Publication for Capella User Manual

Table of Contents

Getting started

Get Publication for Capella

Refers to the installation guide.

Requirements

Details the requirements needed to run the product according to the scaling objectives.

	Client	Server	Comment
Memory Usage	-Xmx4g	n/a	
CPU	2 Ghz 64-bit (x64) processor	n/a	
Networking	n/a	n/a	
Scalability	n/a	n/a	

Using Publication for Capella Web Client

Glossary

Model

A model is a set of objects with properties and links between each other. These properties and links define a semantic that can be understood by knowledgeable users or software. In general, these properties and links conform to a set of rules defined by a meta-model.

Account

A user account that makes it possible for a given person or software to connect to the *Publication Server*.

Team

A team groups a number of users to give them specific access rights to a specific project.

Project

A project groups any number of models stored by the *Publication Server*. It is the unit of exposure of data to the OSLC API of the *Publication Server*. Access Control can only be defined on a per-project basis.

OSLC

Open Services for Lifecycle Collaboration. OSLC is a standard for interoperability between repositories. For more details, refer to the official OSLC page at https://open-services.net/ .

Login

Any access to a page will cause the login page to be displayed if the user is not already logged in.

Logout

Users can log out of the server by clicking on the icon on the top right of any page header and clicking on logout.

Web Client Welcome Page

The server will by default display the Project List page.

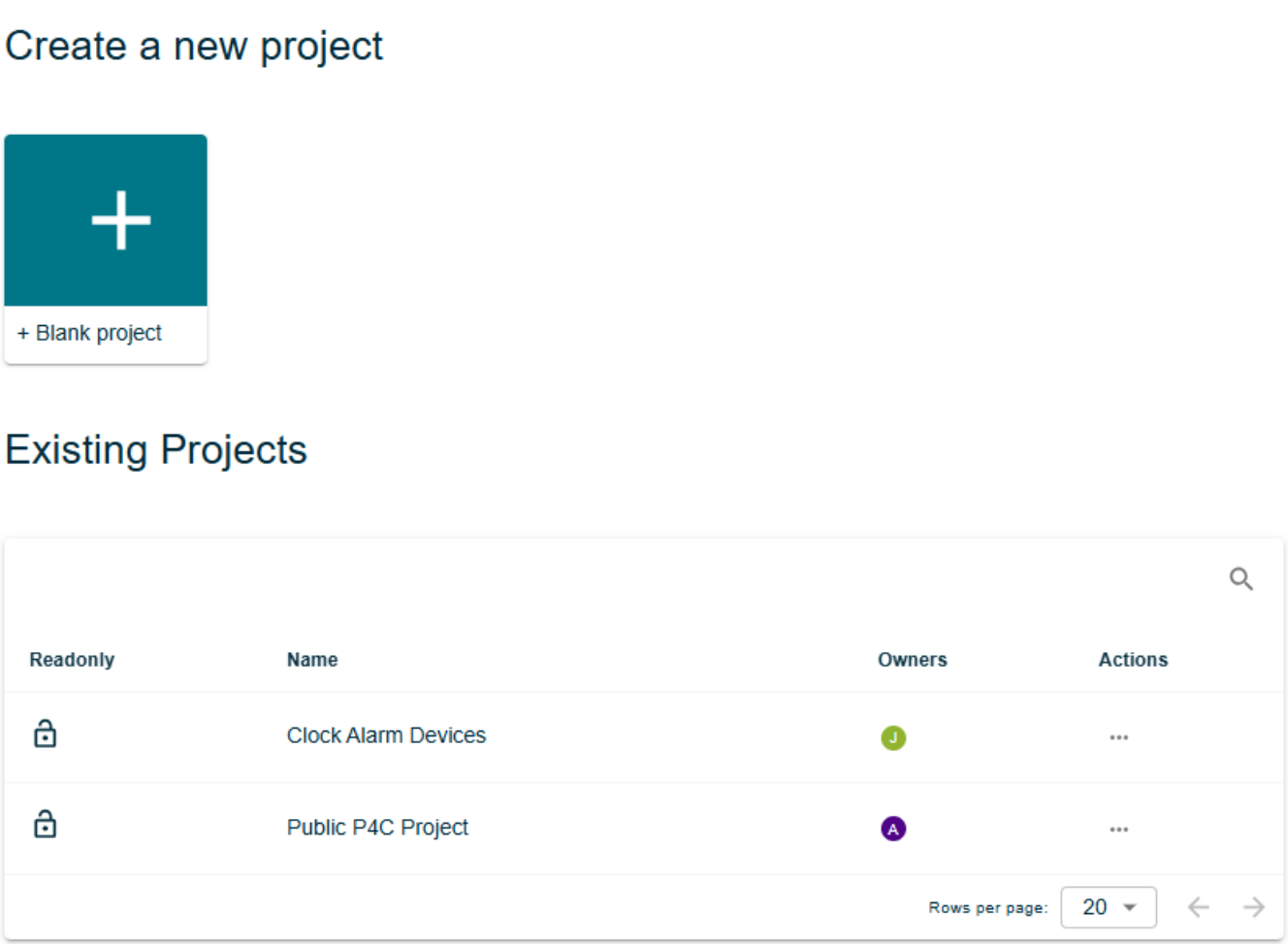


Figure 1. Welocme Page after login

Project List

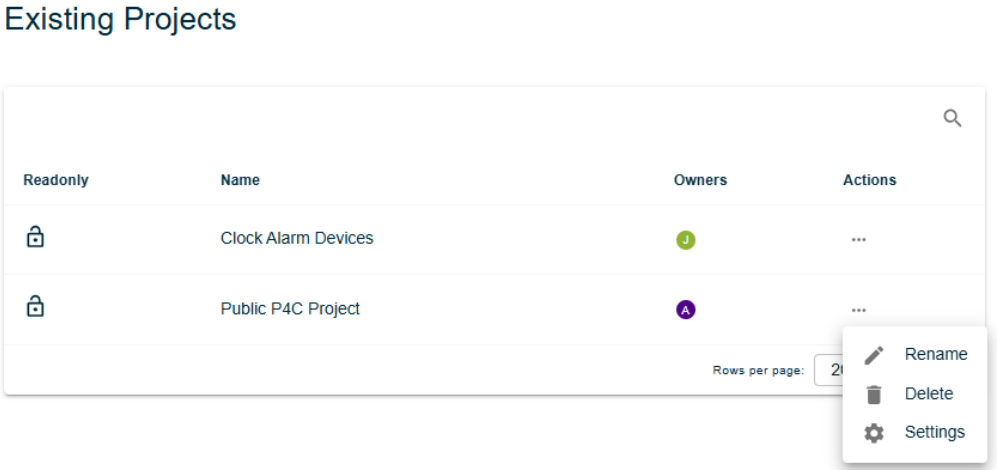


Figure 2. Project List Contextual Menu

Click on the project name to navigate to the project's page described in section Project Page.

Click on the '+' button to create a new project. See Project Creation for more details. Any user can create new projects.

Click on the three dots to the right side of a project to display a pop-up menu with the following actions:

- Rename - see Project Renaming
- Delete - see Project Deletion

Project Creation

To create a new project, click on the '+' button in the Web Client Welcome Page. This displays the following form:

Figure 3. Project Creation Page

The name of a project must contain less than 1024 characters. If the name is not valid, a warning message is displayed and the 'Create Project' button is disabled.

Project Renaming

To rename a project, open the project contextual menu in the Web Client Welcome Page and select 'Rename'.

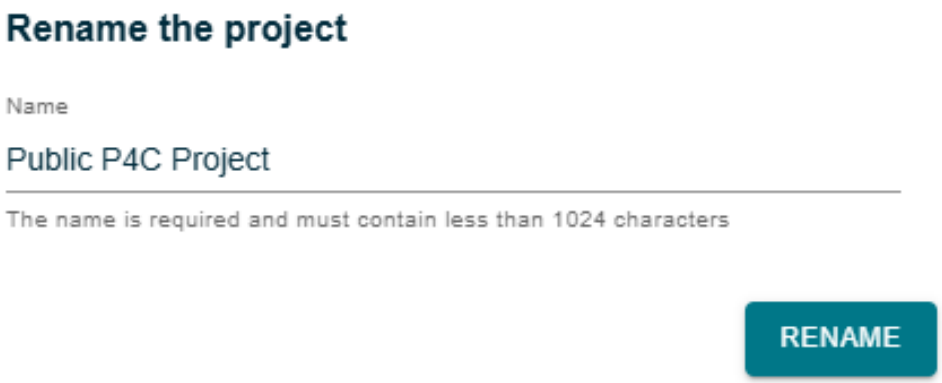


Figure 4. Rename Project Dialog

Change the name and click on 'Rename' to rename the projects, or click on the background (or press 'Escape') to cancel the renaming.

Project Deletion

To delete a project, open the project contextual menu in the Web Client Welcome Page and select 'Delete'.

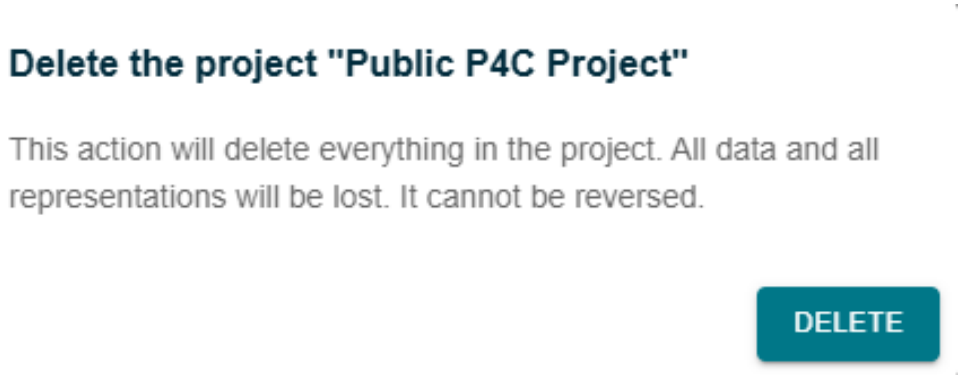


Figure 5. Delete Project Dialog

Click on 'Delete' to confirm the deletion, or click on the background (or press 'Escape') to cancel the deletion.

Project Page

The Project Page is accessible from the Project List once a user is logged in, by clicking on the name of a project in the list.

The Project Page is the entry point to browse the models it contains.

It is made of:

- A header that contains:
  - The Obeo icon, which can be clicked on to navigate to the Web Client Welcome Page;
  - The project name;
  - A hamburger menu for this project, with the following entries:
    - Rename;
    - Settings - see Project Settings Page;
    - Delete.
  - The logged-in user name and menu.
- A main panel, split in three columns:
  - The left column contains the explorer, that displays the models exposed by this project, and their structure;
  - The middle panel displays information on the selected element;
  - The right panel displays the details (properties) of the selected element.

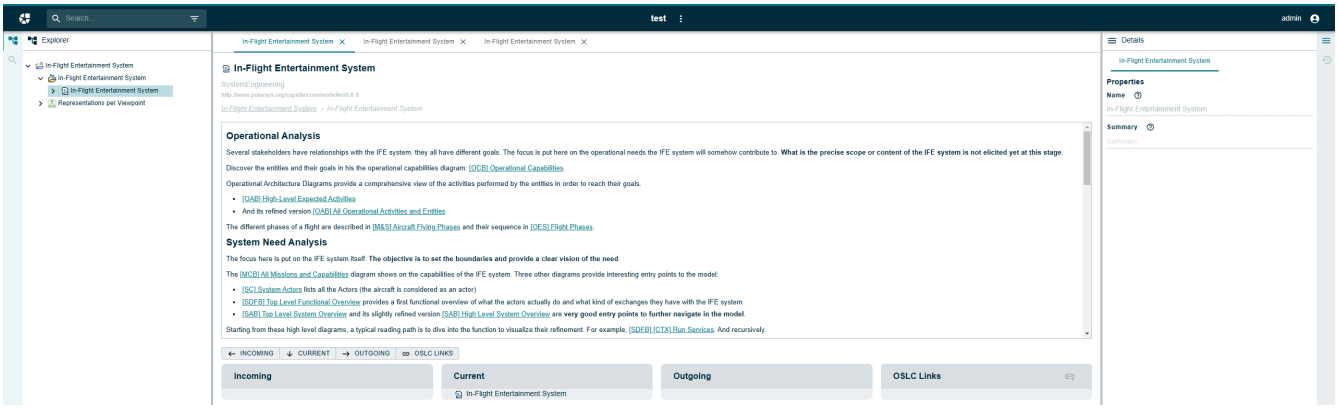


Figure 6. Project page

Model Explorer

The model explorer, in the left panel, can be used to navigate inside the structure of the model(s) available in the current project.

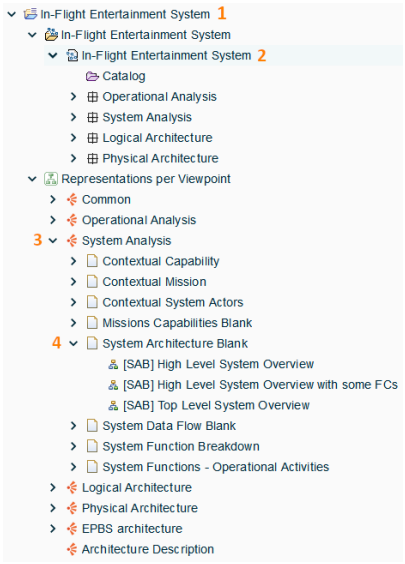


Figure 7. Explorer structure

The top-level elements in the explorer correspond to the models themselves (1). Inside of it the user can find the actual model elements (2), which correspond to the content of the `.capella` model, and all the exported diagrams (3) organized by category (4).

Clicking on any element in the explorer will select it and display its corresponding information in the main panel, and details in the right panel. The information displayed in the main panel depends on the type of element selected. See Main Panel - Model Navigation for more details.

When an element is selected, the browser's URL can be bookmarked to reopen this specific object. This URL can also be provided to other users so that they can access the details of this particular page, provided they have the appropriate permissions.

Search

Model Objects within a project can be searched in two ways:

- From the Quick Search, available in the project header;
- From the Filtered Search View, available in a dedicated panel of the project workbench.

Quick Search

The quick Search feature makes it possible to search for a given String in Model Object names, in a case insensitive way, within the current project. It also looks for OSLC links whose target title or URL matches.

This pattern is searched 'as is', there is no support yet for *wildcard* characters such as `'*'` or `'_'`. Any value containing the exact search terms will match.

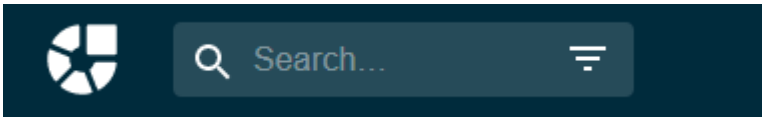


Figure 8. Quick Search

The search is launched each time a character is typed in its field and only displays the first 10 results.

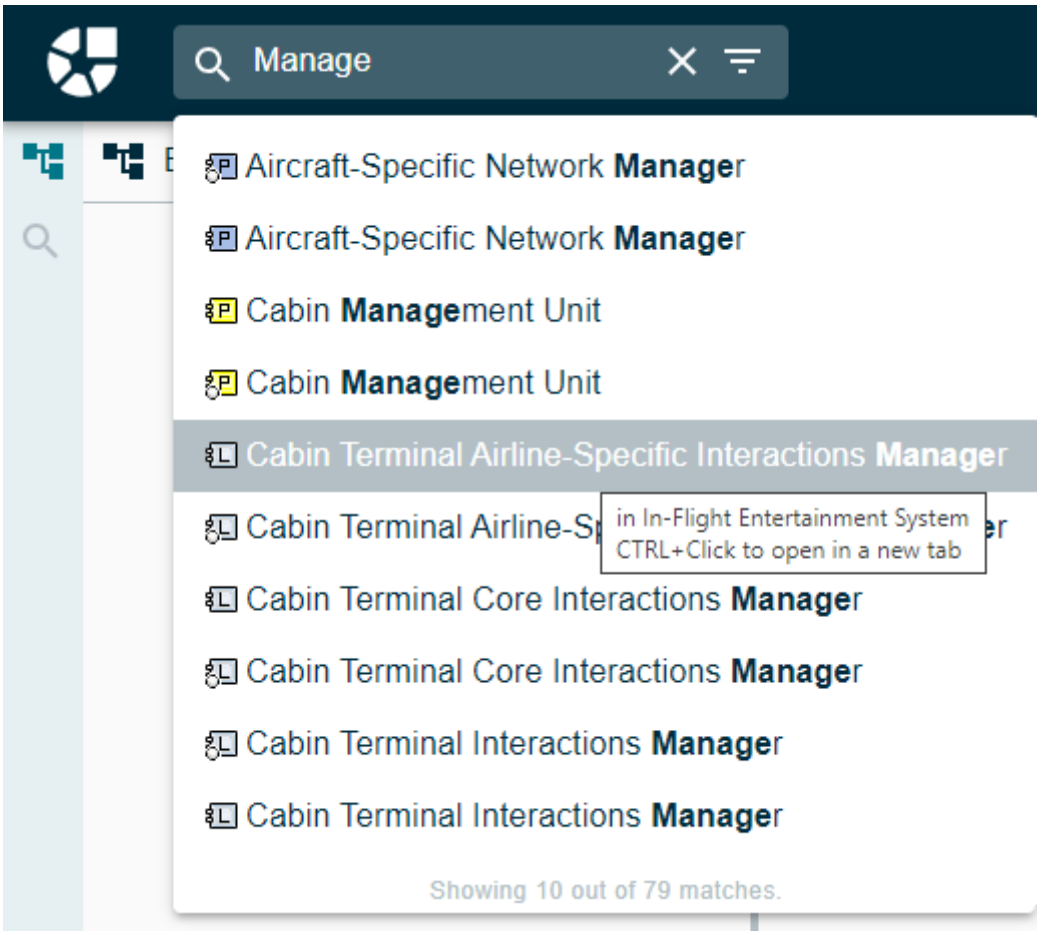


Figure 9. Quick Search Results

Clicking on a result will set the current selection to the object. The object page can be opened in a new tab using `Ctrl+Click`.

Results can also be browsed using the keyboard up/down arrows and selected with `Enter`.

`Ctrl+Enter` will open the object page in a new tab.

Filtered Search View

If more results are needed or there is a need to use more criteria for the search than the name only, the filtered search view provides a more complete search feature. It can be accessed directly from the search bar by clicking on this icon:

Figure 10. Open search view from search bar

Or it can be opened/closed the same way the explorer is by clicking on this icon:

Figure 11. Open/Close search view

The filtered search view can be used through the following form:

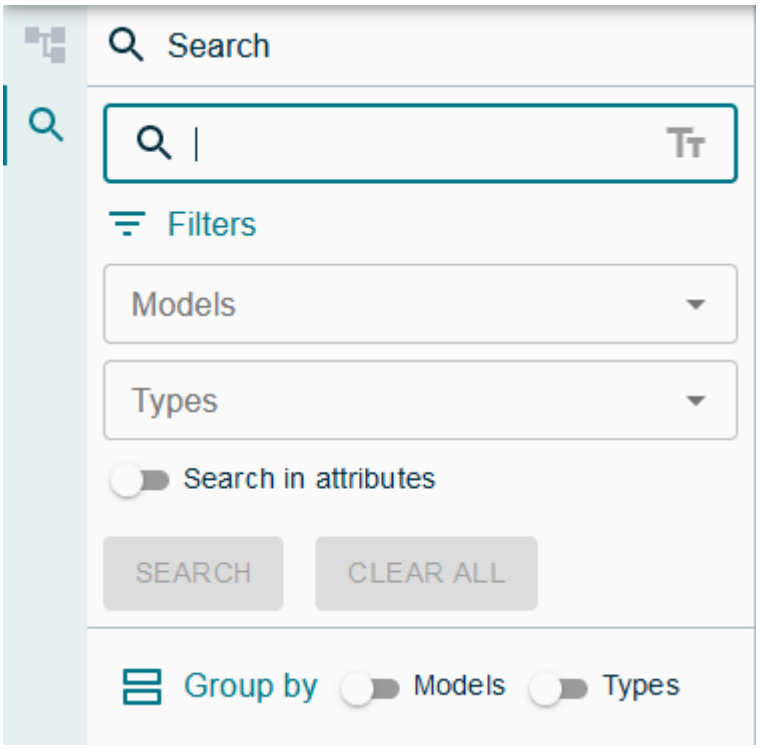


Figure 12. Filtered search view

The form accepts the following search parameters:

- A text field which accepts a pattern to find in model objects names or attributes. This pattern is also looked for in the *title*, *URL*, or *identifier* of OSLC link artifacts. It makes it easy to find model elements linked to a particular artifact when the artifact's title or ID is known.

This pattern is searched 'as is', there is no support yet for *wildcard* characters such as `'*'` or `'_'`. Any value containing the exact search terms will match.

- A model filter, which limits the search to the specified models. If left blank, all models in the

projects are considered by the search.

- A type filter, which limits the search to the specified types. If left blank, any type will be considered by the search.
- A 'search in attributes' switch, which activates the search in all model objects attribute values, instead of only in their names. If activated, the search will also look in OSLC identifiers. This option can have a significant impact on the duration of the search.

Unlike the Quick Search, a filtered search is launched when the 'Search' button is clicked or when 'Enter' is pressed. This button will be enabled only if either a pattern or a type is set.

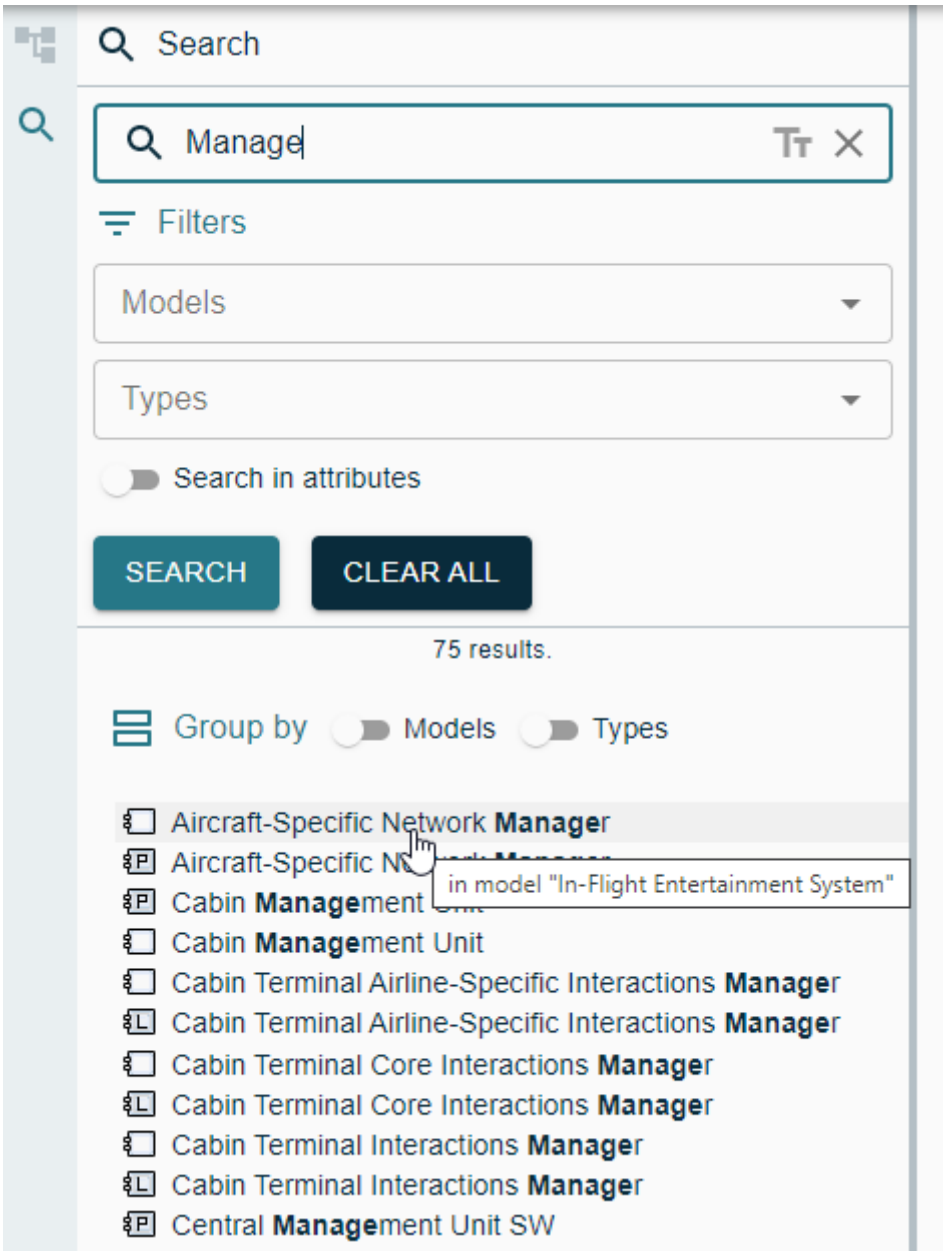


Figure 13. Filtered search view results

Each model object in the result list can be clicked, which will open the model object page as a new tab within the current page.

To ease the reading of the result of a search, it is possible to group the results by model and/or by type. It will turn the list of results into a tree grouping the results by their respective model and/or type (depending on the selected grouping options). A badge above the group name shows how

many results are under the group.

To help with the selection of the right element among the search results, it is possible to hover any result to display its Preview.

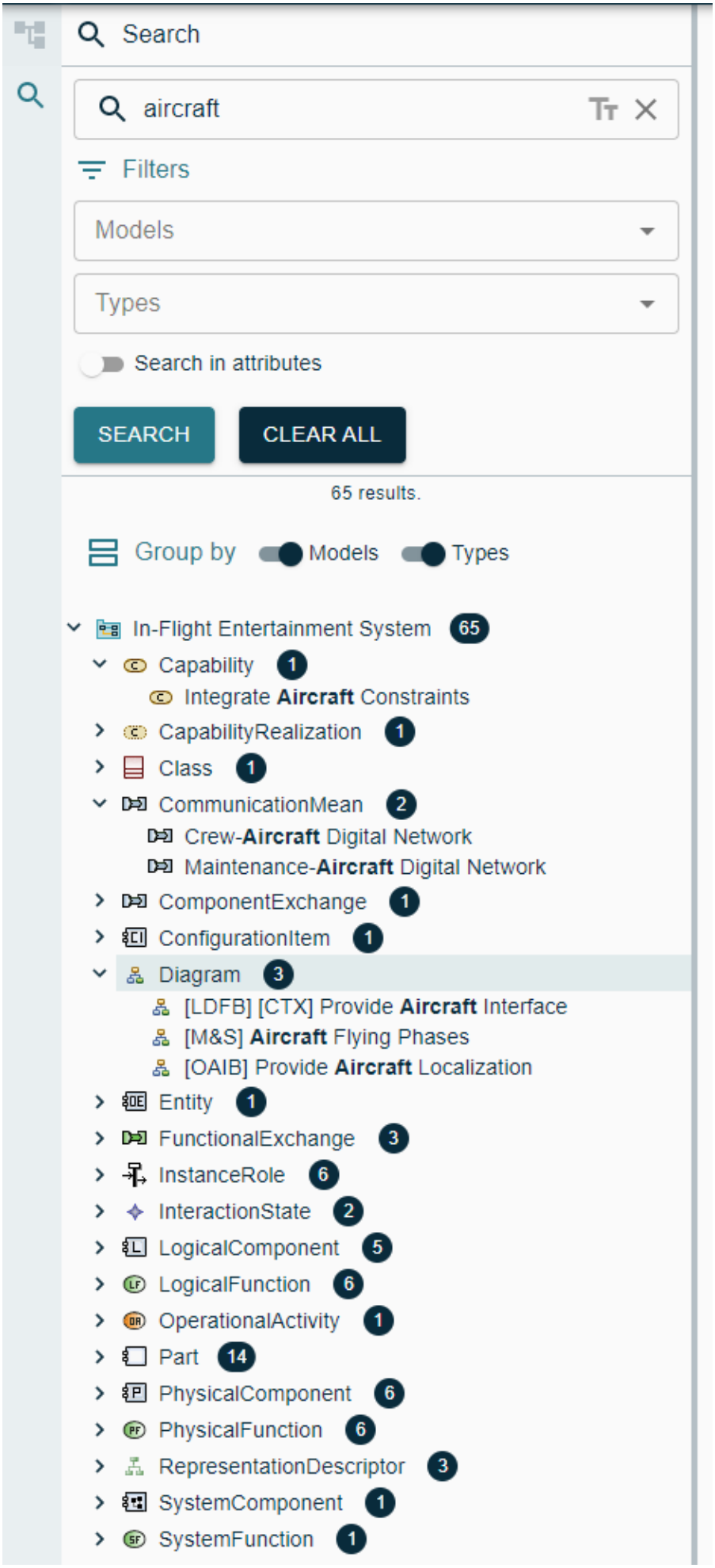


Figure 14. Grouped search results

The results shown in the view are paginated, which means only the first 30 results of each group are shown. To see more results, a 'More results' link appears at the bottom of a list of results (if there are more to show).

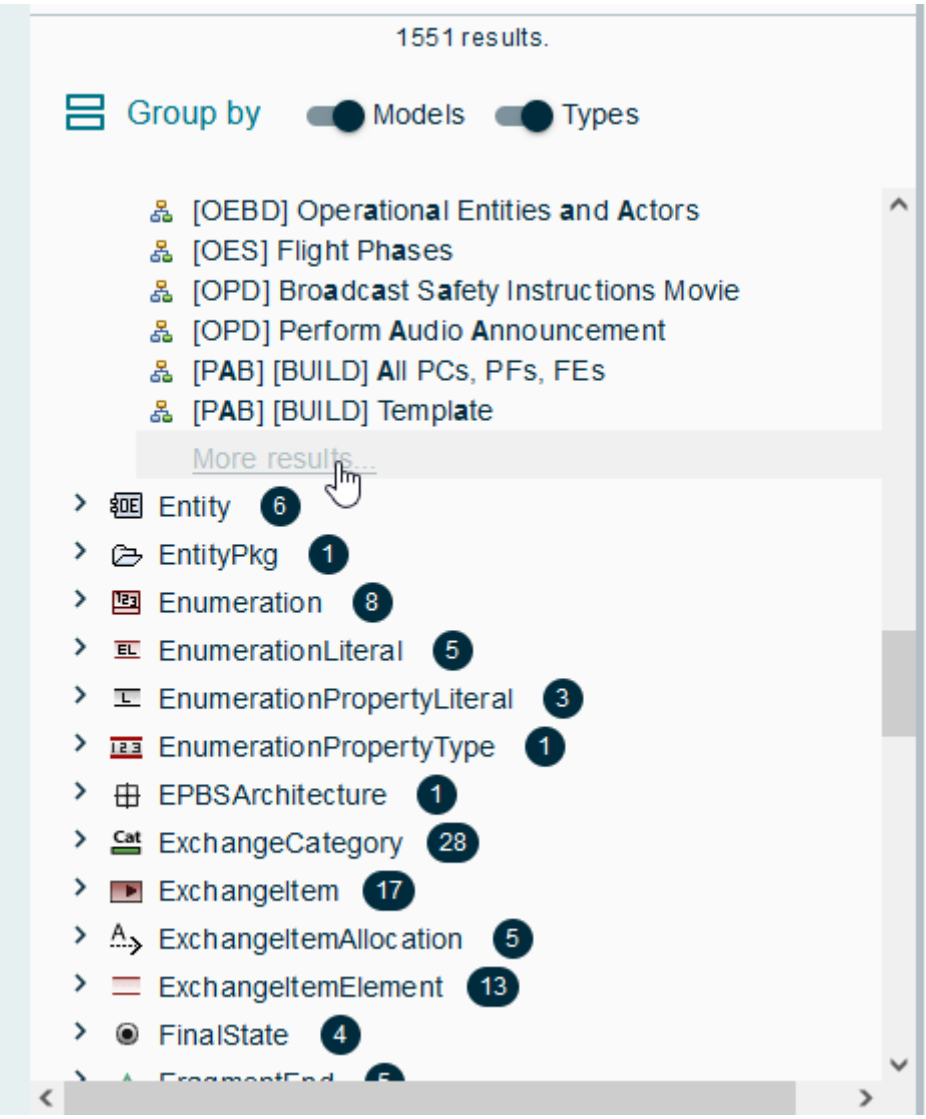


Figure 15. Browsing results

The 'Clear all' button will reset the form to its initial state. All search parameters will be left blank and options reset to their default (no result grouping, no search in attributes).

Main Panel - Model Navigation

The main panel displays useful information about a particular object, which can be a 'Model' or a 'Model Object', including a special kind of of model object that is a 'Diagram'.

Models

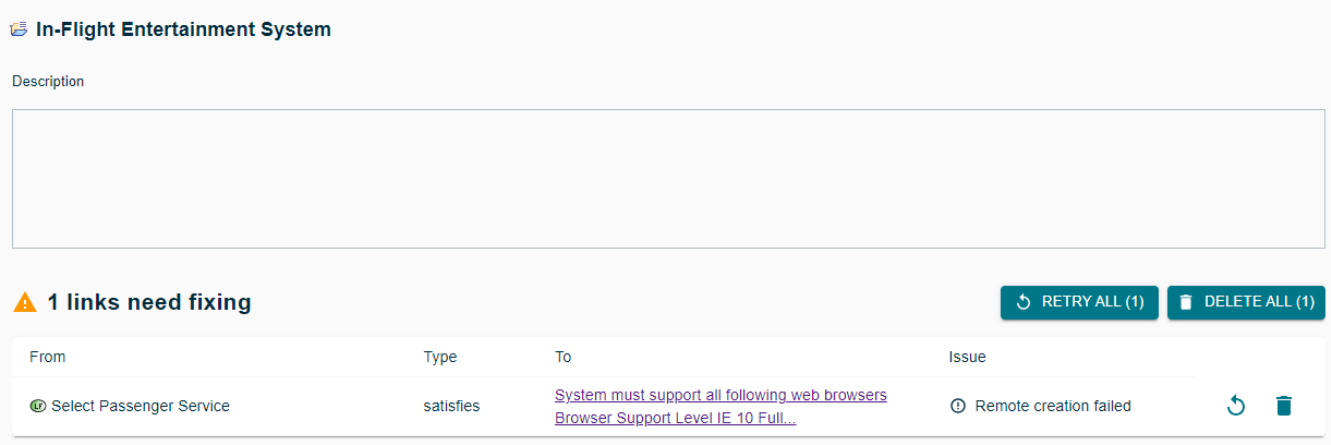


Figure 16. Model information

For Models, the main panel displays:

- The model Name;
- Description;
- An optional section to fix invalid OSLC links (for more details, see Fixing Invalid Links).

Model Objects

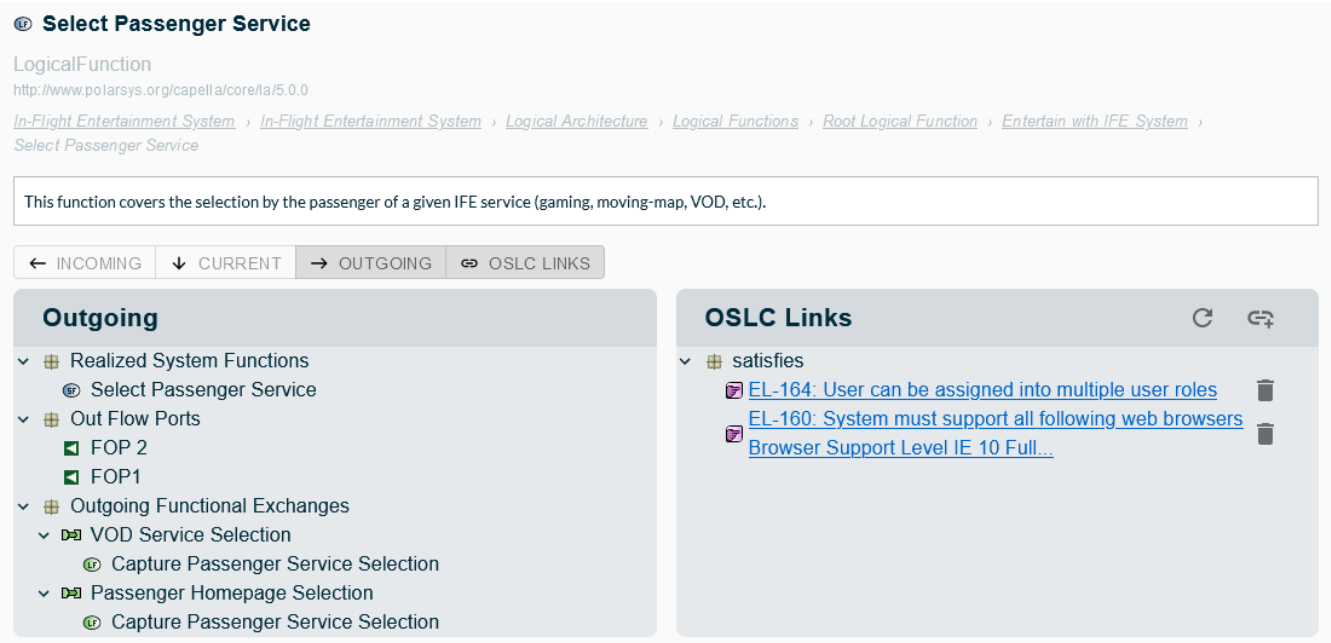


Figure 17. Model object with semantic browser with 4 panels, 2 of them filtered

The model object pages display:

- The object's icon and label;
- The object type name (its meta-class name);
- The namespace of the object's meta-class (useful to configure OSLC Linkings);
- The 'Bread Crumbs', which displays the list of parents of the object in the model hierarchy;
  - The parent objects in the 'Bread Crumbs' can be navigated by clicking on them.
- The object's rich text description;

    - The description may contain links to other model elements or diagrams, which can be used to navigate within the model. It is possible to hover any link that refers to a model element or diagram to display its Preview.
- The semantic browser. See next section for more information about the semantic browser.

Semantic Browser

The semantic browser of Perseus is organized as follows:

- 3 optional panels display the Capella semantic browser, as it is in Capella.
    - These panels are only present if the model is a Capella model, and has been published with the 'semantic browser' option.
- One panel 'OSLC links' that displays the links to third-party repositories, if any.
    - This panel makes it possible to view and edit links between the current object and artifacts exposed by Friend OSLC repositories.
    - The types of the links that can be created depends on the OSLC Linking Settings.

The semantic browser has a 'title bar' made of buttons, one button per available panel. This title bar acts a a filter to display or hide panels at will. Each Panel can be hidden or displayed by clicking on its name in this title bar.

The selected panels will be remembered between pages for each user, using the navigator local storage.

If access to the browser's local storage is forbidden, then the selected panels will be reset each time the user navigates.

OSLC Links

OSLC Links are displayed in a specific panel within the semantic browser. When a model object page is loaded, the semantic browser is automatically displayed. The *OSLC Links Panel* can appear with limited information.

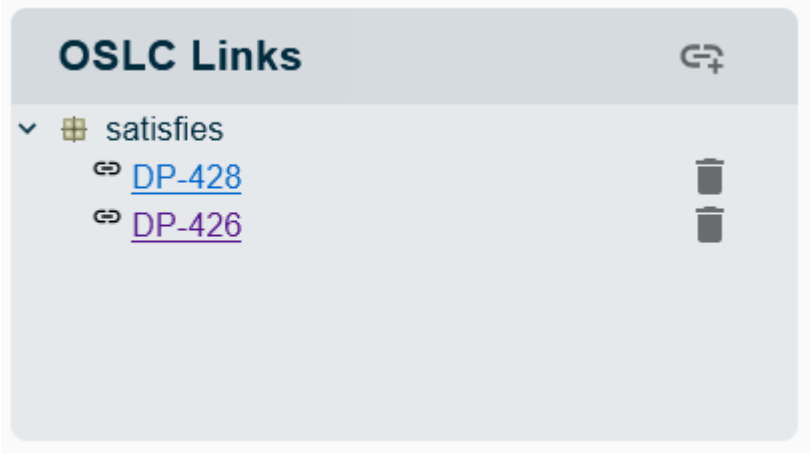


Figure 18. OSLC Link Panel (before loading information from OSLC friend server)

Then, the OSLC Panel will be refreshed automatically when they receive up-to-date information

from the friend server(s). That information includes:

- The icon of the remote artifact
- The label of the links

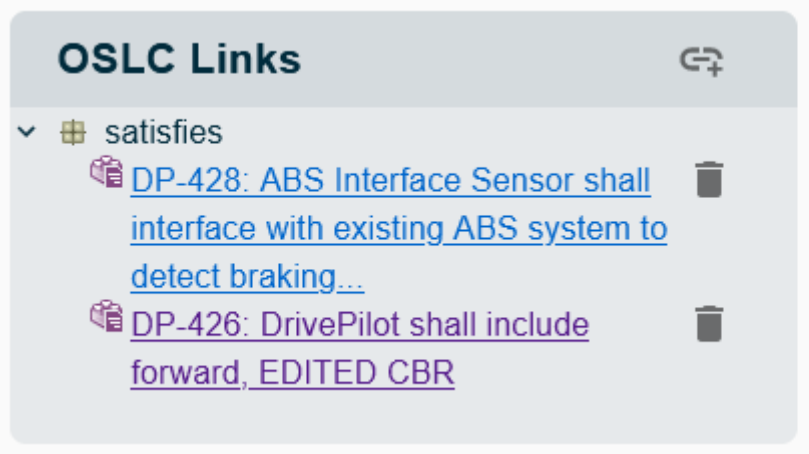


Figure 19. OSLC Links Panel

The OSLC links can be clicked to navigate to this object on the remote repository.

The *OSLC Links Panel* displays:

- A button to refresh the panel;
- A button to create a new link in this panel;
- A tree of existing links, grouped by type. For each link, the panel displays:
  - An icon;
  - A label (as a navigable URL, that opens in a different page);
  - A delete button, to delete that link.

Once they are created, links are displayed in the OSLC Links Panel to every user who can *read* the contents of the project that contains the model. There is no check that the current user has any kind of access or authorization to the third-party server. This can lead to revealing the existence of artifacts to users who are not authorized to see these artifacts on their remote repositories.

Add an OSLC Link

- **Pre-Requisites**
  - The user must have the right to *edit* the project.
  - The user must have an account on the targeted remote repository, and whatever rights are needed on that repository to create links.
  - An OSLC Association (see OSLC Associations) must have been configured on the project that contains the current model.
- ***Modus Operandi***
  - Click on the 'add link' button in the OSLC Links Panel header. That will open the OSLC

Link Creation Dialog.

