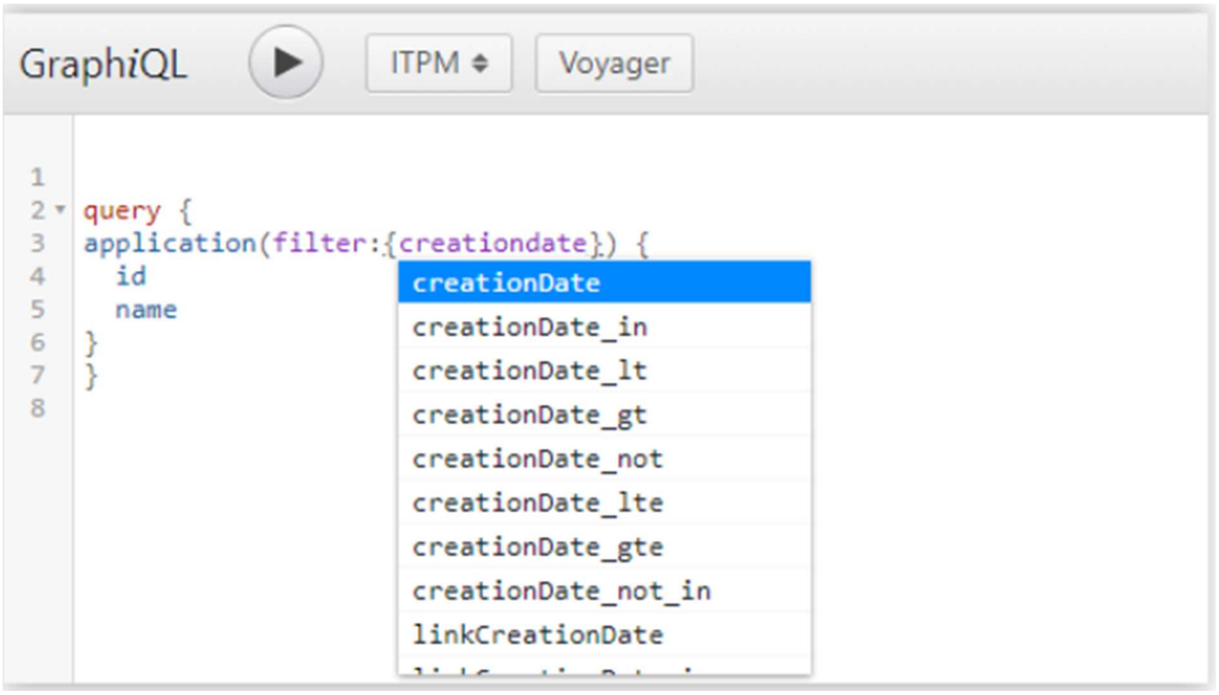


```
query {
application(filter:{name}) {
  id
  name
}
}
```

9.3.2. Date or DateTime filters

All date fields propose the following filter criteria. For instance for **Creation Date**:

creationDate: DateTime # matches all fields with exact value
creationDate_not: DateTime # matches all fields with different value
creationDate_in: [DateTime!] # matches all fields with value in the passed list
creationDate_not_in: [DateTime!] # matches all fields with value not in the passed list
creationDate_lt: DateTime # matches all fields with lesser value
creationDate_lte: DateTime # matches all fields with lesser or equal value
creationDate_gt: DateTime # matches all fields with greater value
creationDate_gte: DateTime # matches all fields with greater or equal value



```
query {
application(filter:{creationdate}) {
  id
  name
}
}
```

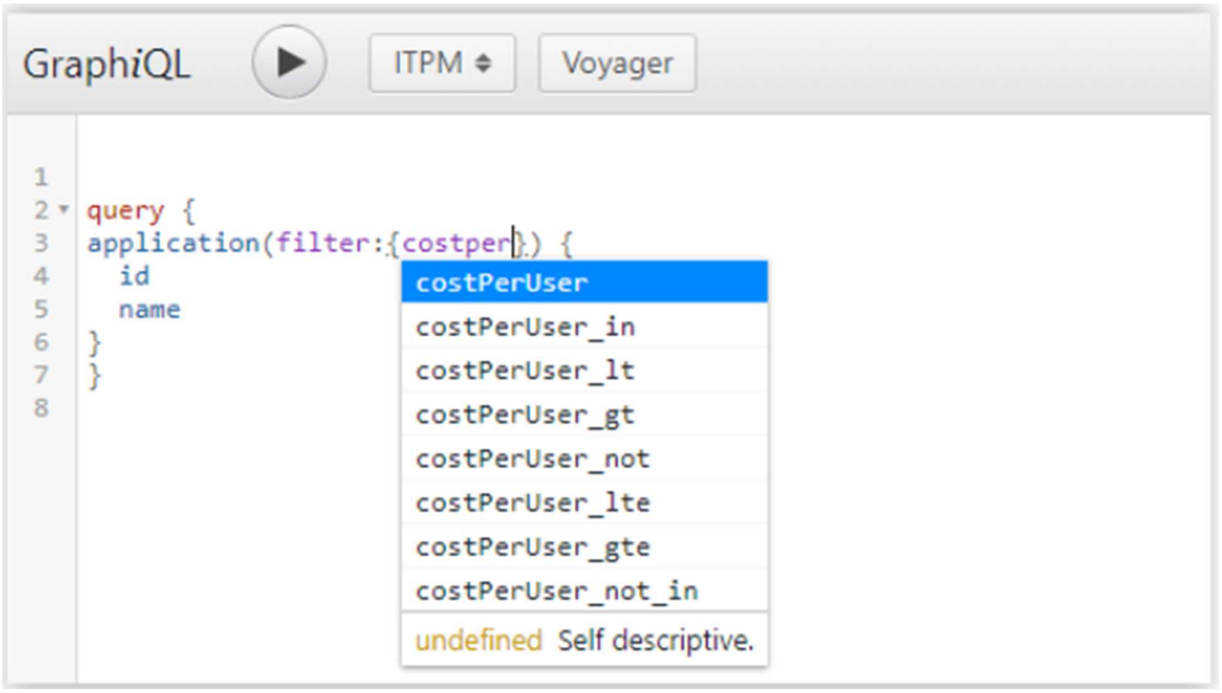
```
creationDate
creationDate_in
creationDate_lt
creationDate_gt
creationDate_not
creationDate_lte
creationDate_gte
creationDate_not_in
linkCreationDate
```

9.3.3. Numbers filters

All numbers (int, float, double) fields propose the following filter criteria.

For example for **Cost Per User**:

costPerUser: Number # matches all fields with exact value
costPerUser_not: Number # matches all fields with different value
costPerUser_in: [Number!] # matches all fields with value in the passed list
costPerUser_not_in: [Number!] # matches all fields with value not in the passed list
costPerUser_lt: Number # matches all fields with lesser value
costPerUser_lte: Number # matches all fields with lesser or equal value
costPerUser_gt: Number # matches all fields with greater value
costPerUser_gte: Number # matches all fields with greater or equal value



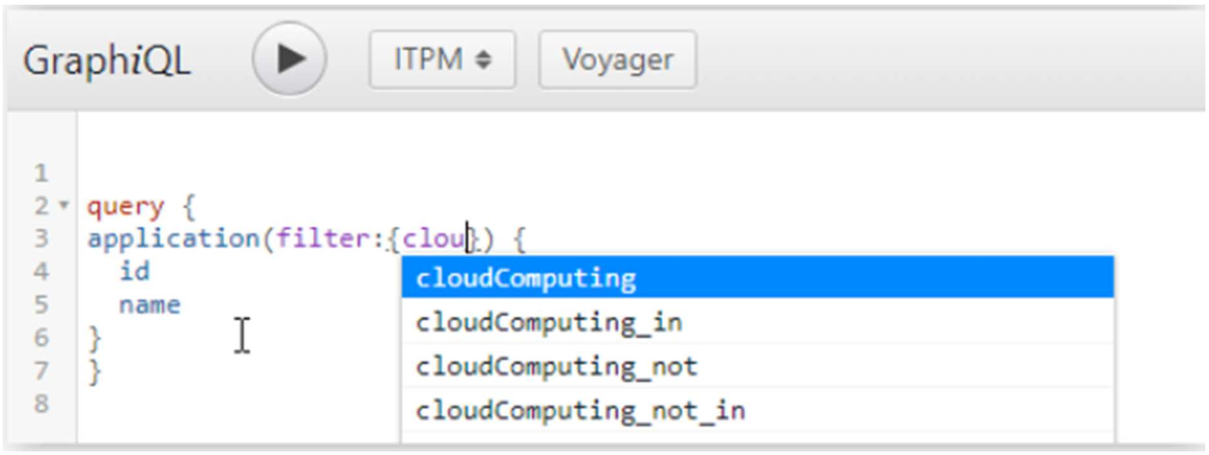
```
query {
application(filter:{costper}) {
  id
  name
}
}
```

```
costPerUser
costPerUser_in
costPerUser_lt
costPerUser_gt
costPerUser_not
costPerUser_lte
costPerUser_gte
costPerUser_not_in
undefined Self descriptive.
```

9.3.4. Enumeration filters

All enumeration fields (MetaAttribute Value) propose the following filter criteria. For instance for **Cloud Computing**:

cloudComputing: cloudComputing # matches all fields with exact value
cloudComputing_not: cloudComputing # matches all fields with different value
cloudComputing_in: [cloudComputing] # matches all fields with value in the passed list
cloudComputing_not_in: [cloudComputing] # matches all fields with value not in the passed list



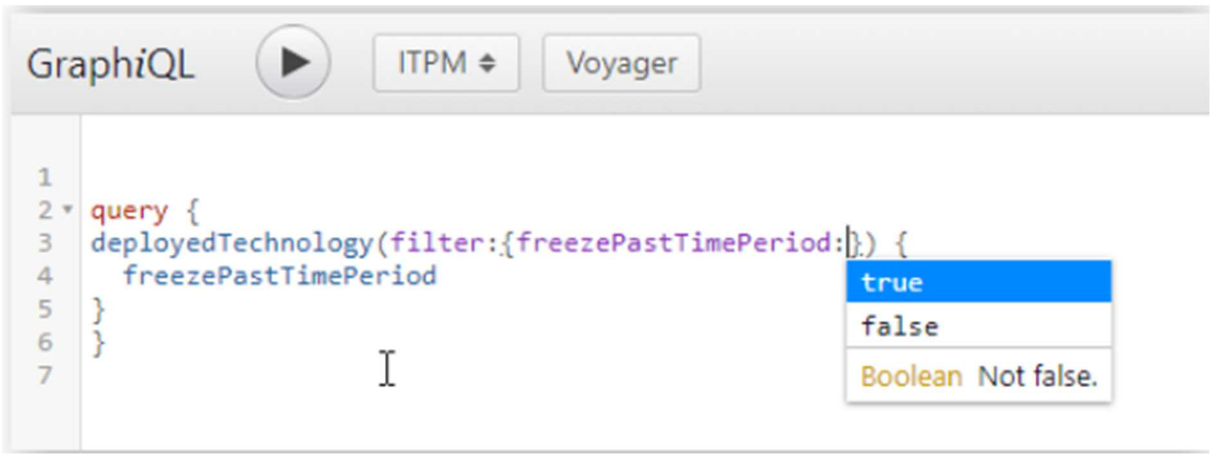
```
query {
application(filter:{clou}) {
  id
  name
}
}
```

```
cloudComputing
cloudComputing_in
cloudComputing_not
cloudComputing_not_in
```

9.3.5. Boolean filters

All boolean propose the following filter criteria. For instance for **Freeze Past Time Period**:

FreezePastTimePeriod: Boolean # matches all fields with exact value
FreezePastTimePeriod_not: Boolean # matches all fields with different value



```
query {
deployedTechnology(filter:{freezePastTimePeriod:}) {
  freezePastTimePeriod
}
}
```

10. Mutation: Absolute/External/Temporary - Identifier

With GraphQL you can make query or mutation on object of the repository. When syncing with an external tool you may want to use the identifier of the other tool to create or connect object.