- Select the link type in the left drop-down menu:

Figure 20. OSLC Link Creation Dialog - Select Link Type

- Then select the OSLC Action Dialog to use for the selection or creation of the remote artifact. The contents of this drop-down is dynamically loaded from the Friend server:

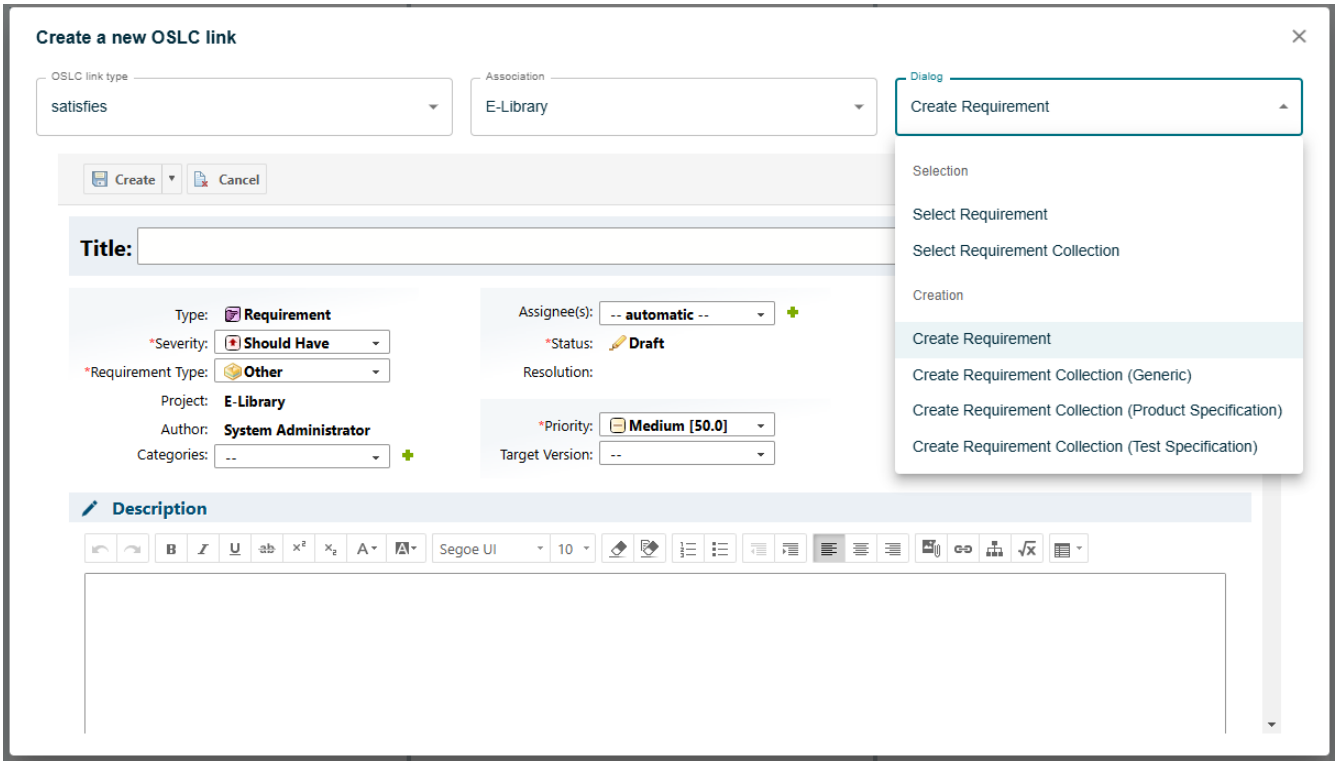


Figure 21. OSLC Link Creation Dialog - Select Action Dialog

- The relevant delegated UI will then be displayed within the *OSLC Link Creation Dialog*. Here is an example using Polarion:

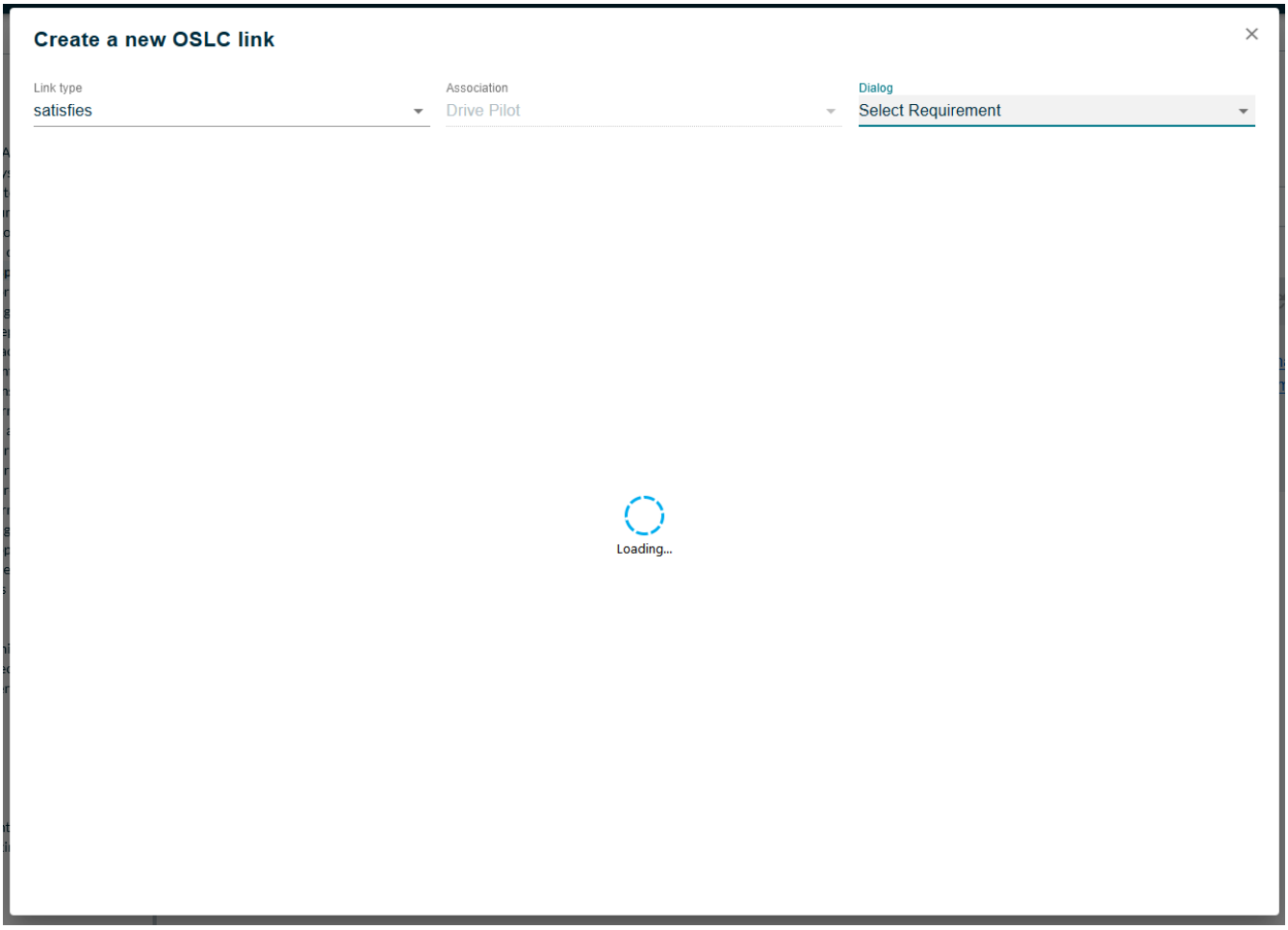


Figure 22. OSLC Link Creation Dialog - Load Action Dialog

- Use the delegated UI to select or create the target of the link on the remote Friend server:

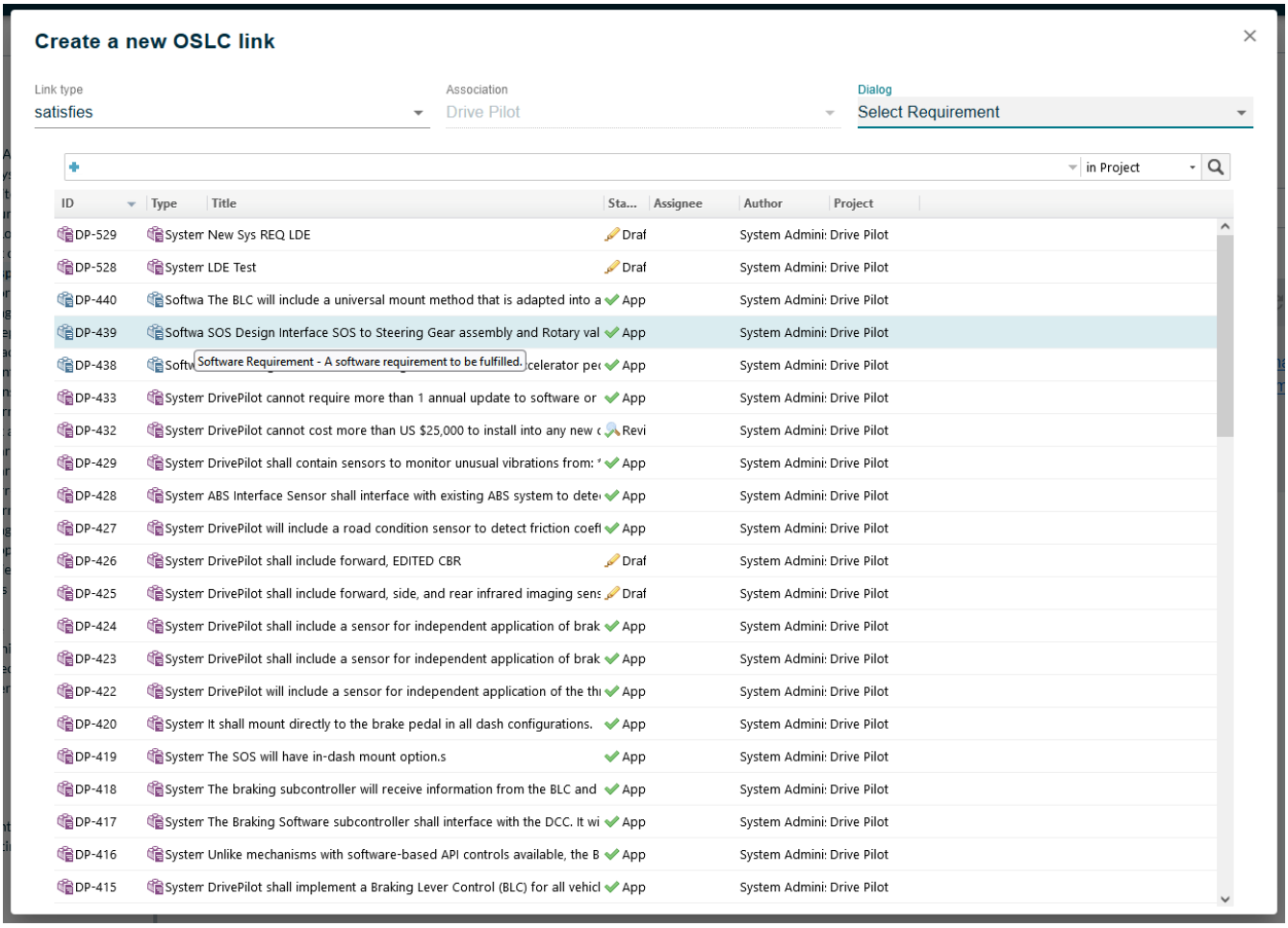


Figure 23. OSLC Link Creation Dialog - Use Action Dialog

- The *OSLC Link Creation Dialog* remains open in case the user wants to create several links in a row. To close it, click outside of it anywhere in the page.
- The *OSLC Links Panel* is automatically updated with the new link or links.

During this process, it is possible that the Friend server demands a confirmation to the Publication for Capella user that they agree on Publication for Capella accessing their data on their behalf on the friend server. Refer to the OAuth section of this documentation for more details.

Any creation of link done in Publication for Capella server will be effective in the corresponding Capella model after the "Update from server" action described in the Getting Changes from the Publication Server chapter.

Delete an OSLC Link

- **Pre-Requisites**
  - The user must have the right to *edit* the project.
  - The user must have an account on the targeted remote repository, and whatever rights are needed on that repository to create links.
  - An OSLC Association (see OSLC Associations) must have been configured on the project that contains the current model.

- ***Modus Operandi***
  - Click on the 'delete' button of the OSLC link to delete. That will trigger the deletion of the link. A progress indicator appears on the panel until the deletion is done.

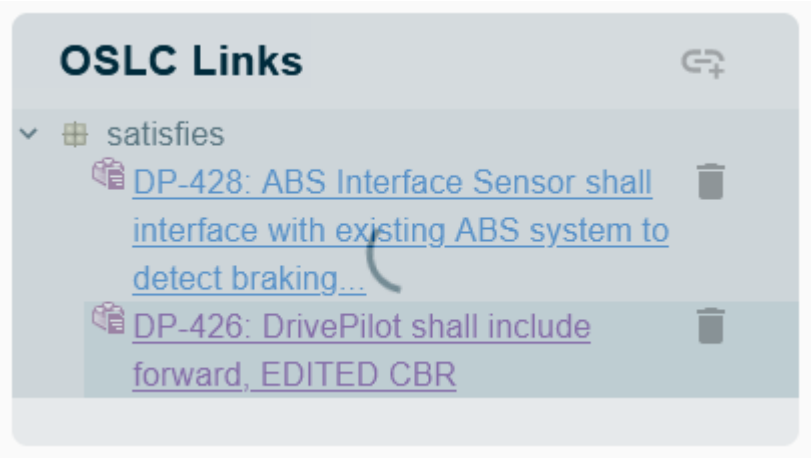


Figure 24. OSLC Link Deletion in Progress

- On success, the *OSLC Links Panel* is automatically updated and the deleted link disappears.
- It may happen that the delete is partial, which means that the deletion of the corresponding backlink on the remote server failed. In such a case, the link will appear in the 'Invalid link' widget of the containing model which offers solutions to resolve the problem, as described in the Fixing Invalid Links section.

During this process, it is possible that the Friend server demands a confirmation to the Publication for Capella user that they agree on Publication for Capella accessing their data on their behalf on the friend server. Refer to the OAuth section of this documentation for more details.

Any deletion of link done in Publication for Capella server will be effective in the corresponding Capella model after the "Update from server" action described in the Getting Changes from the Publication Server chapter.

Diagrams

The information displayed for diagrams is mostly identical as for any other model element.

The only difference is that the diagram's image is displayed between the 'Bread Crumbs' and the description.

This image can be used to navigate in the model by clicking on the objects displayed in the diagram, except the object links.

It is possible to hover any object to display its Preview.

In some cases, a shape in a diagram represents several model elements. In such a case, a small pop-up menu is displayed and the user can choose to which element he or she wants to navigate.

The button in the top right of the diagram image is used to open the diagram in full size, using all

the space available in the browser's window.

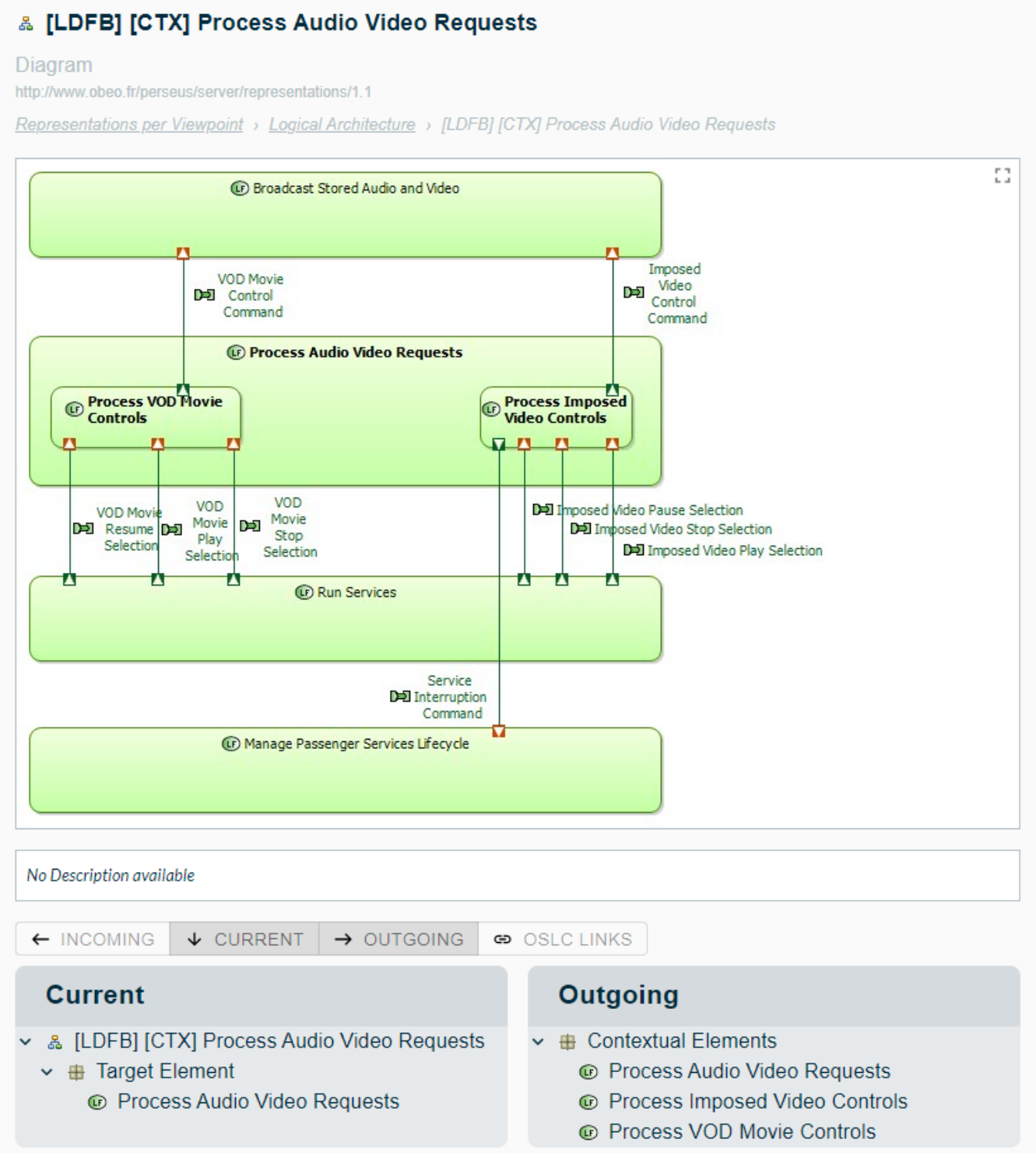


Figure 25. Diagram information

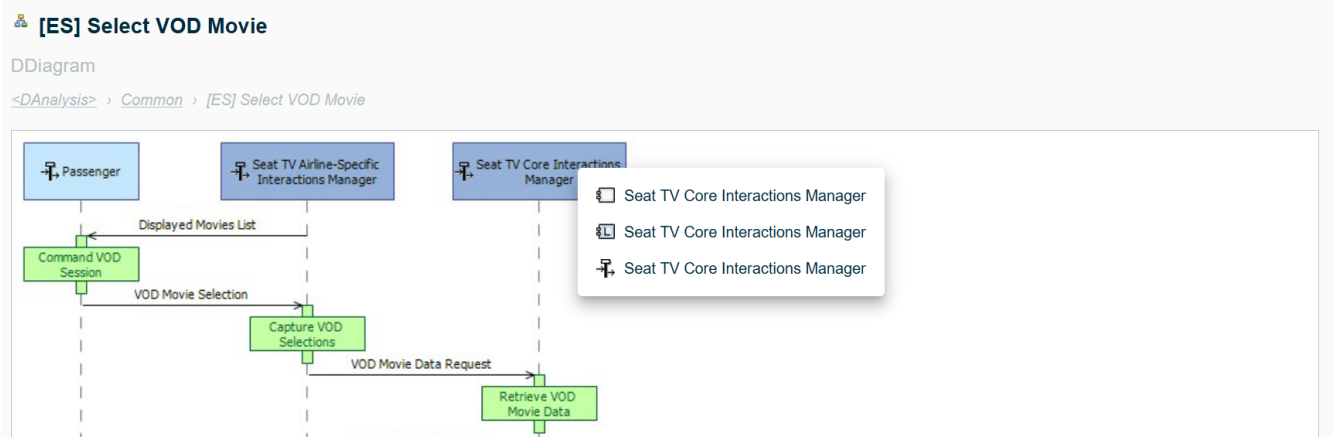


Figure 26. Model diagrams support navigation

Known limitation: Object links in the diagrams cannot be navigated by clicking on them.

Diagrams are actually standard model objects, with a type of `DDiagram`, mainly to match the diagram concept of Sirius, which are stored in `*.aird` files. However, diagrams in Publication for Capella just own an image of the corresponding Capella diagram, they do not contain all the information of the `DDiagrams` contained in the `*.aird` files.

Details Panel

The details display the selected object's attributes, as defined by its meta-class.

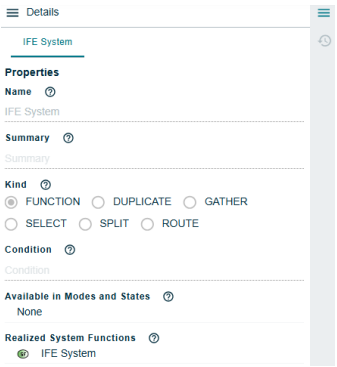


Figure 27. Details

The value of the properties cannot be edited. If the user attempts to edit them, an error message will be displayed.

This view is somehow similar to the properties view in Capella. However, it is much more simplistic and not configurable (yet). Consequently, the details are displayed in a similar manner for all objects.

History Panel

The history panel displays a list of the changes related to the current selection. Changes are displayed in a compact way but leaving the mouse over them displays a pop-up with detailed

information.

Model History Panel

When the selected object is a *Model*, then the history panels displays the complete history of this model, including any change to any element it contains.

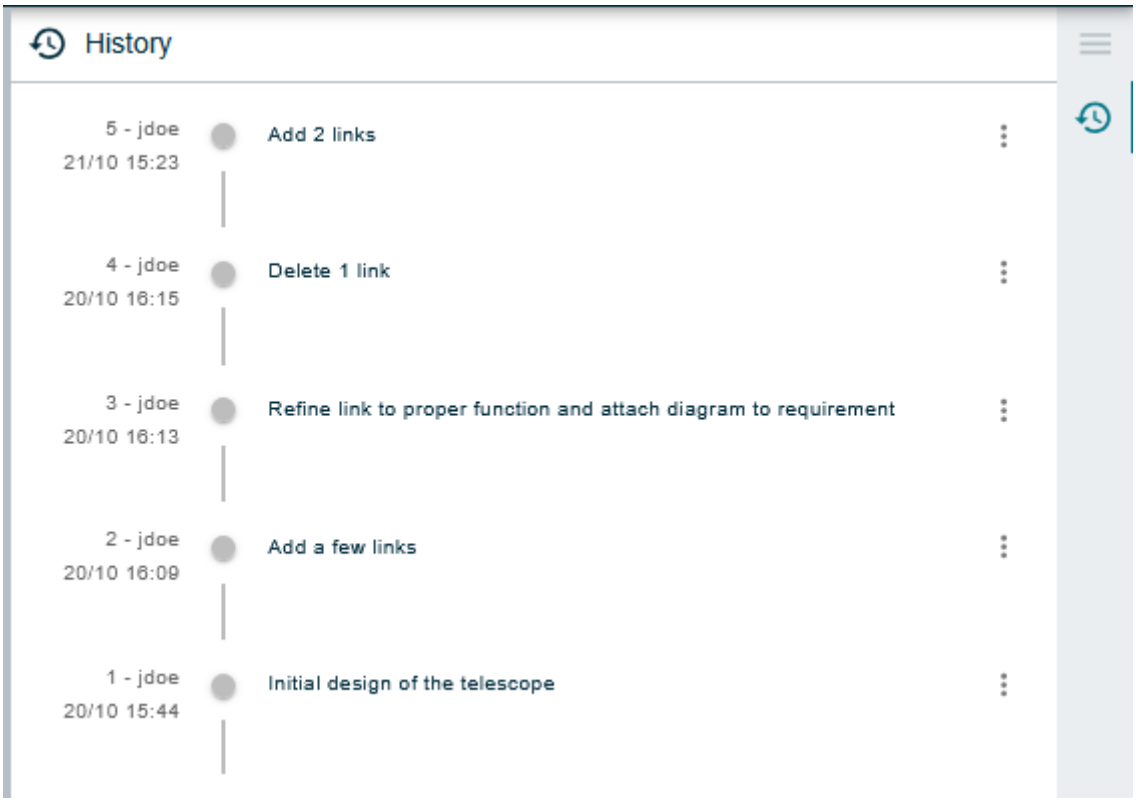


Figure 28. Model History

Hovering any commit in the model history view will display a tooltip that provides some details about the commit in question:

Figure 29. Hovering a Commit in the Model History

Several actions are available in the history panel when a model is selected, *via* the contextual menu:

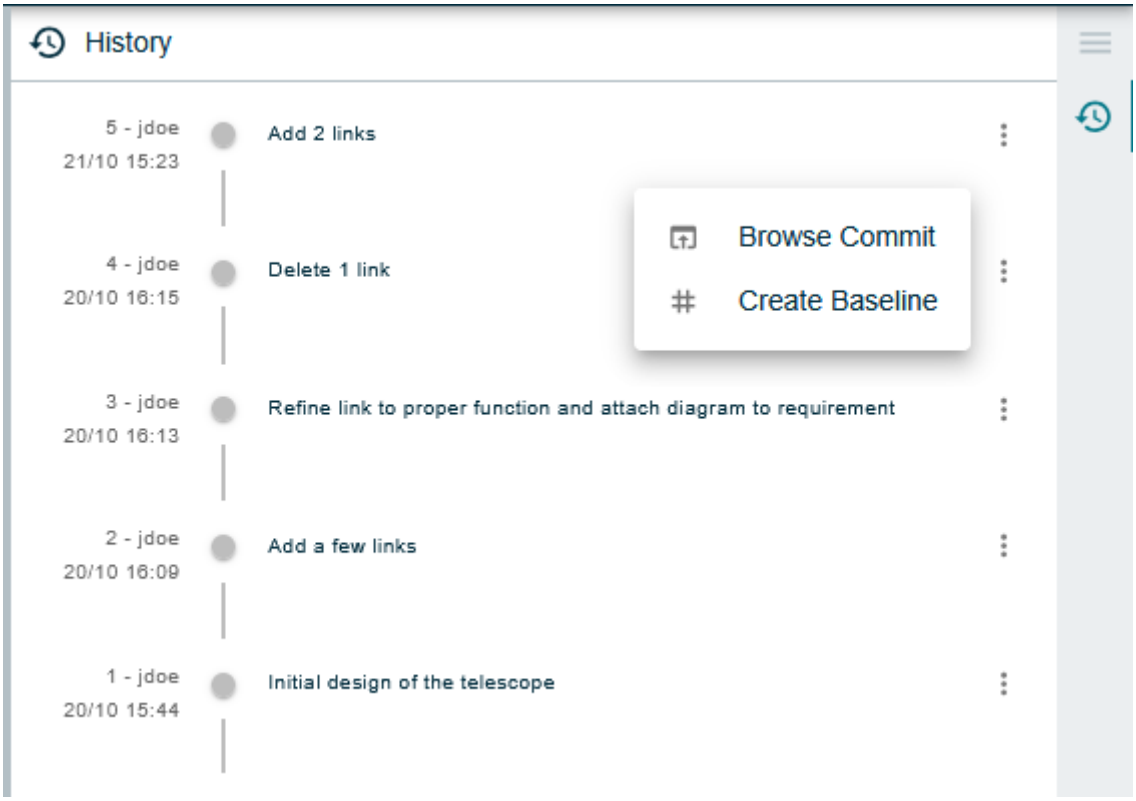


Figure 30. Model History Pop-Up Menu

- Create a baseline (if the logged-in user is at least a contributor to the project).
- Delete a baseline (if the logged-in user is administrator of the project).
- Browse the model on any baseline or even commit.
  - To browse the model on an existing baseline, just click on the baseline tag in the history view.
  - To browse the model on any commit, just select the action 'Browse Commit' on the desired commit.

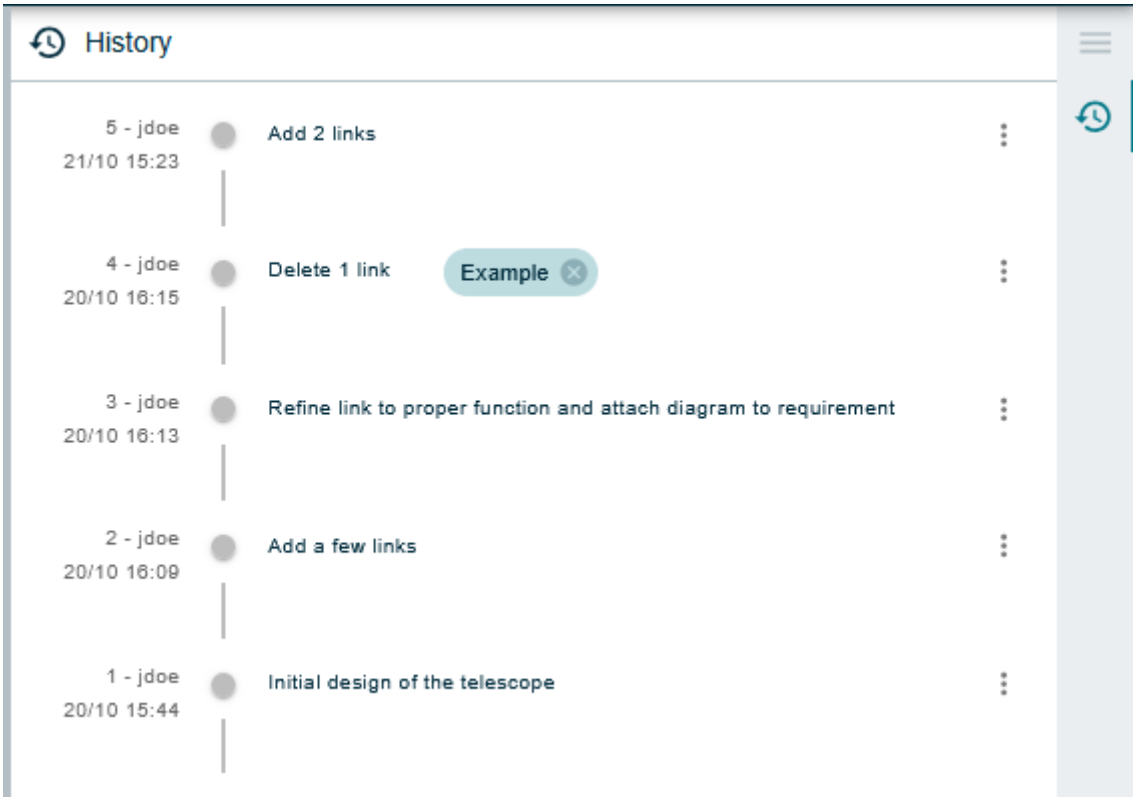


Figure 31. Model History with Baseline

To delete an existing baseline, click on the cross icon on the baseline chip.

A confirmation dialog is displayed:

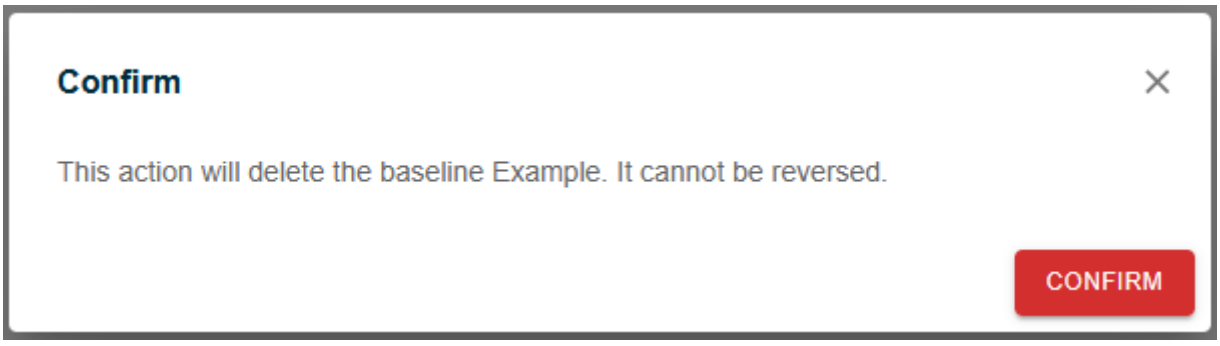


Figure 32. Delete Baseline Confirmation Dialog

Click on 'Confirm' to delete the baseline.

Creating a baseline requires being logged in as a contributor to the parent project. Deleting a baseline requires being logged in as an administrator of the parent project.

Model Object History Panel

When a model object is selected, only the changes impacting this object are listed in the history panel.

Hovering any commit in the model object history displays a preview of the selected object, as it was in this commit. If the selected object is a diagram, the preview displays the diagram, as it was in the hovered commit.

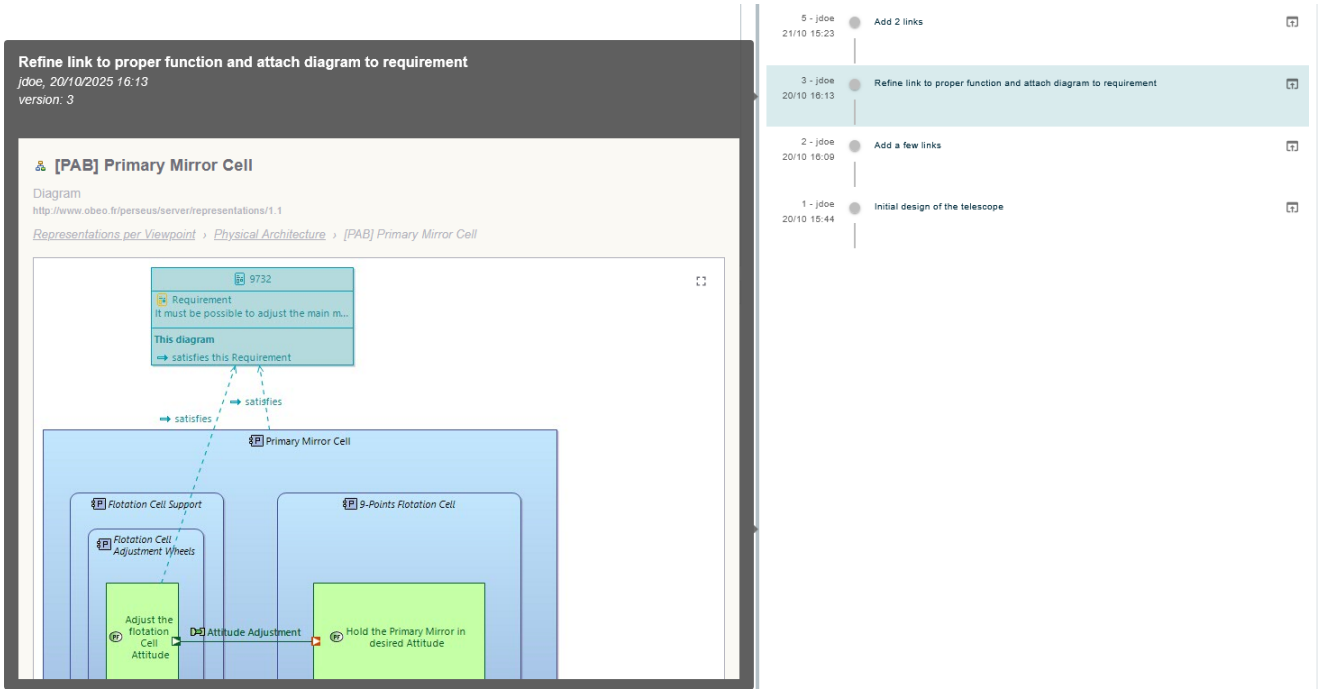


Figure 33. Model Object History with Diagram Preview

Several actions are available in the history panel when a model object is selected, *via* the contextual menu:

- Delete an existing Baseline (if the logged-in user is administrator of the project).
- Browse the model on any visible Baseline or commit. Activating the browsing will land on the currently selected object's page, in the context of the desired Baseline or commit.

Creating a baseline can only be be done from the Model History Panel.

Preview

Previews of model objects and remote artifacts linked through OSLC are available when hovering several elements:

- Breadcrumbs items;
- Search results;
- Model elements in the semantic browser;
- Invalid links table elements;
- Model elements, artifacts, and some edge labels in the diagrams;
- Links that refer to model elements or diagrams in the rich text descriptions.

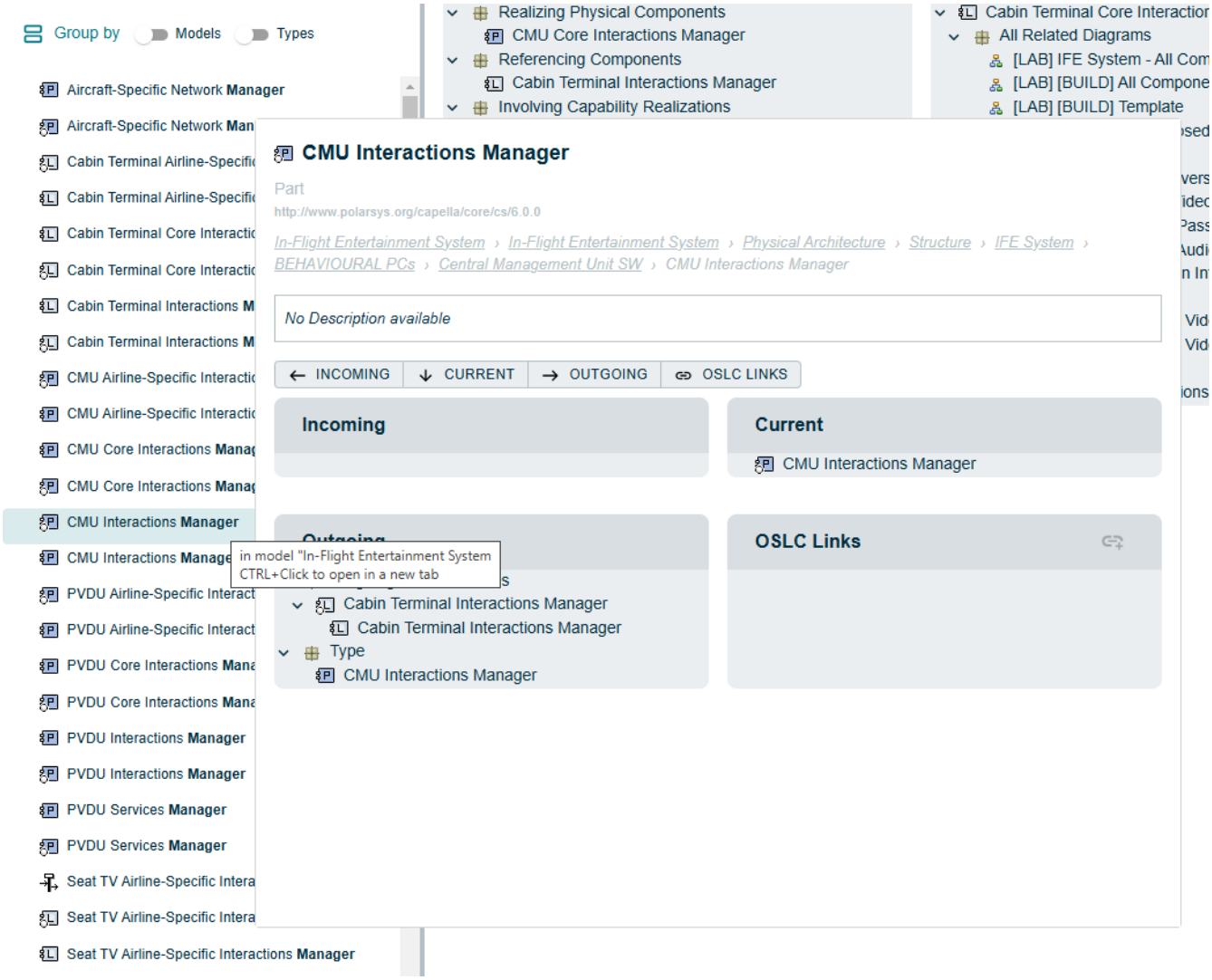


Figure 34. Model Object Preview

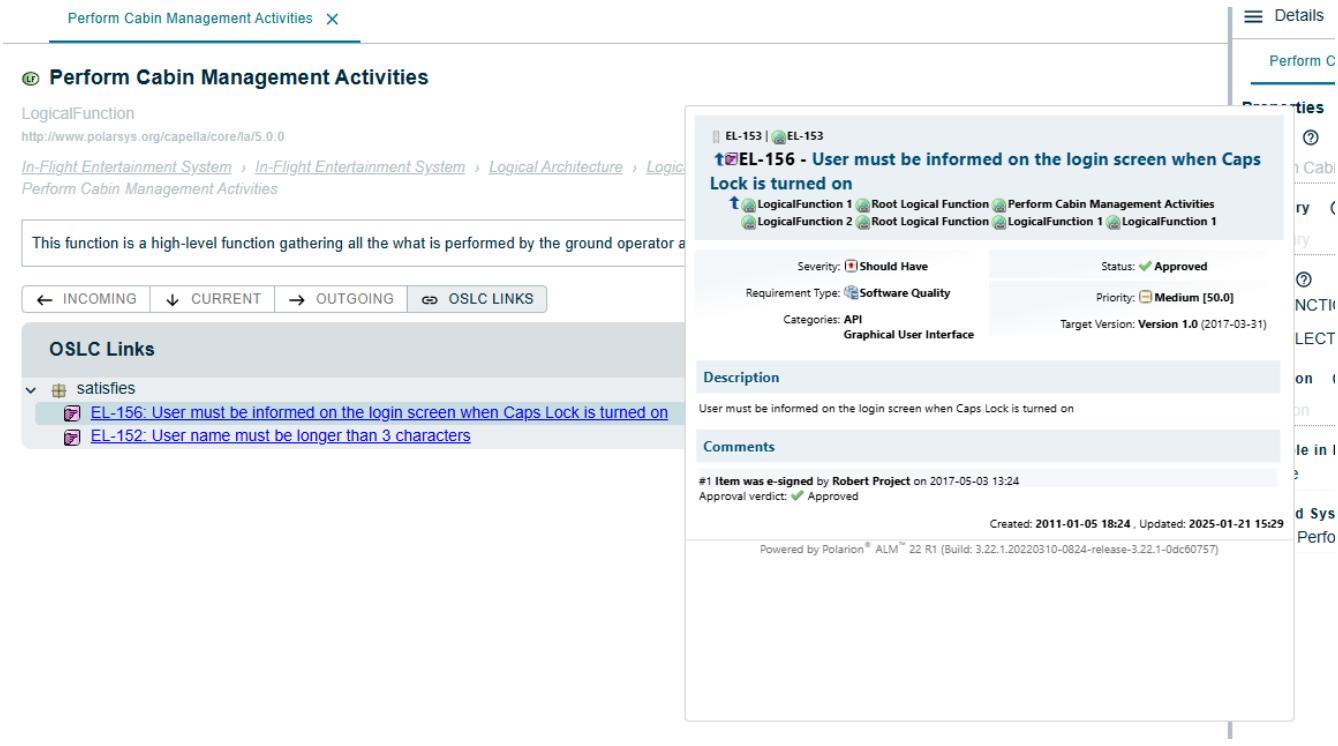


Figure 35. Remote Artifact Preview

Previews are currently unavailable in the model explorer and in diagrams.

Project Settings Page

The settings page of a project makes it possible to edit the settings of this project. It displays the following tabs:

- Visibility - See Project Visibility
- Associations - See OSLC Associations
- OSLC Linking - See OSLC Linking Configuration
- Global Conf. - See Activating *Global Configuration* on a Project

Project Visibility

The project settings visibility tab makes it possible to edit a project's visibility (either `private` or `public`).

Projects marked as `public` can be accessed for reading by any logged-in user with a `READ` permission license token (See Licenses Management). This simplifies access management to non-sensitive projects.

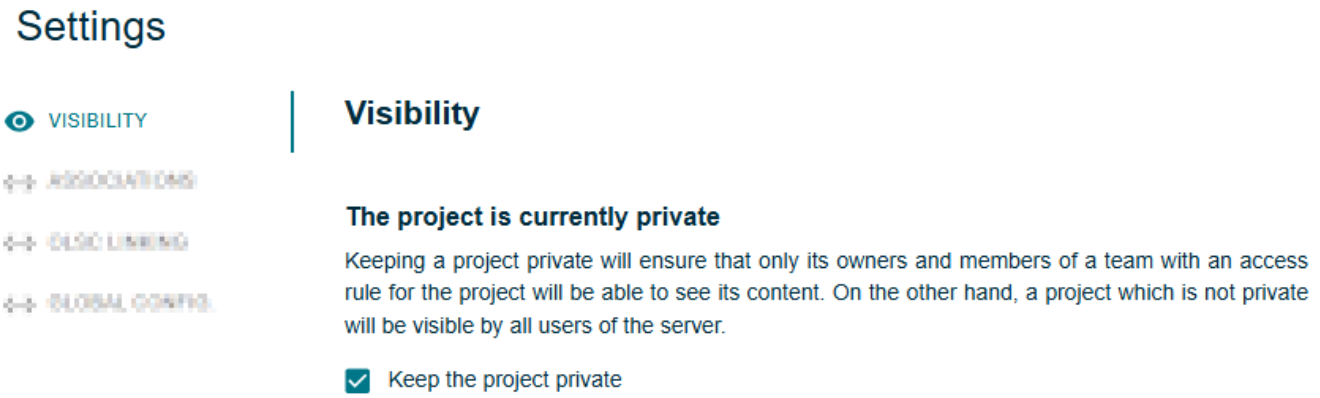


Figure 36. Project Settings: Visibility

Teams Management

Project owners can create and edit Teams and assign access rights to users.

The teams management page can be accessed by clicking on the account icon:

Figure 37. Account Icon

Then in the menu, by clicking on "Teams":

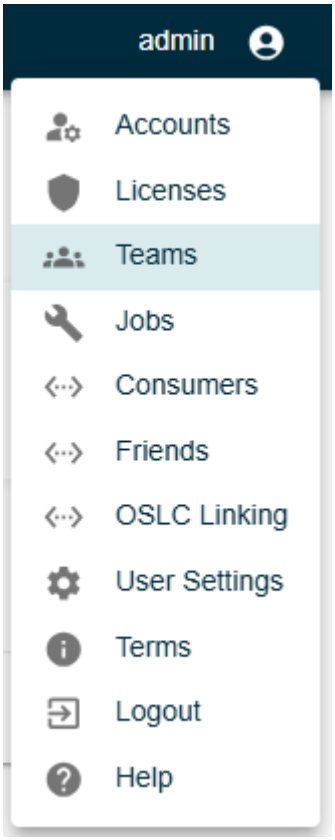


Figure 38. Teams Menu

This action will lead to the Teams page:

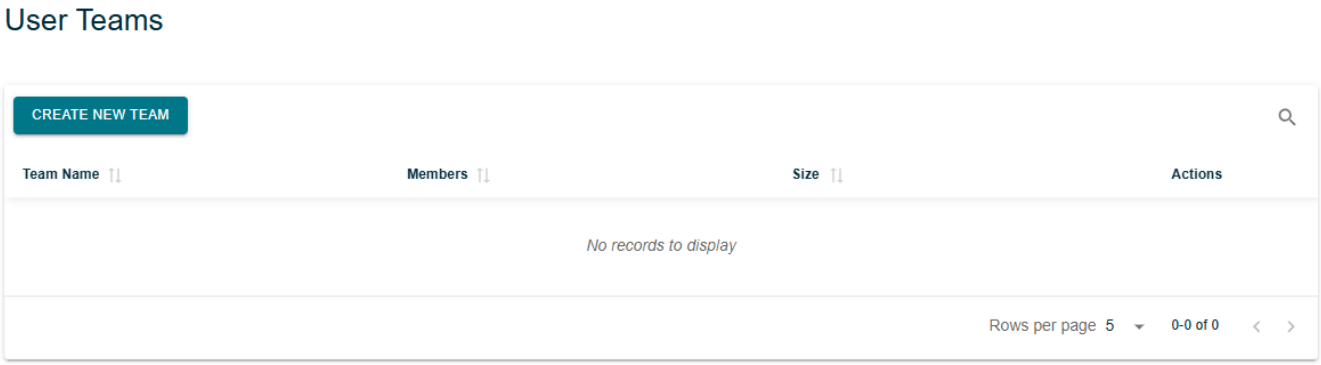


Figure 39. Teams Page

Team Creation

You can create a new team using the 'Create New Team' button.

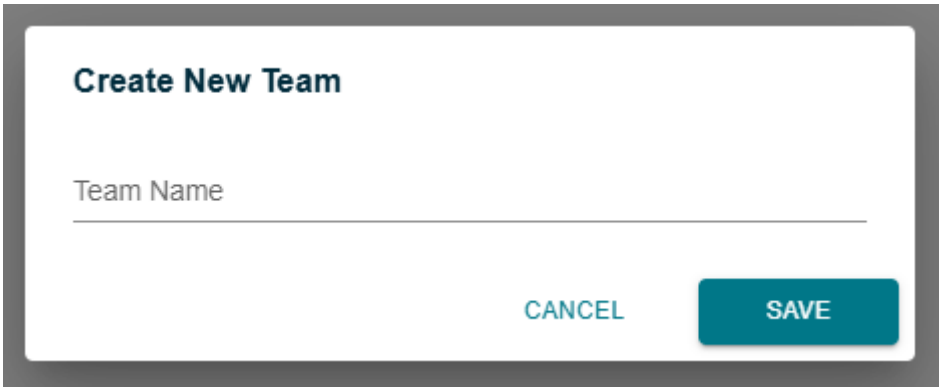


Figure 40. Team Creation Dialog

You can then rename the team using the pen icon or delete it using the trash icon.

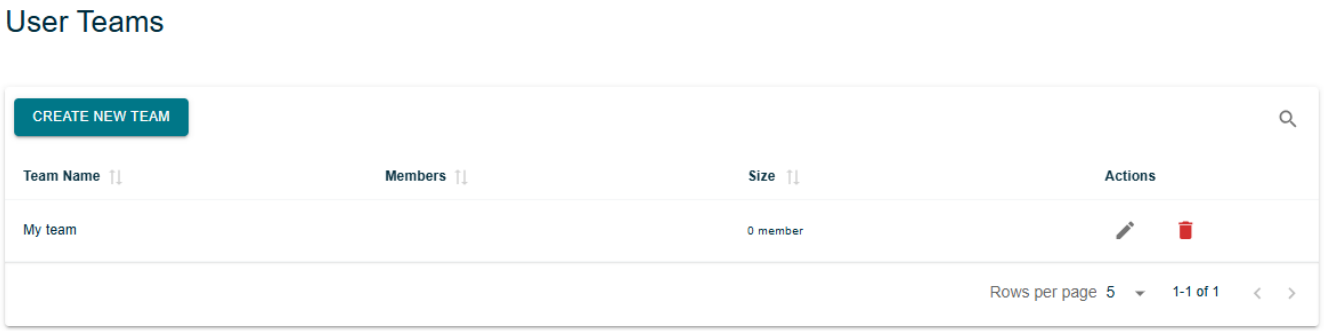


Figure 41. Created Team

To add members and access rules to the team, click on the team name to access the team page.

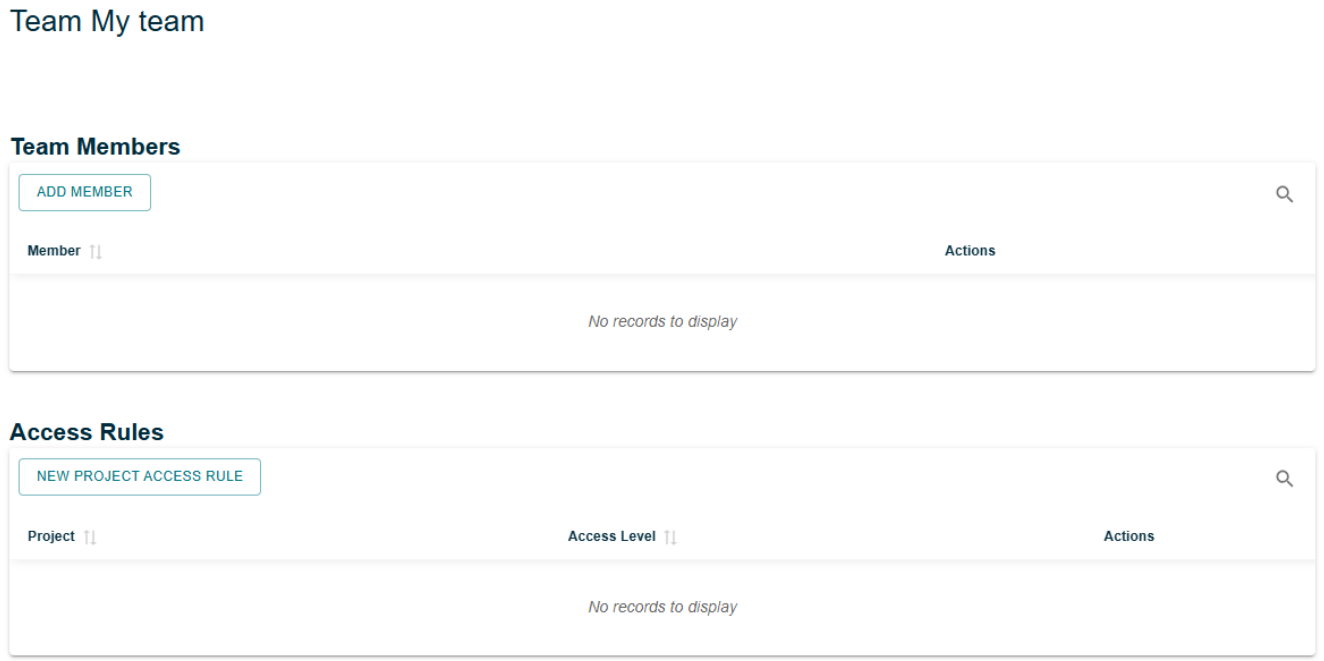


Figure 42. Team Page

Team Members

To add a member to a team, use the 'Add member' button. Enter his or her user name in the field 'Member' and click on the 'Save' button.

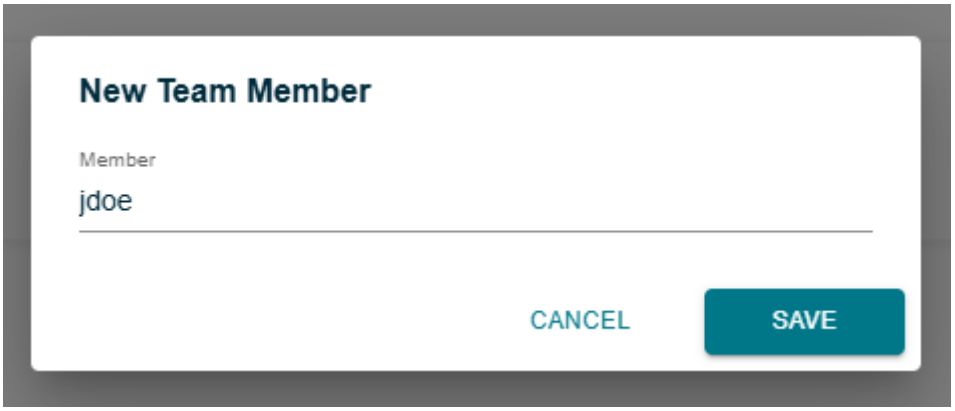


Figure 43. Add Member Dialog

To remove a member from a team, use the cross next to the user in the team page.

Access Rules

The team system allows to parametrize three access levels:

- `READ` access level: Members of this team can 'view' the models exposed by this project.
- `WRITE` access level: Members of this team can 'view and publish' models in the project (or publish updates of models already attached to this project), rename the project.
- `ADMIN` access level: Members of this team can 'edit this project', configure OSLC associations on it and delete the project.

Any access level will be restricted to the license privileges of the user, e.g. a user with `ADMIN` access on a project but with a `WRITE` license privilege will only gain a `WRITE` access on the project.

To create a project access rule in a team, use the 'New Project Access Rule' button.

- Select the access level you want to grant the members of the team.
- Select the project you want to grant access to.

Click on the 'Grant access' button.

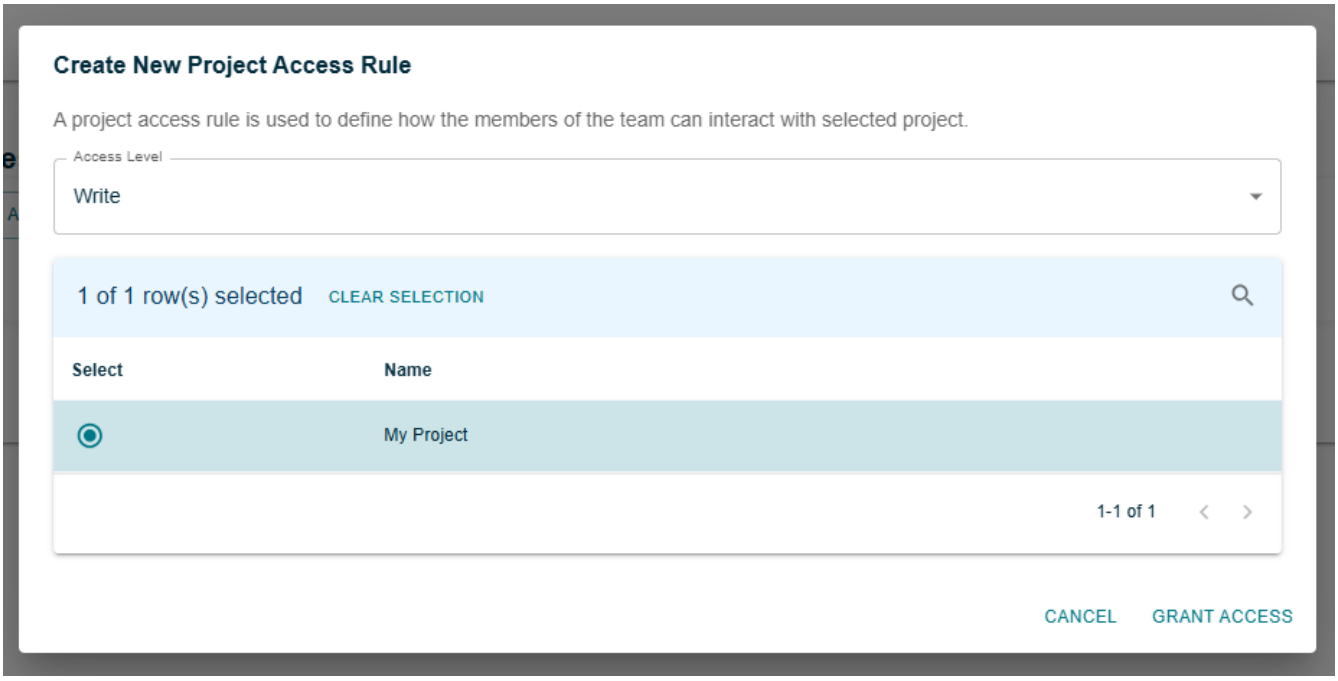


Figure 44. Add Access Rule

Once granted, an access rule can be deleted using the trash icon or updated by creating a new access rule on the same project.

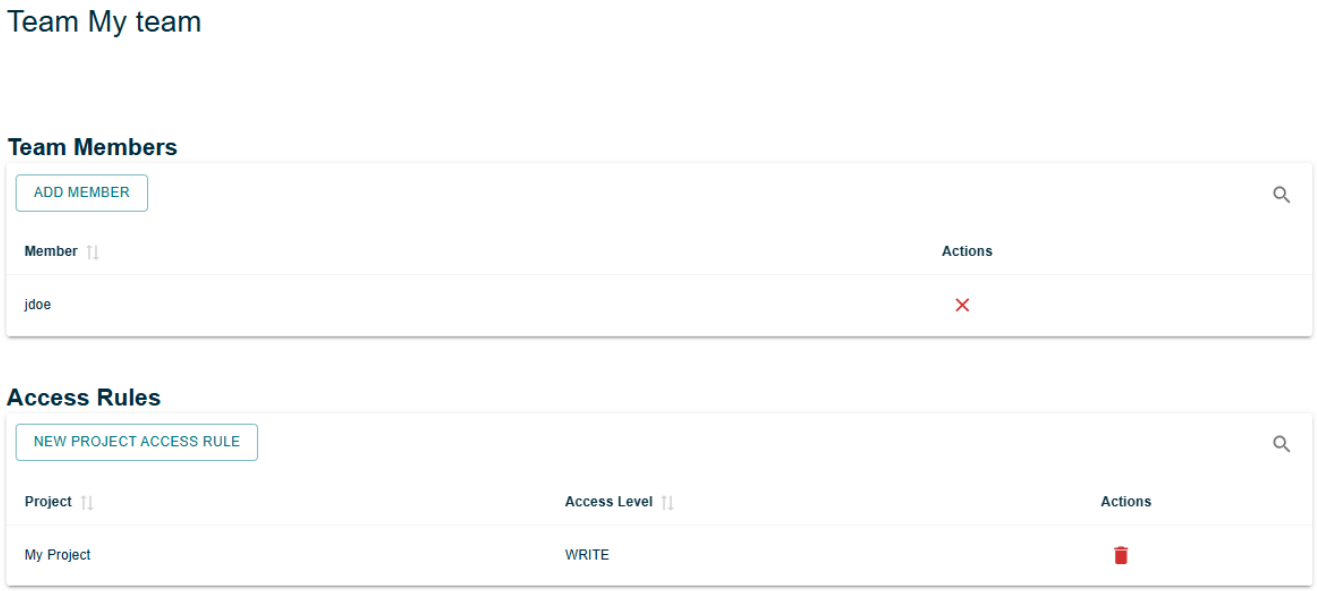


Figure 45. Access Granted

The team list is managed on the server perimeter so the list is common to all projects present inside the Publication for Capella server. They are not "inside" a project. So name of teams should be carefully defined to distinguish teams across the different projects.

User Settings Page

The settings page of a user makes it possible to edit the settings. It displays the following tab:

- Explorer Filters - See Explorer Filters

Explorer Filters

The model explorer view is filter according to the criteria defined in these settings. Each user can have its own settings, and they all inherit the server default settings if they don't explicitly change these settings.

The server administrator can edit the server default settings and reload these default settings so that they become active for every user that has not explicitly configured their own settings.

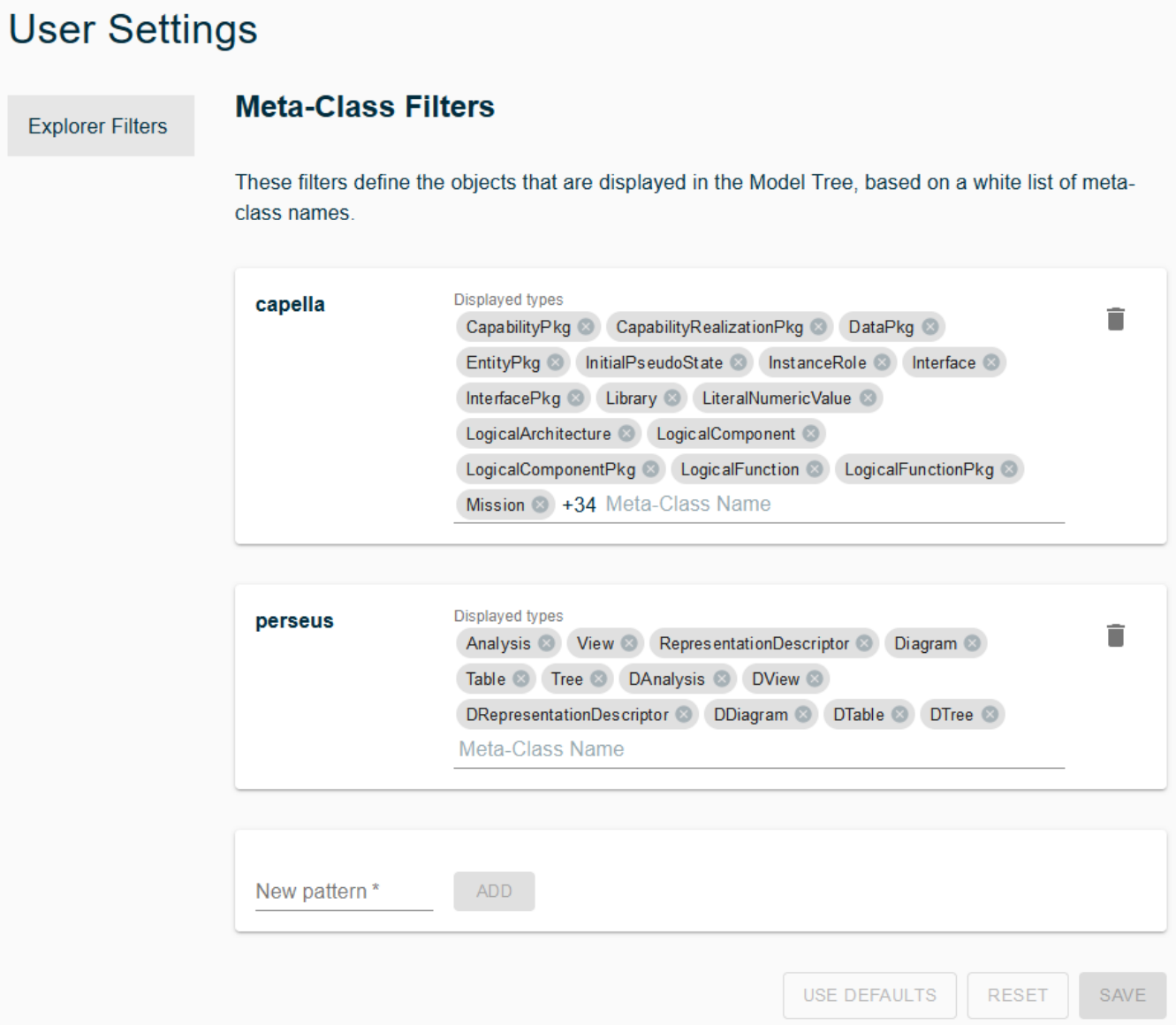


Figure 46. Meta-Class Filter Settings

The settings are based on two simple concepts:

Meta-Model Pattern

A String that will be used to match meta-models. Each meta-model whose `NsURI` contains the pattern will match.

Meta-Class Name

A String that identifies a meta-class by its name.

To be displayed in the Model Explorer, an object must have a type (i.e. a meta-class) that complies

with the following constraints:

- The meta-class name must be among those present in the settings.
- The meta-class' meta-model must match the associated pattern.

For example, the default settings define two patterns:

- `capella`: This pattern matches all the Capella meta-models, since their meta-model `NsURI` all contain the String `capella`. For example, The Logical Architecture meta-model in Capella 6 has the following `NsURI`: `http://www.polarsys.org/capella/core/la/6.0.0`.
- `perseus`: This pattern matches all the Publication for Capella meta-models.

The list of meta-classes to display in the explorer can be edited for each pattern. The editor offers completion features to facilitate the selection of the relevant type.

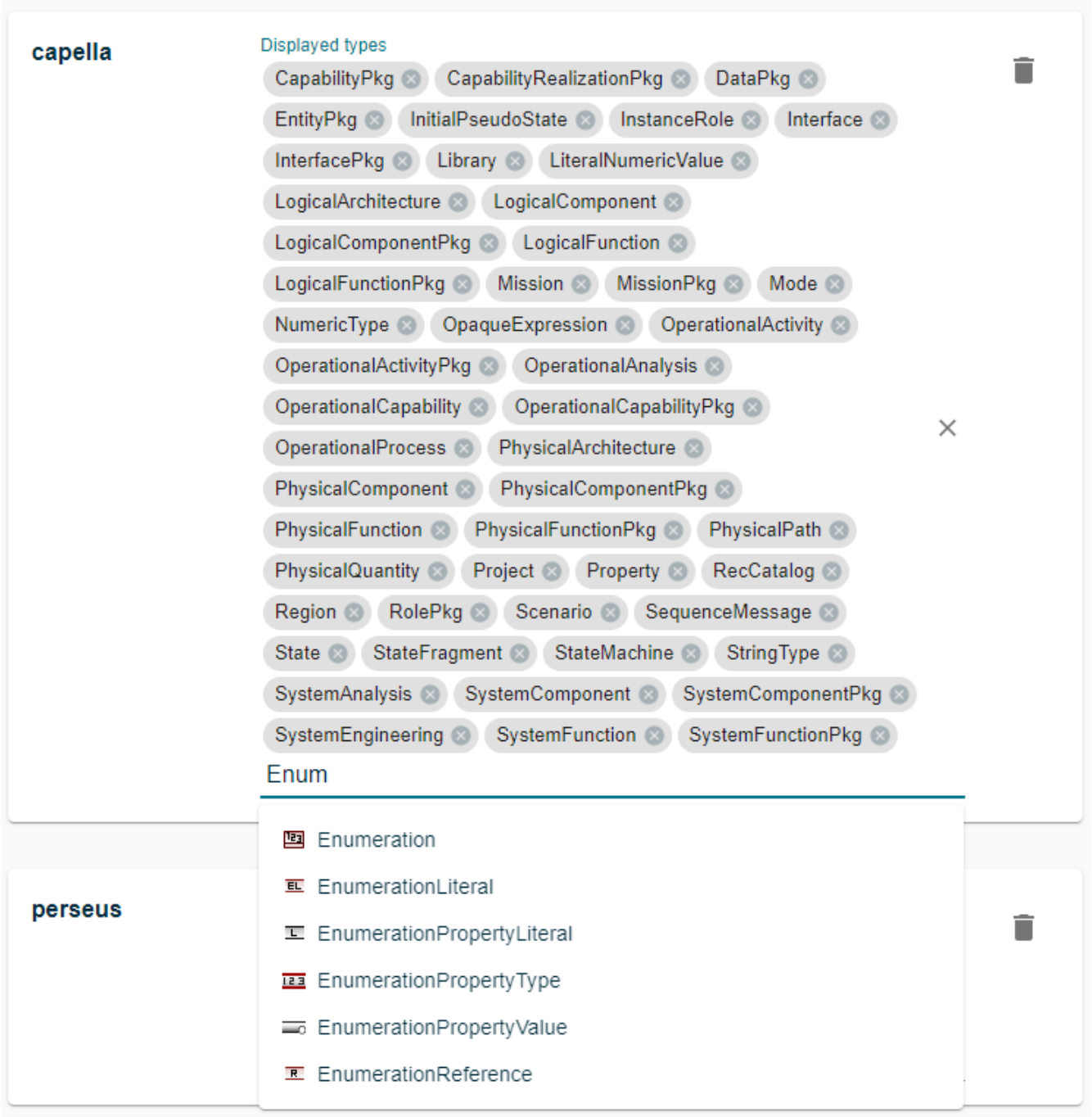


Figure 47. Meta-Class Name Completion

It is also possible to add or delete a pattern. The bottom text Field labeled 'New Pattern' can be used to enter the name of a new pattern to add. Then, the user must click on the 'Add' button.

The addition of a pattern is only possible if the new pattern does not overlap with an existing pattern. For example, trying to add a pattern `cap` when there already is a pattern `capella` is forbidden because any meta-model that matches `capella` would also match `cap`.

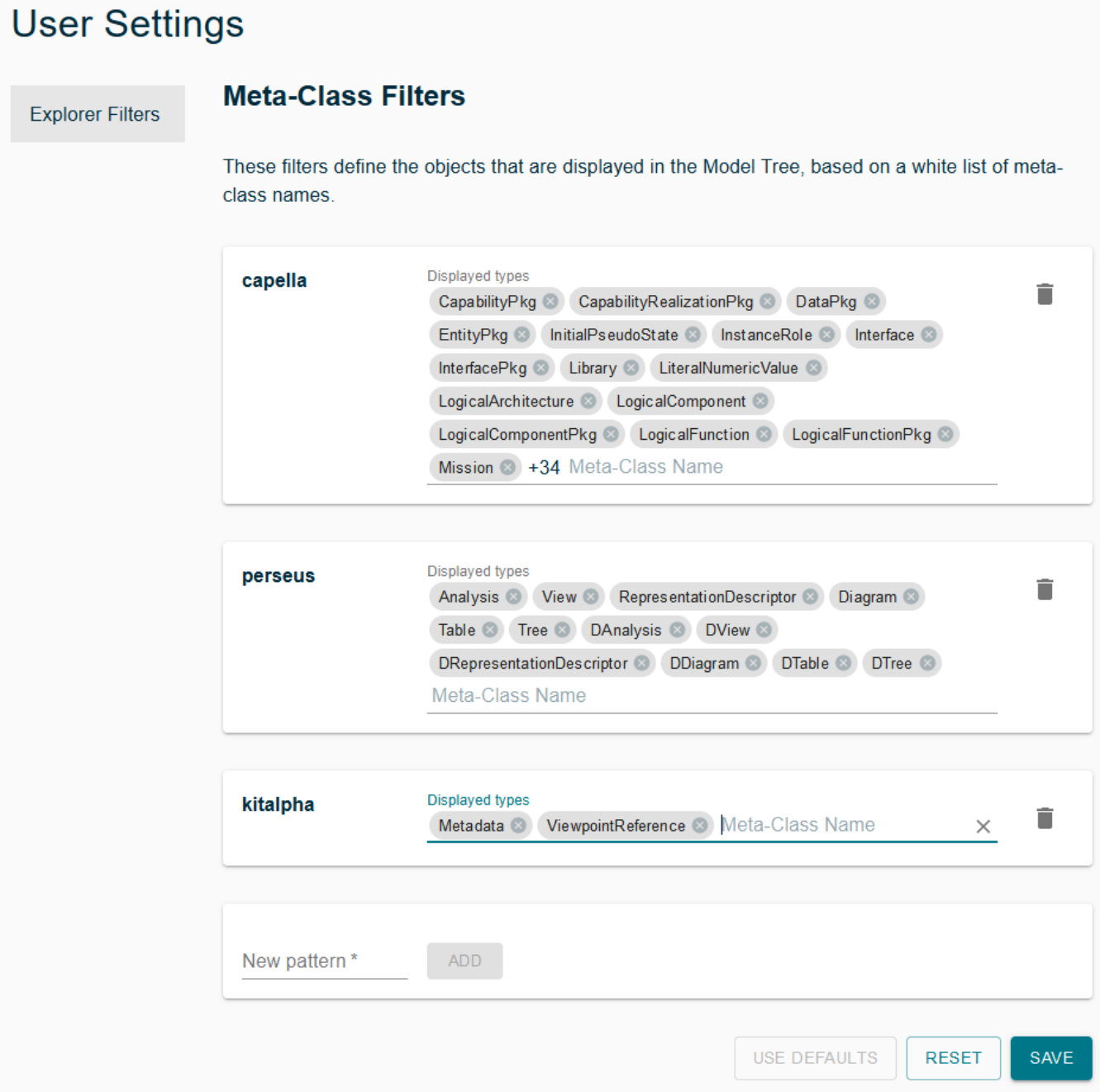


Figure 48. Meta-Model Pattern Addition

A pattern and the associated list of meta-class names can be deleted from the settings by clicking on the delete icon of the corresponding panel.

Three buttons are available at the bottom of the page:

- `Use Defaults`: This button is only active if the current user's settings is not the server defaults. Clicking on this button will erase the user's settings and reset them to the server's defaults.
- `Reset`: This button is only active if some modification has been made to the settings. Clicking on this button will reset the form to its saved state, discarding any change that was made by the user in the settings. The settings will not be changed.
- `Save`: This button is only active if some modification has been made to the settings. Clicking on this button will persist the user's settings, so that they will be taken into account afterwards.

Explorer Filters Default Settings

The default settings of the Explorer Filters can be edited by the administrator in an external file.

This file must be referred to by URL in the `perseus.metaclass.filter.default` property that must be set in the server's `application.properties` settings file.

Example settings for default filter settings file

```
perseus.metaclass.filter.default:file:///d:/metaclass-filter.json
```

After this file has been modified, it must be reloaded for these changes to be taken into account on the server.

Only the *Server Administrator* can reload this settings file.

It can be achieved from the User Settings page of the administrator account, by clicking on the 'Reload Defaults' button. This button is only visible by *Server Administrators*.

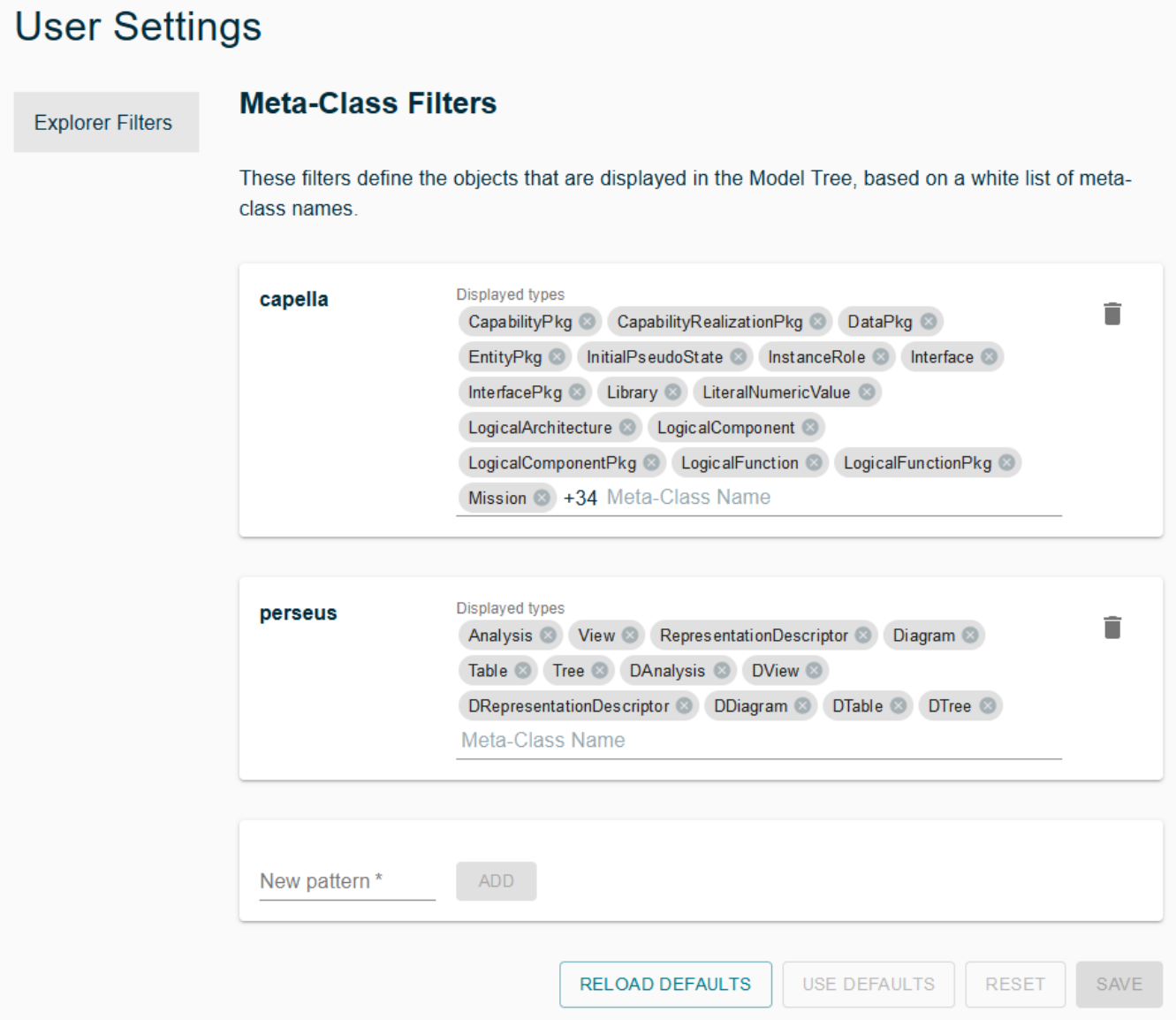


Figure 49. Meta-Class Filter Settings

Jobs Page

As some processes can take a long time to complete, they are launched as jobs on the server. This is the case with model publications for instance. The jobs page displays the list of current and past jobs (all the jobs that are not finished, plus the jobs that finished in the last 7 days). It can be accessed by clicking on the account icon:

Figure 50. Account Icon

Then in the menu, by clicking on "Jobs":

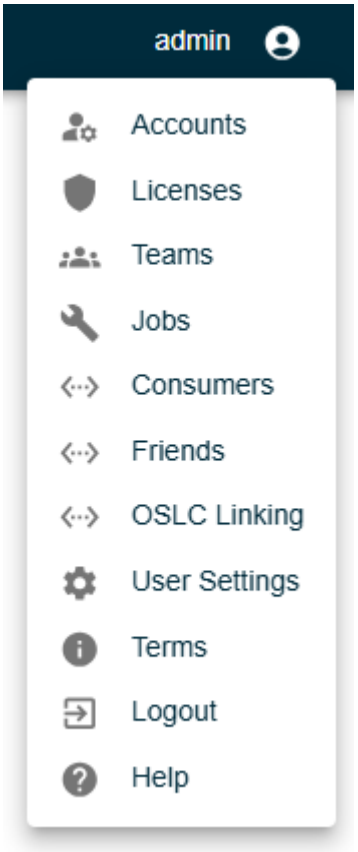


Figure 51. Administrator Menu

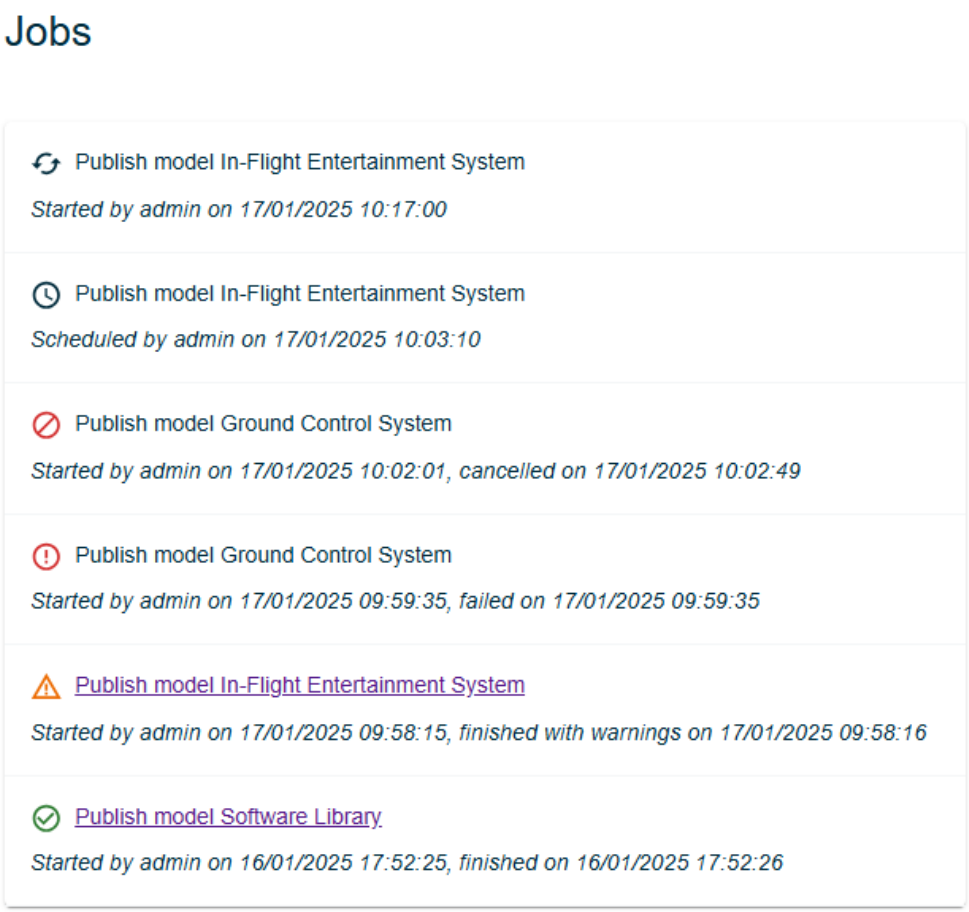


Figure 52. All Kinds of Jobs

For each job, the following information is available:

- The jobs status, as an icon (scheduled, running, complete, warning, failed, canceled);
- The job label, which is a clickable link if the job produced usable results. For instance, a publication job link will have its label linked to the published model;
- The username of the account used to create the job;
- One or two dates:
  - The job scheduling date, if the job has not started yet, or the start date, if the job has begun;
  - The end date, if the job has finished.

A server administrator can view all the jobs, created by any user. A non-administrator user can only view the jobs he or she created.

OAuth Consumers Management

The Publication for Capella server supports OAuth 1.0 and OAuth 1.0a for authentication from third-party servers (like Polarion or IBM Jazz applications). To be able to interact with Publication for Capella, administrators of third-party servers must first issue a request to be associated with the Publication for Capella server (friend request). Administrators of the Publication for Capella server must then accept the friend request.

It is not necessary, and actually impossible, to declare consumers for **Jama**

Connect or **Codebeamer** friend servers.

Managing Consumers and Friend Requests

Administrators (and administrators only) can manage the registered or provisional consumers (friend servers) on a dedicated page. That page can be accessed *via* the administrator user menu, by clicking on the menu item `Consumers`:

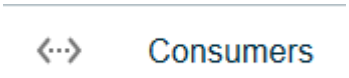


Figure 53. Consumers Menu Item

When a third-party server has issued a friend request and it has not yet been validated by a server administrator, the request is 'provisional', which is materialized by the yellow question mark icon.

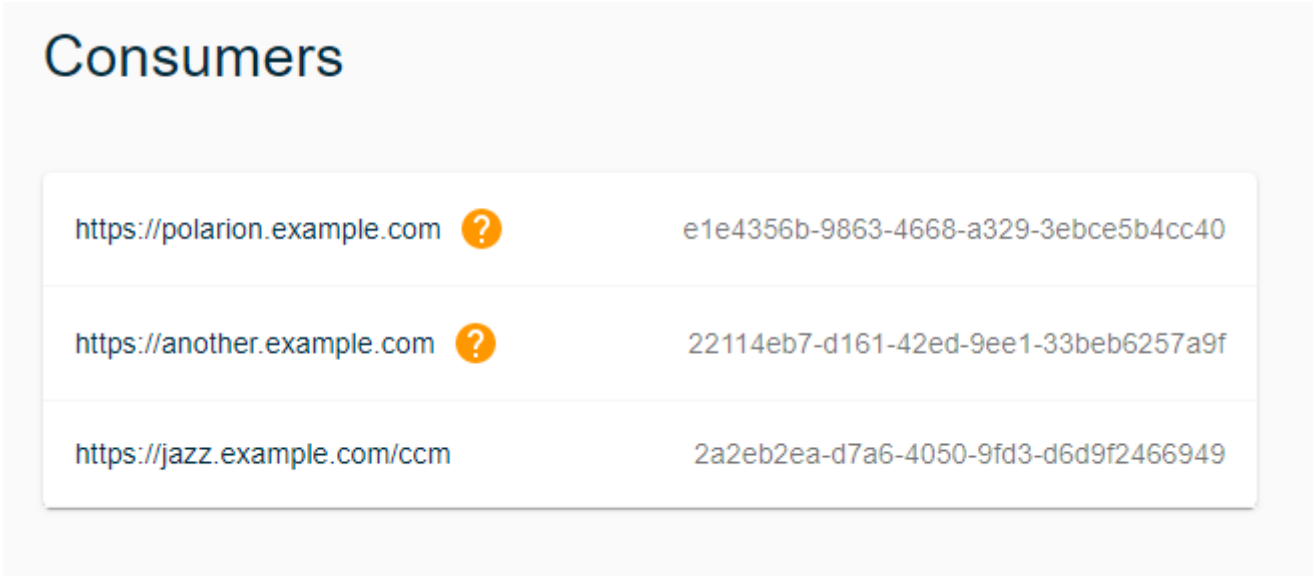


Figure 54. Consumers View

Clicking on a consumer leads to its dedicated consumer page:

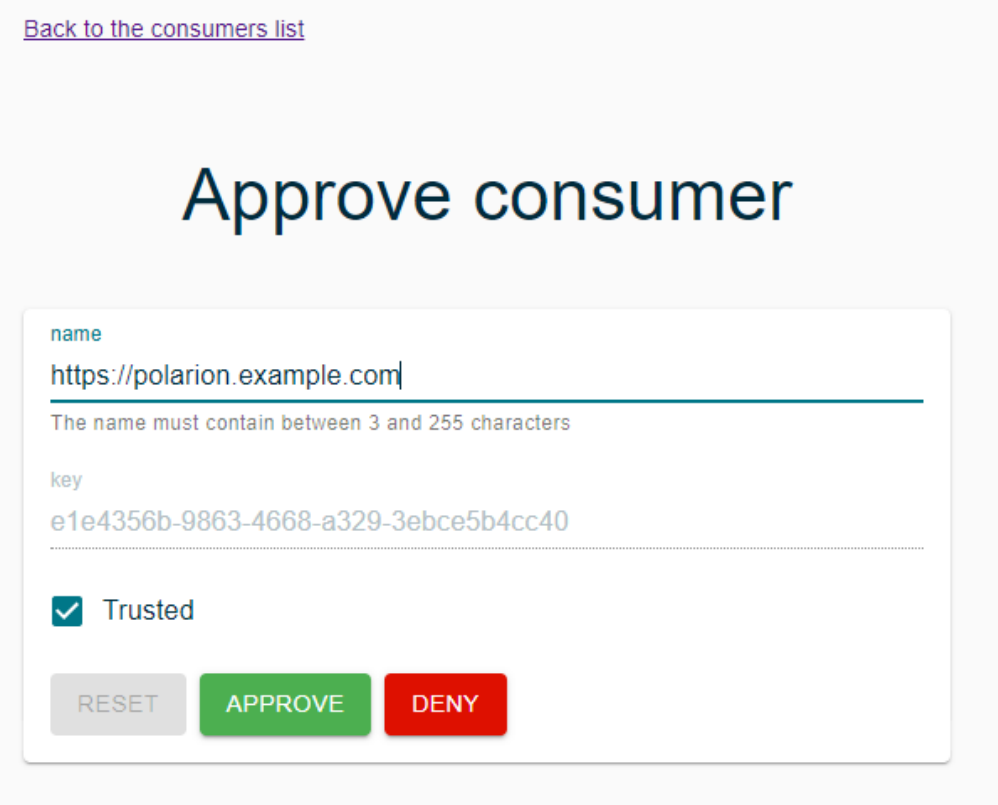


Figure 55. Action Menu for provisional OAuth Consumers

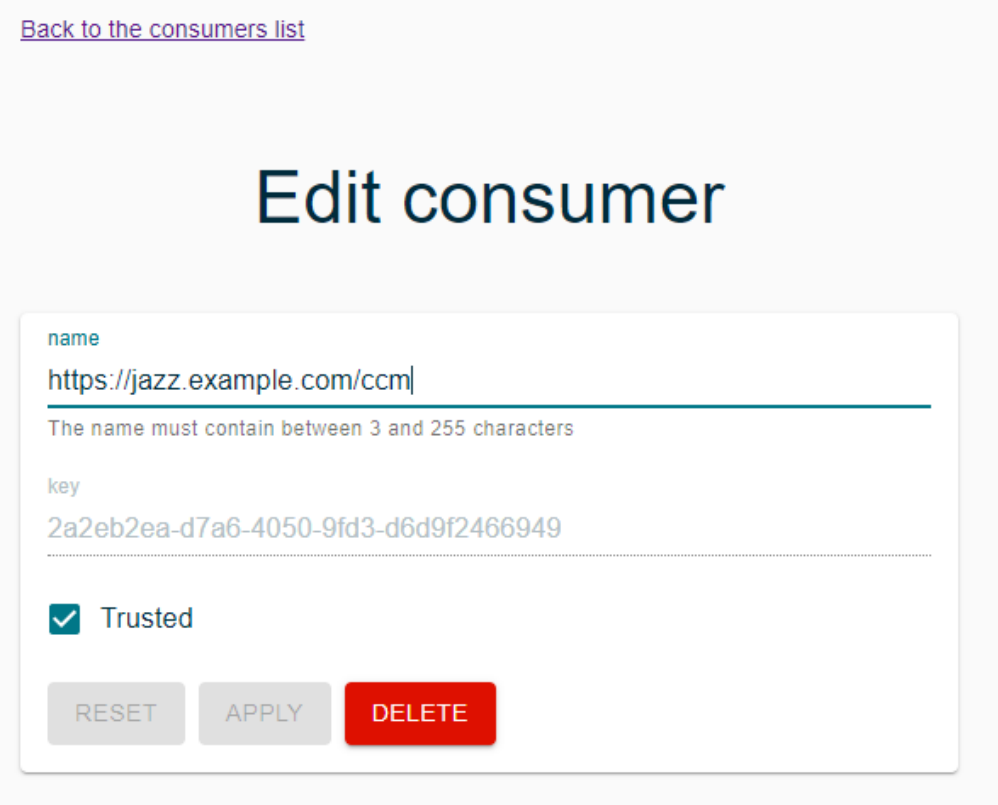


Figure 56. Action Menu for Registered OAuth Consumers

Administrators can edit consumers as follows:

- On provisional consumers (marked with a question mark icon):
  - Approve the consumer — in which case the consumer is marked as approved and can then be used for OSLC scenarios.

- Deny the consumer — in which case the consumer is deleted, and OSLC communication from the requesting server remains not allowed.

- On all consumers:
  - Rename the consumer — so that it can be easily recognized. The new name must then be save with the Apply button.
  - Delete the consumer — This does not warn the consumer's remote server in any way. It does not either delete links associated to this consumer. However, the association between this consumer and this Publication for Capella instance will be broken, and behavior is unspecified on both ends. So, removing a registered consumer should be done with caution, and is generally discouraged in production. However, it can be useful when the associated server is no longer in use, for example.

Connecting to Remote Repositories with OSLC

Publication for Capella is an OSLC *Provider* that exposes model elements using the OSLC AM (Architecture Management) vocabulary.

Publication for Capella is also an OSLC *Consumer* that can connect to third-party repositories using OSLC. It has been tested with Polarion(tm) and IBM DOORS Next(tm).

This section describes how the connection to such third-party repositories is established and used.

Friend Servers

The first operation that needs to be performed to connect to a third-party repository is to emit a so-called 'Friend Request' to this repository.

View the Publication for Capella server's Friends

This page is only accessible to a *Server Administrator*.

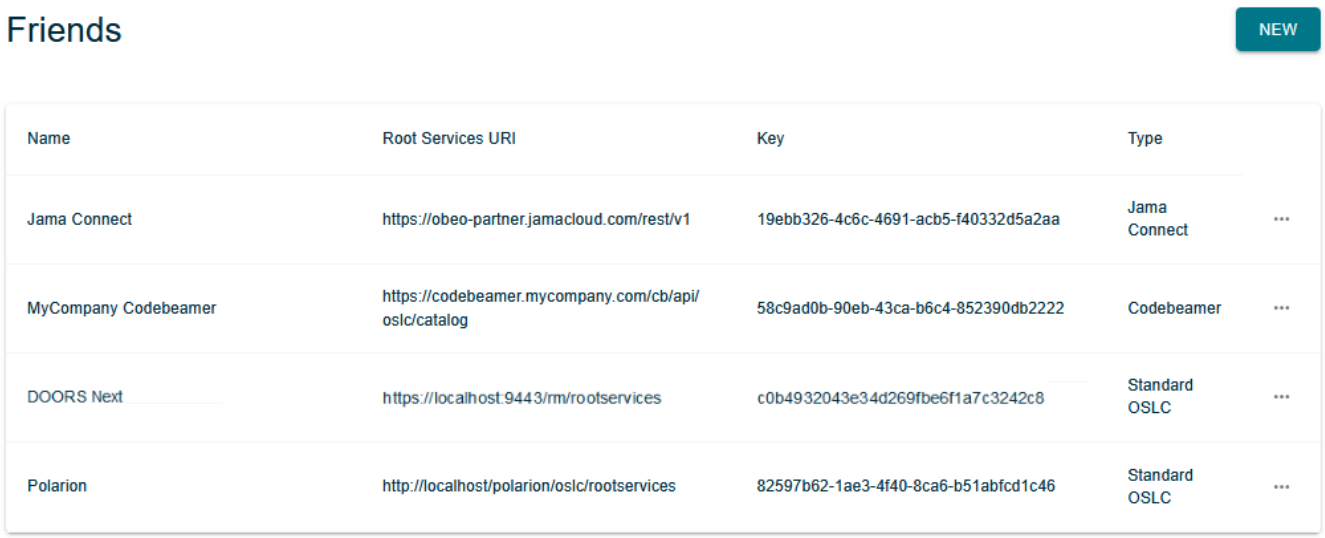


Figure 57. Friend List

The page of the server's Friends display the list of known Friend servers with the following details:

- ***Name***: The user-friendly name of the friend (editable).
- ***Root Services URI***: The URL of the friend, used as the main entry point for all communications with that friend. If the friend is a standard OSLC server, this is its root services URL. Otherwise, please refer to the specific integrations documentation below.
- ***Key***: The key used for authenticating with this friend.
- ***Type***: The type of the friend server, one of:
  - `Standard OSLC (Polarion, IBM Jazz)`: The default value;
  - `Jama Connect`;
  - `Codebeamer`.
- The last column contains actions to rename or delete friend servers.