GraphQL provides 3 types of identifiers:

- **Absolute Identifier:** this is the ID generated by HOPEX when a new object is created. The value cannot be set by the API. It can be used when making query to get the HOPEX unique identifier global.
  - This field **cannot be blank** or null.
  - This field is composed of **12 characters** composed of letter and digit and special characters.
  - This field is **invariant** overtime.
- **External Identifier:** this is an ID that can be given when a new object is created. The uniqueness is limited to the MetaClass for which you are creating an object. This ID is used when importing data from another system to be able to retrieve, at a later stage, the object created from this system.
  - This field **can be blank** or null
  - This field as a maximum length of **1024 characters**
  - This field is **not invariant** overtime

2. **Temporary Identifier:** this is the same principal as the External Identifier. The main difference is that the validity of this ID is limited to the mutation you are doing. The information will not be store in the database and cannot be reused at a later stage.

10.1. Making a Query

A query can only be made on:

- the Absolute Identifier shortly called "**id**" or
- the External Identifier shortly called "**externalId**".

Note: As the Temporary identifier is not physically stored it cannot be queried.

"**id**" and "**externalId**" identifiers can be used to:

- filter query
- ensure that the retrieved object is the expected one.

10.1.1. Example: query with a result containing the identifiers

Query 1	Result 1
query AppID {	{

application {
  id
  externalId
  name
 }
}
	"data": {
	 "application": [
	  {"id": "IubjeRlyFfT1", "externalId": "My ID App1", "name": "Account Management"},
	  {"id": "iSS7AQY6NrgU","externalId": "My ID App2","name": "Account Payable"
	  }
	]
	 }
	}

10.1.2. Example: query containing a filter on the external identifier

Query 2	Result 2

query AppID {
 application
  (filter:{externalId_contains:"My ID"}) {
  id
  externalId
  name
 }
}
	{
	 "data": {
	  "application": [
	   {"id": "IubjeRlyFfT1", "externalId": "My ID App1", "name": "Account Management"},
	   {"id": "iSS7AQY6NrgU","externalId": "My ID App2","name": "Account Payable"
	   }
	  ]
	 }
	}

10.2. Making a Mutation

10.2.1. Creation without an Identifier

The default behavior is to let the system give an absolute identifier to an object. The identifier is unique in the whole HOPEX repository. If you create several objects, even with the same characteristics, each object has a unique.

Query	Result

mutation basicCreation {
 createApplication(application:{
	{
	 "data": {

```
    name:"my new app"
  }) {
    id
    name
  }
}
```
	```
	    "createApplication": {
	      "id": "magoMpShXLDE",
	      "name": "my new app"
	    }
	  }
	}
	```

10.2.2. Creation with an External Identifier

The only way to control the identifier is to use the External Identifier. When the object is created you can specify the value. This value is **unique but limited** to the MetaClass object you are creating. Moreover this external identifier is **not invariant** so you can modify it after the object was created.

CAUTION: the profile you use must be granted to right to read and write this MetaAttribute. This is not always the default right in standard HOPEX.

Query	Result

```
mutation basicCreation {
  createApplication(id:"My ID App3"
    idType:EXTERNAL
    application:{
    name:"my new app 3"
  }) {
    id
    externalId
    name
  }
}
```
	```
	{
	  "data": {
	    "createApplication": {
	      "id": "RcgoRyShXXFE",
	      "externalId": "My ID App3",
	      "name": "my new app 3"
	    }
	  }
	}
	```

Should you want to update the External Identifier you need to first query the object with the Absolute Identifier.

Query	Result

```
mutation updateExtId {
  updateApplication(id:"RcgoRyShXXFE"
    idType:INTERNAL
    application:{
    externalId:"my new app 3 -updated "
  }) {
    id
    externalId
```
	```
	{
	  "data": {
	    "updateApplication": {
	      "id": "RcgoRyShXXFE",
	      "externalId": "my new app 3 -updated ",
	      "name": "my new app 3"
	    }
	  }
	```

`    name`
`  }`
`}`
	`}`

If you do several mutations in a row you can use this External Identifier as a mean to connect object together.

For example: the following mutation performs

1. Creation of an Application.
2. Creation of a Busines Process.
3. Connection of the two created objects.

Query	Result

`mutation multiple {`
`  createApplication(id:"MyAPPID5" idType:EXTERNAL`
`    application:{name:"Application 5"}) {`
`    name`
`  }`
 	
`  createBusinessProcess(id:"My Process 5" idType:EXTERNAL`
`    businessProcess:{name:"Process 5"`
`      application:{`
`        action:ADD`
`        list:[{id:"MyAPPID5" idType:EXTERNAL}]`
`      }`
`    }) {`
`    name`
`    application {`
`      name`
`      externalId`
`    }`
`  }`
`}`
	`{`
	`  "data": {`
	`    "createApplication": {`
	`      "name": "Application 5"`
	`    },`
	`    "createBusinessProcess": {`
	`      "name": "Process 5",`
	`      "application": [`
	`        {`
	`          "name": "Application 5",`
	`          "externalId": "MyAPPID5"`
	`        }`
	`      ]`
	`    }`
	`  }`
	`}`

10.2.3. Creation with a Temporary Identifier

The temporary Identifier works as the external identifier. The only difference is that the value of this identifier is not stored in the database.

11. Asynchronous versus Synchronous Web service call

Each of the GraphQL endpoints, to query the repository, can return results in synchronous or asynchronous way.

To avoid confusion, it is important to make the difference between:

- What does the endpoint return ? Synchronous/Asynchronous **response**
- How the endpoint is called ? Synchronous/Asynchronous **call**

11.1. What does the endpoint return?

There are two cases:

1. **synchronous** (response): the endpoint returns always a response (good or bad) with the result regardless of the time it took to compute the result. The URL like this one {{server_url}}/HOPEXGraphQL/api/ITPM are synchronous.
2. **asynchronous** (response): the endpoint returns a task ID to use to recall the endpoint to get the result. The URL like this one {{server_url}}/HOPEXGraphQL/api/**async**/ITPM are asynchronous.

The **first** case: is particularly useful to get immediate results when calling the API. The drawback is that if it takes time to get a result, the caller is hanging and wait for the response. In case of response taking several seconds, timeout may occur on the network or web browser.

The **second** case: is particularly useful for requests that are time consuming to compute. A first call is made to the API. This call returns a **x-hopex-task** and a **x-hopex-sessiontoken**. The status code of the response will be HTTP **206 Partial response.** This information is then used to recall, in a pooling mechanism, the same endpoint until it returns **200 success**.

11.2. Calling the API in asynchronous way

1. Make a first call to the async endpoint {{server_url}}/HOPEXGraphQL/api/**async**/ITPM request header should contain: **x-hopex-context** and **x-hopex-wait**
2. If the time to compute the query is:
   a. lower than x-hopex-wait time: the API returns the x-hopex-task and x-hopex-sessiontoken in a 206 status
   b. higher than x-hopex-wait time: the API returns the body JSON response in a 200 status
3. In case of 206 status code: make several calls to the end point with in the header x-hopex-task and x-hopex-sessiontoken
   a. While you get 206 status the result is not available and the API is still computing.

b. As soon as you get 200 status the result is available in the body of the response.

The mechanism for asynchronous call is **web pooling.** As of now we do not provide any webhook mechanism.

11.3. How the endpoint is called?

Regardless of the endpoint called (sync or async) each programming language has its own way to make an HTTP request.

Then HTTP request can be made via a synchronous or asynchronous **call**. For instance, the programming can be "stuck" and wait for the response before to proceed to the next step of the programming; or the programming can "continue" and get triggered when the response is available.

As a result you can end up in the following situation:

- A synchronous call to the synchronous endpoint
- A synchronous call to the asynchronous endpoint
- An asynchronous call to the synchronous endpoint
- An asynchronous call to the asynchronous endpoint

11.4. Which is the best option?

The best option depends on the use case:

- Is it a web application?
- Is it a mobile app?
- What is the quality of the network?
- What are timeout options?

Asynchronous are better when the system can deal with the response at a later stage: web application, mobile app for instance.

Synchronous are better when the system needs the information to proceed to the next steps: integration with other systems for instance.

12. Diagram API: dowloading a diagram by REST API

HOPEX contains diagrams that have been designed or generated. Within the Web Front-End you can see and download the diagrams as images. The REST API also provides the ability to export the diagram as a file.

Supported formats are: **PNG**, **JPEG** and **SVG**.

12.1. Use case

You want to expose, as a read only, the diagrams of HOPEX into another application or an external website. This API is for download only, updates on the diagram are not allowed.

To perform this action:

1. **Select** the diagram in HOPEX via the GraphQL API.
2. **Download** the images file via a dedicated endpoint.

Depending on the use case you are, you may call the endpoint in a synchronous or asynchronous way. For details on how to call the API in asynchronous way, see Asynchronous versus Synchronous Web service call.

Note that the API complies with the confidentiality. If the diagram you try to access is confidential with the credential you used you will not be able to get the image.

12.1.1. Downloading a diagram (Metaclass Diagram or System Diagram)

To get your diagram as an image:

1. Make a graphQL query to get the **download URL** of the diagram.
2. Call this URL to get the **binary image** file.

Step 1: Getting the download URL

Execute a *GraphQL query on the endpoint of your choice* ***{{server_url}}****/HOPEXGraphQL/api/****{{schemaName}}*** *:*

Query	*Result*

```
query getDiagram {
  application {
    name
    diagram {
      name
      downloadUrl
    }
```
	```
	{
	  "data": {
	    "application": [
	      {
	        "name": "Excel Checker",
	        "diagram": [
	          {
	```

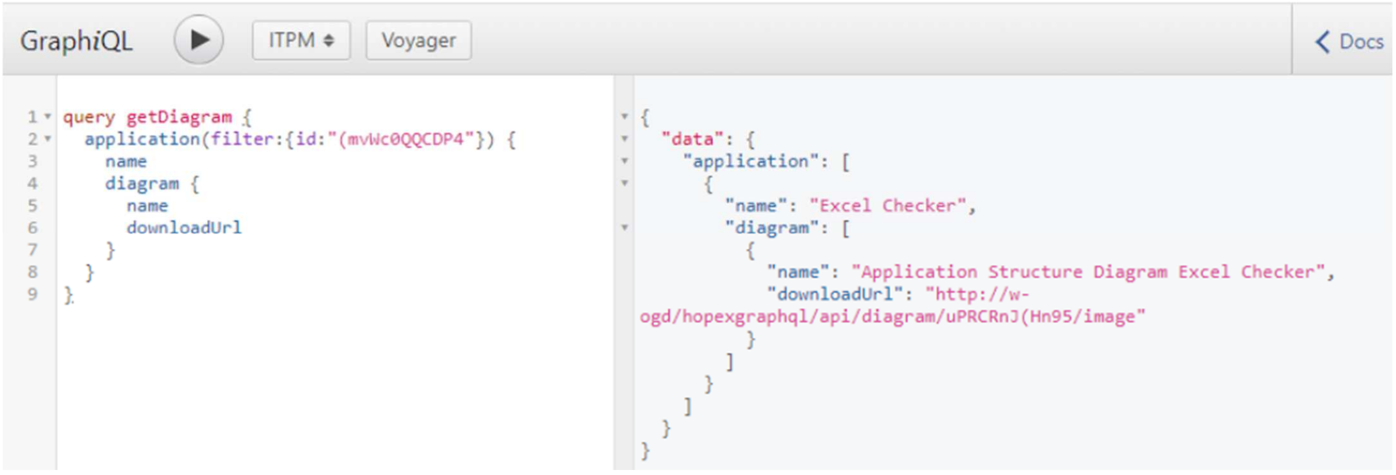
} }	"name": "Application Structure Diagram Excel Checker", "downloadUrl": "http://w-ogd/hopexgraphql/api/diagram/uPRCRnJ(Hn95/image" }] }] } }

The download URL are like:

{{server_url}}/hopexgraphql/api/diagram/**{{diagramId}}**/image

where **{{diagramId}}** represent the absolute identifier of the object in HOPEX. They must be called with **GET** verb.