Create a new Standard Friend

This is the procedure to follow to connect to Polarion, any IBM Jazz application (IBM DOORS Next, IBM EWM, IBM QM, …), Jira, Confluence, or any server that conforms to OSLC.

- **Pre-Requisites**
  - Being logged in as a *Server Administrator*;
  - Navigate to the 'Friends' page;
  - Know the Friend server's OSLC root services URL. That URL depends on the third-party repository you want to connect to, please refer to the relevant documentation of that repository.
- ***Modus Operandi***
  - Click on the button 'New': The OSLC Friend creation dialog is displayed.

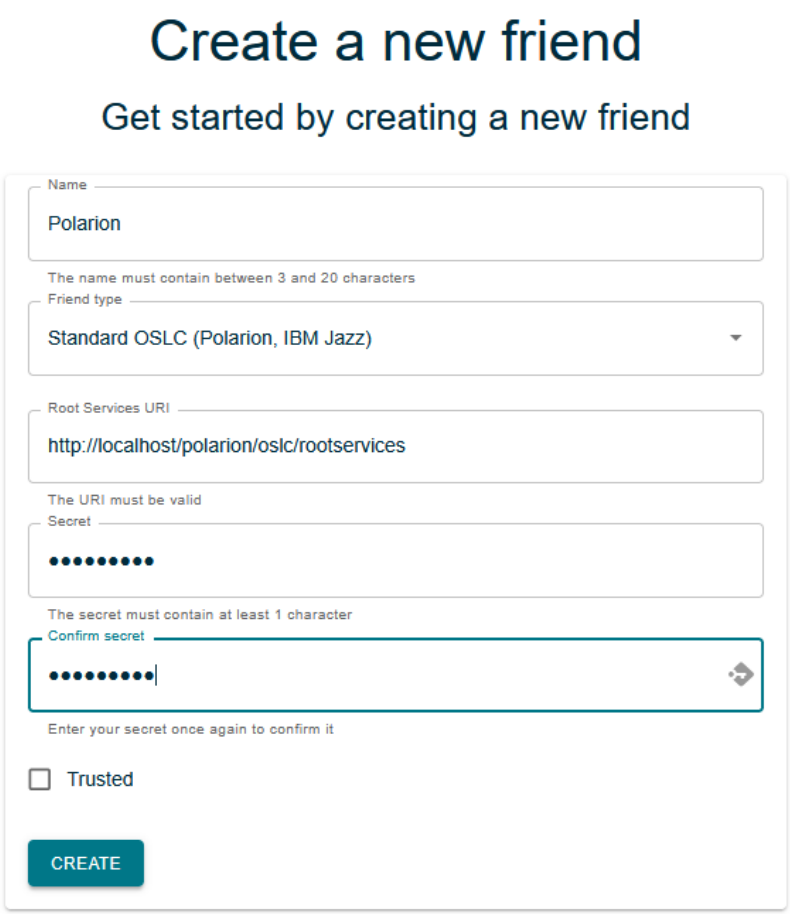


Figure 58. Friend Creation Dialog

- Fill in the form:
  - ***Name***: This is a human-readable name that should make it easy for users to recognize the Friend server.
  - ***Friend type***: Leave it to the default value 'Standard OSLC (Polarion, IBM Jazz)'.
  - ***Root Services URI***: This is the URL of the friend server's root services entry point. That information can only be obtained from the third-party repository's documentation.

- On a standard Polarion server deployed at `https://<polarion.mycompany.com>/polarion`, the `Root Services URI` to use should be: `https://<polarion.mycompany.com>/polarion/oslc/rootservices`.
- For IBM DOORS Next it will `https://<jts.mycompany.com>/rm/rootservices`.
- For IBM EWM it will be `https://<jts.mycompany.com>/ccm/rootservices`.
- On a Jira server deployed at `https://<jira.mycompany.com>` with the required add-on (*OSLC Connect for Jira* by Sodius), the `Root Services URI` to use should be: `https://<jira.mycompany.com>/rest/oslc/1.0/rootservices`
- On a Confluence server deployed at `https://<confluence.mycompany.com>` with the required add-on (*OSLC Connect for Jira* by Sodius), the `Root Services URI` to use should be: `https://<confluence.mycompany.com>/rest/oslc-connect/1.0/rootservices`

- ***Secret***: This should be a complex password, we recommend using at least 12 characters and

mixing lowercase & uppercase letters, digits and special characters.

- ***Confirm secret***: This should be the same as the Secret above.
- ***Trusted***: This is reserved for future use and has no effect.
  - Click on 'Create'
- If successful, the dialog is closed and the friend appears in the list of server friends. In the background, a friend request is issued and sent to the remote server using its root services URL.

This request must be accepted by the third-party repository administrator (or whoever is relevant according to the third-party repository's policy).

- On a standard Polarion server deployed at `https://<polarion.mycompany.com>/polarion`, the URL to use to validate Friends Requests is: `https://<polarion.mycompany.com>/polarion/oslc/services/oauth/approveKey`.
- On IBM Jazz, the consumers should be accepted inside the **server** configuration and "consumers" section.
- Nothing is required in CodeBeamer as authentication is made using HTTP Basic.
- On Jira, the consumers should be accepted from the page `https://<jira.mycompany.com>/plugins/servlet/oslc/consumer`.
- On Confluence, the consumers should be accepted from the page `https://<confluence.mycompany.com>/plugins/oslc-connect/consumers.action`.

- If the root services URL is incorrect, then an error message is displayed on the bottom right of the page.

Create a new Jama Connect Friend

This is the procedure to follow to connect to a *Jama Connect* server.

- **Pre-Requisites**
  - Being logged in as a *Server Administrator*;
  - Navigate to the 'Friends' page;
  - Know the Jama Connect server's Rest API URL.
- ***Modus Operandi***
  - Click on the button 'New': The OSLC Friend creation dialog is displayed.

Select the Friend type 'Jama Connect'.

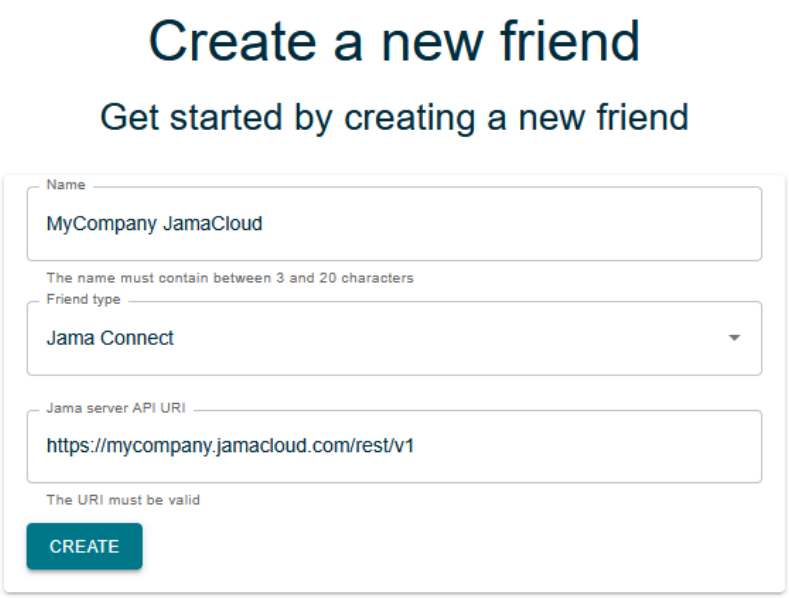


Figure 59. Jama Connect Friend Creation Dialog

- Fill in the form:
  - ***Name***: This is a human-readable name that should make it easy for users to recognize the Jama Connect server.
  - ***Friend type***: Set it to 'Jama Connect'.
  - ***Jama server API URI***: This is the URL of the Jama Connect server's Rest API entry point (Currently, the integration has only been tested with version 1 of the Jama Connect Rest API). For example, `https://<mycompany.jamacloud.com>/rest/v1`.

Configure a Jama Connect Friend Server

Jama Connect Friend server declarations have a specificity: They can be configured.

To access the configuration of a Jama Connect friend server, click on the friend's name in the page View the Publication for Capella server's Friends. The Jama Connect Settings page will open.

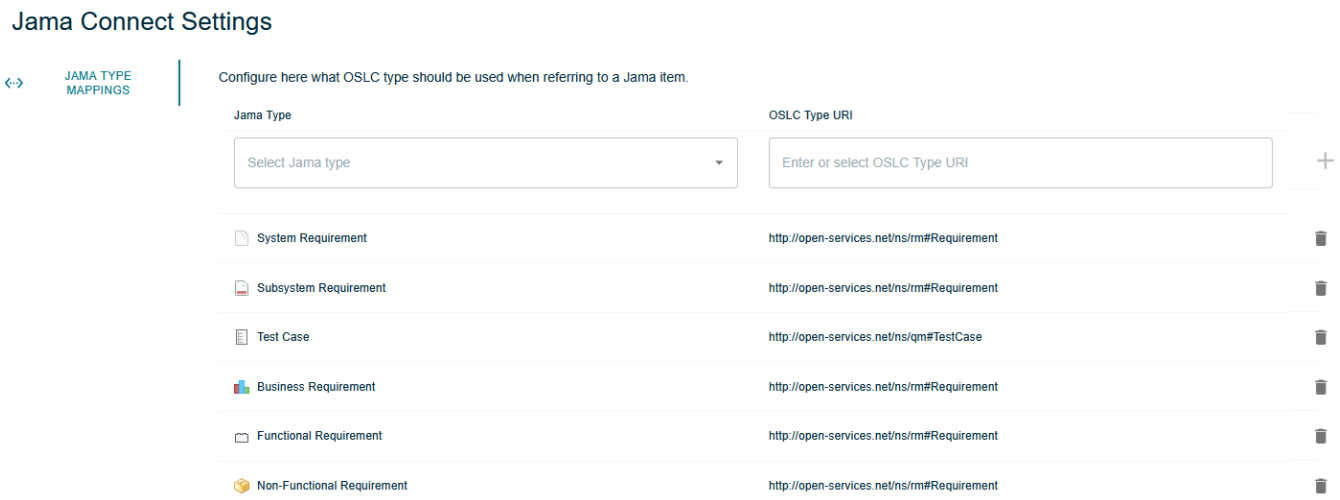


Figure 60. Jama Connect Settings Page

This page makes it possible to configure the mapping of the Jama Connect server's *item types* to OSLC types.

Only the items whose type is mapped to an OSLC type will be recognized by Publication for Capella.

OSLC types are identified by a URI. Standard OSLC types include (non-exhaustive list):

- Requirements Management Domain:
  - `http://open-service.net/ns/rm#Requirement`.
  - `http://open-service.net/ns/rm#RequirementCollection`.
- Quality Management Domain:
  - `http://open-service.net/ns/qm#TestCase`.
  - `http://open-service.net/ns/qm#TestPlan`.
  - `http://open-service.net/ns/qm#TestScript`.
  - `http://open-service.net/ns/qm#TestResult`.
  - `http://open-service.net/ns/qm#TestxecutionRecord`.
- Change Management Domain:
  - `http://open-service.net/ns/cm#ChangeNotice`.
  - `http://open-service.net/ns/cm#ChangeRequest`.
  - `http://open-service.net/ns/cm#Defect`.
  - `http://open-service.net/ns/cm#Enhancement`.
  - `http://open-service.net/ns/cm#Task`.
- Architecture Management Domain:
  - `http://open-services.net/ns/am#Resource`.

However, it is possible to create additional types and use them with Publication for Capella if necessary.

The OSLC types mapped to Jama Connect item types must be defined in the other Publication for Capella settings:

- In the OSLC Linking Configuration.
- In the traceability settings of the Capella models.

To add a mapping:

- Select an item type in the left-hand input.
- Select or type an OSLC type uri in the right-hand input.
- Click on add.

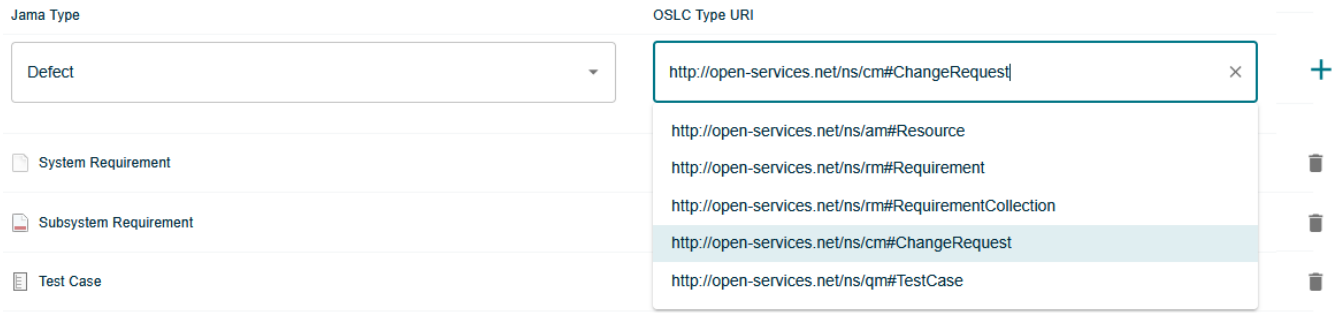


Figure 61. Add a Jama Connect Type Mapping - Step 1

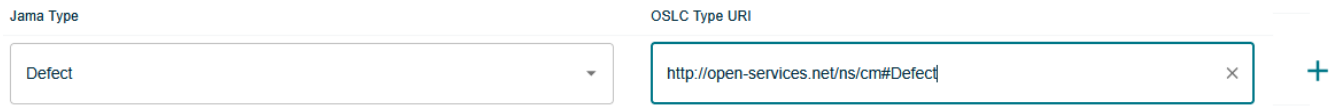


Figure 62. Add a Jama Connect Type Mapping - Step 2

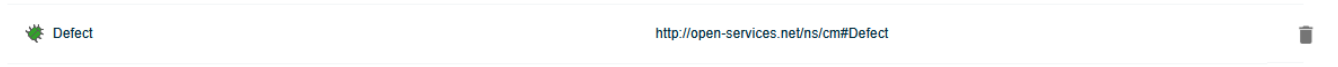


Figure 63. Add a Jama Connect Type Mapping - Step 3

To delete a mapping, click on the trash icon in the mapping line. A confirmation dialog will be displayed.

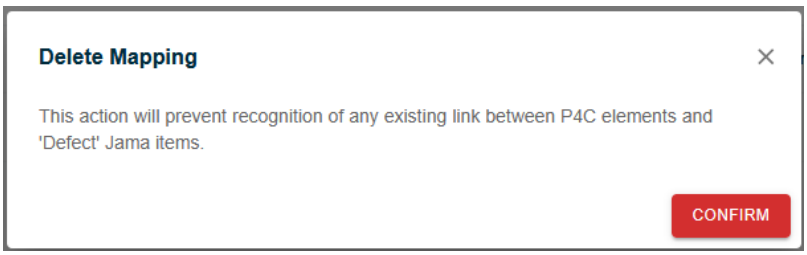


Figure 64. Delete a Jama Connect Type Mapping

Create a new Codebeamer Friend

This is the procedure to follow to connect to a *PTC Codebeamer* server.

- **Pre-Requisites**
  - Being logged in as a *Server Administrator*;
  - Navigate to the 'Friends' page;
  - Know the Jama Connect server's Rest API URL.
- ***Modus Operandi***
  - Click on the button 'New': The OSLC Friend creation dialog is displayed.

Select the Friend type 'Codebeamer'.

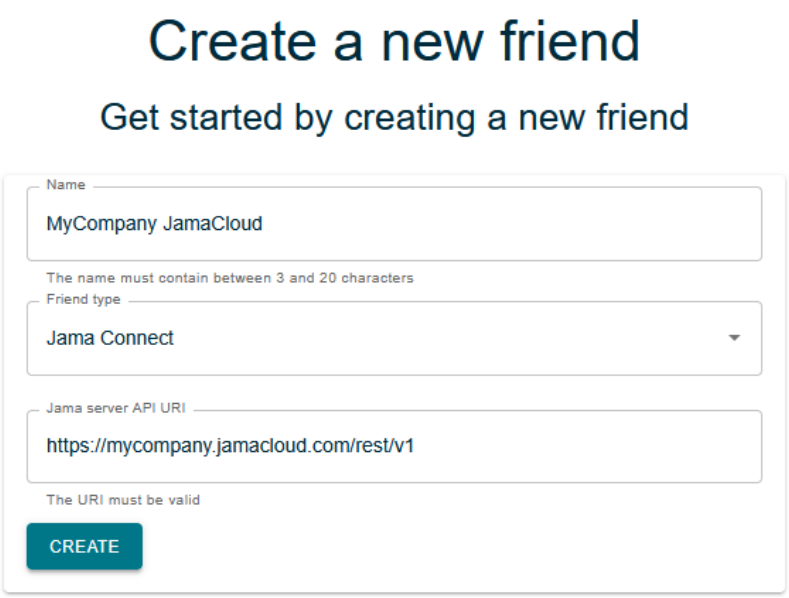


Figure 65. Codebeamer Friend Creation Dialog

- Fill in the form:
  - ***Name***: This is a human-readable name that should make it easy for users to recognize the Codebeamer server.
  - ***Friend type***: Set it to 'Codebeamer'.
  - ***Root catalog URI***: This is the URL of the Codebeamer server's OSLC catalog entry point. For example, on a CodeBeamer server deployed at `https://<codebeamer.mycompany.com>/cb`, the `Root catalog URI` to use should be: `https://<codebeamer.mycompany.com>/cb/api/oslc/catalog`.

Rename a Friend

- **Pre-Requisites**
  - Being logged in as a *Server Administrator*;
  - Navigate to the 'Friends' page;
- ***Modus Operandi***
  - Click on the icon '...' on the line of the Friend to rename, select the 'Rename' action: A renaming dialog is opened.

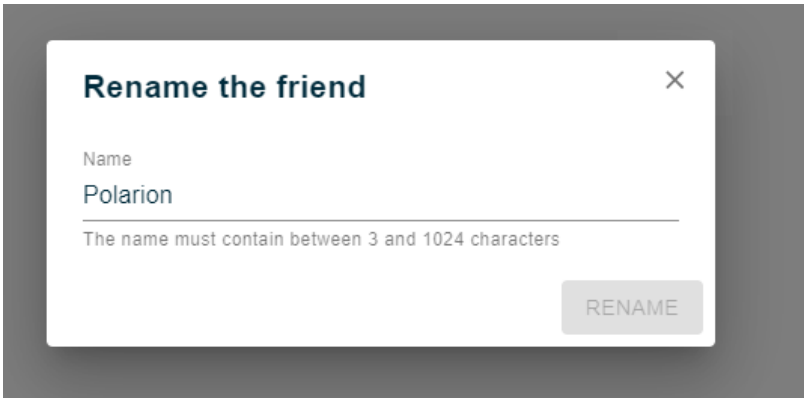


Figure 66. Friend Rename Dialog

Delete a Friend

- **Pre-Requisites**
  - Being logged in as a *Server Administrator*;
  - Navigate to the 'Friends' page;
- ***Modus Operandi***
  - Click on the icon '...' on the line of the Friend to rename, select the 'Delete' action: A confirmation Dialog is opened. The dialog indicates the *OSLC Associations* that will be deleted, if any.

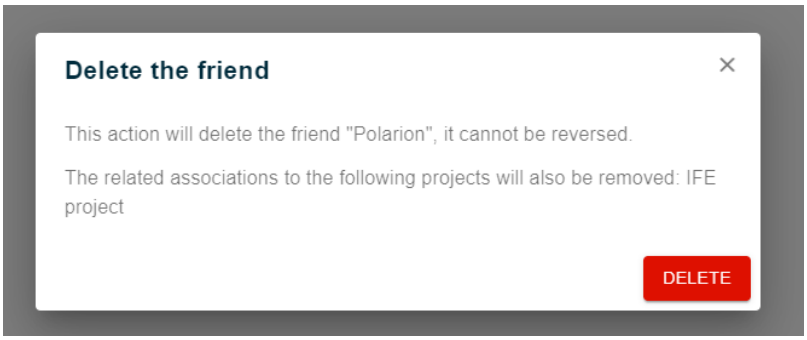


Figure 67. Friend Delete Dialog

OSLC Associations

Once the OSLC 'friendship' is established between the Publication for Capella server and a third-party repository, each *Project Administrator* can define 'OSLC Associations' between their Publication for Capella project and *OSLC Service Providers* of a friend server.

The association configuration page can be found in project settings, under the "Associations" tab:

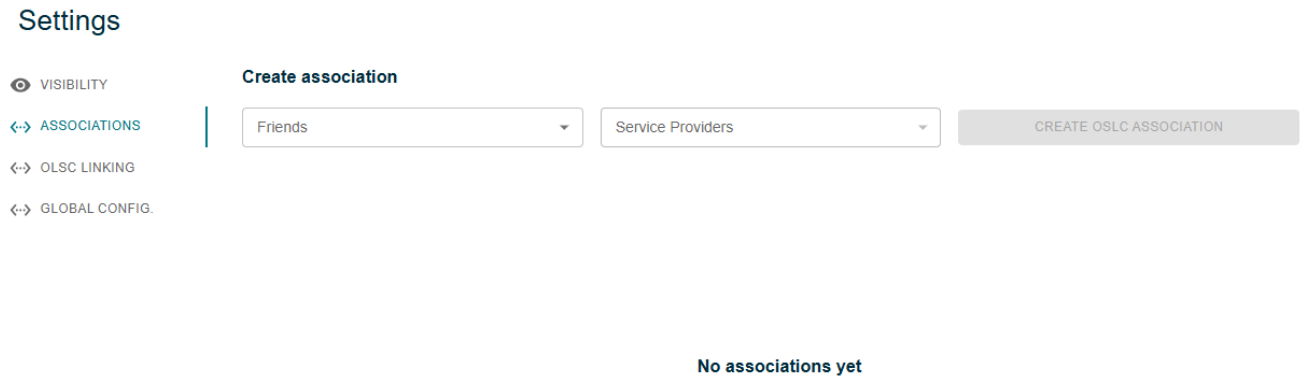


Figure 68. Associations Settings

Once a friend is selected, a service provider can be selected among available ones:

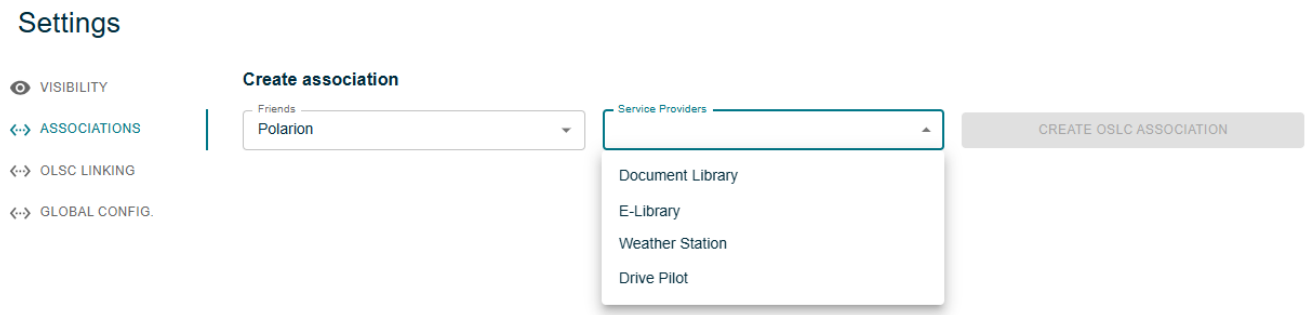


Figure 69. Service Providers List

Clicking on "Create OSLC Association" will add the association to the list. This button is grayed if the association already exists:

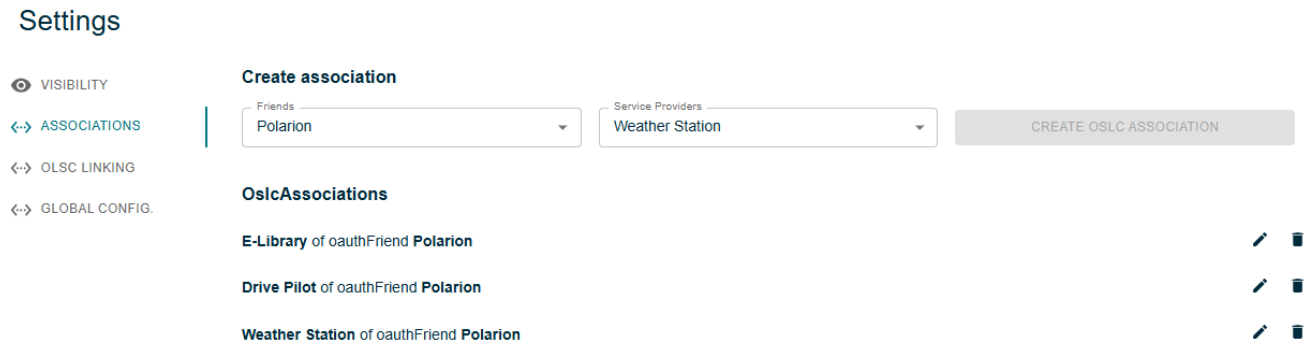


Figure 70. Associations List

An association can be deleted from the list, this will display a warning dialog.

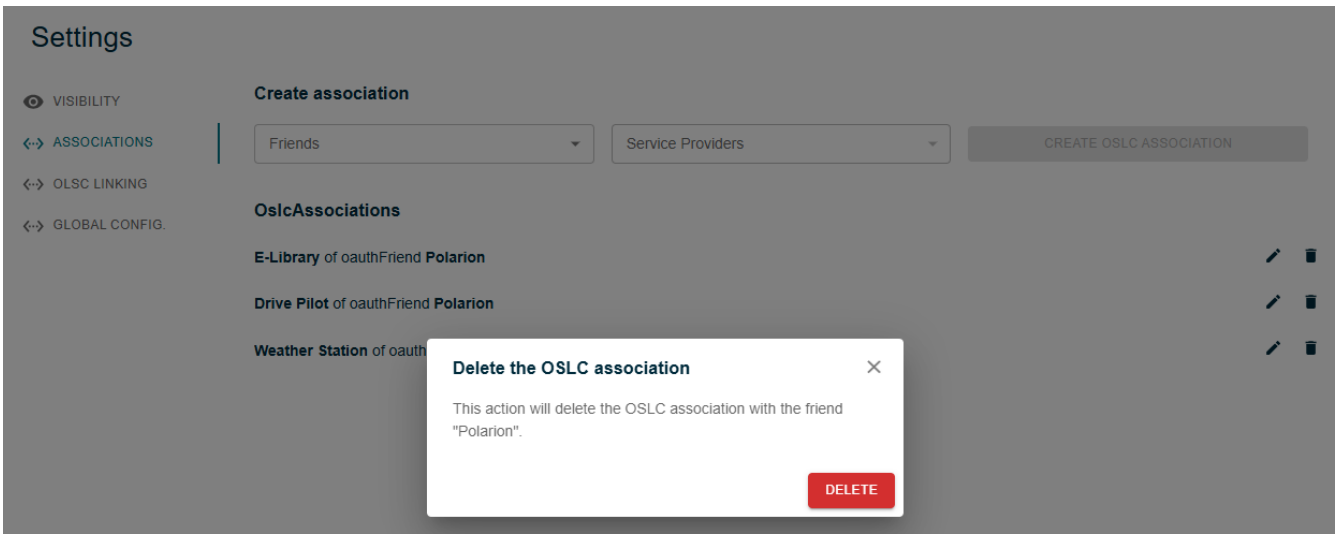


Figure 71. Deleting an Association

An association can be parameterized by clicking on the pen icon next to it. A dialog makes it possible to change the following options:

- "Allow selection dialogs": Unckecking this option will remove the selection dialogs from the list of available dialogs when creating a link.
- "Allow creation dialogs": Unckecking this option will remove the creation dialogs from the list of available dialogs when creating a link.

- "Keep dialogs open after links creation": Unckecking this option will make the link creation dialog close after automatically after use. This option is unchecked by default except when associating with a Polarion server. It is recommended to leave this option unchecked when associating with a third-party that supports multi-selection, such as IBM DOORS Next or Atlassian Jira, for instance.

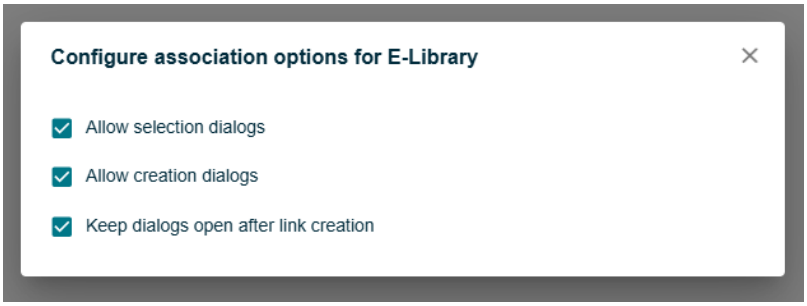


Figure 72. Association Options

OSLC Links

Once at least one OSLC association has been configured on a Publication for Capella project, users that are allowed to *edit* the project can create links to artifacts hosted on the friend repositories.

See sections OSLC Links, Add an OSLC Link, Delete an OSLC Link, or for more details.

OSLC Linking Configuration

The OSLC Linking Configuration makes it possible to configure, to some extent and depending on the third-party repositories capabilities, the kinds of links that can be created to third-party repository objects.

For example, it is possible to allow the creation of links 'satisfies' from Capella *LogicalFunctions* to third party *Requirements*, and of links 'refines' from Capella *Logical Components* to third-party *Requirements*.

Default and Project-Specific Configurations

The *Server Administrator* can configure the default Linking configuration, which will be used by default by all the projects of the server.

Global OSLC Linking configuration

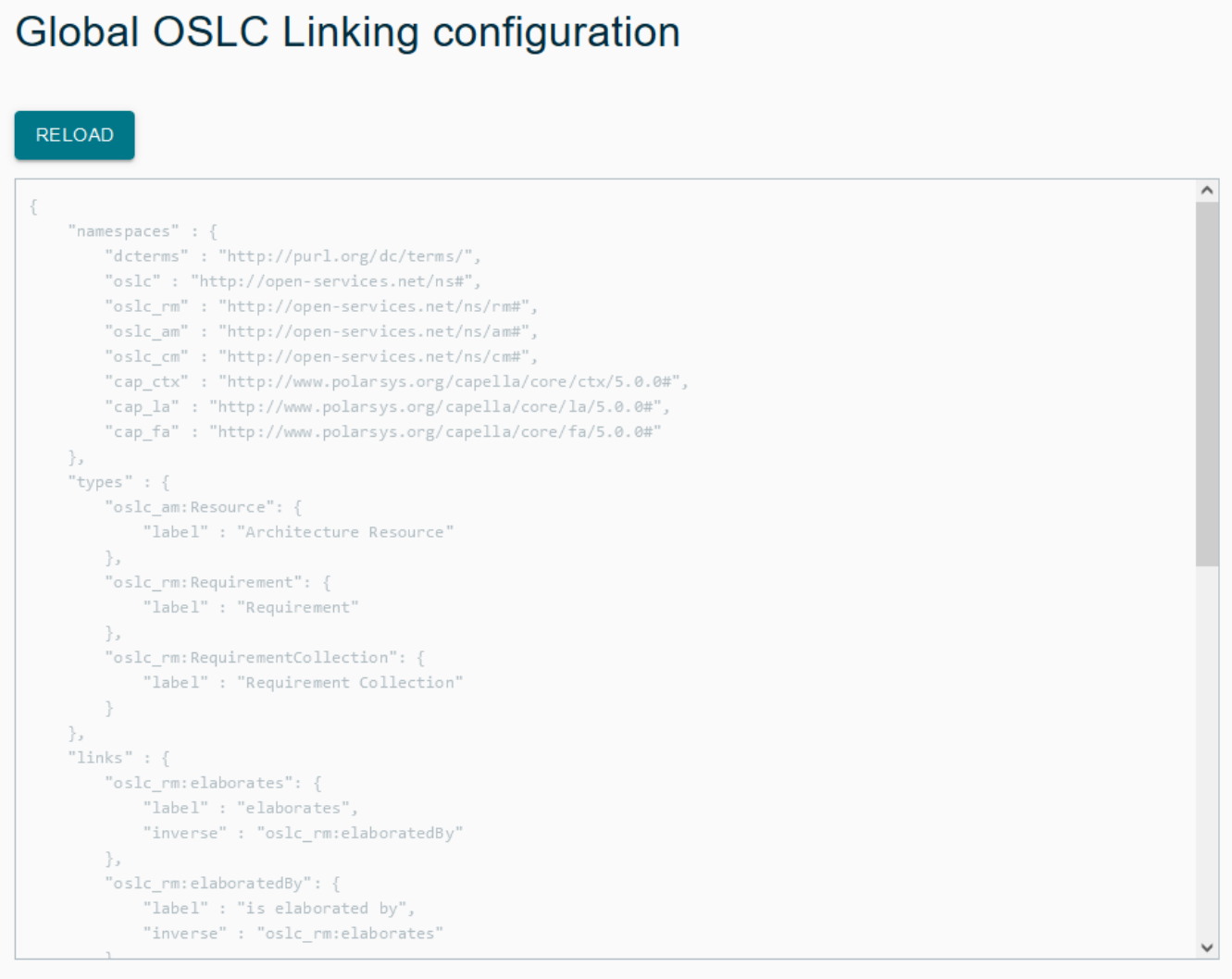


Figure 73. Default Linking Configuration

The *Default Linking Configuration* can be seen by a *Server Administrator* using the administrator user menu and selecting 'OSLC Linking'. To modify it, the *Server Administrator* should manually edit the file containing those configuration then save it. The path and name of this file is defined in `application.properties` file and inside its `perseus.oslc.linking.default` field. Once this file has been modified, the *Server Administrator* must use the 'Reload' button on the default configuration page to reload the contents of the default configuration file so that it is used by the server.

Project-Specific OSLC Linking configuration

Each member of a Project's *Administrator Team* can configure the Project-specific Linking configuration that will be used for this project.

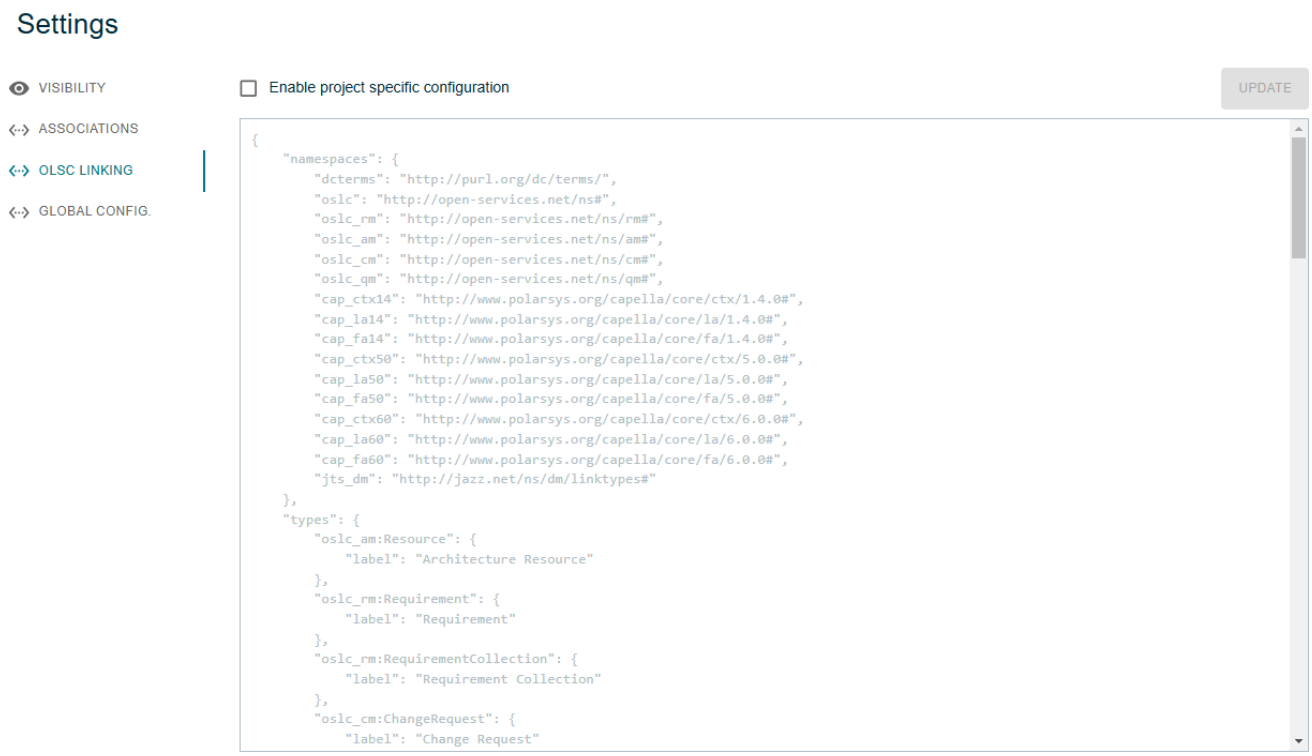


Figure 74. Project Linking Configuration

The *Project Linking Configuration* can be accessed by a *Project Administrator* in the project's settings page.

- Click on the checkbox to enable or disable the project-specific configuration
- If enabled, the configuration can then be edited and saved using the text area in this page, and the 'Update' button.

Configuration Syntax

The configuration of OSLC Linking is specified in JSON format. The example below shows the different concepts that can be used for this configuration.

```
{
    "namespaces" : {
        "dcterms" : "http://purl.org/dc/terms/",
        "oslc" : "http://open-services.net/ns#",
        "oslc_rm" : "http://open-services.net/ns/rm#",
        "oslc_am" : "http://open-services.net/ns/am#",
        "oslc_cm" : "http://open-services.net/ns/cm#",
        "cap_ctx" : "http://www.polarsys.org/capella/core/ctx/5.0.0#",
        "cap_la" : "http://www.polarsys.org/capella/core/la/5.0.0#",
        "cap_fa" : "http://www.polarsys.org/capella/core/fa/5.0.0#"
    },
    "types" : {
        "oslc_am:Resource": {
            "label" : "Architecture Resource"
        },
        "oslc_rm:Requirement": {
```

```
            "label" : "Requirement"
        },
        "oslc_rm:RequirementCollection": {
            "label" : "Requirement Collection"
        }
    },
    "links" : {
        "oslc_rm:elaborates": {
            "label" : "elaborates",
            "inverse" : "oslc_rm:elaboratedBy"
        },
        "oslc_rm:elaboratedBy": {
            "label" : "is elaborated by",
            "inverse" : "oslc_rm:elaborates"
        },
        "oslc_rm:satisfies": {
            "label" : "satisfies",
            "inverse" : "oslc_rm:satisfiedBy"
        },
        "oslc_rm:satisfiedBy": {
            "label" : "is satisfied by",
            "inverse" : "oslc_rm:satisfies"
        },
        "oslc_rm:refines": {
            "label" : "refines",
            "inverse" : "oslc_rm:refinedBy"
        },
        "oslc_rm:refinedBy": {
            "label" : "is refined by",
            "inverse" : "oslc_rm:refines"
        }
    },
    "linking" : {
        "cap_la:LogicalComponent" : {
            "oslc_rm:elaborates": ["oslc_rm:Requirement"],
            "oslc_rm:refines": ["oslc_rm:Requirement", "oslc_am:Resource"]
        },
        "cap_la:LogicalFunction" : {
            "oslc_rm:satisfies": ["oslc_rm:Requirement",
"oslc_rm:RequirementCollection"],
            "oslc_rm:refines": ["oslc_rm:Requirement", "oslc_am:Resource"]
        }
    }
}
```

- ***namespaces***: This is a JSON Object that defines prefixes for URI used in the rest of the document.
  - The `namespace` of the meta-class of a model element hosted in Publication for Capella can be found on this model elements page. Add a `#` character at the end to use it in the OSLC linking configuration.

- ***types***: This is a JSON Object that defines the object types that can be used , and the labels corresponding to these types.
- ***links***: This is a JSON Object that defines the links types that can be used. Each type of link has a 'label' and an 'inverse' that refers to a link type. The 'reverse' link type is used to create a backlink on the Friend server for each link created from the Publication for Capella server, along the lines of the Linked Data philosophy.
- ***linking***: This is a JSON object that defines the types of links that can be created from specific types of model objects stored on the Publication for Capella server to specific types of remote objects on third-party friend repositories.

General Consideration on Configuration Consistency between Repositories

It is important to make sure that the configuration of links in the Publication for Capella server matches the one in the connected repositories.

Especially, it is necessary that the link types and associated reverse link types match. They need to be declared symmetrically in Publication for Capella and in the connected repositories.

For example, Let's assume a link type `oslc_rm:satisfies` is defined in Publication for Capella with a reverse link type of `oslc_rm:satisfiedBy`, and is allowed between Capella Elements and `oslc_rm Requirement` work items exposed by Polarion. Then the Polarion configuration must define a link type `oslc_rm:satisfiedBy` between its work items and `oslc_am:Resource` elements (which are the elements exposed by Publication for Capella via OSLC), with a reverse link type of `oslc_rm:satisfies`.

IBM DOORS Next Specific Configuration

When configuring links to create towards DOORS Next requirements, it is necessary to respect the following constraints:

- In the `namespaces` section, define a namespace prefix for the Jazz-Specific `linktypes` namespace:
  - `jts_dm : "http://jazz.net/ns/dm/linktypes#"`
- In the `links` section, define any of the following link types without any reverse (these are the link types supported by DOORS Next):
  - `jts_dm:trace` (For links labeled as "Traced By Architecture Element" in DOORS Next)
  - `jts_dm:refine` (For links labeled as "Refined By Architecture Element" in DOORS Next)
  - `jts_dm:satisfy` (For links labeled as "Satisfied By Architecture Element" in DOORS Next)
  - `jts_dm:derives` (For links labeled as "Derives Architecture Element" in DOORS Next, the 's' at the end of 'derives' is on purpose)
- In the `linkings` section, assign these link types to relevant object types.

A minimal example of a valid configuration to create links of type `satisfies` to DOORS Next requirements would be:

```
{
```

```
    "namespaces" : {
        "oslc_rm" : "http://open-services.net/ns/rm#",
        "oslc_am" : "http://open-services.net/ns/am#",
        "cap_la" : "http://www.polarsys.org/capella/core/la/5.0.0#",
        "jts_dm" : "http://jazz.net/ns/dm/linktypes#"
    },
    "types" : {
        "oslc_am:Resource": {
            "label" : "Architecture Resource"
        },
        "oslc_rm:Requirement": {
            "label" : "Requirement"
        },
        "oslc_rm:RequirementCollection": {
            "label" : "Requirement Collection"
        }
    },
    "links" : {
        "jts_dm:satisfy": {
            "label" : "satisfies DOORS Next Requirement"
        }
    },
    "linking" : {
        "cap_la:LogicalComponent" : {
            "jts_dm:satisfy": ["oslc_rm:Requirement"]
        },
        "cap_la:LogicalFunction" : {
            "jts_dm:satisfy": ["oslc_rm:Requirement"]
        }
    }
}
```

IBM EWM Specific Configuration

Links that can be created to Change Requests exposed by IBM EWM are constrained. However, Publication for Capella does not currently support dynamically configuring the types of links that can be created based on remote server's configuration.

So, currently, when configuring links to create towards IBM EWM change requests, it is necessary to respect the following steps:

- In the `namespaces` section, define a namespace prefix for the Jazz-Specific `linktypes` namespace:
  - `"jts_dm" : "http://jazz.net/ns/dm/linktypes#"`
- In the `links` section, define the following link type:
  - `jts_dm:elaborates`
    - with the label you wish to use in Publication for Capella,
    - with inverse value `oslc_cm:relatedArchitectureElement` (For links labeled as "Elaborated By Architecture Element" in IBM EWM).

- In the `linkings` section, assign these link types to relevant object types.

Such a configuration makes it possible to create links `jts_dm:elaborates` from Capella elements to EWM change requests, with a reverse link of type `oslc_cm:relatedArchitectureElement` visible in EWM.

A minimal example of a valid configuration to create links to DOORS Next requirements would be:

```
{
    "namespaces" : {
        "oslc_cm" : "http://open-services.net/ns/cm#",
        "oslc_am" : "http://open-services.net/ns/am#",
        "cap_la" : "http://www.polarsys.org/capella/core/la/5.0.0#",
        "jts_dm" : "http://jazz.net/ns/dm/linktypes#"
    },
    "types" : {
        "oslc_am:Resource": {
            "label" : "Architecture Resource"
        },
        "oslc_rm:Requirement": {
            "label" : "Requirement"
        },
        "oslc_cm:ChangeRequest": {
            "label" : "Change Request"
        }
    },
    "links" : {
        "oslc_cm:relatedArchitectureElement": {
            "label" : "Elaborated by Architecture Element",
            "inverse" : "jts_dm:elaborates"
        },
        "jts_dm:elaborates": {
            "label" : "elaborates Change Request",
            "inverse" : "oslc_cm:relatedArchitectureElement"
        }
    },
    "linking" : {
        "cap_la:LogicalComponent" : {
            "jts_dm:elaborates": ["oslc_cm:ChangeRequest"]
        },
        "cap_la:LogicalFunction" : {
            "jts_dm:elaborates": ["oslc_cm:ChangeRequest"]
        }
    }
}
```

IBM ETM Specific Configuration

Links that can be created to Test Cases exposed by IBM ETM are constrained. However, Publication for Capella does not currently support dynamically configuring the types of links that can be

created based on remote server's configuration.

So, currently, when configuring links to create towards IBM ETM Test Cases, it is necessary to respect the following steps:

- In the `namespaces` section, define prefixes for the following namespaces:
  - `"oslc_qm": "http://open-services.net/ns/qm#"`,
  - `"oslc_am": "http://open-services.net/ns/am#"`,
  - `"jts_rqm": "http://jazz.net/ns/qm/rqm#"`
- In the `types` section, define the following artifact type:
  - `"oslc_qm:TestCase": { "label": "Test Case" }`
- In the `links` section, define the following link types:
  - `"oslc_am:validatedBy": { "label": "Validated By", "inverse": "jts_rqm:validatesArchitectureElement" }`,
  - `"jts_rqm:validatesArchitectureElement": { "label": "Validates Architecture Element", "inverse": "oslc_am:validatedBy" }`
  - You can customize the `label` of these types
- In the `linkings` section, assign these link types to relevant object types.

Such a configuration makes it possible to create links `oslc_am:validatedBy` from Capella elements to ETM Test Cases, with a reverse link of type `jts_rqm:validatesArchitectureElement` visible in ETM.

A minimal example of a valid configuration to create links to ETM requirements (from Logical Components and Logical Functions) would be:

```
{
    "namespaces" : {
        "oslc_qm": "http://open-services.net/ns/qm#",
        "oslc_am": "http://open-services.net/ns/am#",
        "jts_rqm": "http://jazz.net/ns/qm/rqm#"
    },
    "types" : {
        "oslc_am:Resource": {
            "label" : "Architecture Resource"
        },
        "oslc_qm:TestCase": {
            "label": "Test Case"
        }
    },
    "links" : {
        "oslc_am:validatedBy": {
            "label": "Validated By",
            "inverse": "jts_rqm:validatesArchitectureElement"
        },
        "jts_rqm:validatesArchitectureElement": {
            "label": "Validates Architecture Element",
```

```
            "inverse": "oslc_am:validatedBy"
        }
    },
    "linking" : {
        "cap_la:LogicalComponent" : {
            "oslc_am:validatedBy": ["oslc_qm:TestCase"]
        },
        "cap_la:LogicalFunction" : {
            "oslc_am:validatedBy": ["oslc_qm:TestCase"]
        }
    }
}
```

Jira Specific Configuration

The following indications are only valid for use with the Jira add-on OSLC Connect for Jira by Sodius.

Jira exposes its artifacts as **Change Requests** form the standard OSLC Change Management vocabulary (`oslc_cm:ChangeRequest`).

The only link type supported between Publication for Capella and Jira is `jts_dm:elaborates`, with an inverse type of `oslc_cm:relatedArchitectureElement`.

A minimal example of a valid configuration to create links to Jira Change Requests (from e.g. Logical Components and Logical Functions) would be:

```
{
    "namespaces" : {
        "dcterms" : "http://purl.org/dc/terms/",
        "oslc" : "http://open-services.net/ns#",
        "oslc_am" : "http://open-services.net/ns/am#", ①
        "oslc_cm" : "http://open-services.net/ns/cm#", ②
        "cap_ctx" : "http://www.polarsys.org/capella/core/ctx/7.0.0#", ③
        "cap_la" : "http://www.polarsys.org/capella/core/la/7.0.0#", ③
        "cap_fa" : "http://www.polarsys.org/capella/core/fa/7.0.0#", ③
        "jts_dm" : "http://jazz.net/ns/dm/linktypes#" ④
    },
    "types" : {
        "oslc_am:Resource": { ⑤
            "label" : "Architecture Resource"
        },
        "oslc_cm:ChangeRequest": { ⑥
            "label" : "Change Request"
        }
    },
    "links" : {
        "jts_dm:elaborates": { ⑦
            "label" : "elaborates",
```

```
            "inverse" : "oslc_cm:relatedArchitectureElement" ⑧
        },
        "oslc_cm:relatedArchitectureElement": { ⑨
            "label" : "is elaborated by",
            "inverse" : "jts_dm:elaborates"
        }
    },
    "linking" : {
        "cap_la:LogicalComponent" : {
            "jts_dm:elaborates": ["oslc_cm:ChangeRequest"] ⑩
        },
        "cap_la:LogicalFunction" : {
            "jts_dm:elaborates": ["oslc_cm:ChangeRequest"] ⑩
        }
    }
}
```

① Prefix for the **Architecture Management** Domain (for Publication for Capella model objects)

② Prefix for the **Change Management** Domain (for Jira items)

③ Prefixes for some Capella 7 meta-models (there could be more if necessary)

④ Prefix for Jazz Team Server Domain, which is used by Jira as well

⑤ Artifact type for Architecture resources exposed by Publication for Capella

⑥ Artifact type for items exposed by Jira

⑦ Link type required by Jira for links from Publication for Capella model objects to Jira items

⑧ And the associated back-link type, required to create the back-links on Jira items

⑨ Declaration of the back-link type

⑩ Allow links with type `jts_dm:elaborates` from Publication for Capella elements to Jira items

Authentication with OAuth 1

Publication for Capella supports OAuth 1.0 to authenticate users with Friend servers (except CodeBeamer, which does not support OAuth 1.0 and with which authentication is made using HTTP Basic).

OAuth 1.0 is an authorization framework that makes it possible to have Publication for Capella communicate with third-party servers, on behalf of a user, without having to know the user's login and password on the friend server.

This authentication can be triggered on different occasions, whenever a user action in the Publication for Capella web site requires authenticated communication with a friend server.

A pop-up frame will be displayed to the user. This frame embed a page from the Friend server that will ensure that the user is properly authenticated on the remote server and allows Publication for Capella to access the friend server on their behalf.

On the user has confirmed that he or she agrees on Publication for Capella accessing the friend

server on their behalf, an access token is securely generated and will be used for further communication between the Publication for Capella server and the friend server. Depending on the friend server's implementation, this access token might have a limited duration. As long a this token is valid, the user will be able to access the friend server from the Publication for Capella server. As soon as this token in invalidated, for whatever reason, the OAuth authentication process described above will be triggered again.

The following figures show how this authentication looks like when accessing a Polarion server from Publication for Capella.

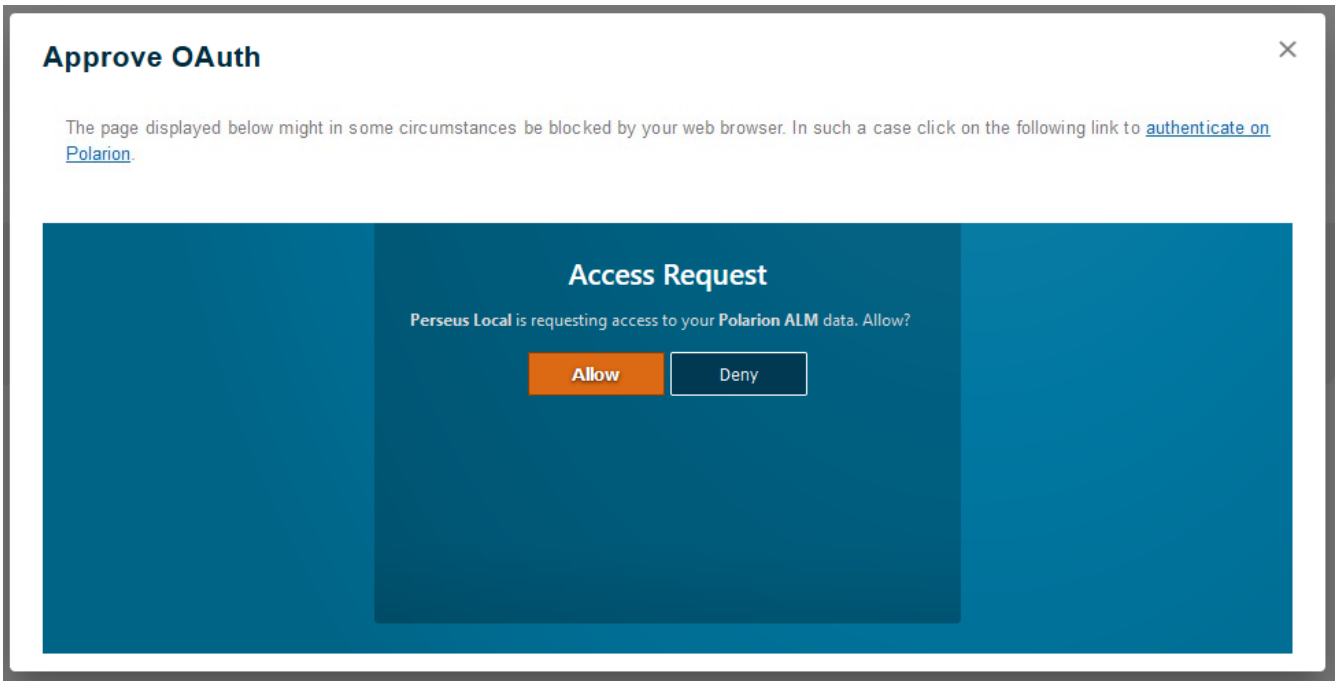


Figure 75. Polarion Requests User Confirmation

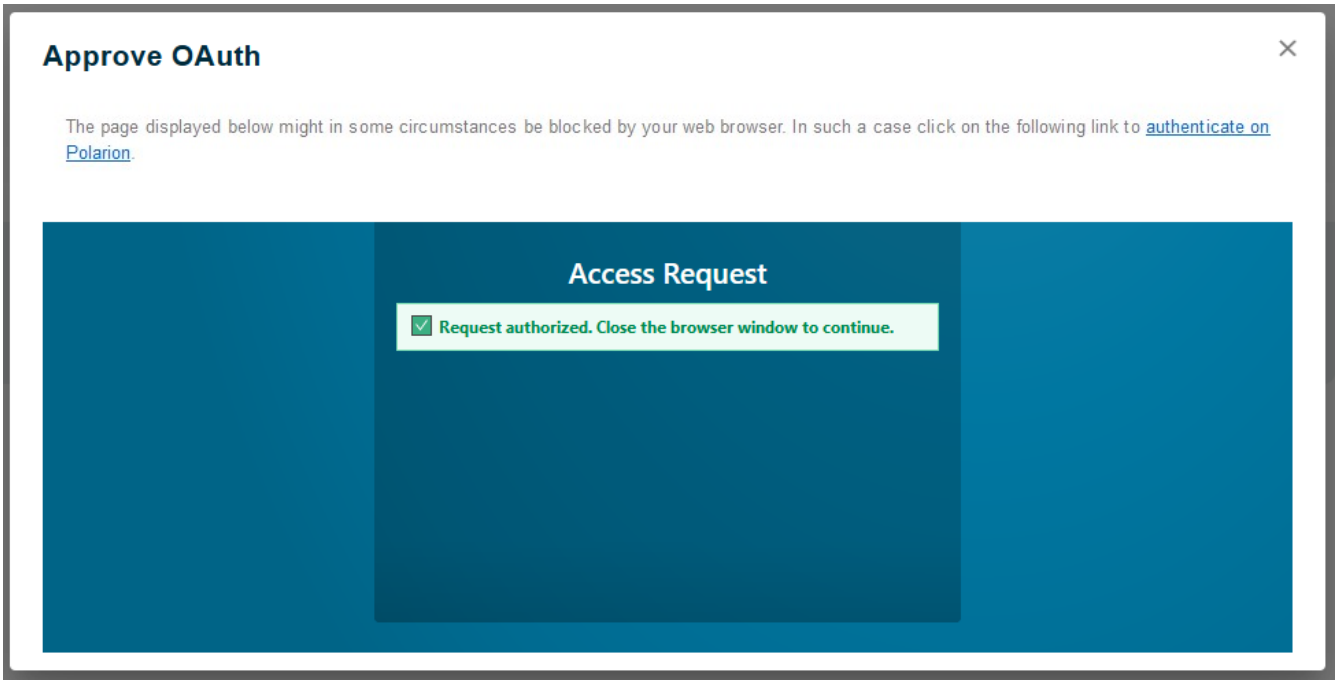


Figure 76. Polarion Accepts User Confirmation

In some cases, the OAuth user interface might be blocked by the web browser. If that happens, the user must click on the dedicated link to open the OAuth user

interface directly within the Friend server's website instead of within an iframe in the Publication for Capella website. The user will then be able to confirm approval, and Publication for Capella will then be able to securely communicate with the Friend server on behalf of the user.

Global Configuration Support

Publication for Capella supports the connection to Jazz Team Server projects that use the *Global Configuration* mechanism.

The support of *Global Configuration* in Publication for Capella is, for the time being, limited. Specifically, we support Scenario Two of the 5 scenarios described by IBM for integrating with *Global Configuration*. We plan to enrich this support in the next versions.

Server Set-Up

To use *Global Configuration*, Publication for Capella must be associated with a friend server that provides Global Configurations. A project will then require an association to a *Global Configuration* service provider of this friend server.

On the friend side, Global Configuration must be activated and at least one configuration must exist.

Activating *Global Configuration* on a Project

In Publication for Capella, the use of *Global Configuration* is decided on a per-project basis. Deciding to use *Global Configuration* on a Publication for Capella Project is a final decision, it cannot be reverted afterwards.

Impacts of Global Configuration

Activating *Global Configuration* on a Project will change the behavior of Publication for Capella for this specific project:

- Links will no longer be bi-directional. Publication for Capella will own the links it creates, and the associated back-links will not be created in the third-party artifacts.
- Links can only be created in the context of a specific *Global Configuration*. The *Global Configuration* must be set on every model for which traceability links are desired.

To activate the use of *Global Configuration* on a project:

- Go to project settings;
- In the *Overview* section, click on the *Activate Global Configuration*;
- Confirm.

Global configuration cannot be activated for a project if active OSLC links are already in use in any of the project models.

Figure 77. Activate Global Configuration

Once activated, the button is no longer available and a message indicates that *Global Configuration* has been activated for this project. A subtitle appears under the project name:

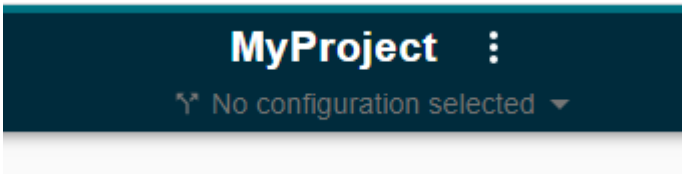


Figure 78. Global Configuration Activated

This action cannot be reverted.

Assigning a *Global Configuration* to a Model

For the time being, each model should correspond to a single stream of work for this model.

Once the Global Configuration mode has been activated for a project, the models it contains can be assigned a *Global Configuration* as follows:

- Click on the model in the Explorer;
- In the model page, click on *Assign Global Configuration*.

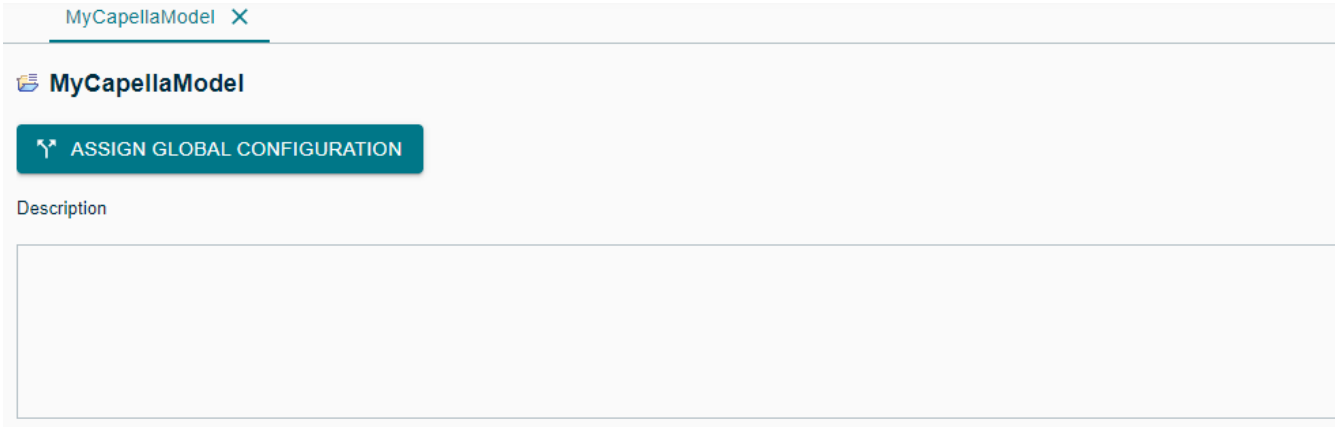


Figure 79. Assign Model Configuration

- A dialog makes it possible to chose the Global Configuration Provider among the configured OSLC associations. If there is only one possible choice the OSLC remote dialog will automatically

be loaded.

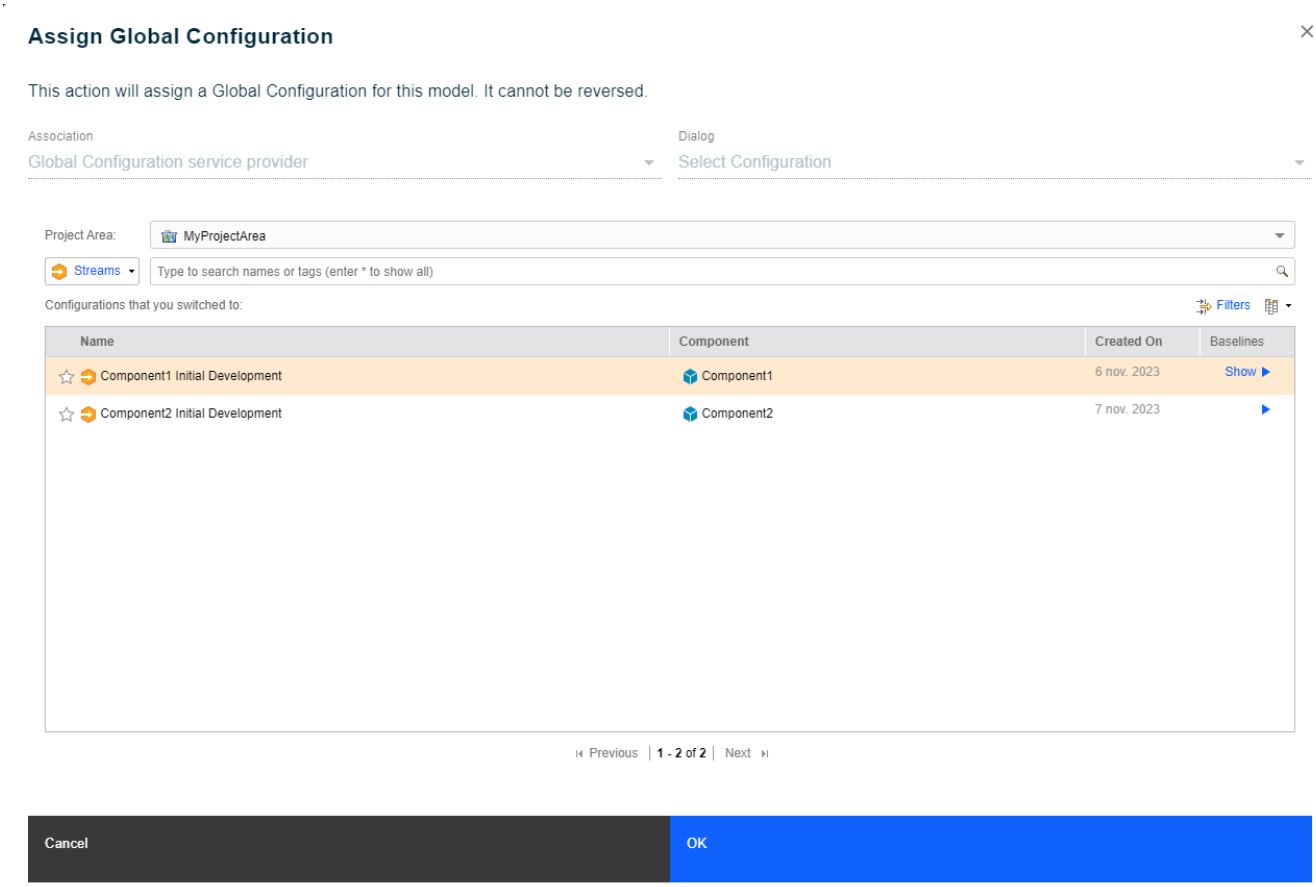


Figure 80. Configuration Assigned

- Select a *Global Configuration* then click *OK*;
- Confirm.

This action cannot be reverted.

Once assigned, any OSLC link created on any of this model's objects will use the configuration to retrieve the remote element data:

- Label & icon of the link will match the remote version related to the assigned configuration.
- Clicking on a link will lead to the remote page within the assigned configuration.
- Once an *Update from server* action has been performed, a Publication for Capella client synchronized with the model will see the data available within the assigned configuration:
  - Refreshing artifacts will fetch the remote version within the assigned configuration.
  - Only artifacts available within the assigned configuration can be dropped within the model.

Filtering a Project by *Global Configuration*

The project can be filtered to only display information related to a specific *Global Configuration* as follows:

- Click on the label under the project title, which shows *No configuration selected* by default.
- This will open the configuration selection dialog.

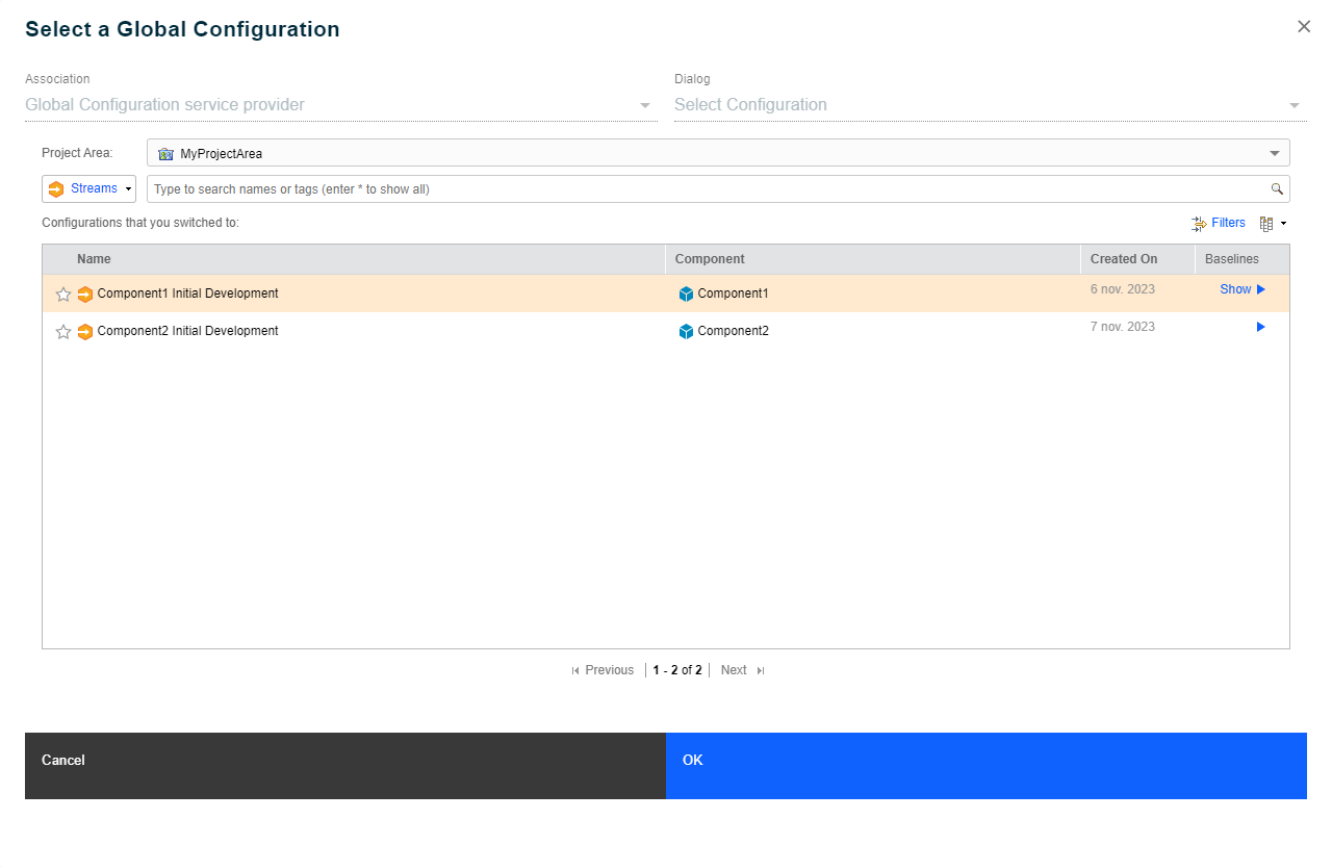


Figure 81. Configuration Selection Dialog

- Once a configuration is selected, its title appears under the project name.
- The filter can be reset by clicking on the cross icon on the right of the configuration title.

Figure 82. Configuration Selected

Traceability Support

The version 2023.3.0 introduces the capability to manage traceability links to third-party artifacts directly from within Capella.

Such traceability links are stored in a dedicated model tied to the Capella model of interest.

The primary means of creating such links is by drag & drop. Just dragging an artifact URL from your browser into the Capella will create a link between a model element and this artifact. Such links are stored in the model. The traceability information is thus available within the model in the modeling workbench.

This traceability information is published to the Publication for Capella server whenever a

publication takes place, along with the rest of the model. It is then available in the usual 'OSLC Links' section, described in OSLC Links.

For more information about the management of such links in Capella, refer to the Publication for Capella user manual, chapter 'Using Publication for Capella Contributor Client'.

Traceability Maintenance

Some operations can cause links to become technically invalid. That is the case if the remote server does not respond properly to a link change request, whatever the reason (the server could be in maintenance, or overloaded, or anything else).

This section describes the maintenance tools that can be used to act and hopefully repair such invalid links.

Fixing Invalid Links

Creating a link to a third-party artifact leads to the creation of the matching reverse link on the remote friend, as deleting a link in Publication for Capella will delete that reverse link on the remote friend.

However, creation and deletion of links to an OSLC friend depends on the availability of said friend so in some cases, a link can be created/deleted locally in Publication for Capella but the related remote operation can fail.

A model with invalid links will appear with a Warning icon in the Explorer.

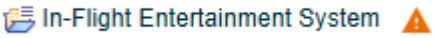


Figure 83. Invalid links model warning

A Publication for Capella server administrator or a project contributor can fix such situations with the help of a dedicated panel that is available on the model page.

This invalid links panel will only appear if there are invalid links in the database.

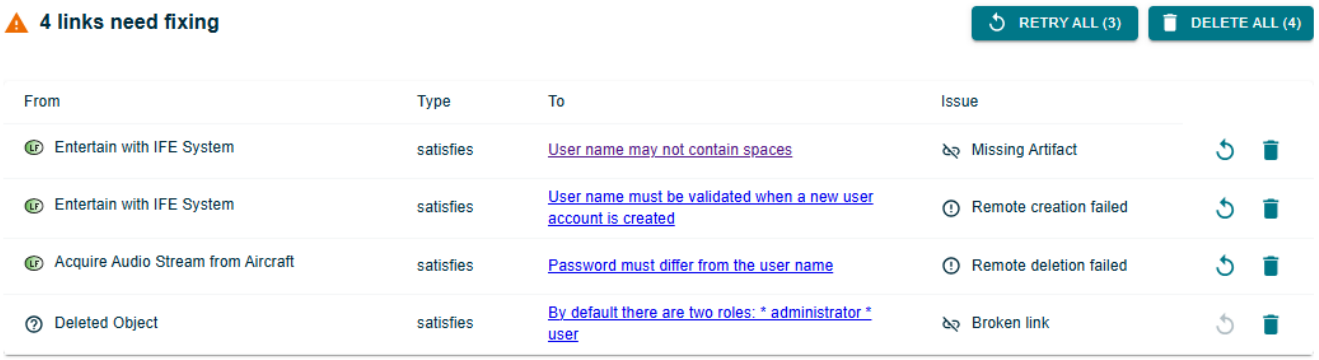


Figure 84. Invalid Links Panel

Several kinds of issues can make a link invalid:

- **Missing artifact:** the target of the link has been deleted.

- **Broken link:** the source of the link has been deleted.
- **Remote creation failed:** the creation of the reverse link on the remote friend was attempted but failed with an error.
- **Remote deletion failed:** the deletion of the reverse link on the remote friend was attempted but failed with an error.

For each link the administrator can:

- Retry the operation: either retry to create or delete the reverse link.
- Delete the link so it will no longer appear in the incomplete link interface. Before deleting an invalid link, the administrator should make sure the remote backlink, if it exists, is properly deleted as well.

The "Retry" action can require an OAuth authentication which will be proposed by a dialog, the operation will be cancelled and need to be retried afterwards.

If the link is broken, "Retry" will be disabled unless it is in *Remote deletion failed* state.

"Retry all" and "Delete all" actions will apply the same fix for all links, any error or OAuth authentication request that happens during that process will halt it.

Known Limitations - Web Client

- The server does not support pagination of responses. This may cause performance or UI issues in some circumstances.
- Tabular representations (tables and cross-tables) are not supported. The only representations that can currently be displayed are diagrams.
- Object links in diagrams don’t support navigation by clicking on them.
- The support of the OSLC Query API is partial, despite improvements made in version 2023.8.0.
- CodeBeamer Integration Limitations:
  - CodeBeamer has no delegated dialogs for creating links. Consequently, it is not possible to create links to CodeBeamer artifacts online, it has to be done in Capella.
  - CodeBeamer cannot display links stored in an associated Publication for Capella server.

Using Publication for Capella Contributor Client

Pre-requisites

- The Publication for Capella *Contributor Client* must be installed in Capella - Refer to the 'Publication for Capella Installation Guide'.
- It is compatible with Capella versions between 1.4.0 and 7.0.

Contributor Client Settings

Publication Servers

One or several *Publication for Capella Servers* must be registered, with specific authentication policies.

The settings are made in the Publication Preference Page.

- Go to *Window* > *Preferences* and select the 'Publication' entry:

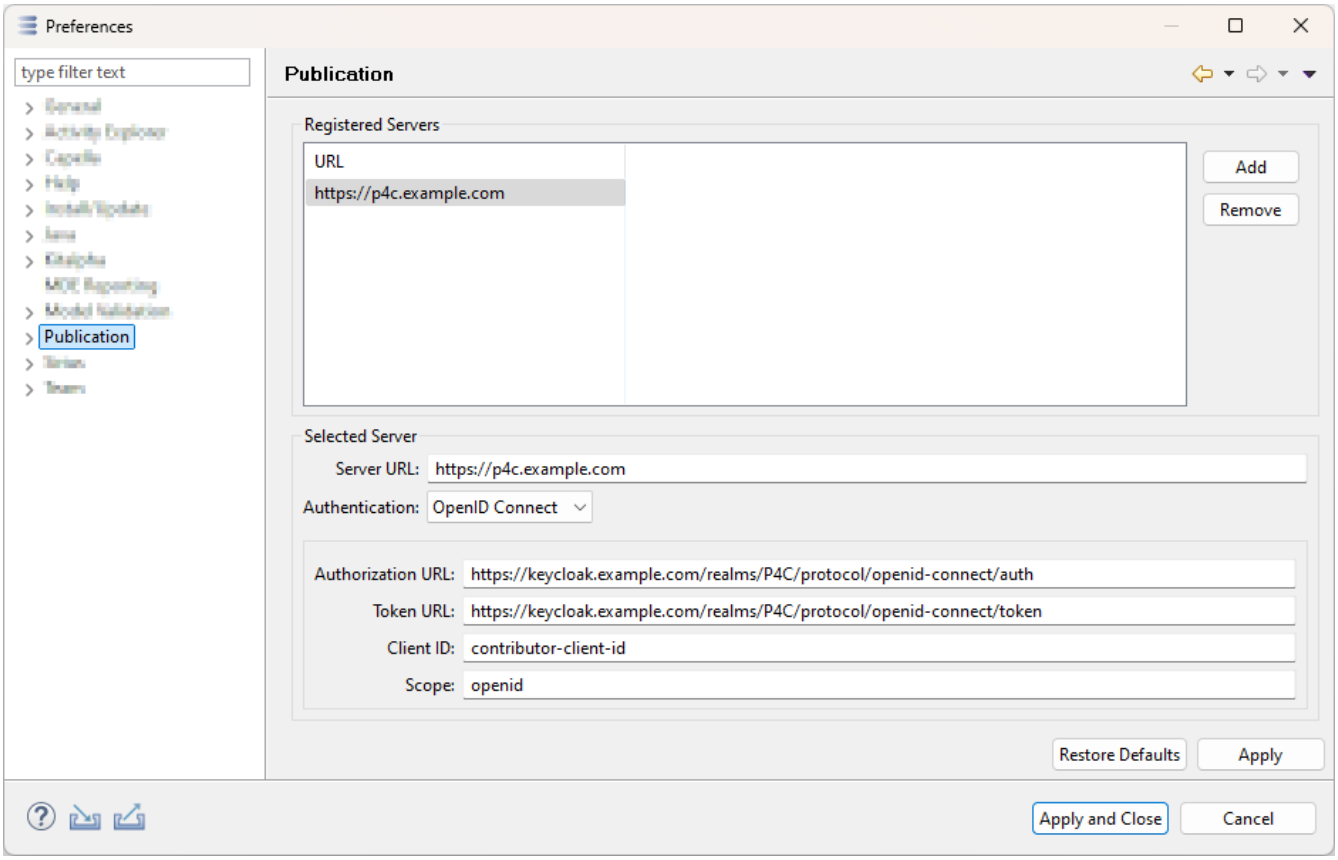


Figure 85. Publication for Capella Preference Page

- Click on 'Add' to register a new *Publication for Capella Server*:

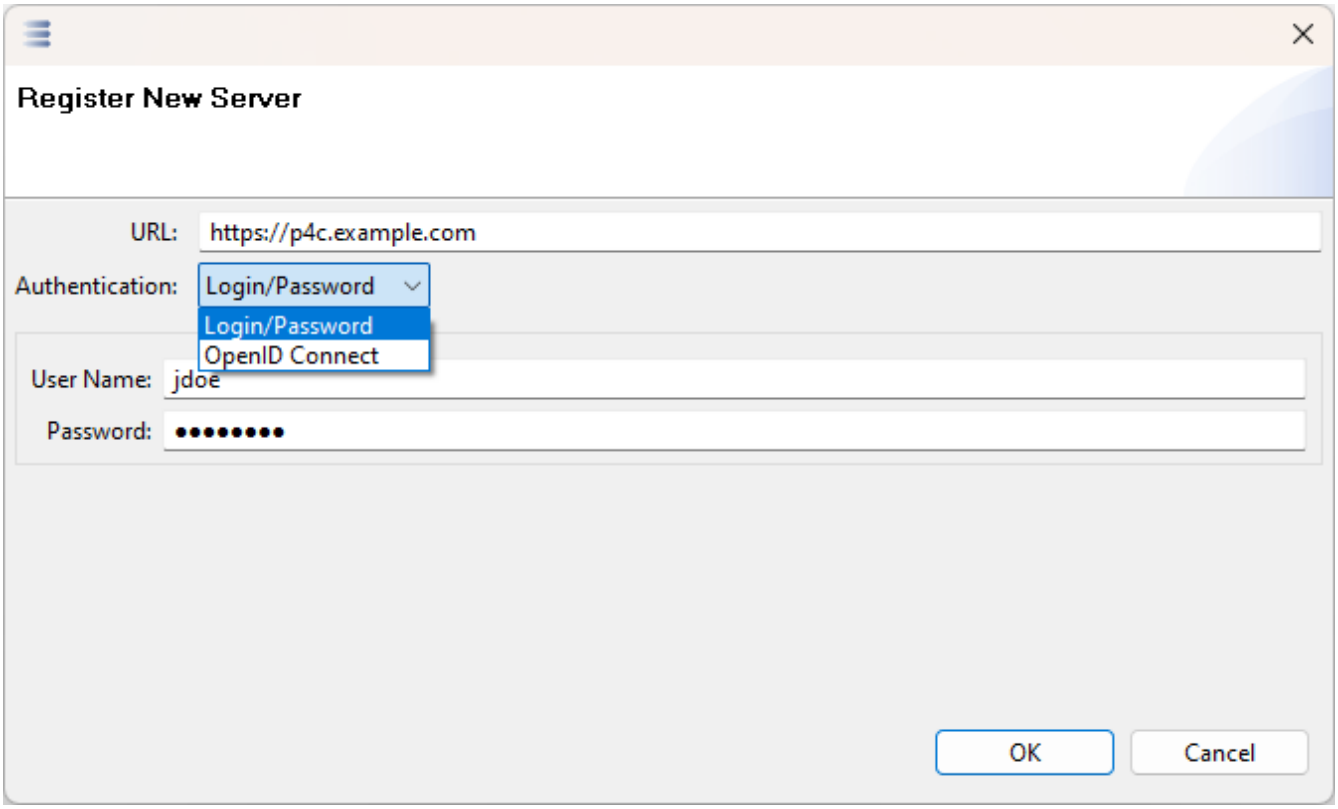


Figure 86. Register Publication Server with Login/Password Authentication

- The `URL` field must be set to the *Publication for Capella Server* URL (e.g. `https://publication.mycompany.com:9443` if Publication for Capella is deployed on a server called `publication.mycompany.com` on port `9443`)

There are two authentication modes available:

- **Login/Password**
  - The `User Name` field should be filled in with the user's login.
  - The `Password` field should be filled in with the user's password.
- **OpenID Connect**
  - The `Authorization URL` field should be filled in with the OpenID Connect authentication provider's authorization URL.
  - The `Token URL` field should be filled in with the OpenID Connect authentication provider's token URL. These two URLs should be requested from knowledgeable IT. The screenshot provides an example based on an imaginary KeyCloak server hosted at `keycloak.mycompany.com`.

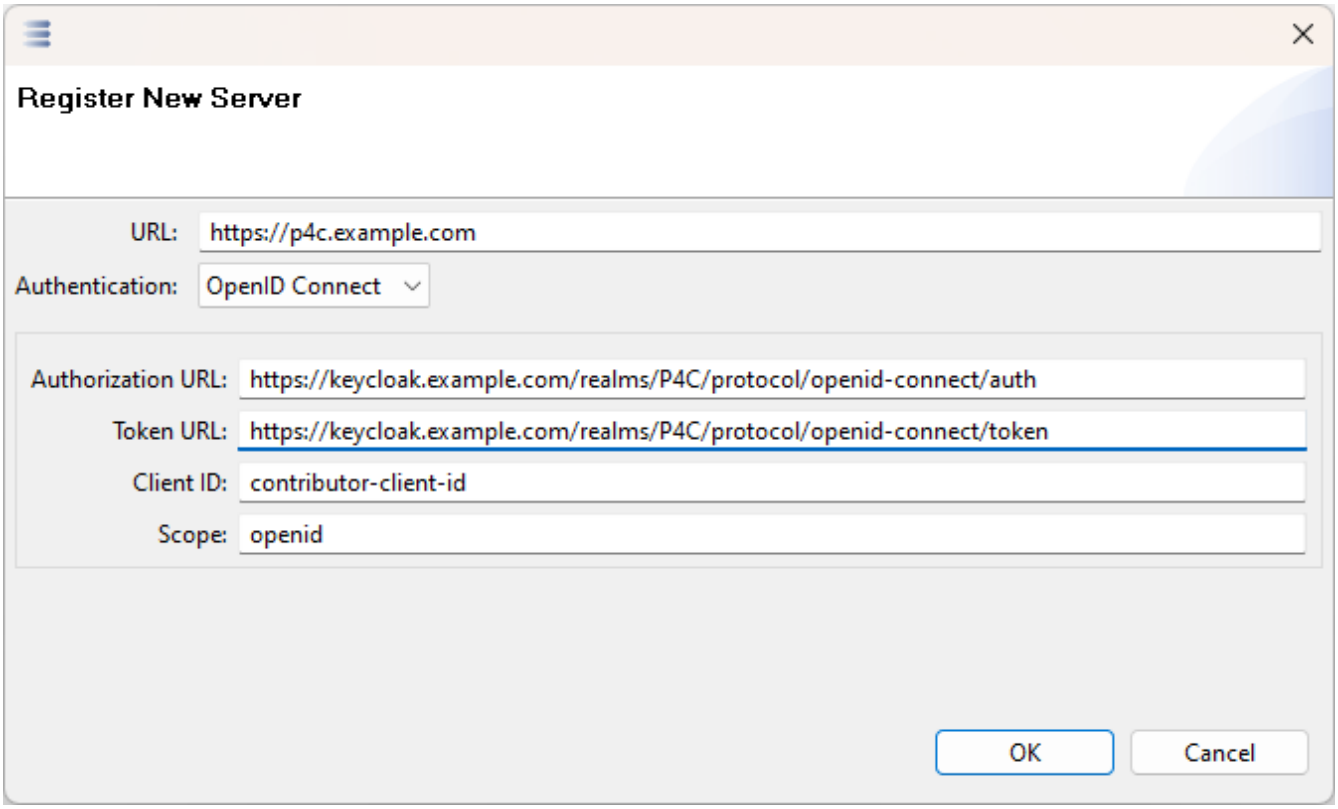


Figure 87. OpenID Connect Settings

Error messages are displayed if necessary, for instance if the requested URLs are not valid.

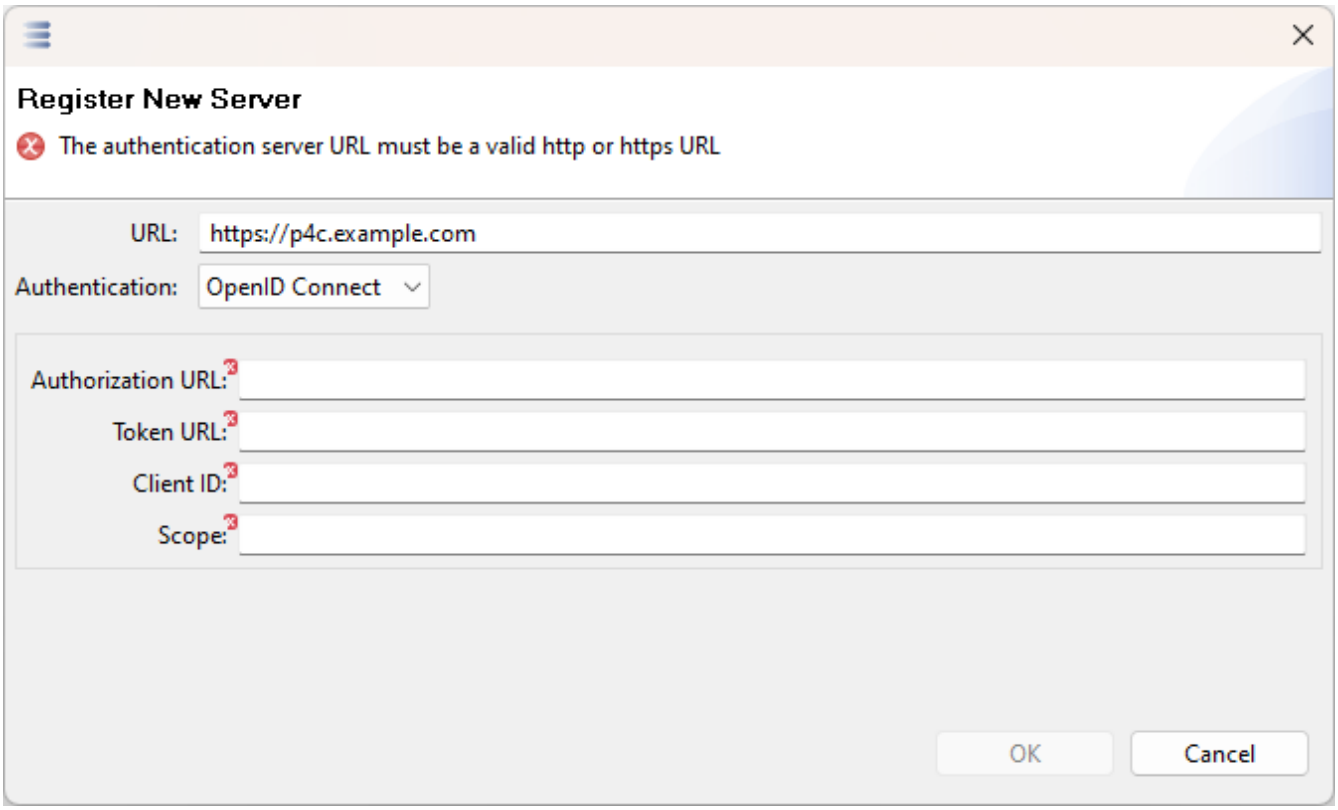


Figure 88. Publication Server Error Messages

Mapping of URLs for Link Creation

In order to support the creation of Traceability Links (See Managing Traceability) using drag and drop or copy/paste, it is necessary to configure the URLs that should be considered relevant for link

creation.

The settings are made in the traceability model, for each artifact repository.

Automatic recognition

When creating a repository using the Initializing Traceability Model action, if its URL is recognized among the Known URL patterns, default URL patterns will automatically be created along with the Artifact Repository.

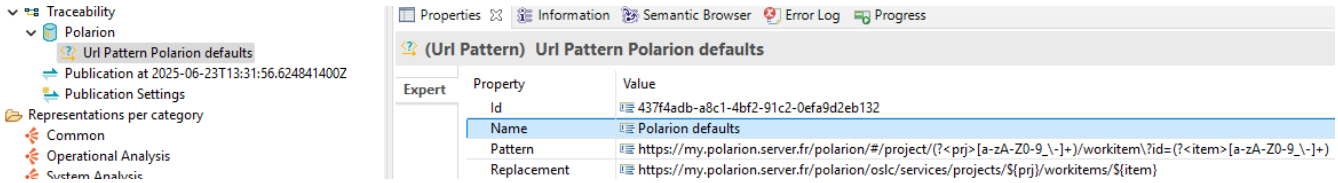


Figure 89. Polarion default URL pattern

After that it will still be possible to modify the default pattern and to recreate it using the *Add Default URL Patterns* action.

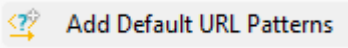


Figure 90. Add default URL patterns

Manual addition

It is possible to create custom patterns by manually creating them:

- On a given *Artifact Repository* in the model tree, right click and select 'Add Capella Element > Url Pattern'.

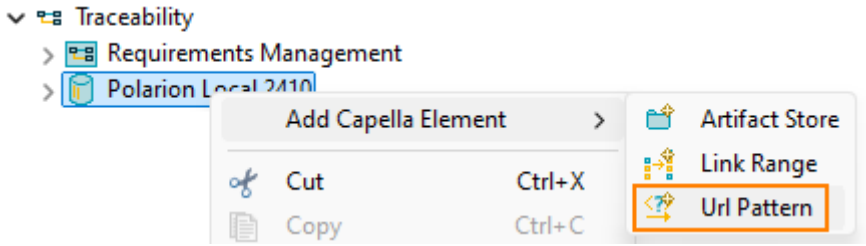


Figure 91. Add a URL Mapping to an Artifact Repository

- Edit the URL Mapping .Edit a given URL Mapping image::perseus-client/artifact-repo-add-url-pattern.png[URL Mapping Properties]

A URL mapping rule has the following properties:

- **ID:** Not editable, auto-generated.
- **Name:** A human-readable name supposed to describe the URL Mapping.
- **pattern:** A regular expression that describes the URL being dropped or pasted. Any URL that matches this pattern will be considered as being a valid candidate to create a link by drag and drop or copy/paste.
- **Replacement:** A String that describes what the matched URL should be transformed into. URLs that can be dropped or pasted from Artifact Repository web sites generally differ from the

associated OSLC URL. A mapping thus makes it possible to convert a web site URL into an OSLC URL.

Known URL patterns

There are known URL patterns for the different third-party repositories Publication for Capella is compatible with. These patterns are documented hereafter for the sake of completeness but should be initialized automatically thanks to the Automatic recognition of third-party repositories.

Polarion Configuration

To deal with **Polarion** URLs, use the following values:

- Pattern: `https://example.com/polarion/#/project/(?<prj>[a-zA-Z0-9_\-]+)/workitem\?id=(?<item>[a-zA-Z0-9_\-]+)`.
- Replacement: `https://example.com/polarion/oslc/services/projects/${prj}/workitems/${item}`.