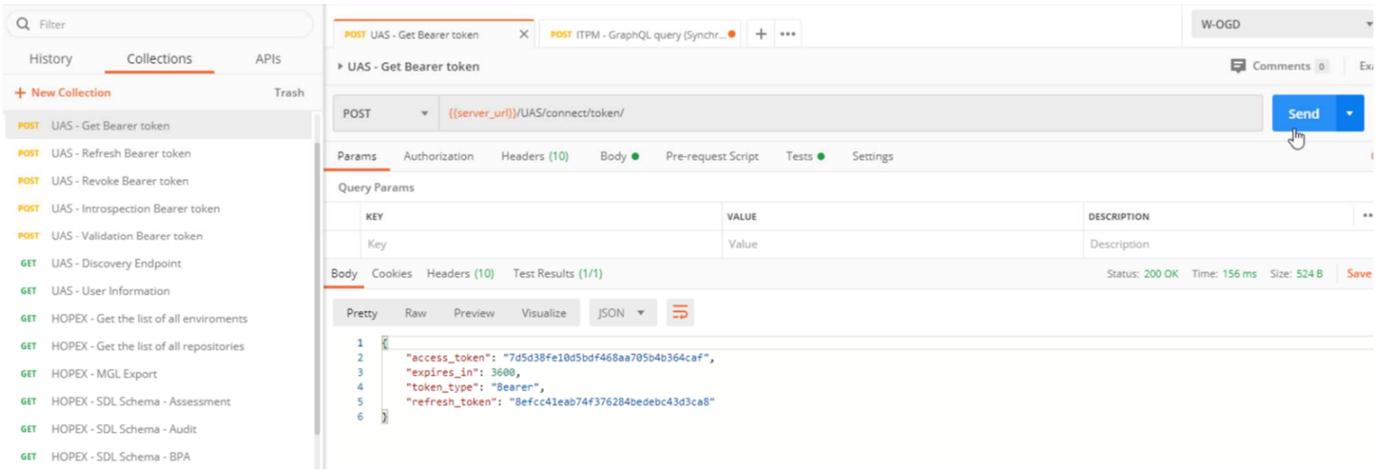
- Example with GraphiQL:



```
query getDiagram {
  application(filter:{id:"(mvWc0QQCDP4"}) {
    name
    diagram {
      name
      downloadUrl
    }
  }
}
```

```
{
  "data": {
    "application": [
      {
        "name": "Excel Checker",
        "diagram": [
          {
            "name": "Application Structure Diagram Excel Checker",
            "downloadUrl": "http://w-ogd/hopexgraphql/api/diagram/uPRCRnJ(Hn95/image"
          }
        ]
      }
    ]
  }
}
```

- Examples with Postman:

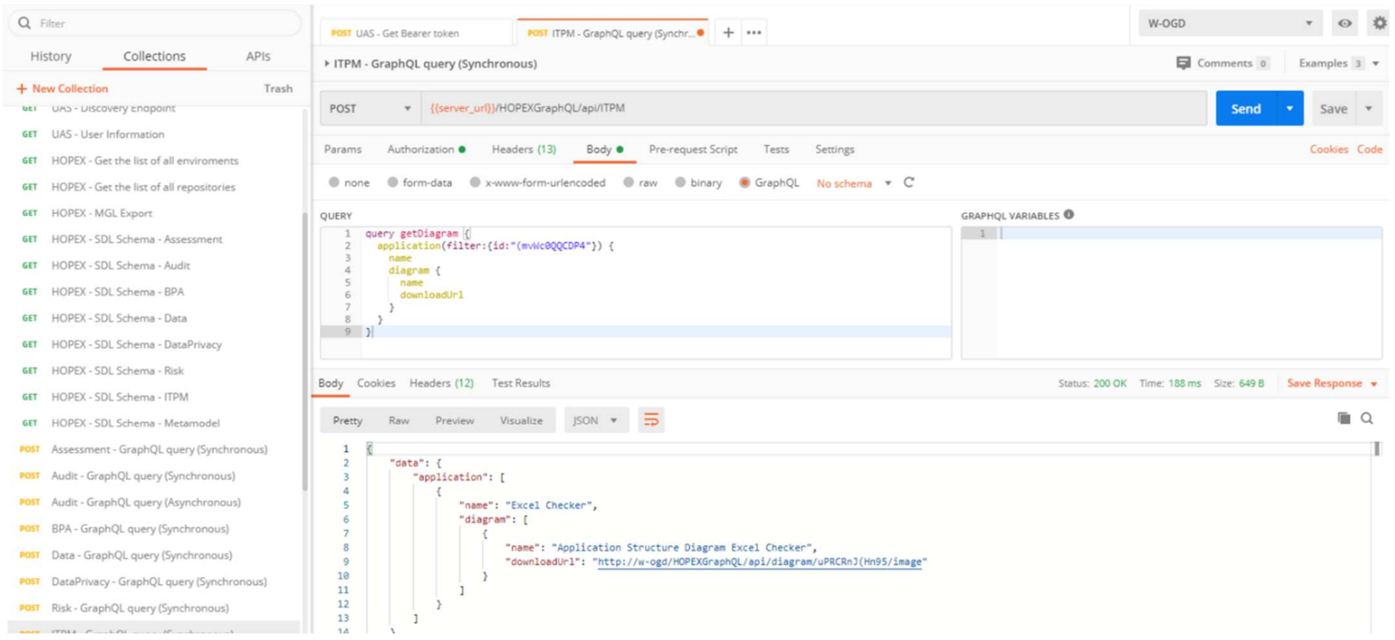

Filter
History
Collections
APIs
New Collection
Trash
POST UAS - Get Bearer token
POST UAS - Refresh Bearer token
POST UAS - Revoke Bearer token
POST UAS - Introspection Bearer token
POST UAS - Validation Bearer token
GET UAS - Discovery Endpoint
GET UAS - User Information
GET HOPEX - Get the list of all enviroments
GET HOPEX - Get the list of all repositories
GET HOPEX - MGL Export
GET HOPEX - SDL Schema - Assessment
GET HOPEX - SDL Schema - Audit
GET HOPEX - SDL Schema - BPA
POST UAS - Get Bearer token
POST ITPM - GraphQL query (Synchr...
W-OGD
UAS - Get Bearer token
Comments 0
Send
Params
Authorization
Headers (10)
Body
Pre-request Script
Tests
Settings
Query Params
KEY
VALUE
DESCRIPTION
Key
Value
Description
Body
Cookies
Headers (10)
Test Results (1/1)
Save
Pretty
Raw
Preview
Visualize
JSON

POST {{server_url}}/UAS/connect/token/

Status: 200 OK Time: 156 ms Size: 524 B

```
{
    "access_token": "7d5d38fe10d5bdf468aa705b4b364caf",
    "expires_in": 3600,
    "token_type": "Bearer",
    "refresh_token": "8efcc41eab74f376284bedebc43d3ca8"
}
```

Get a bearer



History | Collections | APIs

+ New Collection | Trash

- GET UAS - Discovery Endpoint
- GET UAS - User Information
- GET HOPEX - Get the list of all enviroments
- GET HOPEX - Get the list of all repositories
- GET HOPEX - MGL Export
- GET HOPEX - SDL Schema - Assessment
- GET HOPEX - SDL Schema - Audit
- GET HOPEX - SDL Schema - BPA
- GET HOPEX - SDL Schema - Data
- GET HOPEX - SDL Schema - DataPrivacy
- GET HOPEX - SDL Schema - Risk
- GET HOPEX - SDL Schema - ITPM
- GET HOPEX - SDL Schema - Metamodel
- POST Assessment - GraphQL query (Synchronous)
- POST Audit - GraphQL query (Synchronous)
- POST Audit - GraphQL query (Asynchronous)
- POST BPA - GraphQL query (Synchronous)
- POST Data - GraphQL query (Synchronous)
- POST DataPrivacy - GraphQL query (Synchronous)
- POST Risk - GraphQL query (Synchronous)

POST UAS - Get Bearer token | POST ITPM - GraphQL query (Synchr... | W-OGD

ITPM - GraphQL query (Synchronous)

POST {{server_url}}/HOPEXGraphQL/api/ITPM

Params | Authorization | Headers (13) | Body | Pre-request Script | Tests | Settings

none | form-data | x-www-form-urlencoded | raw | binary | [✓ GraphQL] | No schema

QUERY

```
query getDiagram {
  application(filter:{id:"(mvWc0QQCDP4"}) {
    name
    diagram {
      name
      downloadUrl
    }
  }
}
```

GRAPHQL VARIABLES

Body | Cookies | Headers (12) | Test Results — Status: 200 OK Time: 188 ms Size: 649 B

```
{
    "data": {
        "application": [
            {
                "name": "Excel Checker",
                "diagram": [
                    {
                        "name": "Application Structure Diagram Excel Checker",
                        "downloadUrl": "http://w-ogd/HOPEXGraphQL/api/diagram/uPRCRnJ(Hn95/image"
                    }
                ]
            }
        ]
```

query and result

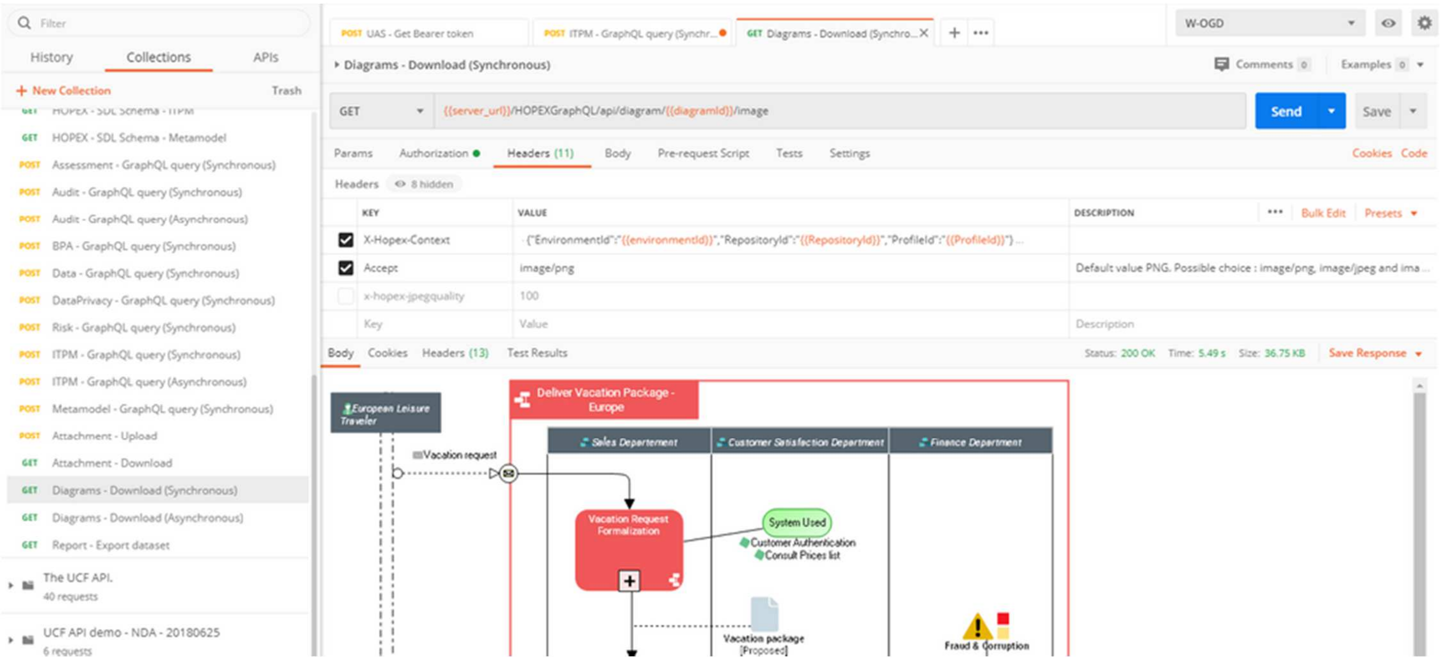
Step 2: Getting the image

When you call the download URL the **header** should contains, in addition to default headers:

- **Accept**: which defines the format of the image you expect to get. If not defined, the default value is PNG.
  - image/png
  - image/jpeg
  - image/svg+xml

As a result of the call the body contains the picture of the diagram.

Example with Postman:



History | Collections | APIs

+ New Collection | Trash

- GET HOPEX - SDL Schema - ITPM
- GET HOPEX - SDL Schema - Metamodel
- POST Assessment - GraphQL query (Synchronous)
- POST Audit - GraphQL query (Synchronous)
- POST Audit - GraphQL query (Asynchronous)
- POST BPA - GraphQL query (Synchronous)
- POST Data - GraphQL query (Synchronous)
- POST DataPrivacy - GraphQL query (Synchronous)
- POST Risk - GraphQL query (Synchronous)
- POST ITPM - GraphQL query (Synchronous)
- POST ITPM - GraphQL query (Asynchronous)
- POST Metamodel - GraphQL query (Synchronous)
- POST Attachment - Upload
- GET Attachment - Download
- GET Diagrams - Download (Synchronous)
- GET Diagrams - Download (Asynchronous)
- GET Report - Export dataset
- The UCF API. 40 requests
- UCF API demo - NDA - 20180625 6 requests

POST UAS - Get Bearer token | POST ITPM - GraphQL query (Synchr... | GET Diagrams - Download (Synchro... | W-OGD

Diagrams - Download (Synchronous)

GET {{server_url}}/HOPEXGraphQL/api/diagram/{{diagramId}}/image

Params | Authorization | Headers (11) | Body | Pre-request Script | Tests | Settings

Headers (8 hidden)

	KEY	VALUE	DESCRIPTION
✓	X-Hopex-Context	{"EnvironmentId":"{{environmentId}}","RepositoryId":"{{RepositoryId}}","ProfileId":"{{ProfileId}}"} ...	
✓	Accept	image/png	Default value PNG. Possible choice : image/png, image/jpeg and ima ...
	x-hopex-jpegquality	100	
	Key	Value	Description

Body | Cookies | Headers (13) | Test Results — Status: 200 OK Time: 5.49 s Size: 36.75 KB

European Leisure Traveler
Deliver Vacation Package - Europe
Sales Departement
Customer Satisfaction Department
Finance Department
Vacation request
Vacation Request Formalization
System Used
Customer Authentication
Consult Prices list
Vacation package [Proposed]
Fraud & Corruption

12.1.2. Result of the API

- **Status 200**: the diagram has been found and is returned in the body
- **Status 500**: an error has occurred with a body that contains the following message.
    - the absolute identifier you gave is not valid
    - the absolute identifier yet the diagram you are trying to acces is not visible with your credentials.

Example of error message:

```
{
    "Message": "An error has occurred.",
    "ExceptionMessage": "Empty Object Invocation:Only default and GetID methods are
available on a Empty Object",
    "ExceptionType": "System.Exception",
    "StackTrace": null
}
```

13. Attachment API: Uploading or Downloading Business Document

In HOPEX, you can store attachments in the concept called **Business Document**.

Supported files (e.g: DOCX, XLSX, JPEG, PNG, PDF) can be uploaded/downloaded from the web interface. You may want to access this information by REST API to perform the same upload/download action.

See video: https://youtu.be/nS7WYsDtr0Q.

13.1. Use case

The Attachment API allows you to upload and query file attachments. You can upload or retrieve a single file with each request. The Attachment API complies with HOPEX limitations on uploaded files, such as maximum file size and allowed attachment types.

The API supports the following features:

- **Upload:** it manages new business documents or new version of an existing document
- **Download:** it allows to download the latest version or a selected version of a business document.

To perform this action:

1. Create or select the object in HOPEX via the GraphQL API.
2. Upload/download the binary file via a dedicated endpoint.

13.2. Downloading an attachment (Business Document)

To download an attachment with the REST API:

1. Create a GraphQL query to get the download URL for your document.
2. Call the URL to download the file.

*GraphQL query executed on the **{{server_url}}**/HOPEXGraphQL/api/**{{schemaName}}** endpoint*

Query	*Result*

```
query {
  businessDocument{
    id
    name
    downloadUrl
  }
}
```
	```
	{
	  "data": {
	    "businessDocument": [
	      {
	        "id": "Dbm4QHHbM5vH",
	        "name": "Audit Report",
	        "downloadUrl": "http://w-ogd/hopexgraphql/api/attachment/Dbm4QHHbM5vH/file"
	```

	```
	      }
	    ]
	  }
	}
	```

The download URL are like:

{{server_url}}/HOPEXGraphQL/api/attachment/**{{documentId}}**/file

where **{{documentId}}** represents the absolute identifier of the object in HOPEX. They must be called with **GET** verb.

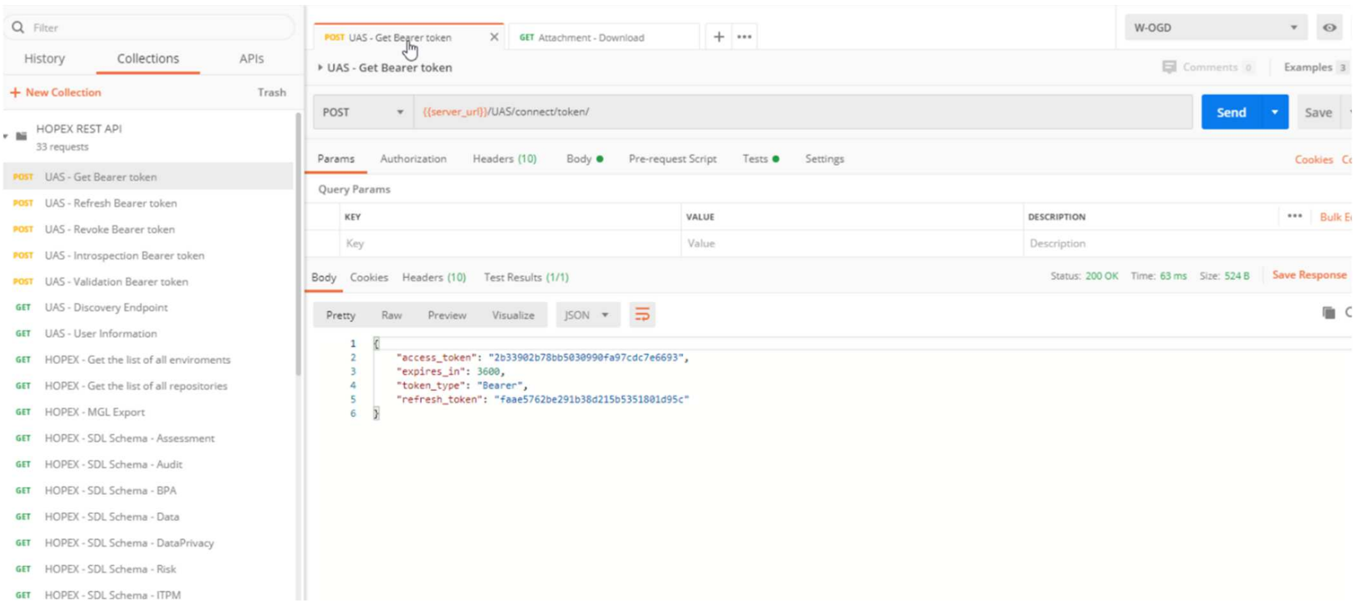
Should you want to access a specific version of the business document, use the absolute identifier of the version you want to download. You can make a graphQL query on the business document version.

Query	*Result*

```
query {

businessDocument(filter:{id:"Dbm4QHHbM5vH"}){
    id
    name
    downloadUrl

businessDocumentVersion_DocumentVersions {
      id
      name
      documentVersion
      downloadUrl
    }
  }
}
```
	```
	{
	  "data": {
	    "businessDocument": [
	      {
	        "id": "Dbm4QHHbM5vH",
	        "name": "Audit Report",
	        "downloadUrl": "http://w-ogd/hopexgraphql/api/attachment/Dbm4QHHbM5vH/file",

	"businessDocumentVersion_DocumentVersions": [
	          {
	            "id": "sLD2611MNbrE",
	            "name": "Audit Report v2.docx",
	            "downloadUrl": "http://w-ogd/hopexgraphql/api/attachment/sLD2611MNbrE/file",
	            "documentVersion": "2"
	          },
	          {
	            "id": "Kcm4RHHbMTzH",
	            "name": "Audit Report v1.docx",
	            "downloadUrl": "http://w-ogd/hopexgraphql/api/attachment/Kcm4RHHbMTzH/file",
	            "documentVersion": "1"
	          }
	        ]
	```

	}
	]
	 }
	 }

Example with Postman:

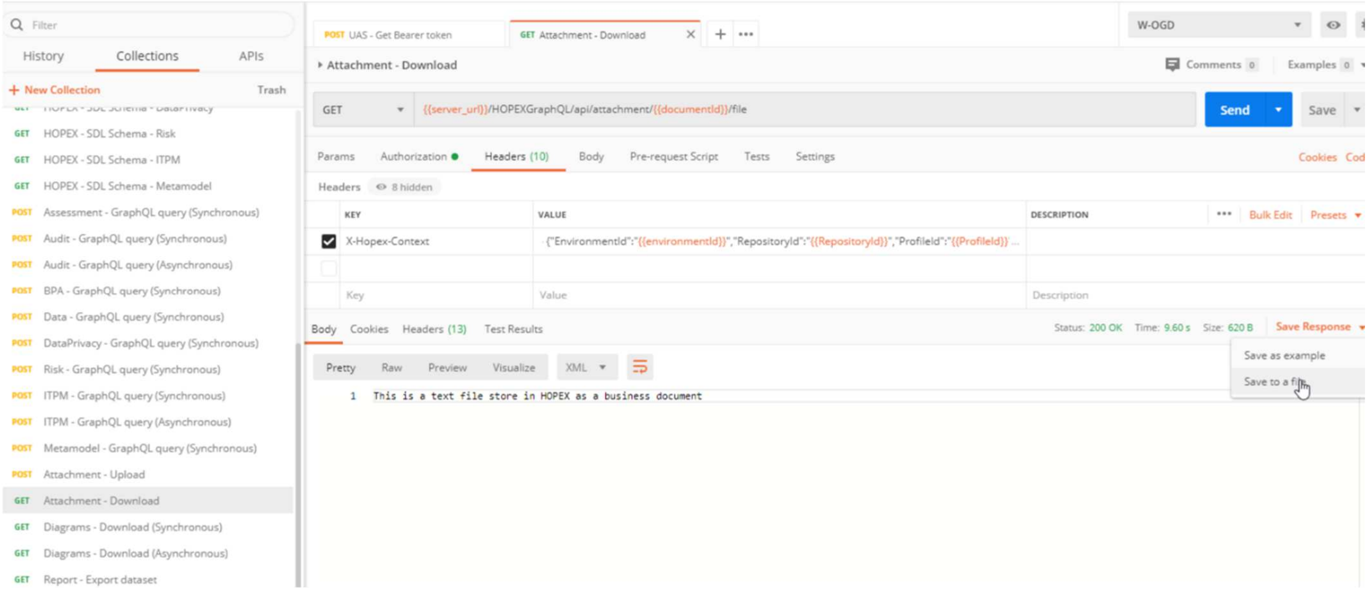

Filter
History
Collections
APIs
New Collection
Trash
HOPEX REST API
33 requests
POST UAS - Get Bearer token
POST UAS - Refresh Bearer token
POST UAS - Revoke Bearer token
POST UAS - Introspection Bearer token
POST UAS - Validation Bearer token
GET UAS - Discovery Endpoint
GET UAS - User Information
GET HOPEX - Get the list of all enviroments
GET HOPEX - Get the list of all repositories
GET HOPEX - MGL Export
GET HOPEX - SDL Schema - Assessment
GET HOPEX - SDL Schema - Audit
GET HOPEX - SDL Schema - BPA
GET HOPEX - SDL Schema - Data
GET HOPEX - SDL Schema - DataPrivacy
GET HOPEX - SDL Schema - Risk
GET HOPEX - SDL Schema - ITPM
POST UAS - Get Bearer token
GET Attachment - Download
W-OGD
UAS - Get Bearer token
Comments 0
Examples 3
Send
Save
Params
Authorization
Headers (10)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk E
Key
Value
Description
Body
Cookies
Headers (10)
Test Results (1/1)
Save Response
Pretty
Raw
Preview
Visualize
JSON

POST {{server_url}}/UAS/connect/token/

Status: 200 OK Time: 63 ms Size: 524 B

```
{
    "access_token": "2b33902b78bb5030990fa97cdc7e6693",
    "expires_in": 3600,
    "token_type": "Bearer",
    "refresh_token": "faae5762be291b38d215b5351801d95c"
}
```

Query


Filter
History
Collections
APIs
New Collection
Trash
GET HOPEX - SDL Schema - Risk
GET HOPEX - SDL Schema - ITPM
GET HOPEX - SDL Schema - Metamodel
POST Assessment - GraphQL query (Synchronous)
POST Audit - GraphQL query (Synchronous)
POST Audit - GraphQL query (Asynchronous)
POST BPA - GraphQL query (Synchronous)
POST Data - GraphQL query (Synchronous)
POST DataPrivacy - GraphQL query (Synchronous)
POST Risk - GraphQL query (Synchronous)
POST ITPM - GraphQL query (Synchronous)
POST ITPM - GraphQL query (Asynchronous)
POST Metamodel - GraphQL query (Synchronous)
POST Attachment - Upload
GET Attachment - Download
GET Diagrams - Download (Synchronous)
GET Diagrams - Download (Asynchronous)
GET Report - Export dataset
POST UAS - Get Bearer token
GET Attachment - Download
W-OGD
Attachment - Download
Comments 0
Examples 0
Send
Save
Params
Authorization
Headers (10)
Body
Pre-request Script
Tests
Settings
Cookies
Headers
8 hidden
Bulk Edit
Presets
Body
Cookies
Headers (13)
Test Results
Save Response
Save as example
Save to a file
Pretty
Raw
Preview
Visualize
XML

GET {{server_url}}/HOPEXGraphQL/api/attachment/{{documentId}}/file

	KEY	VALUE	DESCRIPTION
✓	X-Hopex-Context	{"EnvironmentId":"{{environmentId}}","RepositoryId":"{{RepositoryId}}","ProfileId":"{{ProfileId}}" ...	
	Key	Value	Description

Status: 200 OK Time: 9.60 s Size: 620 B

```
This is a text file store in HOPEX as a business document
```

Result

13.3. Uploading an attachment (Business Document)

When you upload an attachment you need to know if the business document already exists or not in HOPEX. Below you will find the 2 cases.

Case 1: Upload for a new business document

You must:

1. Creat the business document object via GraphQL
2. Upload the binary content in this newly create via the dedicated endpoint

Query	*Result*

`mutation newBusinessDocument {`
`  createBusinessDocument(`
`    businessDocument:{`
`      name:"My new Document"`
`    }`
`  ) {`
`    id`
`    uploadUrl`
`  }`
`}`
	`{`
	`  "data": {`
	`    "createBusinessDocument": {`
	`      "id": "Sjot25kbUvI3",`
	`      "uploadUrl": "http://w-ogd/hopexgraphql/api/attachment/Sjot25kbUvI3/file"`
	`    }`
	`  }`
	`}`

You now have the ID and upload URL for your document. The upload URL are like this:

{{server_url}}/HOPEXGraphQL/api/attachment/**{{documentId}}**/file

where **{{documentId}}** represent the absolute identifier of the object in HOPEX. They must be called with **POST** verb.

Then follow the instruction described in final step below.

Case 2: Update of an existing business document

You must:

1. Get the business document upload URL.
2. Upload the binary content and decided to update or create a new version of the binary content.

Query	*Result*

`query {`
`businessDocument(filter:{id:"Dbm4QHHbM5vH"}){`
`    id`
	`{`
	`  "data": {`
	`    "businessDocument": [`
	`      {`

```
    name
    uploadUrl
  }
}
```

```
        "id": "Dbm4QHHbM5vH",
        "name": "Audit Report",
        "uploadUrl": "http://w-ogd/hopexgraphql/api/attachment/Dbm4QHHbM5vH/file"
      }
    ]
  }
}
```

Final step:

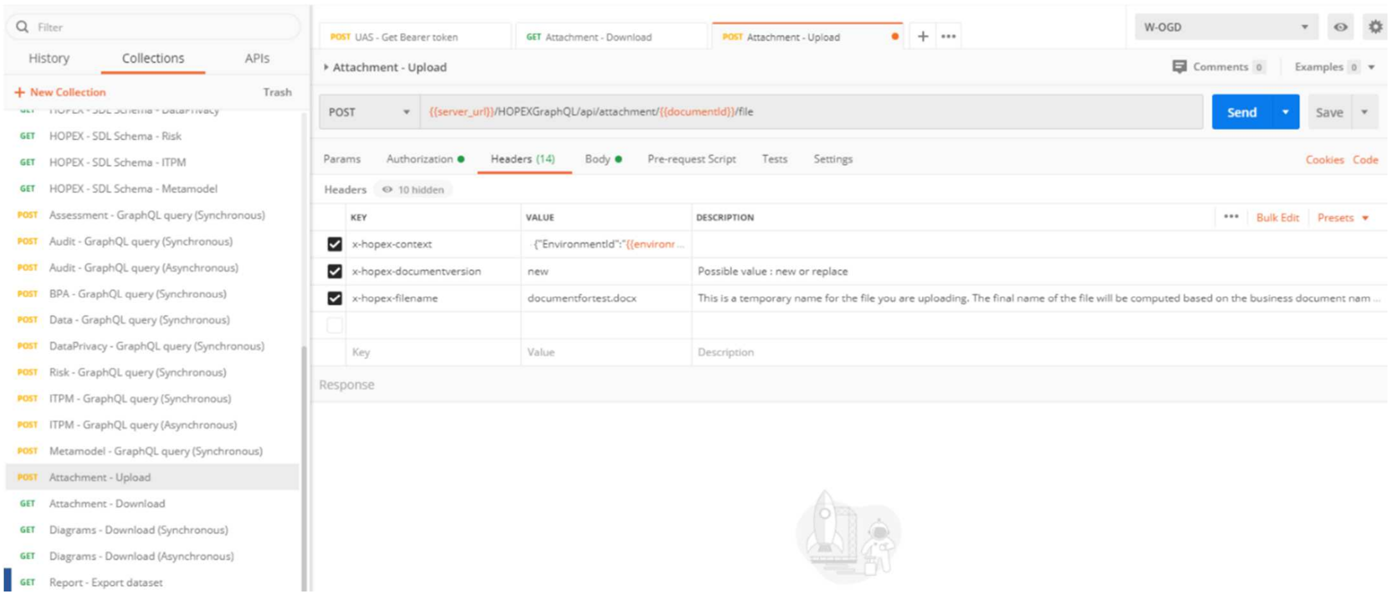
When you call the upload URL the **header** should contains:

- The document version: x-hopex-documentversion with possible value **new** or **replace**
  - **New:** will upload a new binary content and thus create a new version of business document version
  - **Update:** will upload a binary content and replace the current existing binary content for the latest business document version
- The document file name: x-hopex-filename that contain the name of the file with the extension
  - Only the extension of the file is important as it will be used to defined the file content type

When you call the upload URL the **body** should contains:

- the binary content of the attachment

Example with Postman:



POST UAS - Get Bearer token | GET Attachment - Download | POST Attachment - Upload

Attachment - Upload

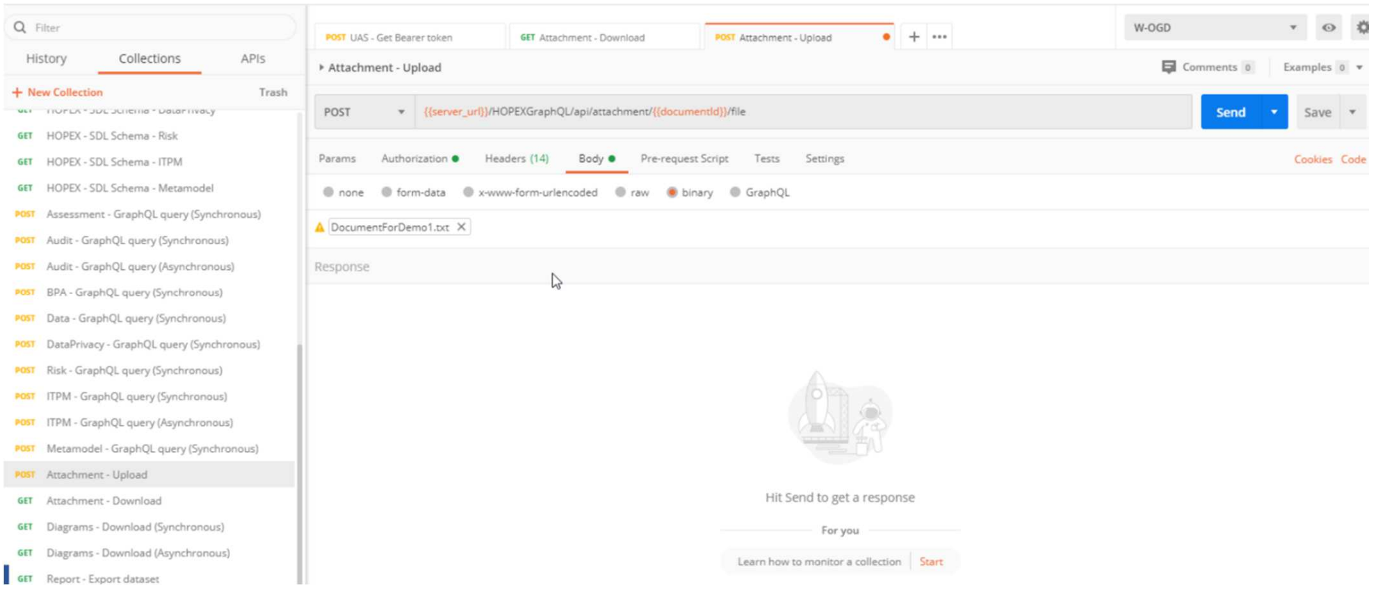
POST {{server_url}}/HOPEXGraphQL/api/attachment/{{documentId}}/file

Headers (14) — 10 hidden

	KEY	VALUE	DESCRIPTION
✓	x-hopex-context	{"EnvironmentId":"{{environr ...	
✓	x-hopex-documentversion	new	Possible value : new or replace
✓	x-hopex-filename	documentfortest.docx	This is a temporary name for the file you are uploading. The final name of the file will be computed based on the business document nam ...