Where `https://example.com` should be replaced by a value that matches the Polarion server URL.

DOORS Next Configuration

To deal with **DOORS Next** URLs, use the following values:

- Entry 1 (necessary to support link creation by drag & drop):
  - Pattern: `\Qhttps://<jts.mycompany.com>/rm/web#\E(?:[^&]+&(:?amp;)?)*\QartifactURI=https%3A%2F%2F<jts.mycompany.com>%3A9443%2Frm%2Fresources%2F\E(?<item>[a-zA-Z0-9\-\_]+).*`.
  - Replacement: `https://<jts.mycompany.com>/rm/resources/${item}`.
- Entry 2 (necessary to support link creation by copy/paste):
  - Pattern: `https://<jts.mycompany.com>/rm/resources/(?<item>[a-zA-Z0-9\-\_]+).*`.
  - Replacement: `https://<jts.mycompany.com>/rm/resources/${item}`.

Where all occurrences of `<jts.mycompany.com>` should be replaced by the relevant value (the DNS name of the IBM Jazz Team Server).

Note that in the second part of the Pattern, `:` is replaced by `%3A` and `/` is replaced by `%2F`. That is due to the way DOORS Next handles its web pages URLs.

Jazz EWM Configuration

To deal with **Jazz EWM** URLs, use the following values:

- Pattern: `\Qhttps://<IBM_jts.mycompany.com>/ccm/web/projects/Drive%20Pilot%20(Change%20Management)#\E(?:[^&]+&(:?amp;)?)*id=(?<item>[0-9]+).*`.
- Replacement: `https://<IBM_jts.mycompany.com>/ccm/resource/itemName/com.ibm.team.workitem.WorkItem/${item}`.

Where `https://<IBM_jts.mycompany.com>/ccm` should be replaced by a value that matches the Jazz EWM server URL.

Note the use of `\Q` and `\E`: These are used to quote what's between them without escaping special characters.

Note also the use of `Drive%20Pilot%20(Change%20Management)`: This must match a project name from your Jazz EWM server. Note that the project name is encoded to be valid in a URL, with spaces replace by `%20` and so on. Only artifacts from this project can then be dropped in Capella.

Jazz Quality Manager is not supported yet

Drag and drop from Jazz Quality Manager is not yet supported. The reason is that the OSLC URL of Jazz QM artifacts cannot be computed from the corresponding end-user URLs. This limitation should be resolved in a future version of Publication for Capella.

In the above examples, the syntax `(?<item>[a-zA-Z0-9\-\_]+)` can be decomposed as follows:

- `(?<item>` ... `)`: This defines a group of characters to match and gives a name to this group to reuse it easily in the replacement pattern. Here the name is 'item'. This makes it possible to reuse the captured group of characters in the replacement pattern with the syntax `${item}`.
- `[` ... `]+`: Will match at least one to any number of characters among those listed between the brackets.
- `a-zA-Z0-9\-_`: Lower-case `a-z` or upper-case `A-Z` letters, digits `0-9`, plus dashes `-` and underscores `_`. Dashes and underscores need to be escaped with a backslash character `\`.

Giving more details on regular expressions is out of the scope of this documentation. Refer to any online documentation regarding regular expressions in java to learn more.

Jira Configuration

The following indications are only valid for use with the Jira add-on OSLC Connect for Jira by Sodius.

To deal with **Jira** URLs, use the following values:

- Pattern: `https://jira.example.com/browse/(?<item>[A-Z\_0-9]+\-[0-9]+)`
- Replacement: `https://jira.example.com/rest/oslc/1.0/cm/issue/${item}`

Where `https://jira.example.com` should be replaced by a value that matches the Jira server URL.

With recent versions of Jira and confluence, you must set an additional property in the file `config/application.properties`: `perseus.http.client-headers[X-Atlassian-Token] = no-check`

Without this property, the Atlassian server will reject API calls with a message `403 Forbidden` with a message `XSRF check failed`.

Authentication

Every operation that involves data exchange between the Contributor Client and the Publication server is performed on behalf of a user with Write access on the project. Authentication is performed as configured in the preferences, which is driven by how the Publication server is configured.

Login/Password Authentication Flow

If the server is configured with Login/Password authentication (not recommended), then the Contributor Client will behave as follows:

- The contributor client will use the credentials registered in the preferences.
  - If there are no credentials registered, the contributor client will display a pop-up requesting the login and password whenever necessary, and make up to 3 attempts to login (if the credentials provided are rejected).
  - In this case, the credentials registered in the preferences will be updated.

OpenID Connect Authentication Flow

If the server is configured with OpenID Connect, then the Contributor Client will behave as follows:

- The contributor client will open a web browser page on the relevant OpenID Connect authentication provider's page.
  - If the user logged in recently with the OpenID Connect authentication provider, this step might be skipped.
- Once the authentication is processed by the OpenID Connect authentication provider, the web browser will be redirected to `http://127.0.0.1:<random_port>/login/auth?<openid-connect data>`. The page will show a message indicated that the authentication has been successfully processed, in which case the web page can be closed. The page may also indicate that the authentication has failed, in which case the page can also be closed.
  - If the authentication has been successful, the operation started in Capella can proceed.

Antiviruses or other security systems might react to the fact that the Contributor Client will start a (very lightweight) local server process to listen on the local host, what's called the loopback interface at address 127.0.0.1, on a random port.
This behavior is normal and is not a security issue, as it adheres to the current OAuth-2.0 best practices for user authentication from a native application which is to use the Authorization Flow along with the PKCE extension, omitting the client secret from the request, and to use an external user agent to complete the flow. See https://www.oauth.com/oauth2-servers/oauth-native-apps/ for more details.
It is possible to force this local server process to use a specific port rather than a random port. This can be useful if any security mechanism in place requires to know the port of such a process.
The option to set the port is `fr.obeo.perseus.oauth2.local.server.port` and must be set in the `capella.ini` file.

> Refer to section Contributor Client System Properties for more details.

Managing Traceability

Since version **2023.3.0**, the *Contributor Client* makes it possible to manage traceability within Capella models.

Configuring a Project to Support Traceability

Traceability Viewpoint Activation

The activation of the Traceability viewpoint can be done with contextual menu 'Viewpoint Selection' available on `.aird` files.

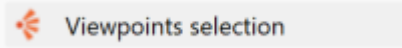


Figure 92. Viewpoint Selection Menu

The Traceability Viewpoint is automatically activated for new projects but it can be manually activated for projects that were created before installing the *Contributor Client*.

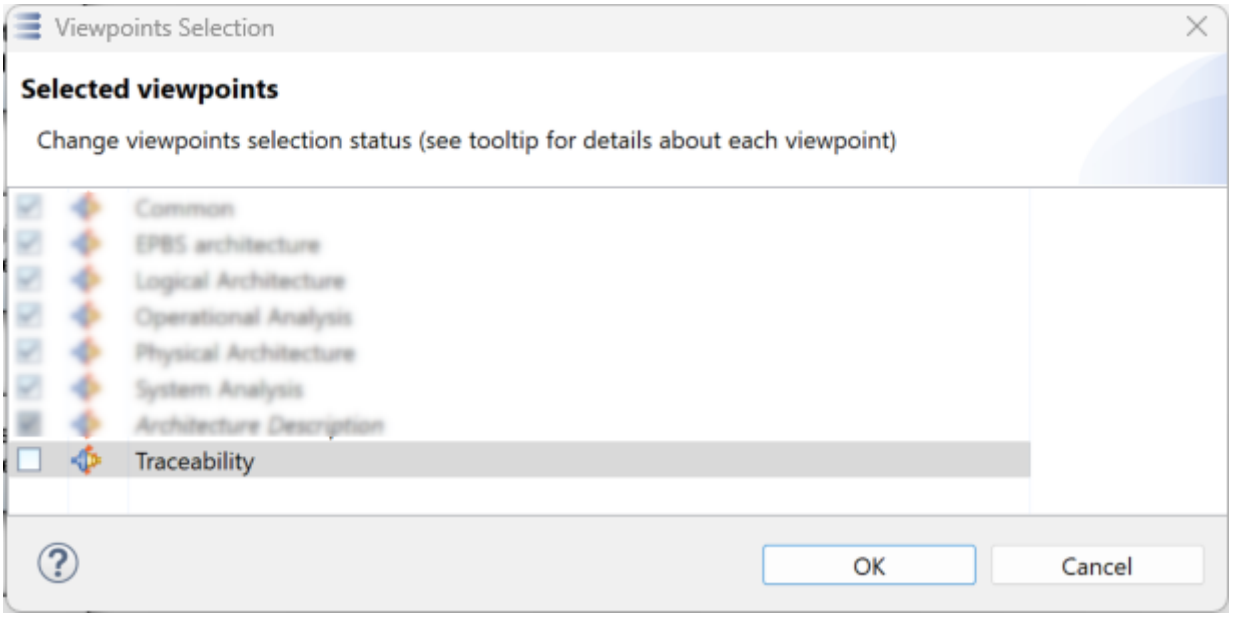


Figure 93. Viewpoint Selection Dialog

Initializing Traceability Model

Initializers are provided to help create and initialize traceability models.

If the Traceability model is not present, running any initializer will create it and add the traceability resource to the session.

There are currently two kinds of initializers:

- Domain Initializers, that create Domains with a set of artifact types and link types.
  - Three Domain Initializer are available, for the three relevant OSLC domains: Requirement Management, Change management, and Quality Management.

- If any of these domains is already configured in the model, the associated initializer will not be displayed in the list of available initializers.
- A Domain Initializer is just an accelerator, all these initializers do could be done manually by the end users. Also, the domains created by these initializers can be modified as needed.

- Repository Initializers, that create Artifact Repositories.
  - Repository Initializers are only available if the model has been published at least once, i.e. if it is tied to a specific URL on the publication server.
  - One initializer is available for each OSLC Friend configured on the server, if the associated Artifact Repository is not already present in the model.
  - After it is created, an Artifact Repository needs to be configured with relevant Link Range objects to define the types of links that can be created on which model element.

To run initializers, right-click on a `*.aird` file and use the menu `Initialize Traceability`.

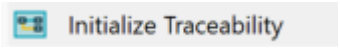


Figure 94. Initialize Traceability Menu

Select the relevant initializers.

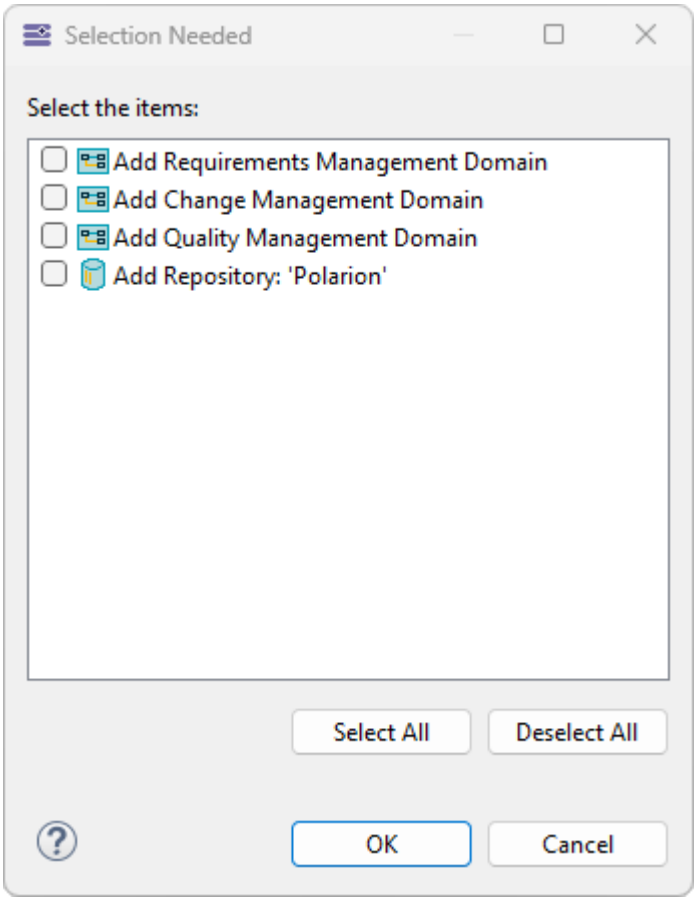


Figure 95. Initialize Wizard Page

Traceability Model Creation

- The session must be opened beforehand, by double-clicking on the `*.aird` file. The initialization menu is not available if the session is not open.

- The IDE offers the possibility to create manually the Traceability model. This is not recommended, the best way to create a traceability model is to use an initializer. When the model is added to the project it is possible to edit it and modify elements as you want.

Traceability Models

It is possible to edit the Traceability model in the 'Package explorer' and the 'Properties view'.

Do not edit Traceability Models outside of a Capella model

Even though an editor exists for `*.traceability` files, the only way to edit traceability models when working with Publication for Capella is within a Capella session, using the tools in the model explorer, in the diagrams and the drag and drop operations.

Capella 6.1 and Above Known Issue

There is a known issue with Capella 6.1 and Above: By default, the capability 'Expert Semantic Properties' is off. This capability must be turned on for the Publication for Capella Contributor Client to work properly.

To activate it, open the preferences (Window > Preferences) and navigate to the section General > Capabilities. Then make sure that this capability, located in 'Capella Advanced Modeling', is turned on.

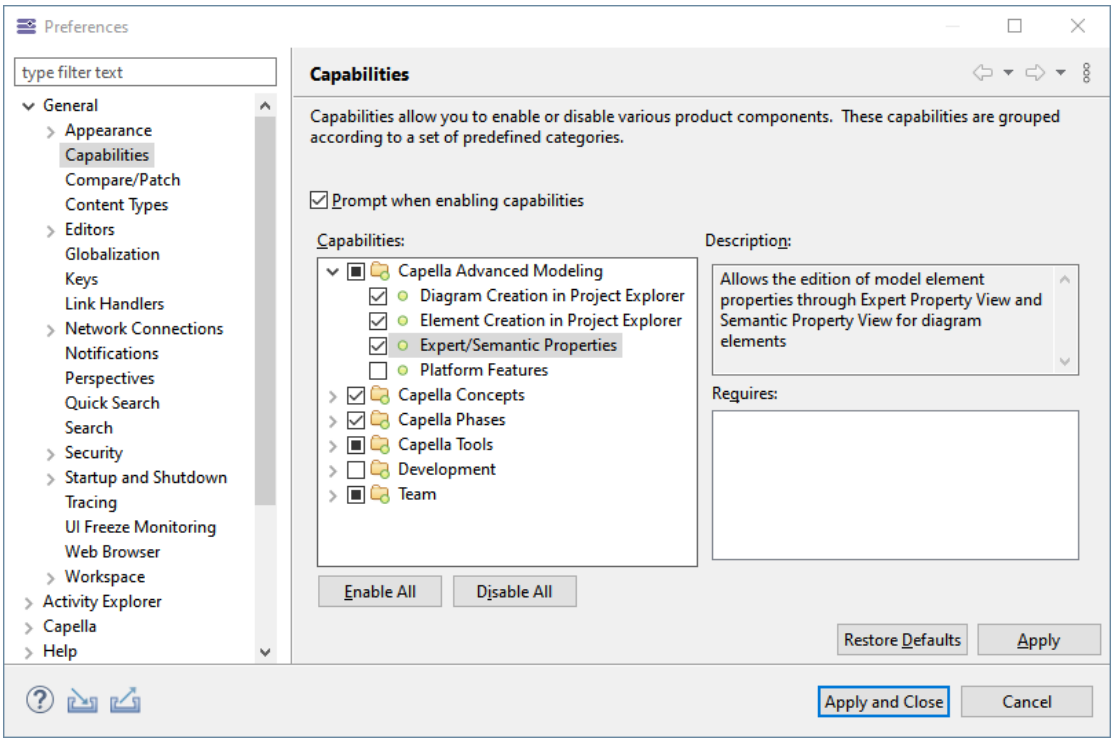


Figure 96. Capability to Turn On in Capella 6.1 and Above

Description of the problem

If this capability is turned off, problems will happen whenever a user wants to edit information in the Traceability model, which is required to configure drag&drop

> from third-party repositories. The properties view will be empty and claim that there is 'no properties available' when the user selects any Traceability model element.
>
> This problem can actually happen in any Capella version if this capability has been manually turned off.

Publication Settings

The traceability models contain some settings related to model publishing. These settings are captured in an object of type `PublicationSettings`.

This object can be created manually with a right-click on the `Traceability` node in the model tree, then 'Add Capella Element > Publication Settings'.

- `Exclude Prefix` (defaults to `[PRIVATE]`): The prefix to use in a diagram name to prevent its publishing.
- `Filter Diagrams to Publish` (defaults to `false`): Activating this option will prevent the publishing of diagrams prefixed with the `Exclude Prefix` value.
- `Publish Diagrams` (defaults to `true`): Activates or deactivates the publishing of diagrams.
- `Publish Semantic Browser` (defaults to `true`): Activates or deactivates the publishing of semantic browser data.
- `Refresh Diagrams` (defaults to `true`): If the option `Publish Diagrams` is set, activating this option will refresh each diagram before publishing it.

See Main Panel - Model Navigation for more information about how the diagrams and the semantic browser are displayed by the server.

Domains

A Domain is a name space that defines a set of artifact types and link types.

- **Domain:** Name space for defining artifact types and link types.
  - Name: **mandatory**, a human-readable name.
  - Prefix: **mandatory**, a short name of the domain, must be a valid XML namespace prefix.
  - Uri: **mandatory**, URL of the domain, **must end with a slash '/' or a hash sign '#'**.

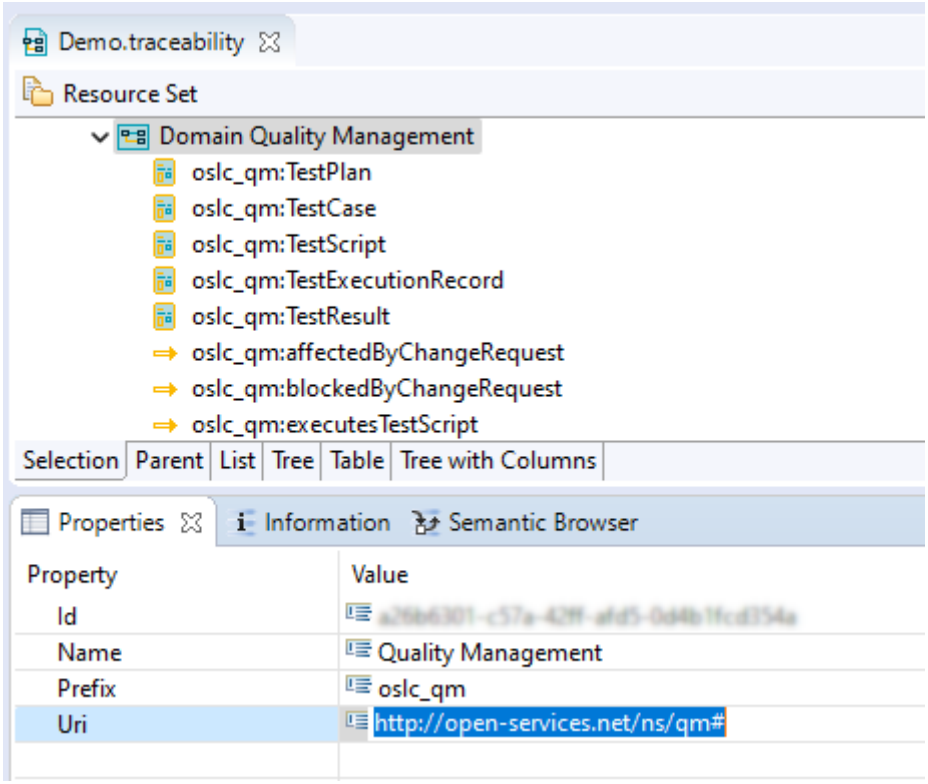


Figure 97. Edit Domain

There is currently no way to retrieve traceability settings from the server into a Traceability model, this has to be done manually.

Domain Constraints

Domain attributes `uri` and `prefix` must be set, otherwise the Domain cannot be used and will be ignored.

All the domains defined in a model must have a distinct **URI** and **Prefix**.

- **ArtifactType:** Specifies the type of artifact
  - Name: **mandatory**, name of the type, *should start with an upper-case letter*.
  - Uri: **not editable**, derived attribute, based on domain URI and name.

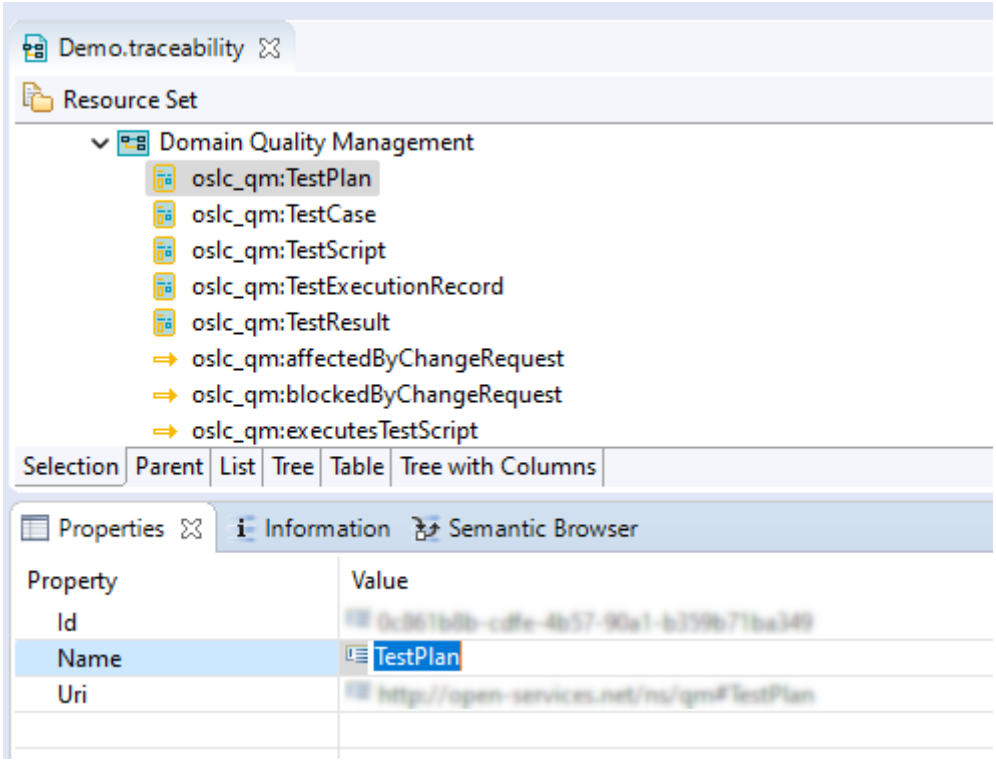


Figure 98. Edit ArtifactType

- **LinkType:** Specifies the type of a link between a model element and an Artifact.
  - Name: **mandatory**, name of the link type, *should start with a lower-case letter*.
  - Uri: **not editable**, derived attribute, based on domain URI and name
  - Artifact range: **optional**, applicable artifact types list - Reserved for future use
  - Impact: **optional**, No impact / Symmetric impact / follows link / Opposite link - Reserved for future use
  - Inverse: **optional**, inverse link type - Reserved for future use

Ignored Properties

The `artifactRange`, `impact` and `inverse` properties are not used. They are reserved for future used and ignored in the current version.

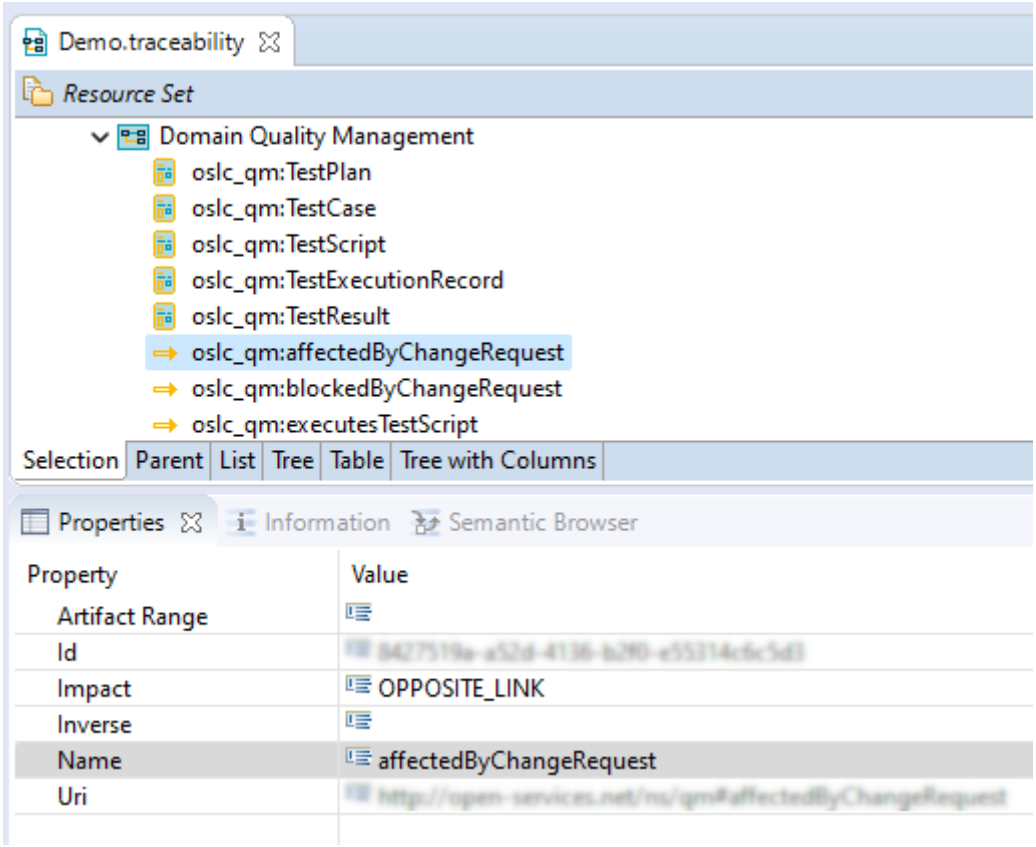


Figure 99. Edit LinkType

Artifact Repositories

- **ArtifactRepository:** One *ArtifactRepository* should be present in the model for each third-party OSLC Friend server configured on the Publication server. An *ArtifactRepository* owns [o..n] Link Ranges, [0..1] Artifact Store, and [0..n] Snaphosts.
  - Name : **optional**, label of the repository.
  - Url : **mandatory**, URL of the server, **must match the URL of an OSLC Friend configured on the Publication server**.
  - Configuration: **optional**, reserved for future use.

ArtifactRepository Mandatory attributes

The `URL` attribute must be set. If it is not set, this repository cannot be used and will be ignored.

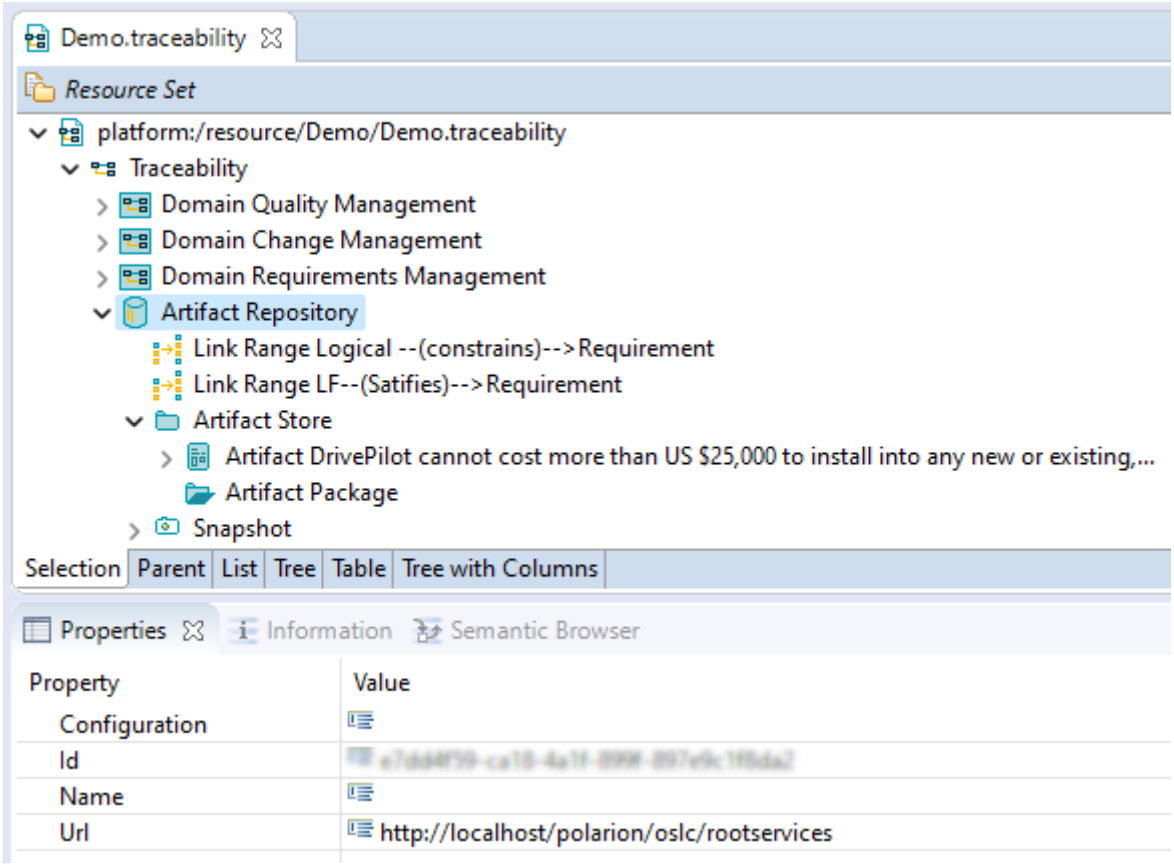


Figure 100. Edit ArtifactRepository

Link Ranges

- **LinkRange:** customization specifying for a given link type the possible source and target objects. It is used to constrain the link creation, by drag & drop or by using the palette in diagrams.
  - `Name`: **optional**, for readability is is recommended to set an easily readable name
  - `Local Range`: **mandatory**, list of Strings. Each value should be the name of a type of element in the local model, e.g. `LogicalFunction`, `LogicalComponent`. Spaces and case are ignored, so entering `logical function` is strictly equivalent to entering `LogicalFunction`. It is now possible to use the name of super-types, in which case every instance of a sub-class will be accepted. For example, entering `Component` will make this link range valid for Operational Entities as well as System, Logical, and Physical Components, because all these extend the `Component` meta-class.

Linking to Diagrams

Since v2025.2.0, to make it possible to create links between diagrams and artifacts, it is necessary to include the value `Diagram` in the `Local Range` property of the **LinkRange**.

- `Link Type`: **mandatory**, link type to constrain.
- `Impact`: **optional**, reserved for future use, possible values:
  - `No impact`
  - `Symmetric impact`

- `Follows link`
- `Opposite link`

- `Direction Constraint`: **mandatory**, possible values:
  - `Both`
  - `Model element to artifact` - This is the default value, and the only one that should be used for the time being.
  - `Artifact to model element`
- `Artifact Range`: **mandatory**, list of authorized types of artifact.

Link Ranges

It is necessary to configure at least one Link Range for each type of link to be created by drag and drop or copy/paste. During a drag and drop operation, Link Ranges are used to determine the possible link types between the artifact and the model element. If no link type is found according to the link ranges, then no link will be created.

The only *Direction constraint* supported in this version is `Model Element to Artifact`. Other values are here for the sake of having a complete data model but are not supported currently.

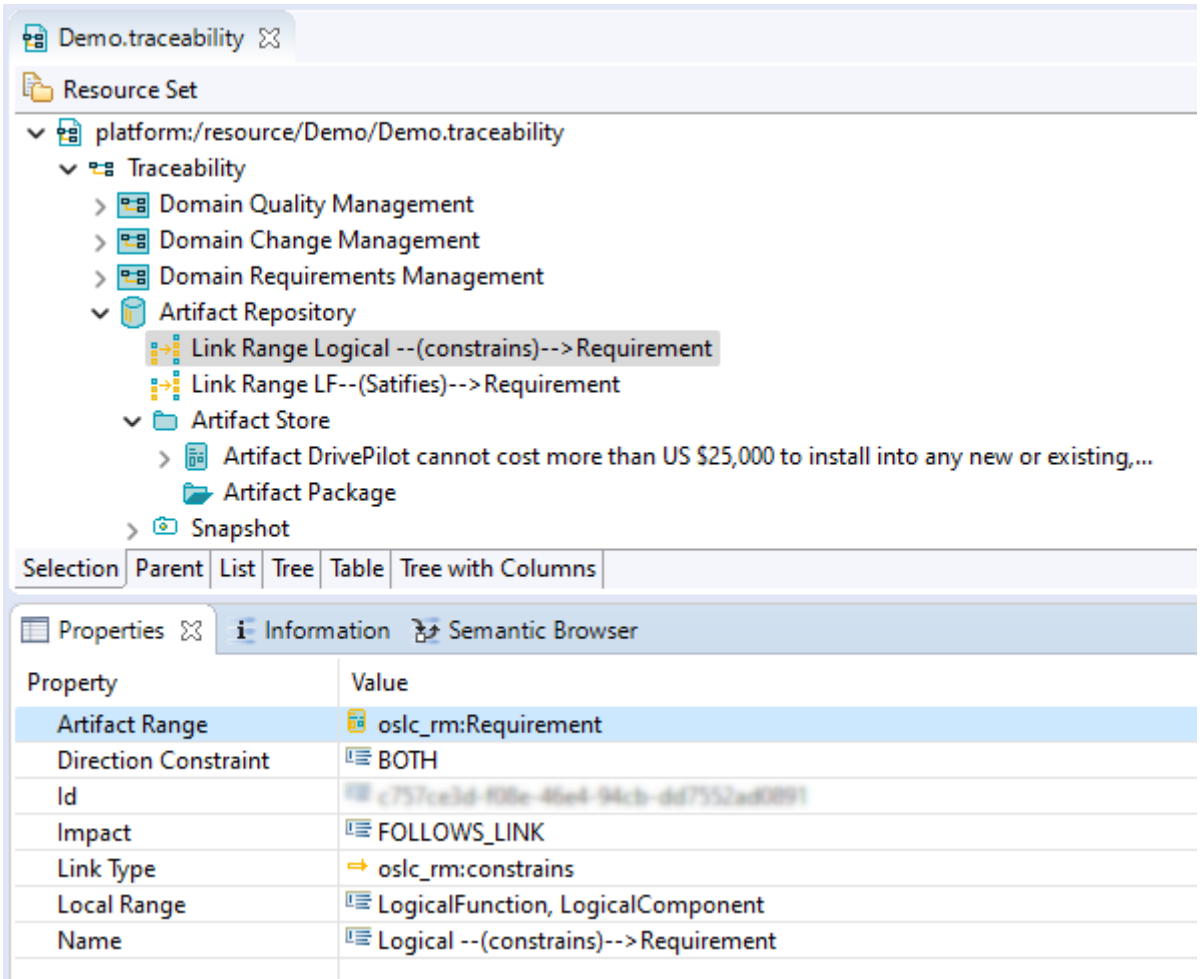


Figure 101. Edit LinkRange

Artifact Store

- **Artifact Store:** Root Package used as the main container for artifacts and packages in a single *ArtifactRepository*.
  - Name: **optional**, name of the package
- **Artifact Package:** Package useful to group artifacts
  - Name: **optional**, name of the package
- **Artifact:** Represents an artifact managed by a third party repository (such as a requirement, a change request, or a test case).
  - should be created when dropped or pasted from third-party server web page on an element in the model.
  - Artifacts can be freely moved into other Artifacts or Artifact packages.

Artifact Creation

Artifacts should never be created manually. Their properties cannot be modified.

Snaphosts

- **Snapshot:** Synchronization point, captures the state of links at the time of a given synchronization with the server. This is a purely technical concept.

Snapshot Constraints

Snapshot elements mustn't be deleted or modified. These elements are mandatory for the publication process but they are not intended to be manipulated by the end user. Modifying or deleting snapshot or snapshot contents is not supported and should absolutely be avoided, otherwise traceability data loss or mismatches may occur.

Creating Links by Drag and Drop

An element configured in the mappings can be dropped on a model element, either in the model tree or in a diagram. If several link ranges can apply to a drop, a selection dialog is open to select the link type to use.

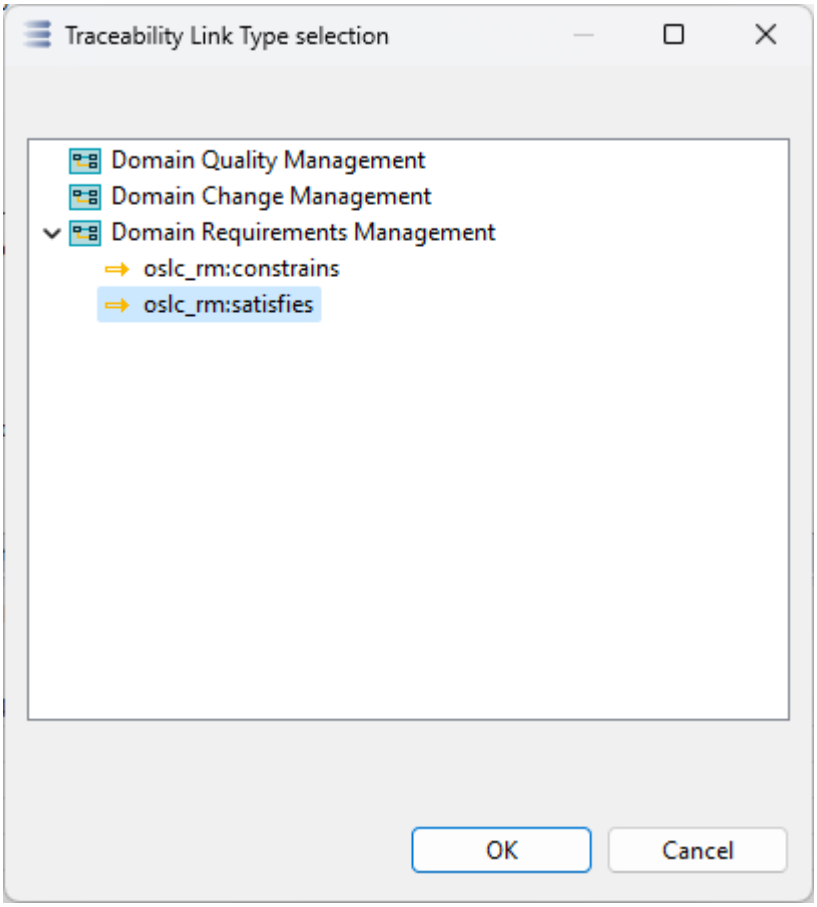


Figure 102. Link Type Selection

The artifact and the link are created. If the artifact was present before the drop, only the link will be created.

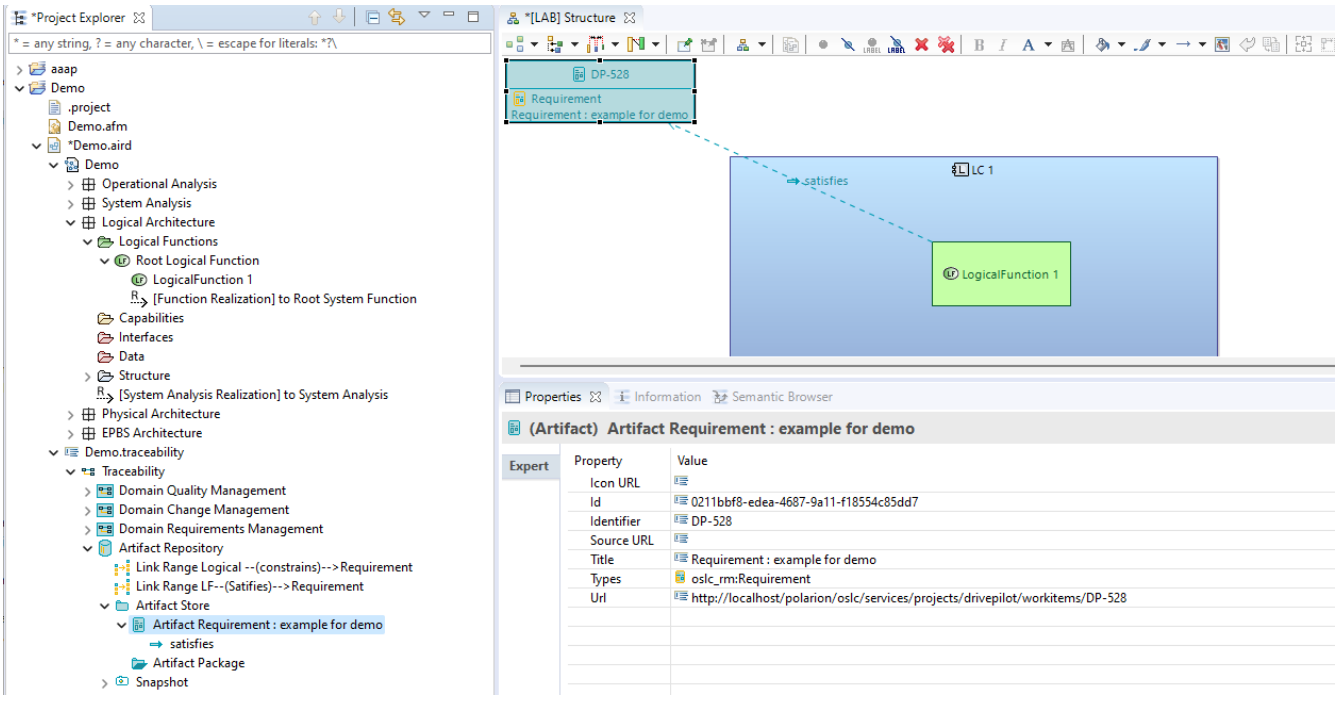


Figure 103. Diagram Result

Drag and Drop Limitations

- Drag and drop can only be achieved if the model was already published to Publication for Capella, and its traceability model is initialized.

- Drag and drop is not supported on object links in diagrams.

Creating Links by Copy Paste

An element configured as described in Mapping of URLs for Link Creation can be pasted on a model element in a diagram. First, copy the artifact URL from a web browser (This step depends on the web browser used and the third party repository, but it should be as simple as right-click on the artifact's URL and select 'Copy Link'). Then, select the target model element in a diagram and hit `Ctrl+V` (or whatever shortcut is defined for the paste operation) to paste the link.

If at least one link type is defined between the target model element and the pasted artifact, the link gets created. If several link types are defined, a selection dialog is opened to select the link type before creating the link.

Copy Paste Limitations

- Creating a link by Copy Paste can only be achieved if the model was already published to Publication for Capella, and its Traceability model is initialized.
- Creating a link by Copy Paste is only supported in diagrams, not in the model tree.

Creating links between Diagrams and Third-Party Artifacts

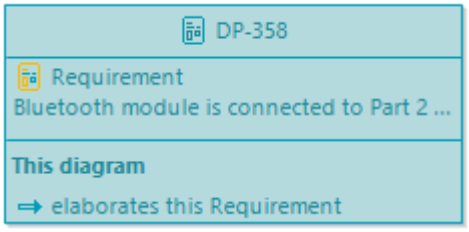
It can be interesting to add links between diagrams and third-party artifact such as Requirements or Test Cases.

Since version 2025.2.0, this is supported using either drag & drop or copy/paste.

Open the diagram and drop the artifact on the diagram's background, or copy/paste the artifact's URL on the diagram's background.

Specific settings must be defined to allow links between diagrams and artifacts, as described in section Link Ranges.

An artifact that is linked to the diagram it is displayed in will have an additional container to make it noticeable that the artifact is really linked to the diagram and not only displayed in it:



Traceability Tools in Diagrams

Traceability tools are available for all Capella diagrams. The Traceability Layer should be selected to view tools.

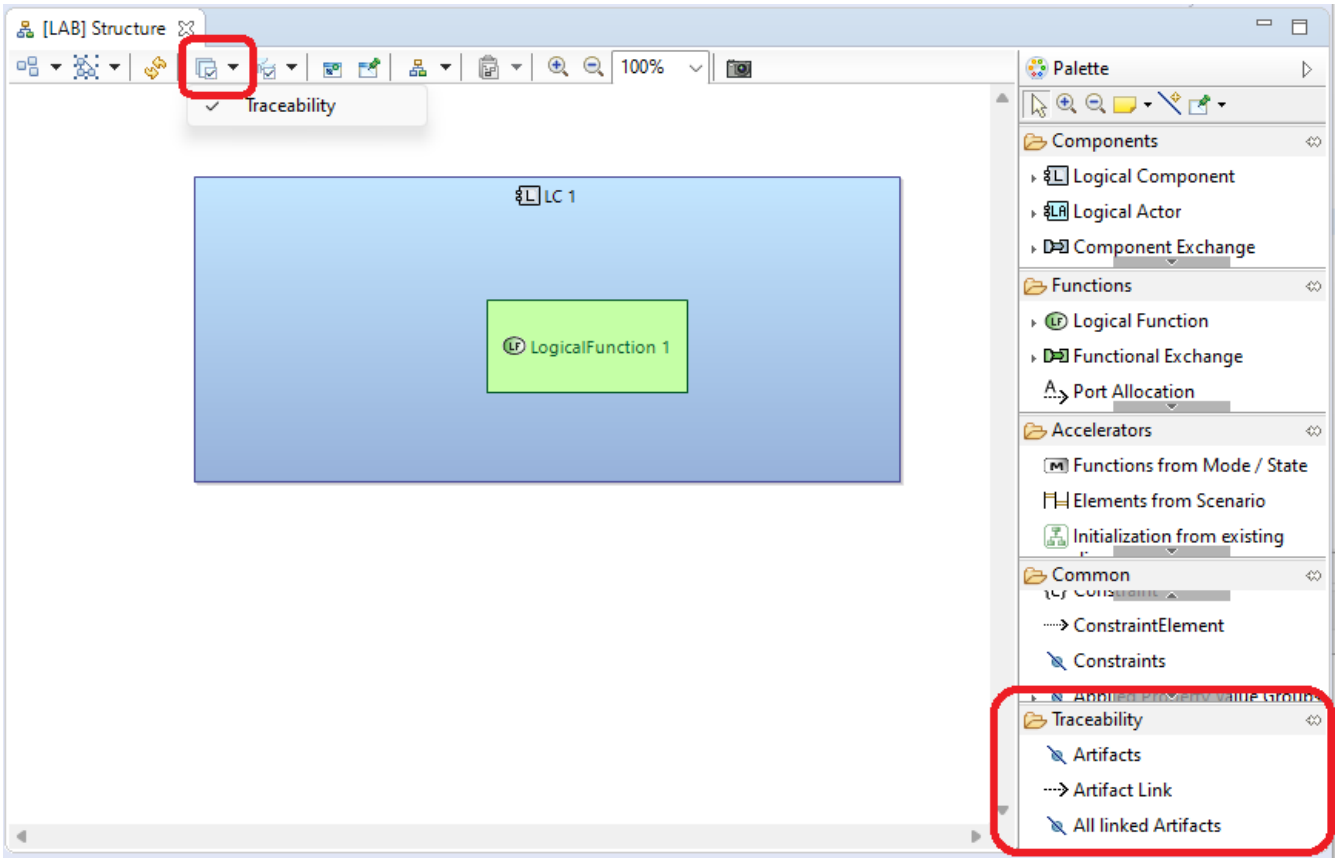


Figure 104. Traceability Layer Tools

- Tool "Artifacts" : This tool can be used everywhere in the diagram. It allow to select already created artifact to add them in the diagram. It open a selection dialog with all the artifacts present in the model.

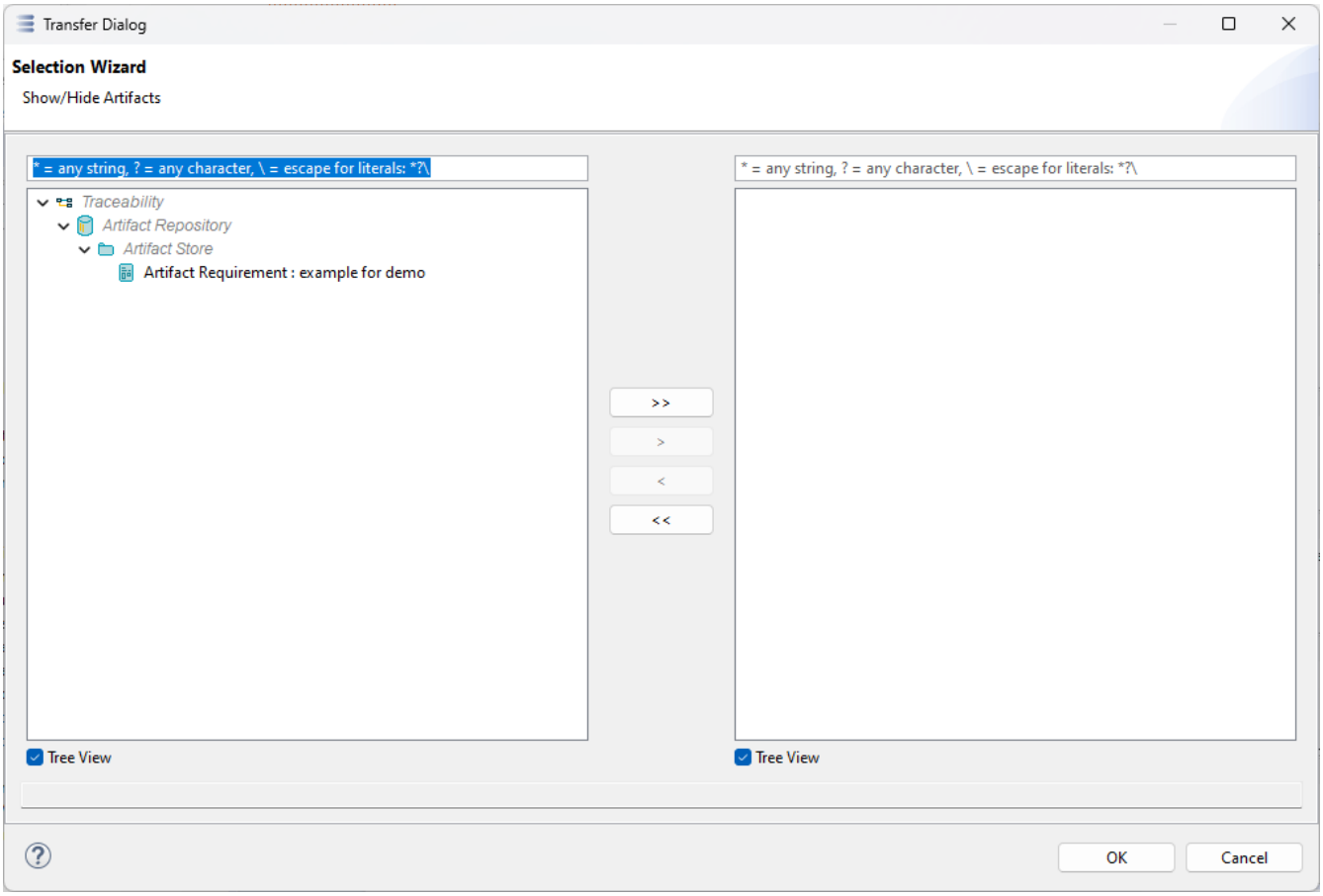


Figure 105. Artifact Selection Dialog

- Tool "Artifact Link": This allow to create link between Artifact present in diagram and element. If several link types are available, a selection dialog is shown to select the type of the new link. Link creation is limited by the linkRange customization which specifies types and direction.

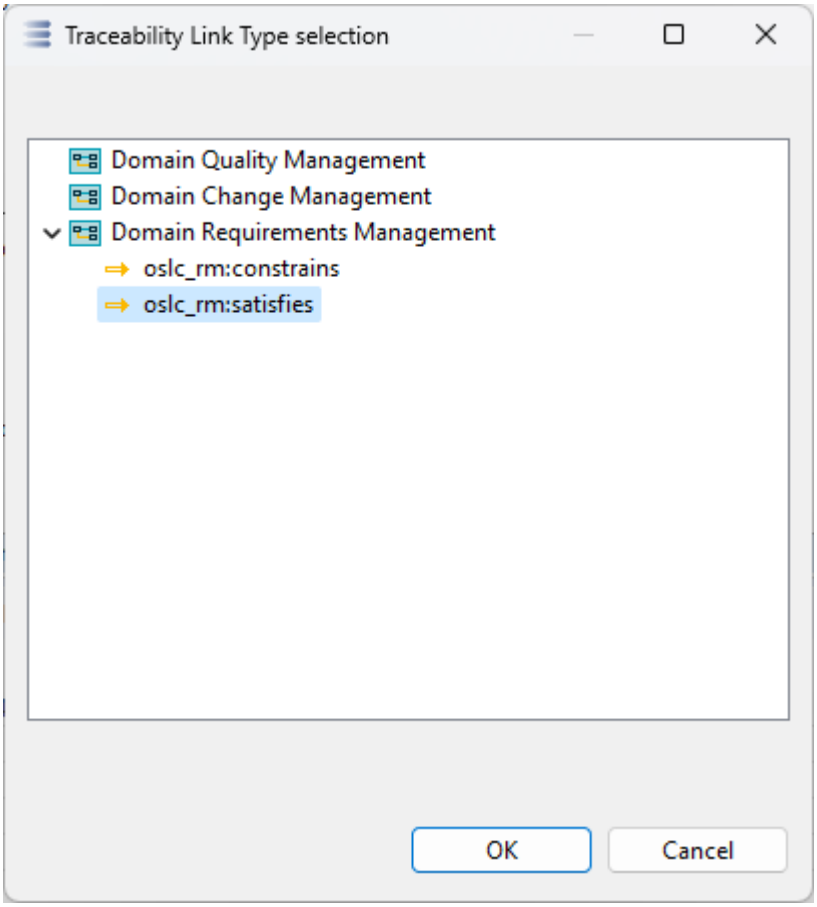


Figure 106. Link Type Selection

- Tool "All Linked Artifacts" : Using this tool for an element present in diagram imports all the artifacts linked to the selected element.

Traceability Tools limitations

Traceability Tools are only present in diagrams defined in the core Capella. Additional diagrams defined by Capella extensions (such as the open-source Requirement add-on, or proprietary add-ons with their own viewpoints) will not have these tools in the Palette. Also, links between artifacts and model elements not defined in the core Capella cannot be displayed in diagrams.

Getting Changes from the Publication Server

There is no live connection between the Contributor Client and the Publication server. It means that the contributor client is not notified when links are modified on the Publication server. To get link changes made on the server, the Contributor Client user must use the appropriate action 'Update from server':

- Right-click on the `*.aird` file of the published model.
  - The model must have been published to the server at least once before.
  - The session must be open.
- Select 'Update from server'.

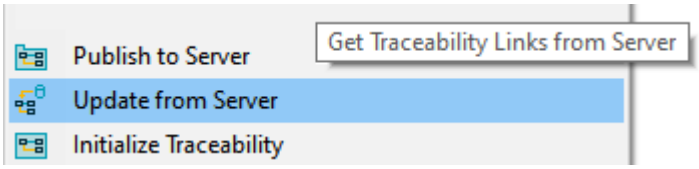


Figure 107. Update from Server Menu

A progress bar will appear while the Contributor client retrieves the link data from the Publication server.

If no modification has been made on the server, then a message will be displayed:

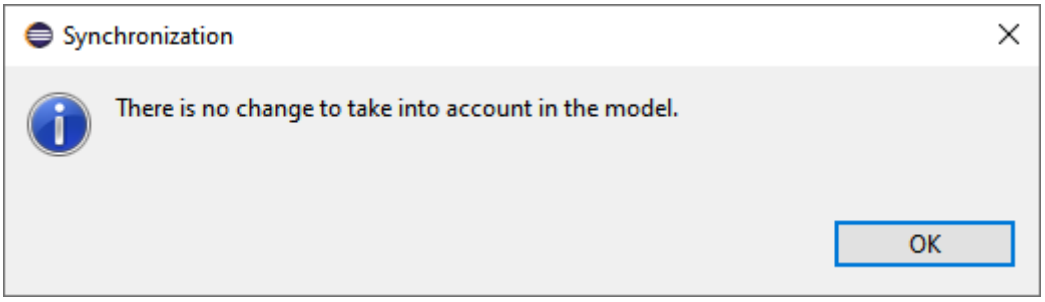


Figure 108. Message Without Server-Side Changes

If there are differences, then a dialog will be displayed to show the differences and apply them to the Capella model.

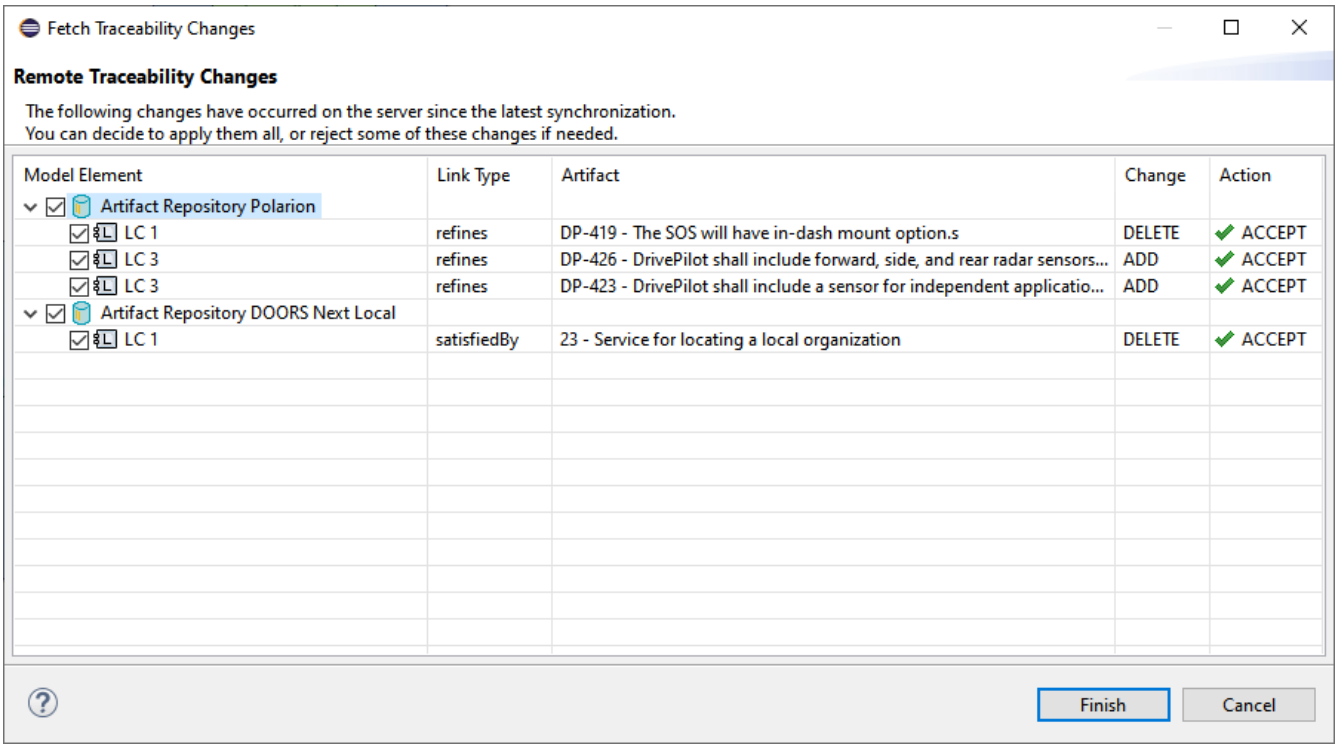


Figure 109. Example of Update with Remote Changes

This dialog shows all the changes that have happened on the Publication server since the last synchronization between the Capella environment and the Publication server.

The changes are grouped by repository, in case several remote repositories are involved.

Traceability and Repositories

> Only the repositories that are configured in the Traceability model are considered for link synchronization. If links are created online to or from a friend server that is not configured as an Artifact Repository in the model, these links won't be retrieved in the model.
>
> If links are missing because their repository is not configured, the missing repository can be added using the Initializing Traceability Model action. The next 'update from server' will then also take added repositories into account, and retrieve all the links involving this repository.

Within each repository, each line represents a link modification: Addition or deletion.

In this dialog, the user can do the following actions:

- **Cancel:** This cancels the update, none of the remote changes received from the server are taken into account in the model.
- **Finish:** This applies the selected changes to the model and stores a new synchronization point.
- **Accept or reject a change:** Changes can be individually rejected or accepted. All the changes received are accepted by default. The user can uncheck the box in front of a link change to 'reject' this change.
- Accepting a change means that this change will be taken into account in the Capella model.
  - However, new links will not appear in diagrams. For that, users need to use the relevant palette tool in the diagrams where they want to display the new link (if any).
  - Deleted links will be removed from diagrams the next time the diagram is opened.
- Rejecting a change means that this change will:
  - NOT be taken into account in the Capella model.
  - Be reverted on the server the next time the model is published:
    - If the link has been created on the server, the next publication will delete it.
    - If the link has been deleted on the server, the next publication will recreate it.

The following screenshot shows an example where the user accepts 2 changes and rejects the 2 others.

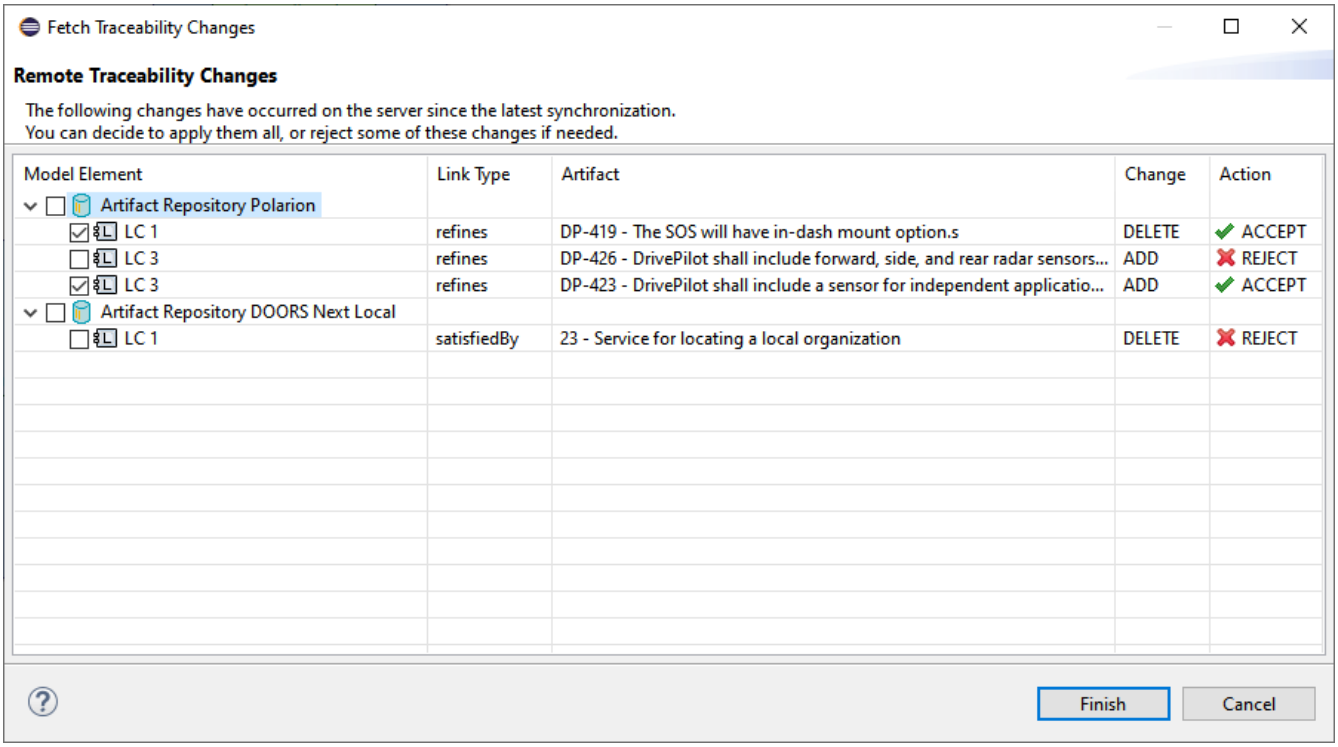


Figure 110. Rejecting Some Link Changes

Update link

- All the artifact types and link types must be initialized according to the Publication for Capella mapping.
- All existing artifacts and links will be imported from Publication for Capella to the Traceability model even if LinkRange are not initialized. It is not possible to create such link in diagram if the customization with LinkRange is not initialized.

Refreshing Artifacts

The Oslc links displayed and stored in Capella are updated only when requested by the user. For performance reasons, the artifact data (especially their titles) are not updated during these link update operations.

However, it is possible to update a single artifact or all the artifacts of a single repository independently from the links.

In Capella, right-click on an *Artifact* or *Artifact Store* in the model tree (in the traceability model), or on an *Artifact* in a diagram, then select 'Update from server'.

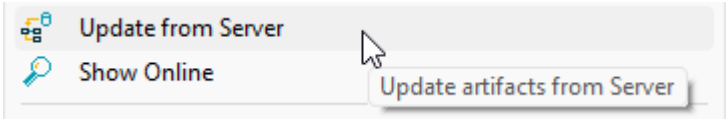


Figure 111. Refresh Artifact Action

It will trigger the update of the remote resource in both the Publication server and Capella. This should only be useful when the remote artifact's title has been modified.

Import / Export Traceability Settings

The traceability settings of a given model can be reused in another. Only the settings aspects of the traceability model will be reused: repositories, domains, link ranges. Snapshots, target model URI, OSLC configuration, are ignored when importing traceability settings.

When initializing the traceability for a model it is possible to import existing settings directly from another model.

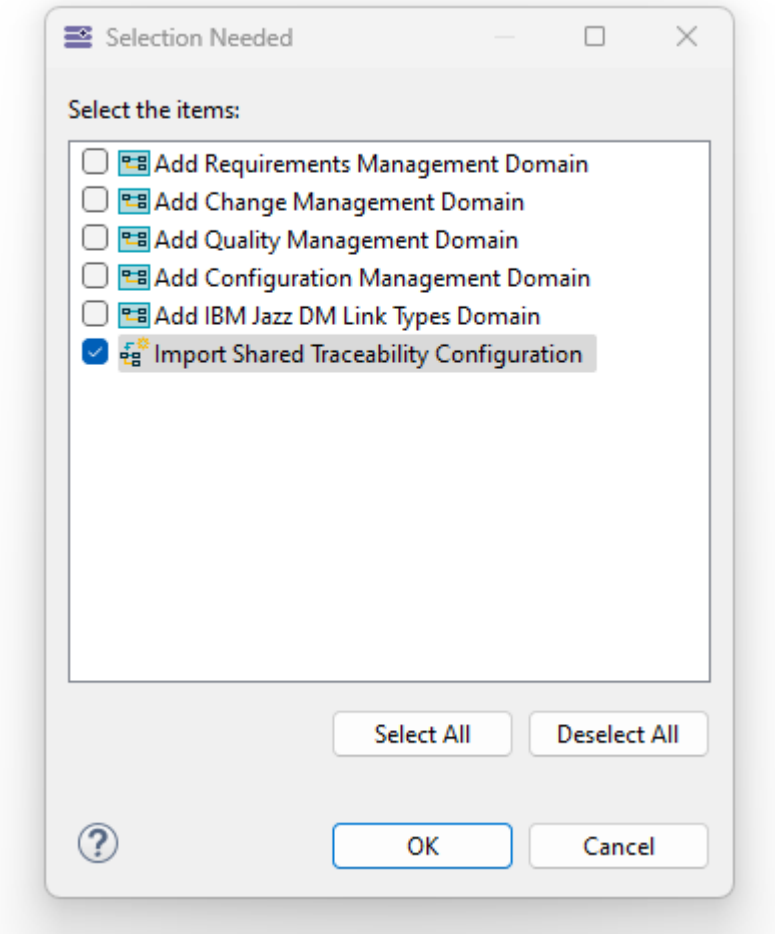


Figure 112. Import Traceability Settings

The dialog displays all the traceability files available within the current workspace.

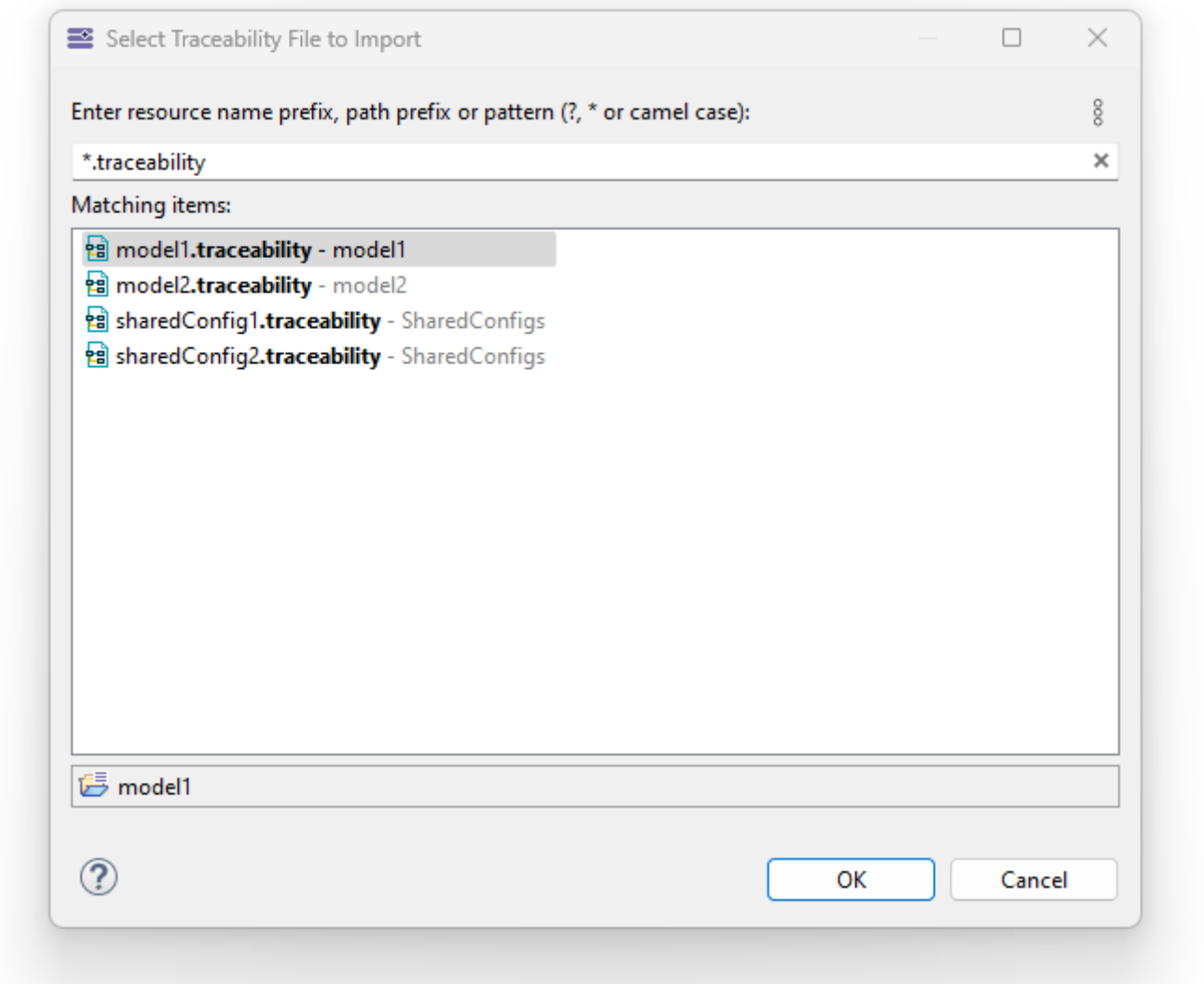


Figure 113. Available Traceability Models

To export traceability settings, right-click on the `*.traceability` file and click on the 'Export traceability Settings' action.

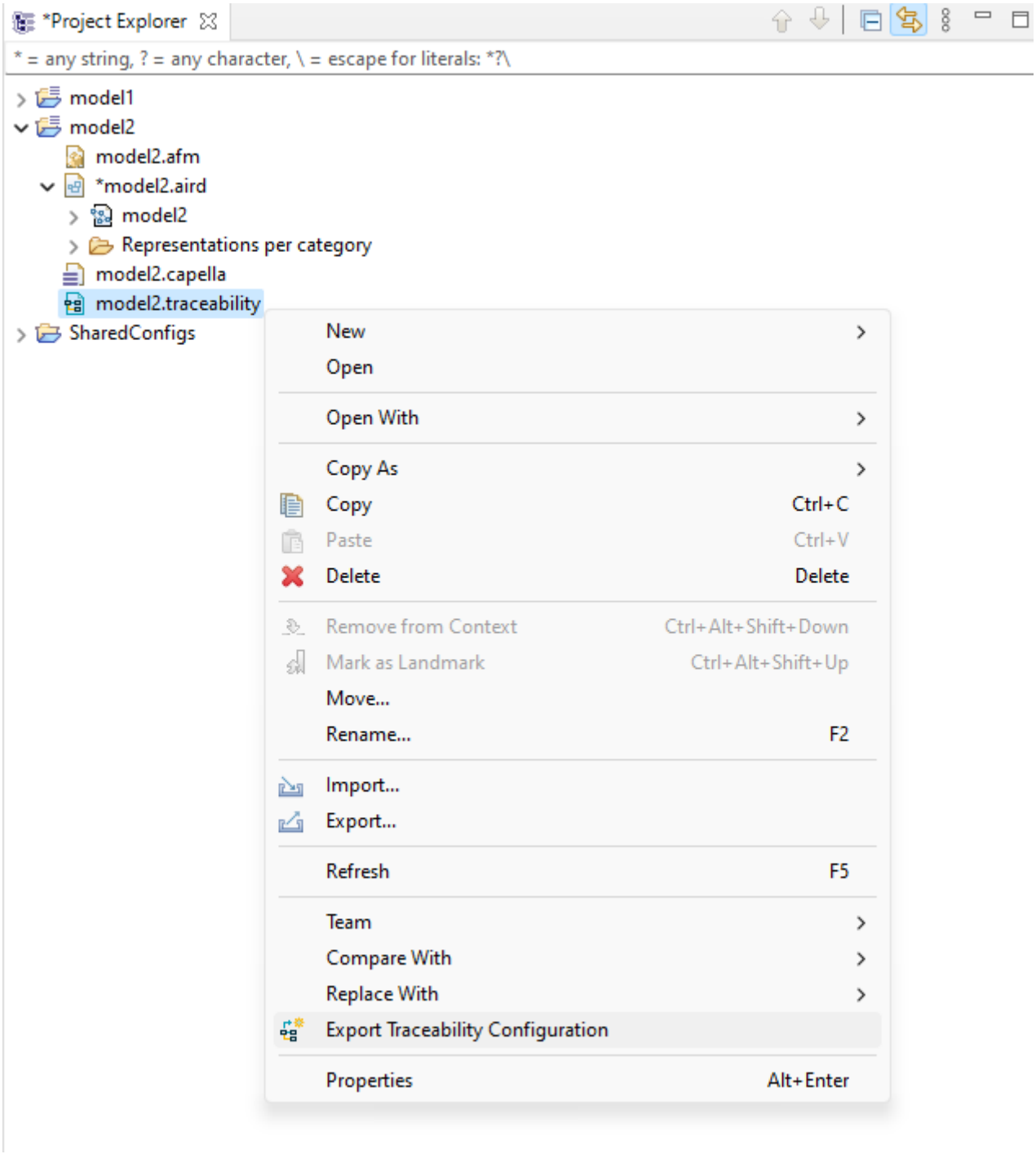


Figure 114. Available Traceability Models to Import From

Exported traceability models only contain settings and can safely be shared between users and teams to facilitate the initialization of traceability in Capella models.

Publishing a Model

Publishing a Model is the operation of sending the model data to the *Publication Server*, where it will become available for browsing and linking with third-party artifacts.

Before publishing, it is necessary to configure at least on Publication server. Refer to section Publication Servers for more details.

Three distinct cases are supported:

- Publishing a new Model (the model does not exist yet on the *Publication Server*). See Publish for the First Time.
- Publishing an update of Model (the model already exists on the *Publication Server*). See Publish an Update of a Model.
- Publishing an update of a Model published with an older version of Publication for Capella (Any version prior to 2022.9.0). See Publish an Update of a Model of a Previous Publication for Capella Version.

Publish for the First Time

The publication of a model is triggered by right-clicking on the session file `*.aird` in the project Explorer.

The session must be opened beforehand, by double-clicking on the `*.aird` file. The publish menu is not available if the session is not open.

The menu is labeled 'Publish to Server'.

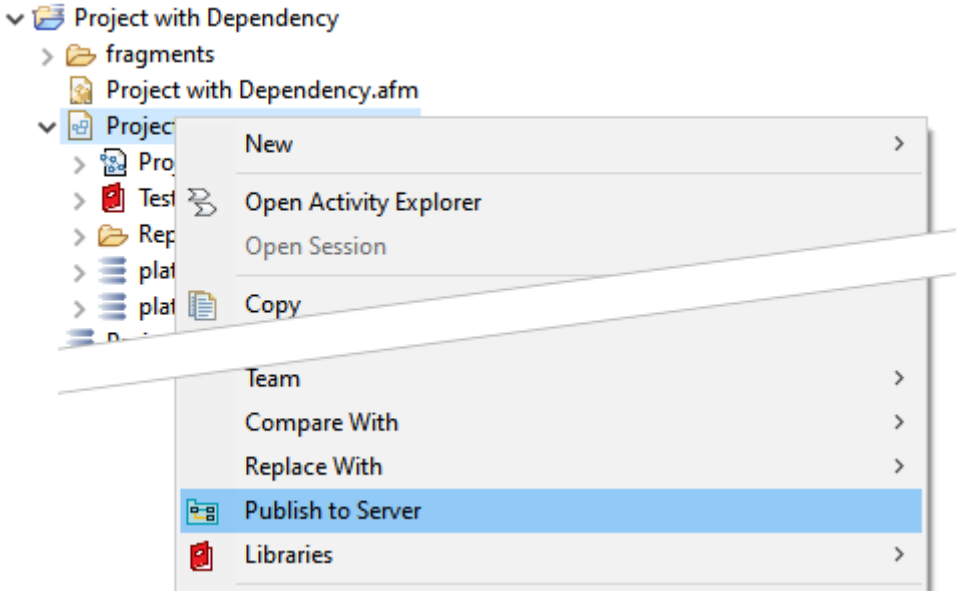


Figure 115. Publication for Capella Publish Menu

- Select the *Publication Server* to which the model should be published. Enter the credentials if needed:

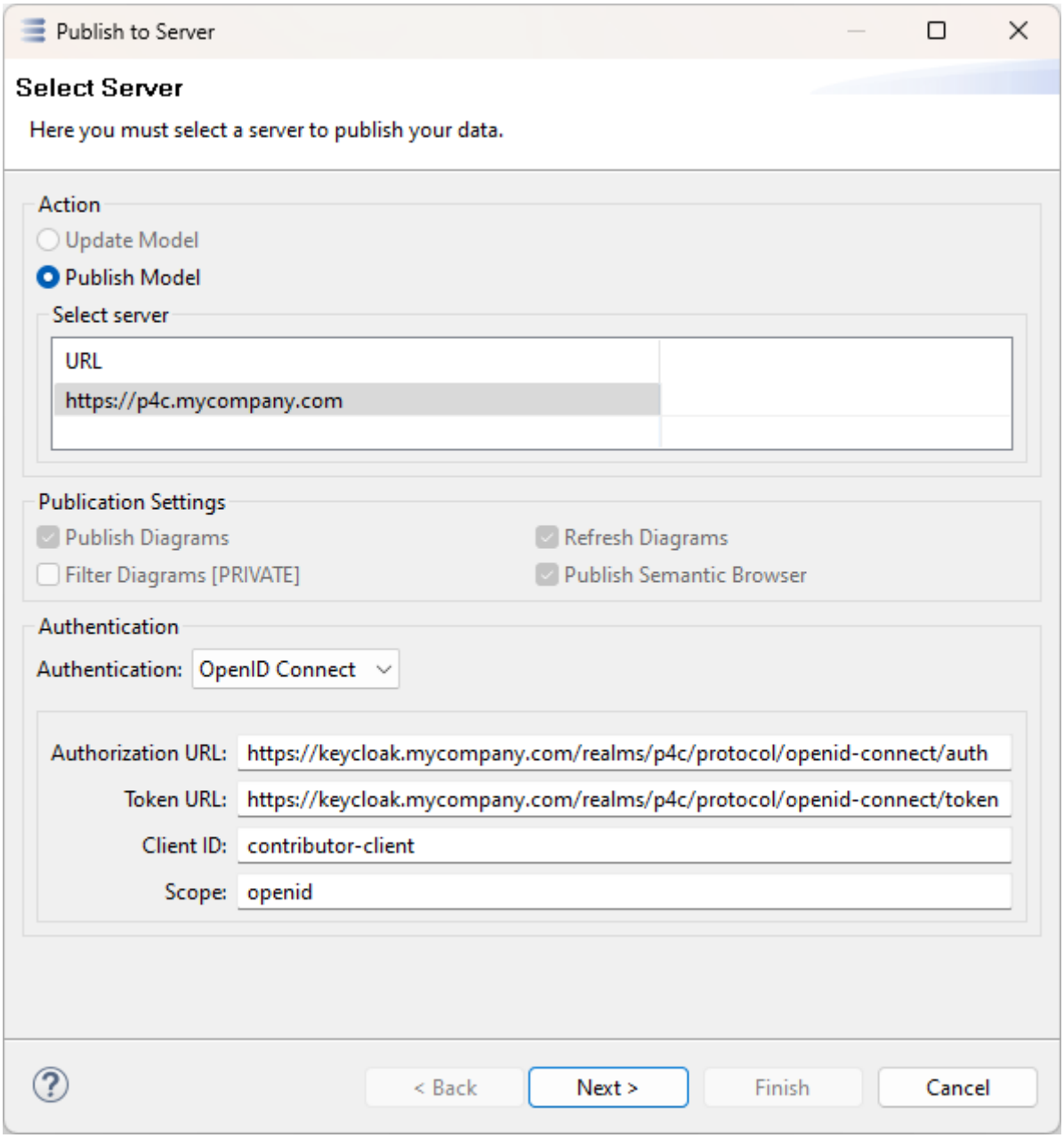


Figure 116. Publication for Capella Publish - Select Server

- Click on 'Next >'.
  - The next page displays the projects available for publication:

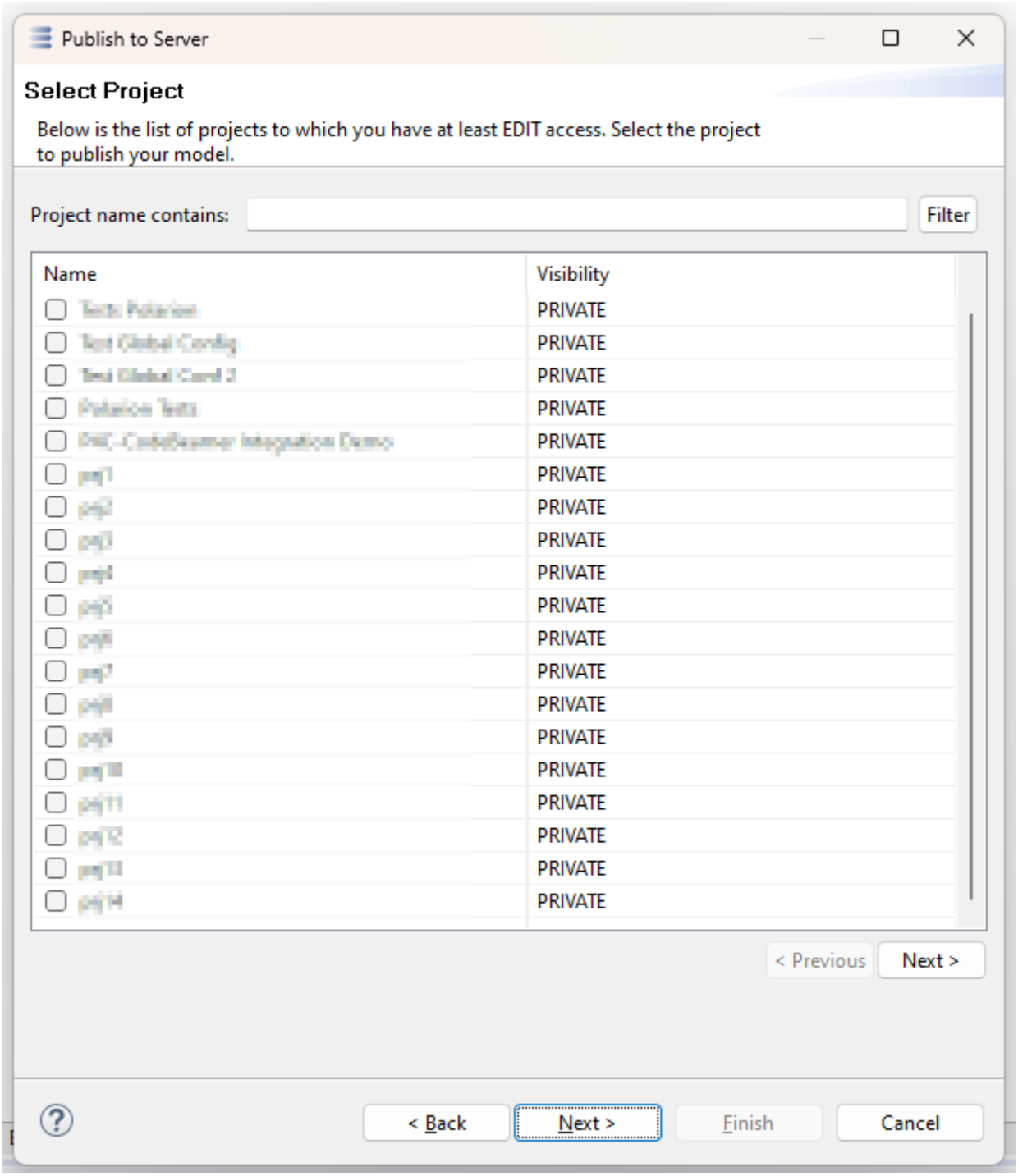


Figure 117. Publication for Capella Publish - Select Project

The list of projects is paginated. It can be filtered with the input at the top and the button 'Filter'. The next or previous page of available projects can be obtained by clicking on the buttons at the bottom of the project list.

Only the projects to which the user has *EDIT* access are displayed. If no project is displayed, then publication cannot proceed. Contact the manager of the targeted project to obtain *EDIT* access.

- Click on 'Next >'.
  - The user is prompted to publish a new model.
    - It is recommended to enter a proper description for the model to publish. This is the

description that will be used in the Main Panel - Model Navigation.

- Enter a commit message, that will appear in the model history (mandatory).

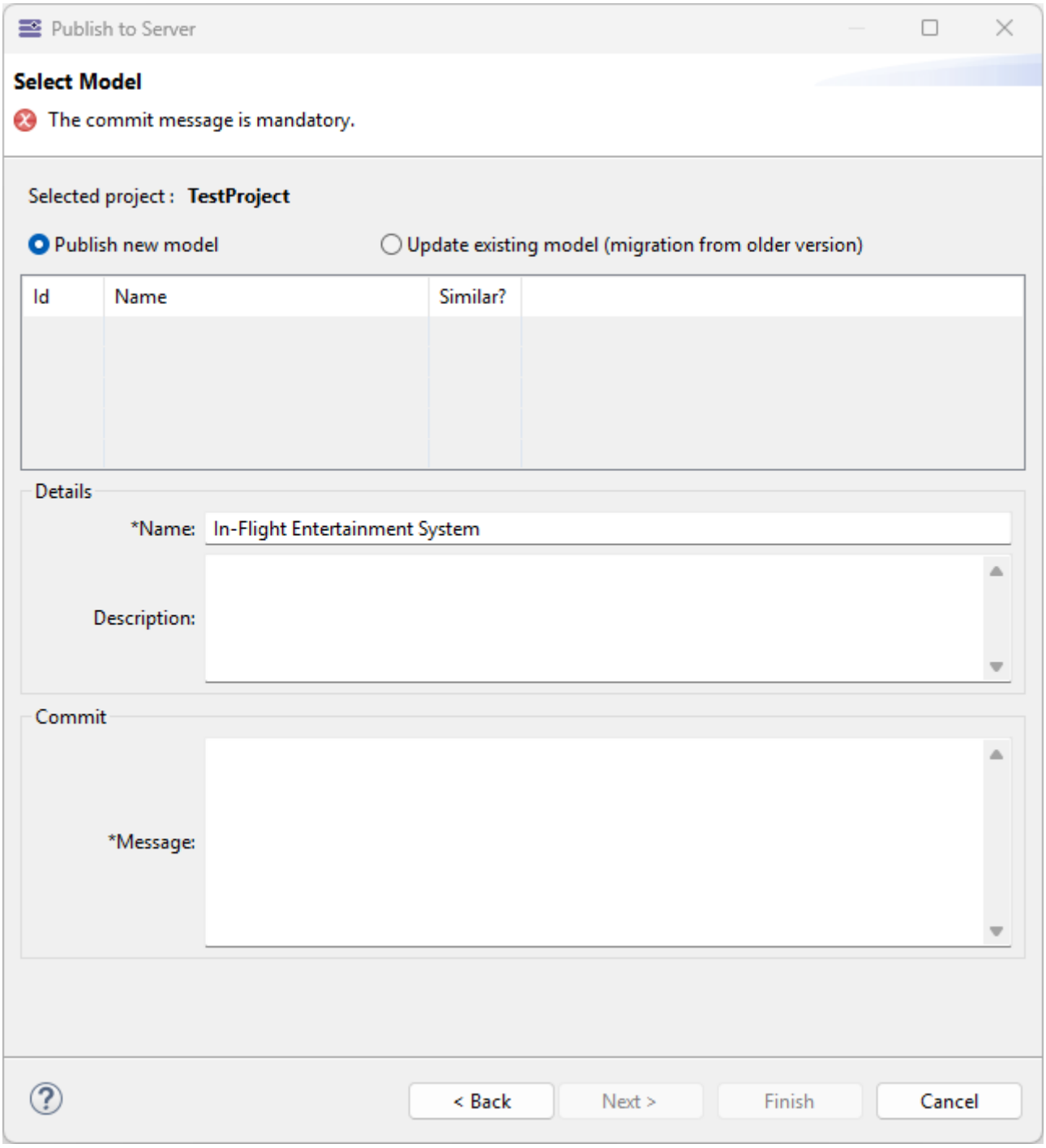


Figure 118. Publish New Model

If at least one similar Capella model already exists on the *Publication Server*, the user is prompted to update an existing model, by default the latest similar model that was updated on the server. This is supposed to be used **only in the case of a migration from a previous version of Publication for Capella**. This makes it possible to 'reconnect' a model published with a previous version of Publication for Capella to the local model. See Publish an Update of a Model of a Previous Publication for Capella Version for more details.

- Click on 'Finish'. The publication starts.

    - Depending on the size of the model, it may take from a few seconds to several minutes to send all the publication data to the server.
    - After the publication data upload is finished, a background process will wait for the publication result. The server will schedule a publication job.
    - On the server side, the jobs page will show the progress of the publication job.
    - Once the job is complete, its label becomes a clickable hyperlink which leads to the published model. The client is notified of the result (success, warning or error) of the publication through a dialog.
- Save the Sirius session to keep track of the Publication traceability information in the local model.

Publish an Update of a Model

Models that have been published once to a *Publication Server* are 'connected' to this server, and keep track of that connection. This connection information is stored in the Sirius session file (the `*.aird` file).

This makes it easier to publish updates of this model.

This also makes it possible to navigate easily from any model object, including diagrams, to their Publication counterpart. See Navigating from Capella to the Publication server for more details.

To publish an update of a previously published model, just use the same menu as for any publication. This opens the publication wizard.