GET HOPEX - SDL Schema - Risk
GET HOPEX - SDL Schema - ITPM
GET HOPEX - SDL Schema - Metamodel
POST Assessment - GraphQL query (Synchronous)
POST Audit - GraphQL query (Synchronous)
POST Audit - GraphQL query (Asynchronous)
POST BPA - GraphQL query (Synchronous)
POST Data - GraphQL query (Synchronous)
POST DataPrivacy - GraphQL query (Synchronous)
POST Risk - GraphQL query (Synchronous)
POST ITPM - GraphQL query (Synchronous)
POST ITPM - GraphQL query (Asynchronous)
POST Metamodel - GraphQL query (Synchronous)
POST Attachment - Upload
GET Attachment - Download
GET Diagrams - Download (Synchronous)
GET Diagrams - Download (Asynchronous)
GET Report - Export dataset

Query


History | Collections | APIs
+ New Collection | Trash

GET HOPEX - SDL Schema - DataPrivacy
GET HOPEX - SDL Schema - Risk
GET HOPEX - SDL Schema - ITPM
GET HOPEX - SDL Schema - Metamodel
POST Assessment - GraphQL query (Synchronous)
POST Audit - GraphQL query (Synchronous)
POST Audit - GraphQL query (Asynchronous)
POST BPA - GraphQL query (Synchronous)
POST Data - GraphQL query (Synchronous)
POST DataPrivacy - GraphQL query (Synchronous)
POST Risk - GraphQL query (Synchronous)
POST ITPM - GraphQL query (Synchronous)
POST ITPM - GraphQL query (Asynchronous)
POST Metamodel - GraphQL query (Synchronous)
POST Attachment - Upload
GET Attachment - Download
GET Diagrams - Download (Synchronous)
GET Diagrams - Download (Asynchronous)
GET Report - Export dataset

POST UAS - Get Bearer token | GET Attachment - Download | POST Attachment - Upload
W-OGD

Attachment - Upload

POST {{server_url}}/HOPEXGraphQL/api/attachment/{{documentId}}/file

Params | Authorization | Headers (14) | Body | Pre-request Script | Tests | Settings

none | form-data | x-www-form-urlencoded | raw | [✓ binary] | GraphQL

DocumentForDemo1.txt

Response

Hit Send to get a response

Result

14. Getting GraphQL Schema as a file (SDL schema file)

When building graphQL query it can be convenient to use auto-completion to navigate the available information in the schema. Thus, useful to write the graphQL to read/write in the REST API.

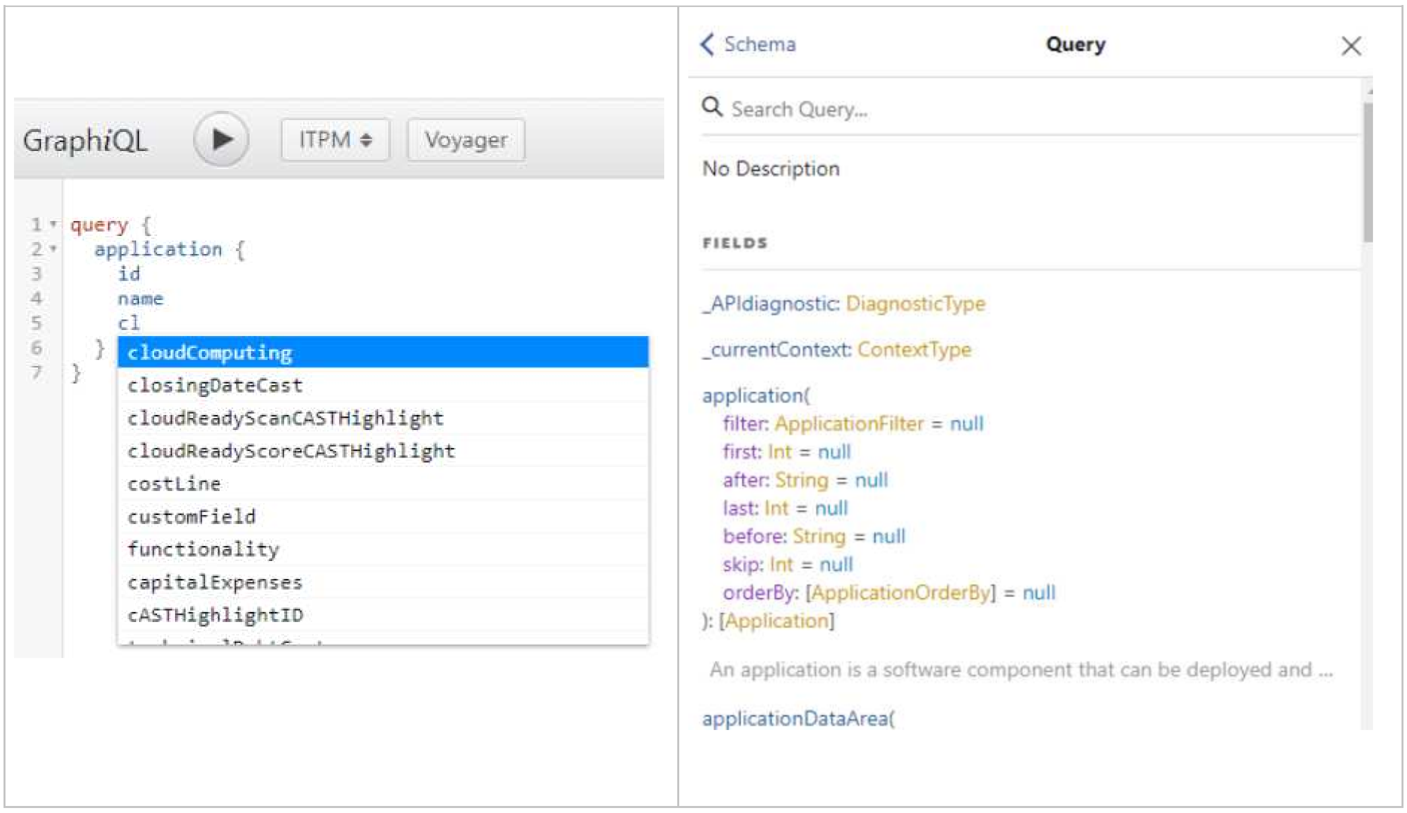
For example, for ITPM to know which MetaClass, MetaAttributes or MetaAssociation are available.

See video: https://youtu.be/gSOTxi3Hh2w.

14.1. In GraphiQL

If you use a tool like GraphiQL the auto completion is native and will be displayed while you are writing. You can also navigate the embedded documentation on the schema to get additional information on the fields available.

For example: for an enumeration field you can access the value of the enum.



```
query {
  application {
    id
    name
    cl
  }
}
```

```
cloudComputing
closingDateCast
cloudReadyScanCASTHighlight
cloudReadyScoreCASTHighlight
costLine
customField
functionality
capitalExpenses
cASTHighlightID
```

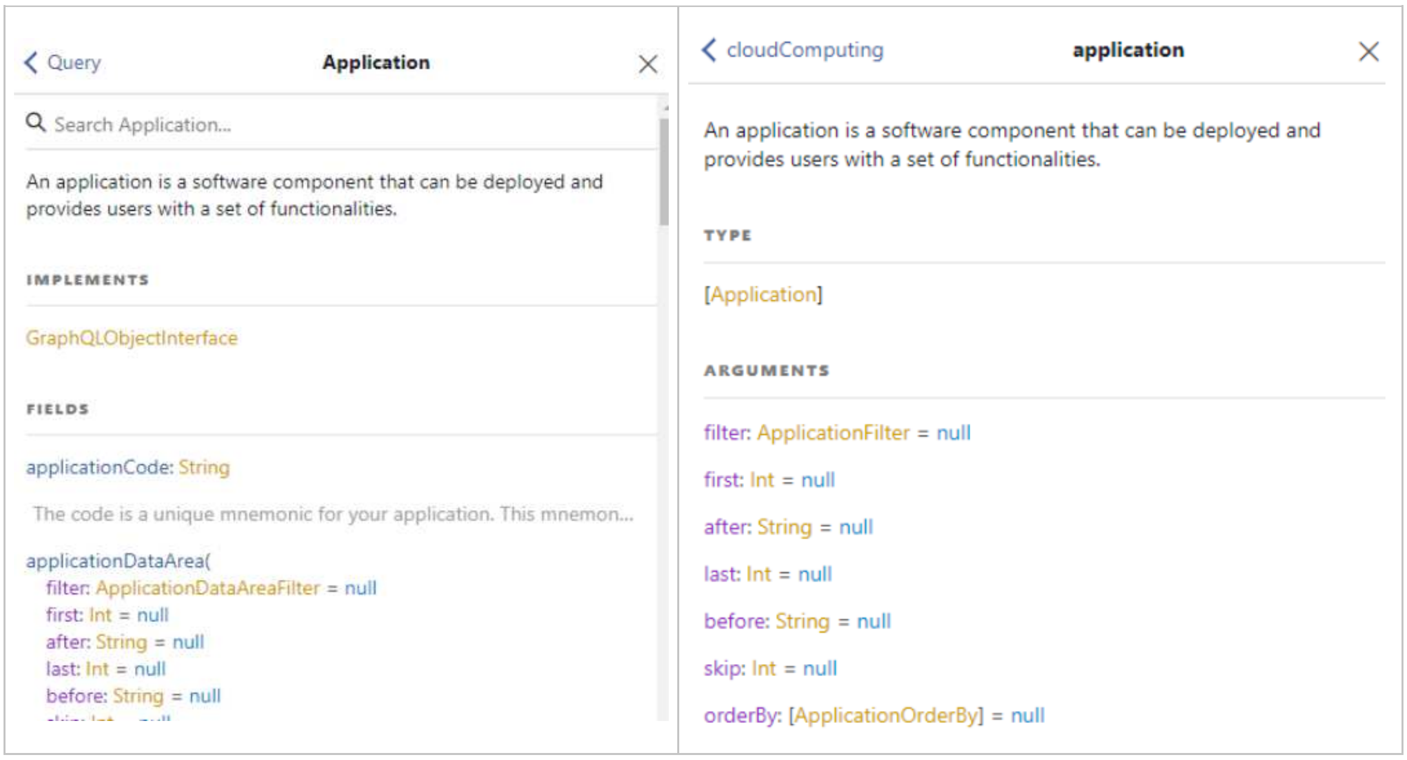
No Description

FIELDS

```
_APIdiagnostic: DiagnosticType
_currentContext: ContextType
application(
  filter: ApplicationFilter = null
  first: Int = null
  after: String = null
  last: Int = null
  before: String = null
  skip: Int = null
  orderBy: [ApplicationOrderBy] = null
): [Application]
```

An application is a software component that can be deployed and ...

applicationDataArea(



Query — Application

Search Application...

An application is a software component that can be deployed and provides users with a set of functionalities.

IMPLEMENTS

GraphQLObjectInterface

FIELDS

applicationCode: String

The code is a unique mnemonic for your application. This mnemon...

```
applicationDataArea(
  filter: ApplicationDataAreaFilter = null
  first: Int = null
  after: String = null
  last: Int = null
  before: String = null
```

cloudComputing — application

An application is a software component that can be deployed and provides users with a set of functionalities.

TYPE

[Application]

ARGUMENTS

```
filter: ApplicationFilter = null
first: Int = null
after: String = null
last: Int = null
before: String = null
skip: Int = null
orderBy: [ApplicationOrderBy] = null
```

In this tool use:

- **On Mouse over** to get high level description of the field
- **Ctrl+Space** to access the possible option.
- **Ctrl+Click** on an item to access the documentation tab in graphiQL

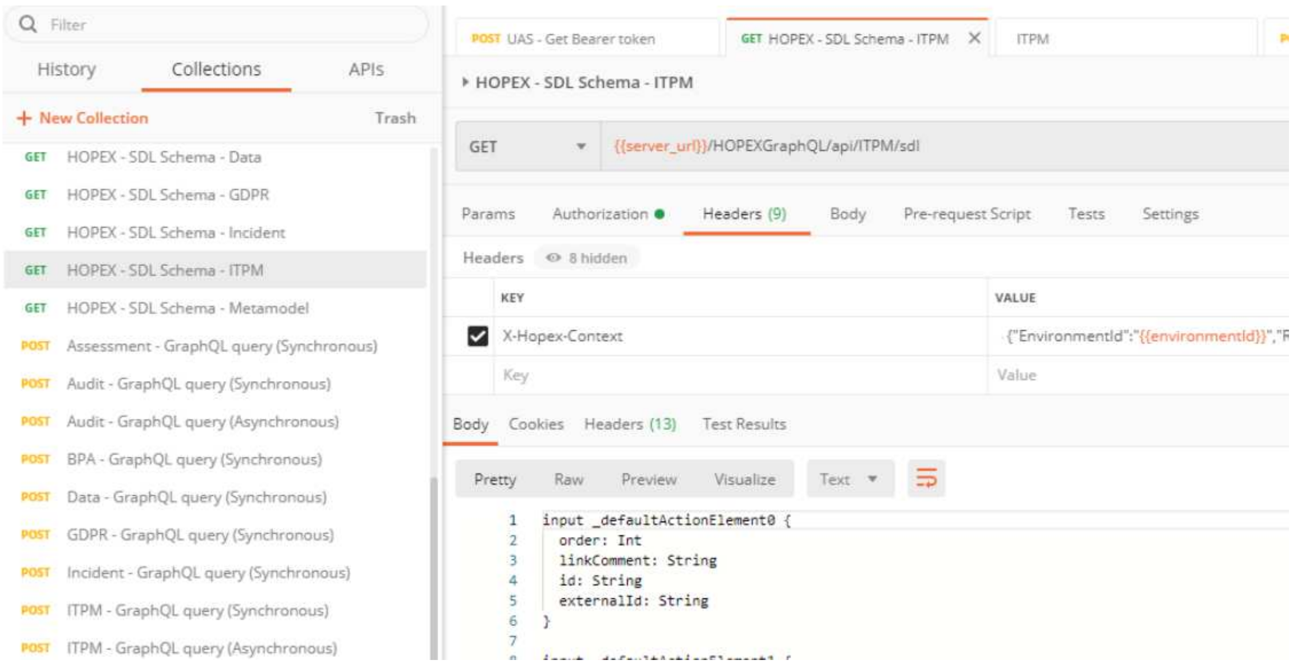
While GraphiQL meets this use case it is not often used by developer who will prefer tools like Postman.

14.2. In Postman

The GraphQL query language has its own schema definition based on a format called "SDL". This SDL format file can be downloaded via the API itself. Once retrieved it can be used in tools that can parse this standard.

For instance, Postman is a developer tool that can handle GraphQL and SDL schema. To access the SDL file follow this steps:

- Get a bearer token
- Call of of the endpoint of the required schema {{server_url}}/HOPEXGraphQL/api/ITPM/**sdl**. The response will be a filenamed "schema".graphql.
- Import this file in Postman
- In the body parameters select GraphQL and in the dropdown your schema



POST UAS - Get Bearer token | GET HOPEX - SDL Schema - ITPM | ITPM

HOPEX - SDL Schema - ITPM

GET {{server_url}}/HOPEXGraphQL/api/ITPM/sdl

Params | Authorization | Headers (9) | Body | Pre-request Script | Tests | Settings

Headers 8 hidden

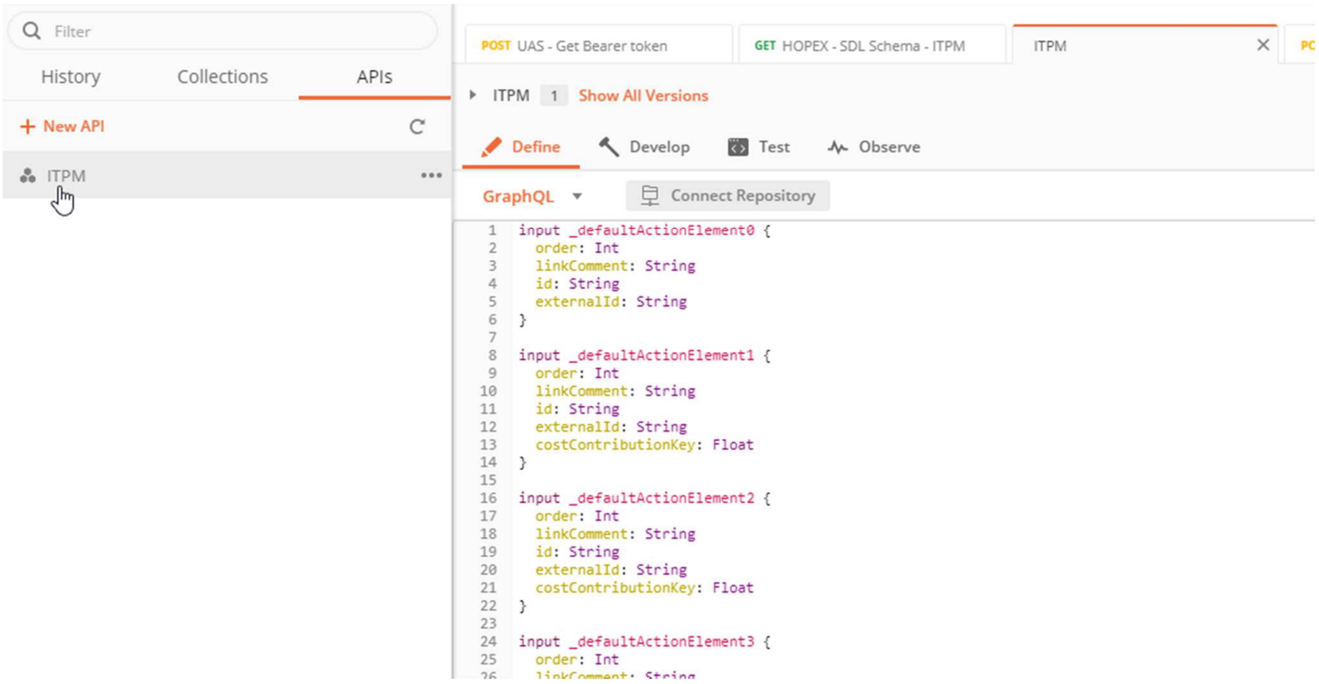
KEY	VALUE
X-Hopex-Context	{"EnvironmentId":"{{environmentId}}","R
Key	Value

Body | Cookies | Headers (13) | Test Results

Pretty | Raw | Preview | Visualize | Text

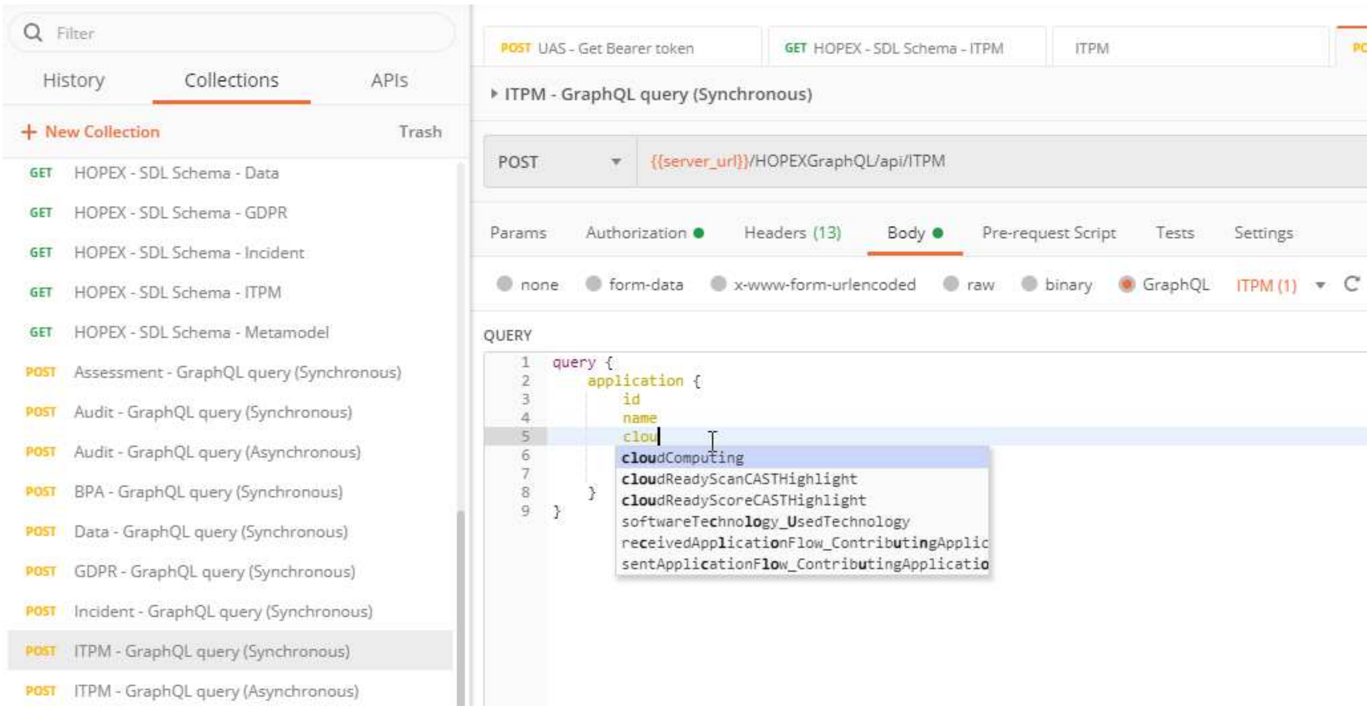
```
input _defaultActionElement0 {
  order: Int
  linkComment: String
  id: String
  externalId: String
}

```

Filter
History
Collections
APIs
New Collection
Trash
GET HOPEX - SDL Schema - Data
GET HOPEX - SDL Schema - GDPR
GET HOPEX - SDL Schema - Incident
GET HOPEX - SDL Schema - ITPM
GET HOPEX - SDL Schema - Metamodel
POST Assessment - GraphQL query (Synchronous)
POST Audit - GraphQL query (Synchronous)
POST Audit - GraphQL query (Asynchronous)
POST BPA - GraphQL query (Synchronous)
POST Data - GraphQL query (Synchronous)
POST GDPR - GraphQL query (Synchronous)
POST Incident - GraphQL query (Synchronous)
POST ITPM - GraphQL query (Synchronous)
POST ITPM - GraphQL query (Asynchronous)



Filter
History
Collections
APIs
New API
ITPM
POST UAS - Get Bearer token
GET HOPEX - SDL Schema - ITPM
ITPM
ITPM 1 Show All Versions
Define
Develop
Test
Observe
GraphQL
Connect Repository

```
input _defaultActionElement0 {
  order: Int
  linkComment: String
  id: String
  externalId: String
}

input _defaultActionElement1 {
  order: Int
  linkComment: String
  id: String
  externalId: String
  costContributionKey: Float
}

input _defaultActionElement2 {
  order: Int
  linkComment: String
  id: String
  externalId: String
  costContributionKey: Float
}

input _defaultActionElement3 {
  order: Int
```


Filter
History
Collections
APIs
+ New Collection
Trash
GET HOPEX - SDL Schema - Data
GET HOPEX - SDL Schema - GDPR
GET HOPEX - SDL Schema - Incident
GET HOPEX - SDL Schema - ITPM
GET HOPEX - SDL Schema - Metamodel
POST Assessment - GraphQL query (Synchronous)
POST Audit - GraphQL query (Synchronous)
POST Audit - GraphQL query (Asynchronous)
POST BPA - GraphQL query (Synchronous)
POST Data - GraphQL query (Synchronous)
POST GDPR - GraphQL query (Synchronous)
POST Incident - GraphQL query (Synchronous)
POST ITPM - GraphQL query (Synchronous)
POST ITPM - GraphQL query (Asynchronous)
POST UAS - Get Bearer token
GET HOPEX - SDL Schema - ITPM
ITPM
ITPM - GraphQL query (Synchronous)
Params
Authorization
Headers (13)
Body
Pre-request Script
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
ITPM (1)

POST {{server_url}}/HOPEXGraphQL/api/ITPM

QUERY

```
query {
    application {
        id
        name
        clou
    }
}
```

```
cloudComputing
cloudReadyScanCASTHighlight
cloudReadyScoreCASTHighlight
softwareTechnology_UsedTechnology
receivedApplicationFlow_ContributingApplic
sentApplicationFlow_ContributingApplicatio
```

Sample for graphQL available here: https://community.mega.com/mega/attachments/mega/technical-product/7814/2/SampleSDL.zip.

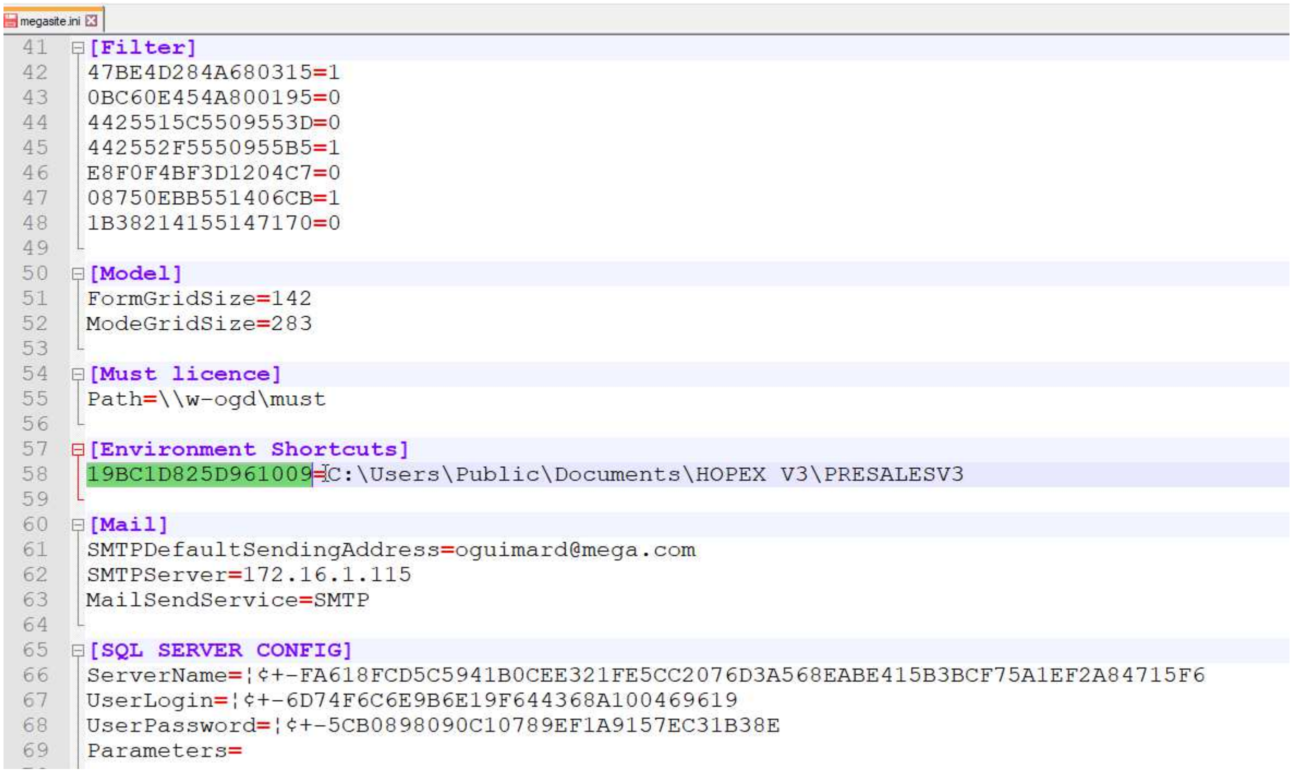
15. ID Converter From HexaIdAbs to Absolute identifier

When calling HOPEX **REST API** you need to provide the absolute identifiers of the environment and the repository. You will find below some explaination on how and where to find this information.

On the **server side** you can access the:

- HexaIdAbs of the environment in the **megasite.ini** located in the installation folder. Default path is **<HOPEX installation>\Cfg**

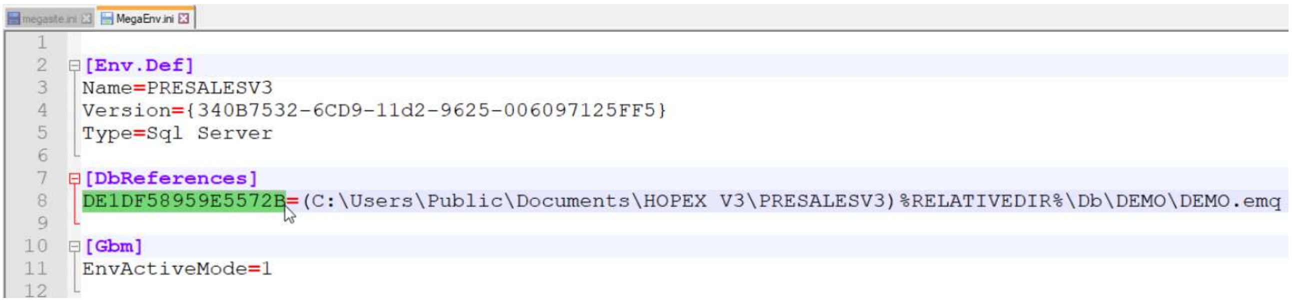
  E.g.: C:\ProgramData\MEGA\Hopex Application Server\5000\.shadowFiles\hopex.core\17.0.0+6637\Cfg



```
megasite.ini
[Filter]
47BE4D284A680315=1
0BC60E454A800195=0
4425515C5509553D=0
442552F5550955B5=1
E8F0F4BF3D1204C7=0
08750EBB551406CB=1
1B38214155147170=0

[Model]
FormGridSize=142
ModeGridSize=283

[Must licence]
Path=\\w-ogd\must

[Environment Shortcuts]
19BC1D825D961009=C:\Users\Public\Documents\HOPEX V3\PRESALESV3

[Mail]
SMTPDefaultSendingAddress=oguimard@mega.com
SMTPServer=172.16.1.115
MailSendService=SMTP

[SQL SERVER CONFIG]
ServerName=¦¢+-FA618FCD5C5941B0CEE321FE5CC2076D3A568EABE415B3BCF75A1EF2A84715F6
UserLogin=¦¢+-6D74F6C6E9B6E19F644368A100469619
UserPassword=¦¢+-5CB0898090C10789EF1A9157EC31B38E
Parameters=
```

- HexaIdAbs of the repository in the **megaenv.ini** located in the environment folder.
  Default path is C:\...\HOPEX Application Server\<HAS instance name>\Repos\<Environment name>

  E.g.: C:\ProgramData\MEGA\Hopex Application Server\5000\Repos\EnvTestsLab

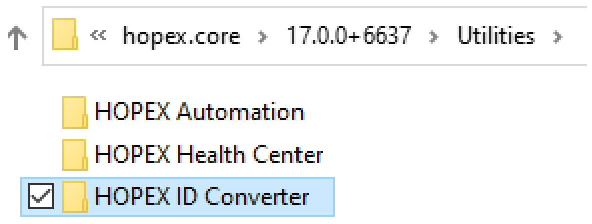


```
megasite.ini   MegaEnv.ini

[Env.Def]
Name=PRESALESV3
Version={340B7532-6CD9-11d2-9625-006097125FF5}
Type=Sql Server

[DbReferences]
DE1DF58959E5572B=(C:\Users\Public\Documents\HOPEX V3\PRESALESV3)%RELATIVEDIR%\Db\DEMO\DEMO.emq

[Gbm]
EnvActiveMode=1
```

You can find the ID converter in the **<HOPEX installation>\Utilities\HOPEX ID Converter** folder.

For example:

C:\ProgramData\MEGA\Hopex Application Server\5000\.shadowFiles\hopex.core\17.0.0+6637\Utilities\HOPEX ID Converter



Perfom the following steps:

1. Launch the **Hopex ID Converter** executable.
2. In the "INI" file copy the HexaIdAbs you want to convert (from the megasite or megaenv).
3. Paste the value in the hexa idabs (16 char).
4. Select the below field cn64 idabs (12 char).
5. Right-click **Copy** and **Paste** it wherever you need to use it

More on yuoutube: https://youtu.be/yujkirLXawY

16. Using Postman to call the REST API

Calling the REST API can be done with a wide range of tools.

In this post we explain how to leverage postman to make HTTP request to the API.

The main steps to use postman with the HOPEX REST API are:

1. Download postman
2. Download postman collection that contains all the endpoints
3. Import the collection in postman (replace or copy when prompted)
4. Download the environments variables (file is located on this post Sample.postman_environment.zip) to be resolve for the endpoints: URL, login, password...
5. Import the environments variables (replace or copy if prompted)
6. Fill(in the variables with the value applicable for your installation
7. Execute the fist HTTP request to get an UAS bearer token
8. Call the SDL schema and save the file to add in postman API
9. Use any of the endpoints to build you graphQL queries

More here https://youtu.be/Db93On4nH7I