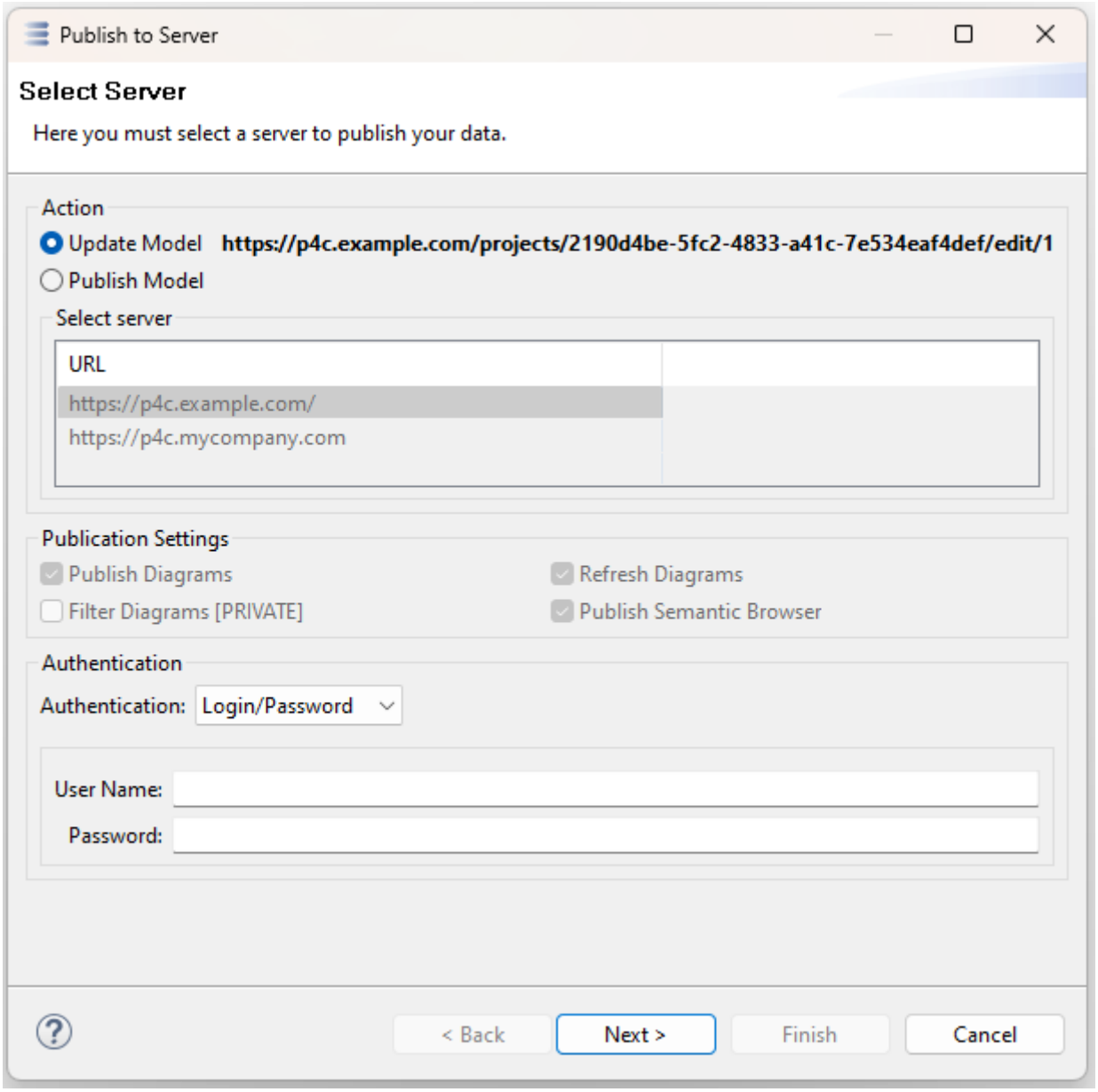


Figure 119. Publication for Capella Publish - Select Server

- On the first page of this wizard:
  - Enter the relevant credentials if needed;
  - Decide whether diagrams must be published;
  - Decide whether semantic browser links must be published.
  - Click on 'Next >'.

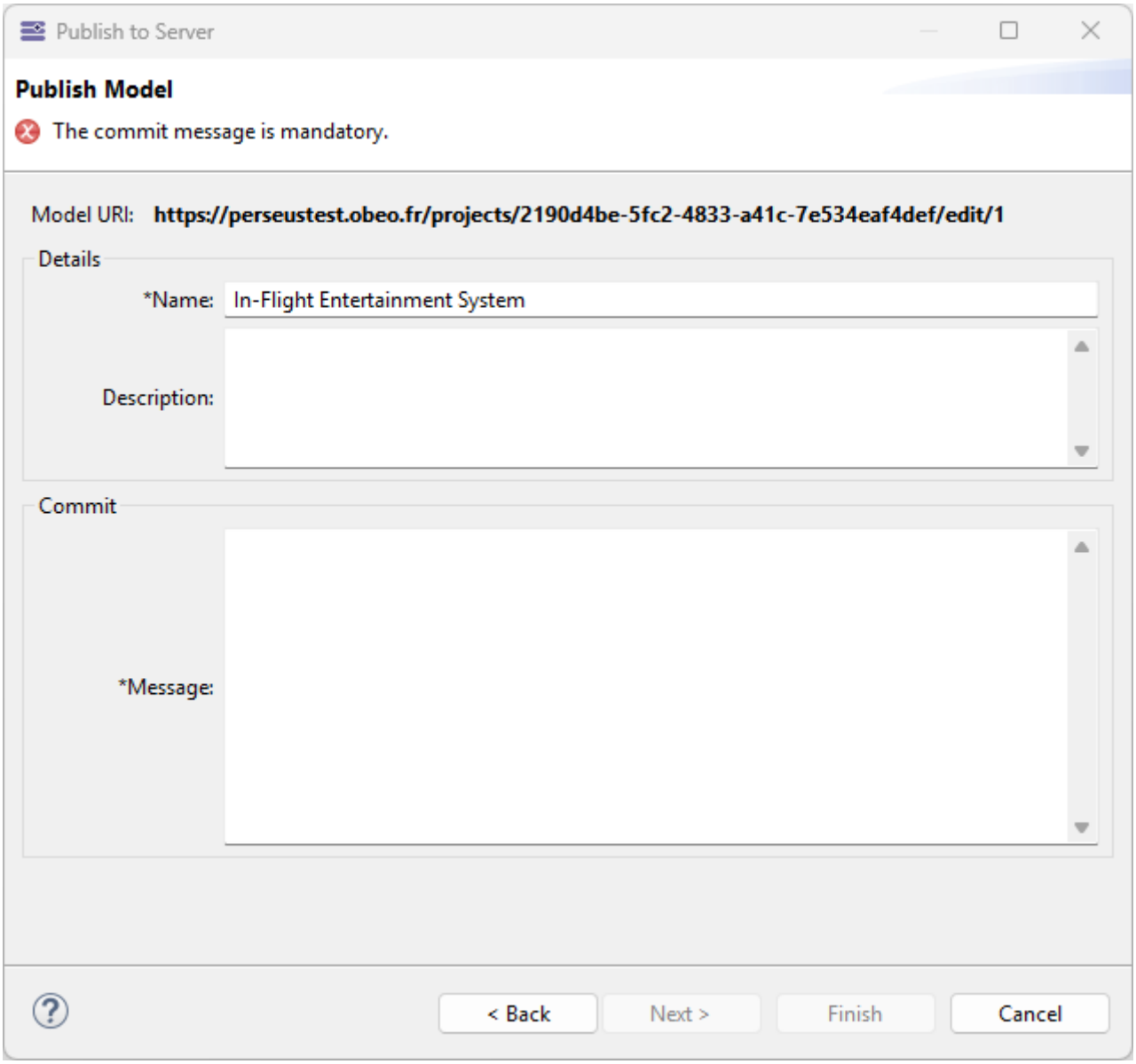


Figure 120. Publication for Capella Publish - Select Server

- On the second page:
  - Modify the name of the model to publish if needed (or leave it as it is)
  - Enter a useful description for that model.
  - Enter a commit message, that will appear in the model history (mandatory). A warning is displayed if the description is blank because a useful description is recommended.

Traceability Links are automatically taken into account and published to the server.

Once the publication result has been received a popup indicates if it was successful or if anything went wrong. If an error occured the publication will be dismissed. If the publication resulted in some links being somehow invalid, a warning will indicate that an action must be performed in the server (see Fixing Invalid Links for more details).

Publish an Update of a Model of a Previous Publication for Capella Version

This section only makes sense if you were previously using Publication for Capella **2022.3.0**, and your Publication environment (servers and clients) has been upgraded to the latest version.

In that case, the Capella models contain no information about their Publication counterpart. To 'reconnect' a project to its Publication counterpart, you must publish it again with the latest version of Publication for Capella.

To publish an update of a previous version model, just use the same menu as for any publication. This opens the publication wizard.

The first step and second step are identical to the Publish for the First Time use-case.

On page 3 of the wizard, the tool should detect that there is an existing model that corresponds to the model you are publishing (See the note on model similarity below). If the tool does not detect that automatically, the user can force the update by selecting the second option labeled 'Update existing model (migration from older version)'

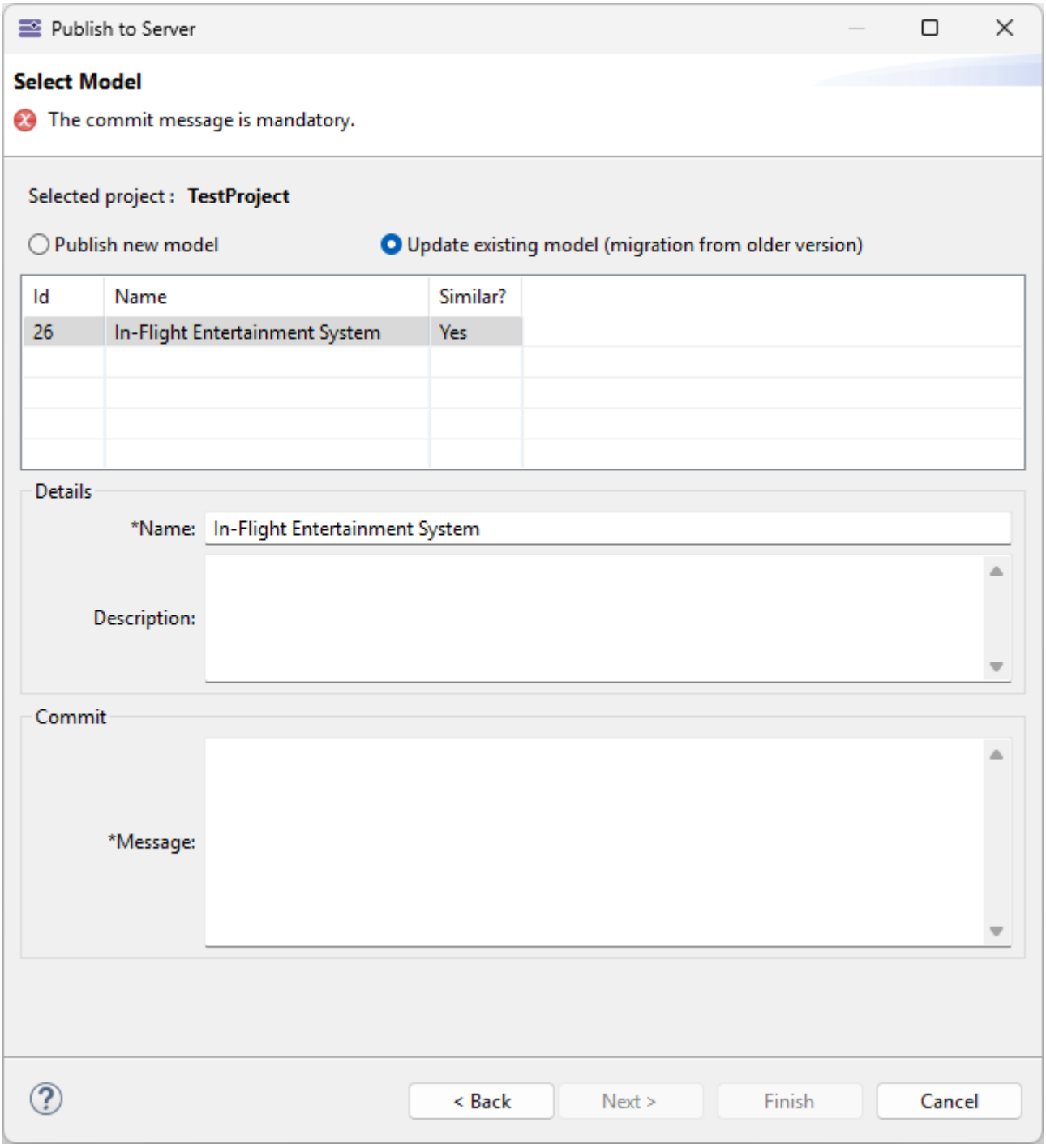


Figure 121. Publication for Capella Publish - Select Project

Click on 'Finish' to launch the publication as usual.

Once that is done, the Sirius session is updated with traceability information and the new *Contributor Client* features are active for that project, especially the navigation feature. As usual, save the Sirius session to keep track of the Publication traceability information.

Incompatibility with Traceability

The process described here must NOT be used with a model that has traceability links. A given Model that contains traceability information must always be published to the same URL in order to always update the same data and not break, duplicate, or corrupt traceability information.

Model Similarity

Similarity is determined by comparing the technical identifiers of the root model objects. That way, two distinct Capella models created independently will never be considered similar. Similar models would rather be different versions of the 'same' Capella model.

It is possible to publish any number of similar models, but then the only way to distinguish between them will be the creation date, last update date, and description.

Latest Synchronization

The traceability model keeps track of ongoing publications and the latest successful publication or update. The latest synchronization point can be visualized inside of the traceability model.

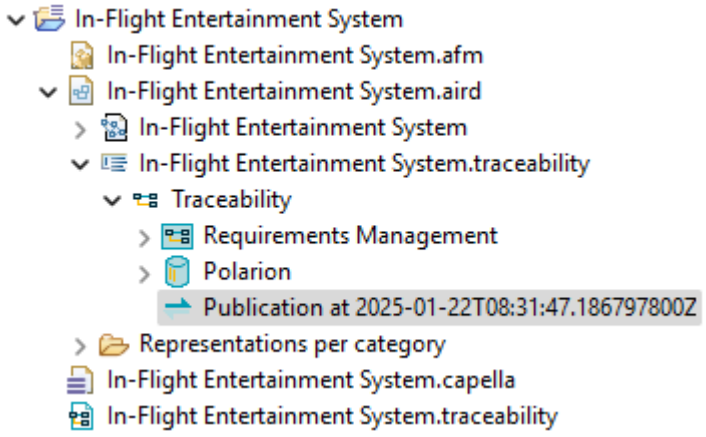


Figure 122. Latest Publication

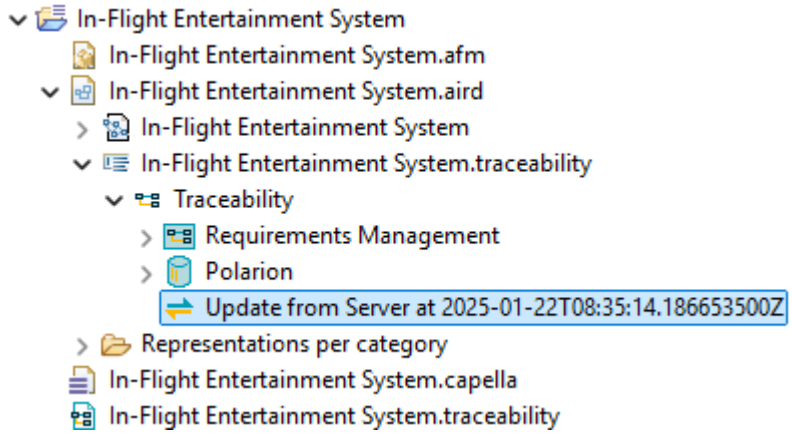


Figure 123. Latest Update

If for some reason the contributor client was closed before a publication ended, on reopening the traceability model will still be waiting for the publication result. To get out of that state, an 'Update publication result' action will fetch the publication result and set the latest publication accordingly.

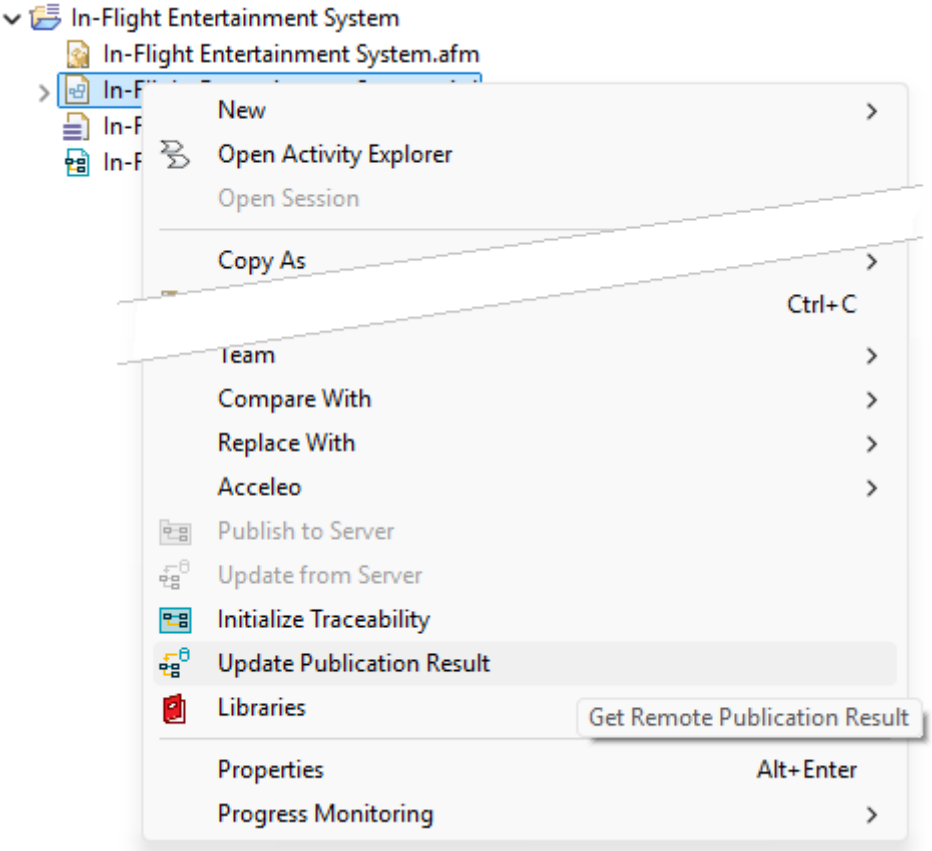


Figure 124. Update Publication Result

Navigating from Capella to the Publication server

Once a model has been published, it is possible to easily navigate from a model element of this model to its online counterpart.

Right-click on any model element, in the Model Tree or in a diagram, and select 'Show in Publication'. That will open the system web browser on the relevant URL on the Publication website.

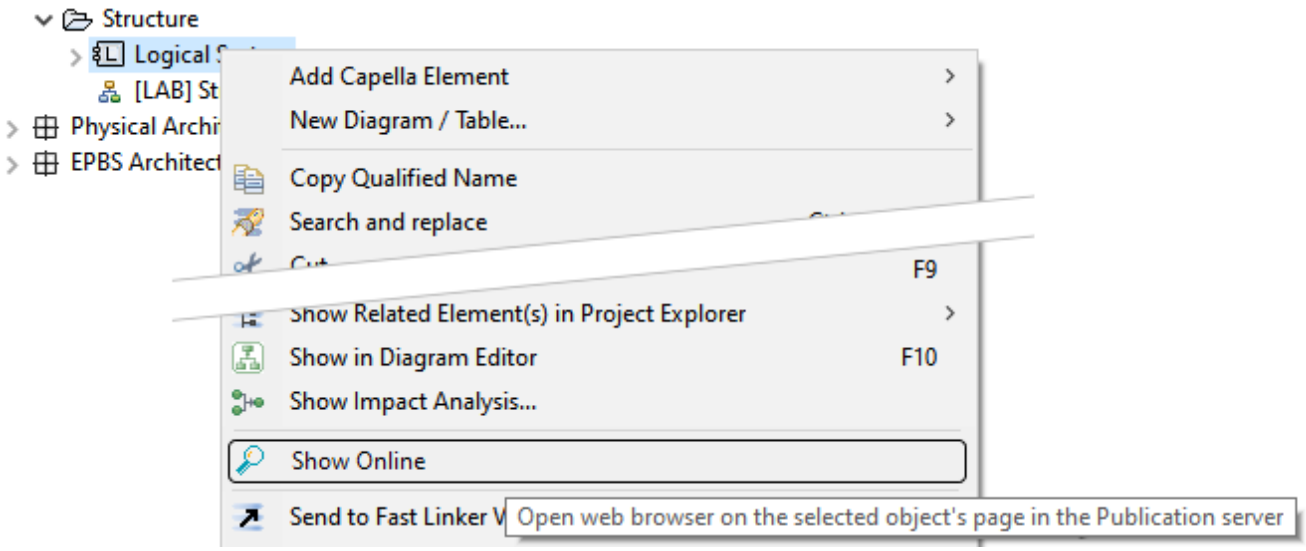


Figure 125. Show in Publication Menu

The user will have to login if they don't have an active session on the *Publication Server* on that web browser.

Semantic Browser Additions

Traceability in the Semantic Browser

Publication for Capella adds a button on the top right of the semantic browser to activate the display of traceability-related elements.

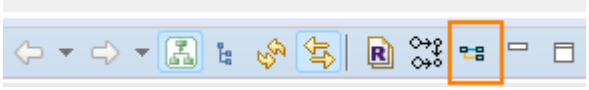


Figure 126. Traceability Button in the Semantic Browser

Once this button is activated, traceability information is displayed in the semantic browser.

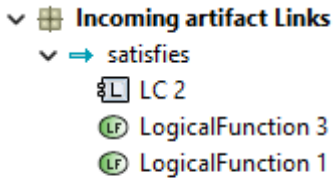


Figure 127. Traceability in the semantic browser panel 'referenced elements'

Improved Property Values in the Semantic Browser

Publication for Capella introduces a specific category in the semantic browser to display all the applied property values of an object in a way that is compatible with the PVMT Capella add-on.

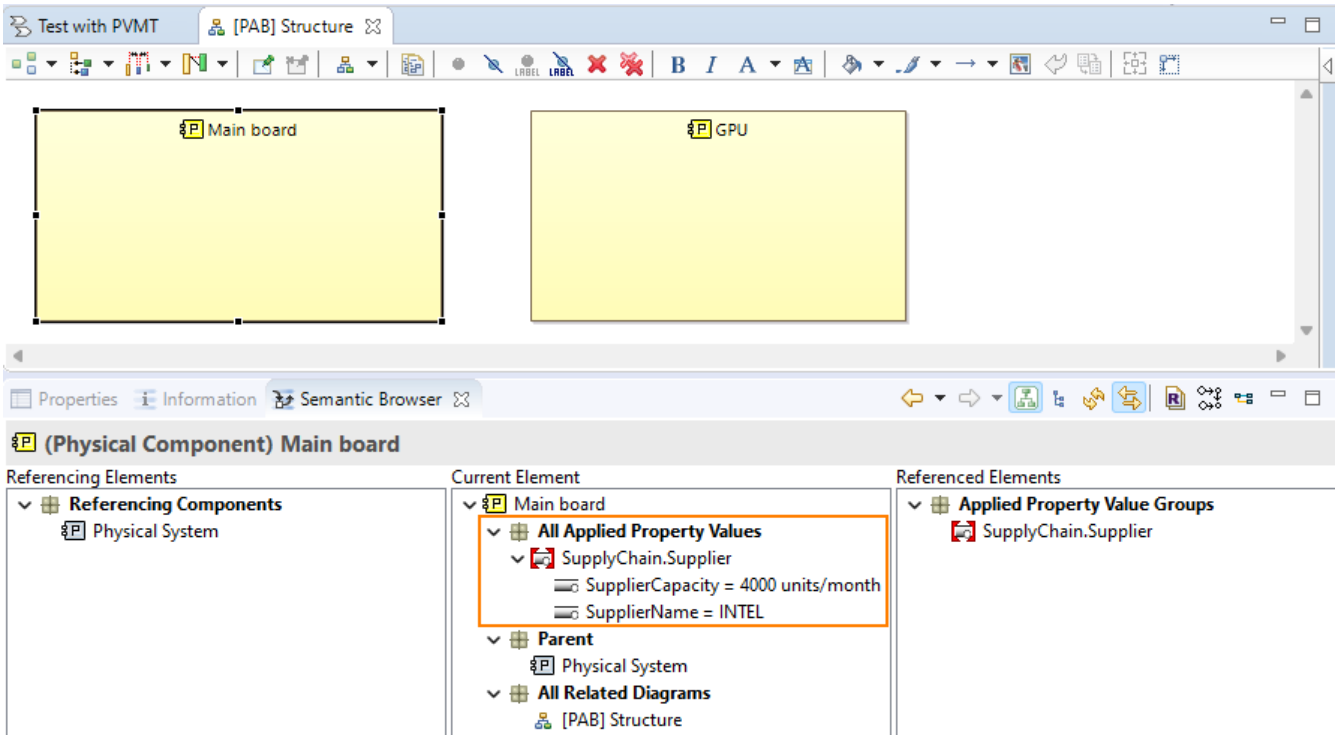


Figure 128. Publication for Capella Category in the Semantic Browser

This provides more information than the standard semantic browser support of property values (visible on the right of the picture for instance). This information is also published to the server so that people who access models online can easily see which property values are applied to a model element.

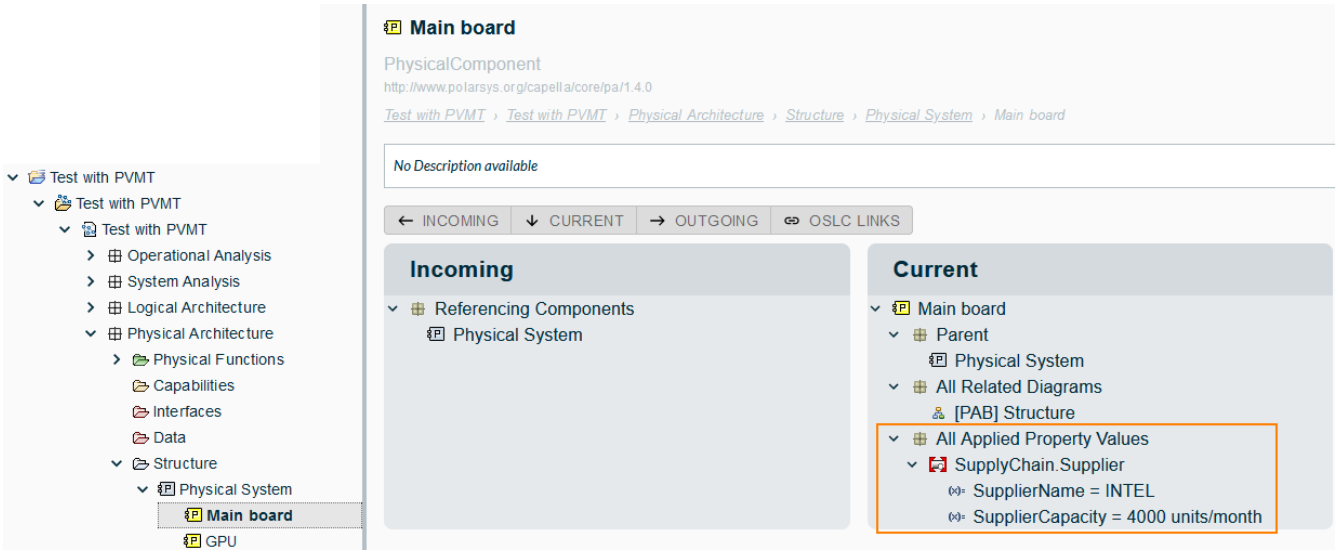


Figure 129. Publication for Capella Category in the Online Semantic Browser

Contributor Client System Properties

This section lists the system properties that can be used to configure the Publication for Capella contributor client.

These system properties can be set in the file `*.ini` of the installation folder of Capella (or other eclipse-based bundle) where the contributor client is installed.

To set a property in the `capella.ini` file, locate the line `-vmargs` and below this line, add a line for

each desired property with the syntax `-Dmy.property.key=desired value`.

For instance, to set the property `fr.obeo.perseus.oauth2.local.server.port` to `55555` (`(⋯)` indicates the possible presence of additional lines):

```
-startup
plugins/org.eclipse.equinox.launcher_1.6.400.v20210924-0641.jar
(...)
-vmargs
(...)
-Dfr.obeo.perseus.oauth2.local.server.port=55555
(...)
```

They have sensible default values and must only be set if the default values don't work.

Property key	Default value	Description
`fr.obeo.perseus.publish.result.fetch.interval.ms`	5000 (5 seconds)	The interval in milliseconds to poll the publication server when a publication is in progress to obtain the publication result.
`fr.obeo.perseus.oauth2.loopback`	`127.0.0.1`	Name of the local server that is use locally to retrieve that OAuth-2.0 authentication token. The best practice is to set it to `127.0.0.1` when IP v4 is used. However, this property makes it possible to use other names, such as `localhost` if the OIDC server doesn't accept redirect URLs with `127.0.0.1`.
`fr.obeo.perseus.oauth2.local.server.port`	`-1`	Port to use for the local server launched by the contributor client to handle the OAuth-2.0/OpenID Connect authentication. Negative values can be used to indicate to use a random port, which is the default behavior. If a specific port must be used, it should be in the range [`49152`–`65535`] (See https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers#Dynamic,_private_or_ephemeral_ports)
`fr.obeo.perseus.oauth2.login.timeout.ms`	`10000`	Timeout in milliseconds of the OAuth-2.0/OpenID Connect login.
`fr.obeo.perseus.oauth2.use.id_token`	`false`	Set this to `true` to use the `id_token` instead of the `access_token`. This is necessary with some third-party authentication providers that don't provide the JWT as the `access_token`, but provide it as the `id_token`.

Known Limitations in the Contributor Client

- The client does not publish tabular representations (such as tables and cross-tables), only diagrams can be published;
- When opening the web browser after publication, the user will not be automatically logged in;
  - If there is no open session on Publication in the browser, the user will be prompted for credentials.
- Initializing a Traceability model must be done with registered initializers. Traceability models can be edited easily to be adapted to the mapping. Create the model with the menu `File\new\other\Example EMF Model Creation Wizards\Traceability Model` must not be used. The model will not be fully initialized and will not be connected to the Capella model. In this case all actions will fail.
- Dropping a URL for link creation requires:
  - An initialized Traceability model.
  - That the model was already published to Publication for Capella.
  - Dropping from *Jazz Quality Manager* is not supported yet.
- Before updating model from server, all the link types and artifact types used in the Publication for Capella server must be created in the Traceability model. If types are missing, the Update might fail.
- There is no automated way of aligning the OSLC linking settings of the Publication server and the Traceability settings of a Traceability model.
- Depending on the Capella version where the *Contributor Client* is installed, diagrams may be improperly displayed and/or exported during publication. In such a case, the currently recommended workaround is to restart Capella with a display resolution set to 100%, and only change the screen resolution once Capella has properly started. See https://forum.mbse-capella.org/t/diagram-graphics-problem/6681 for more details.
- When used within a *Team for Capella* bundle, the contributor client may cause errors when closing a Capella model session. The symptom is that in the Project Explorer view, the node `*.aird` will still display children nodes even though the session is actually closed. The Error Log will contain messages like `org.eclipse.net4j.util.lifecycle.LifecycleException: Not active: CDOTransactionImpl`. However, this problem does not imply loss of data or data corruption. The only known workaround is to close and reopen the project explorer view. This will be fixed in Capella 7 and the associated *Team for Capella* release.
- CodeBeamer Integration Limitations:
  - The URLs of CodeBeamer artifacts do not contain any information about the project that contains these artifacts. Consequently the integration is currently limited to being used with only one CodeBeamer project for a given Capella model.
  - CodeBeamer does not provide the correct URL for its artifacts, which prevents the action 'Show online' from working properly on CodeBeamer artifacts.

Using Publication for Capella OSLC API

Preamble

This document provides examples that assume that the Publication for Capella server is deployed on `http://localhost:8080`. Of course, this is not valid in general, especially not in production.

So to transform the examples in something valid for a given deployment of Publication for Capella, the URLs must be transformed by replacing `http://localhost:8080` with `https://p4c.mycompany.com` where `p4c.mycompany.com` is the actual DNS name of the Publication for Capella server. The port must be included (e.g. `https://p4c.mycompany.com:8443`) if Publication for Capella is deployed on e.g. port 8443 instead of the default 443 HTTPS port.

OSLC API

Logging In

A proper way to login via API has not been implemented yet. Currently the workaround to login to Publication for Capella via API so as to be able to query the contents of published models is to call `GET http://localhost:8080/api/graphql` and pass the header for Basic Authentication and the header `x-perseus-client`.

This will return an error response that, if the credentials given in the Basic Authorization header are correct, contains a `Set-Cookie` header for `JSESSIONID` that must be set for the next HTTP requests.

Example:

```
x-perseus-client: true
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==

GET http://localhost:8080/api/graphql
--->
Set-Cookie: JSESSIONID=7C2D4DC2F34DFC7E8B2EC00B3481EE7; Path=/; SameSite=None; Secure
Allow: POST
(Additional headers)

(JSON content)
```

Model Elements as RDF

Each element of a published model is exposed at a unique and stable URL that is based on:

- The project ID `{project-id}`;
- The model ID `{model-id}`;
- The model object ID `{object-id}`.

Such a URL can be computed easily as follows: `http://localhost:8080/oslc/model/{project-id}/{model-id}/{object-id}`.

The model object is provided in RDF format. Publication for Capella supports the following content types: `application/rdf+xml`, `text/turtle`, `application/xml`, `text/xml`. The desired content type can be requested using the request header `Accept`.

Example:

```
Accept: application/rdf+xml

GET http://localhost:8080/oslc/model/fa9644dd-9c17-4b87-82f8-
cd72b70f4fa1/35/_QHUAIJY7Ed6rH5SGn-UjSg
```

RDF is a widespread data format and there are a number of off-the shelf libraries to parse RDF in many programming languages.

Mapping of Capella Model Element EAttributes and EReferences with RDF Properties

Publication for Capella applies a number of rules to map the attributes and references of Capella model elements into their RDF representation.

Table 1. EMF vs RDF Mappings

Capella	RDF	Query Criterion
`identifier`	`dcterms:identifier`	Yes
`name` (NamedElement)	`dcterms:title`	Yes
Meta-class name	`dcterms:type`	Yes
eContainer (parent object)	`pers:parent`	No
model ID (numeric ID of the model that contains a model object)	`pers:modelId`	Yes
Any EAttribute `attName`	`pers_mm:attName`	Yes
Any EReference `refName`	`pers_mm:refName`	Yes
Latest modification date	`dcterms:modified`	No

These resurces aer all exposed as resources of typ `oslc_am:Resource`, in compliance with the OSLC Architecture Management vocabulary.

OSLC Query

Querying the contents of Publication for Capella projects (a project containing zero to many published models) is achieved by performing a HTTP GET request on the project's OSLC Query URL. This URL is built from the project's identifier as follows: `http://localhost:8080/oslc/model/{project-id}/q?` where `{project-id}` must be replace by the

actual project identifier. The query criteria must be added after the question mark of the URL `?` and encoded using standard URL query encoding.

For more details on the OSLC query syntax, please refer to the OSLC Query Version 3.0 specification.

The query result is provided in RDF format. Publication for Capella supports the following content types: `application/rdf+xml`, `text/turtle`, `application/xml`, `text/xml`. The desired content type can be requested using the request header `Accept`.

Example:

```
Accept: application/rdf+xml

GET http://localhost:8080/oslc/model/3a94f683-ea4b-4514-8583-
3cb688b3da78/q?oslc.where=(...)
```

Here are some examples of `oslc.where` clauses for some use-cases:

- Get all the model elements linked to a specific DOORS Next Requirement:
  - `*=https://localhost:9443/rm/resources/TX_6uRrAAQ1Ee6mray9nPRAQw`
- Get all the non-actor LogicalComponents in a specific model:
  - `pers:modelId=2 and dcterms:type="LogicalComponent" and pers_mm:actor=false`
- Get the realizing PhysicalComponents of a given LogicalComponent:
  - `pers_mm:realizedLogicalComponents=http://localhost:8080/oslc/model/fa9644dd-9c17-4b87-82f8-cd72b70f4fa1/35/_QHUAIJY7Ed6rH5SGn-UjSg`

Invalid OSLC queries

There are a number of restrictions as to what can be queried, either for security reasons or for technical reasons.

Invalid query across projects

One restriction is that it is not allowed to query data across several projects. Queries a run in the context of a specific project and can only access this project's data.

Example:

```
GET http://localhost:8080/oslc/model/3a94f683-ea4b-4514-8583-
3cb688b3da78/q?oslc.where=pers_mm%3ArealizedLogicalComponents%3D%3Chttp://localhost:80
80/oslc/model/fa9644dd-9c17-4b87-82f8-cd72b70f4fa1/35/_QHUAIJY7Ed6rH5SGn-UjSg%3E
```

This request (where `oslc.where` is `pers_mm:realizedLogicalComponents=http://localhost:8080/oslc/model/fa9644dd-9c17-4b87-82f8-cd72b70f4fa1/35/_QHUAIJY7Ed6rH5SGn-UjSg`) gets rejected.

It is rejected because it is run in the context of project `3a94f683-ea4b-4514-8583-3cb688b3da78` but

requests information about a model element located in a different project `fa9644dd-9c17-4b87-82f8-cd72b70f4fa1`.

Administration of Publication for Capella

Account Management

Administrators can create user accounts and set the account privileges.

To create an account or manage existing accounts, the administrator must login and then select the menu entry 'Accounts' in the *User Menu*.

Accounts Page

The Accounts page lists the existing accounts.

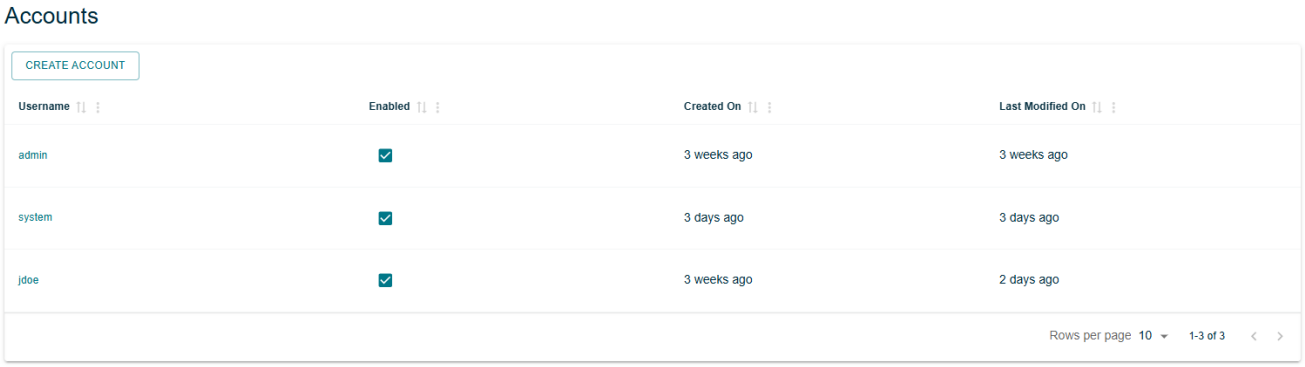


Figure 130. Accounts Page with default accounts

For each account the page lists:

- **Username**: the unique user name used to log in.
- **Enabled**: whether or not the account is currently enabled.
- **First Name** and **Last Name**: the name of the user.
- **Created On**, **Modified On**: the dates when the account was first created, when it was last modified.

Click on 'Create Account' to create a new Account. See Account Creation for more details.

Account Creation

Creating accounts only makes sense when not using OpenID Connect for authentication. If OpenID Connect is used for authentication, then accounts need not and must not be created manually. They will be created dynamically the first time a user logs in and authenticates successfully with the OpenID Connect authentication server.

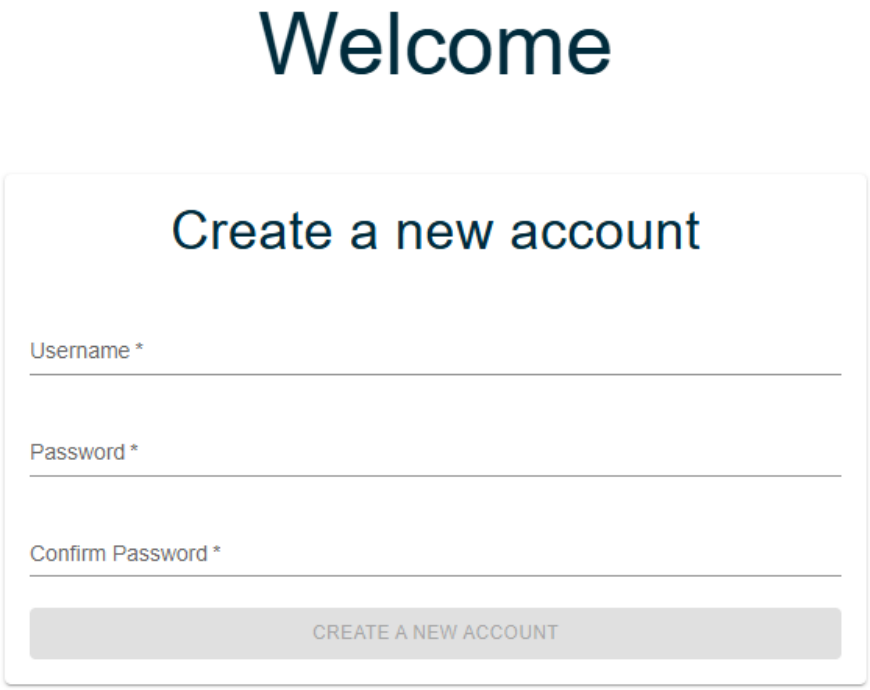


Figure 131. Account Creation Form

Figure 132. Account Creation - Password constraints

The password must contain at least 5 characters.

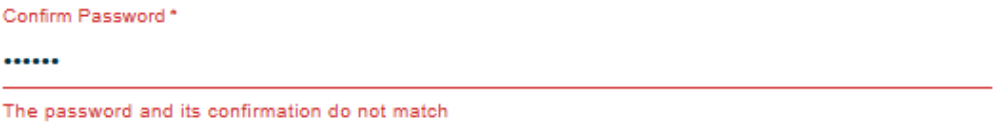


Figure 133. Account Creation - Password confirmation

The confirmed password must be identical to the first password.

Once the form is ready, the 'Create account' button is enabled unless some fields are incorrect. Click on the 'Create account' button to create the new account.

once created, an account cannot be deleted. This is on purpose to keep track of the user's contributions. However, an account that is no longer relevant can be disabled by the server administrator.

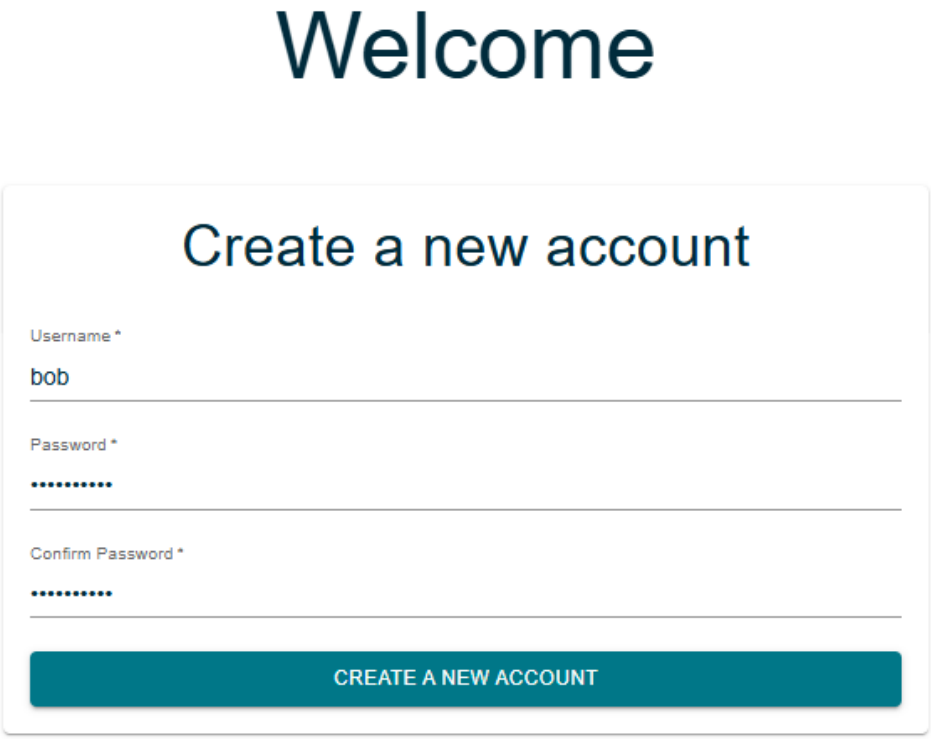


Figure 134. Account Creation ready to be submitted

On successful creation user is redirected to the accounts list page. Newly created accounts are automatically enabled.

Account Enablement

Each account can be enabled or disabled thanks to the check-box in the accounts list. Newly created accounts are enabled by default. When a user account is disabled, the corresponding user will not be able to login anymore but the account is still present and can be re-enabled by an administrator at a later date.

Disabling a user account will only have an effect the next time the user tries to login, but it will not immediately invalidate the user's session if the user is connected.

If OpenID Connect is used, the user account should also be disabled in the OpenID Connect authentication server.

Account Details Page

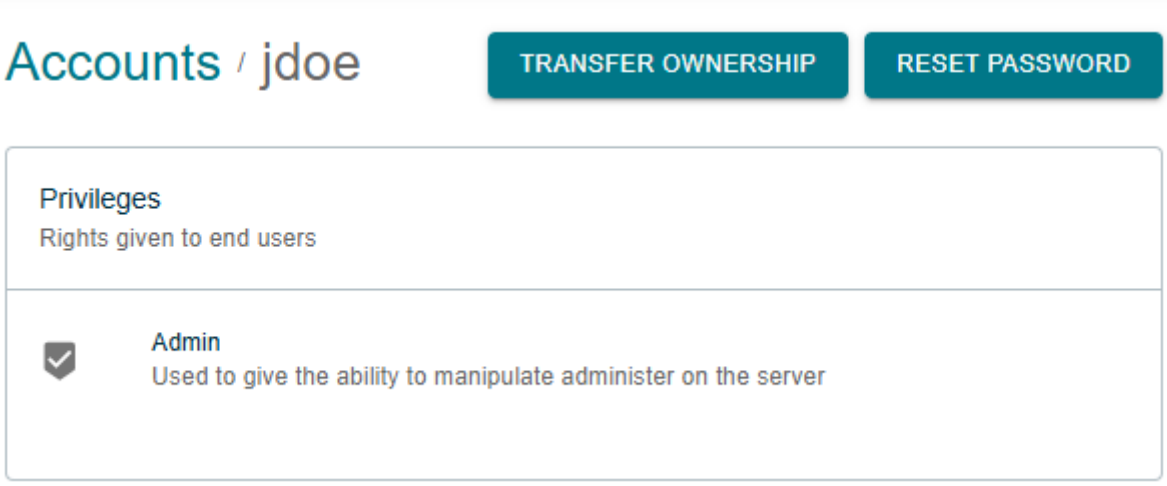


Figure 135. Account Details

The 'Account Details' page displays the privileges granted to the account, if any.

It also proposes the following actions:

- Reset Password;
- Transfer Ownership.

Resetting Account Password

The `admin` user can reset the password of any user account (including his own) that is configured to use username/password credentials.

Resetting the password of accounts configured to login with OpenID Connect is an operation that must be performed in the OpenID Connect Authentication Tool, not in Publication for Capella. Attempting to reset the password of such an account in Publication for Capella will raise an error message and have no effect.

Click on the 'Reset Password' button.

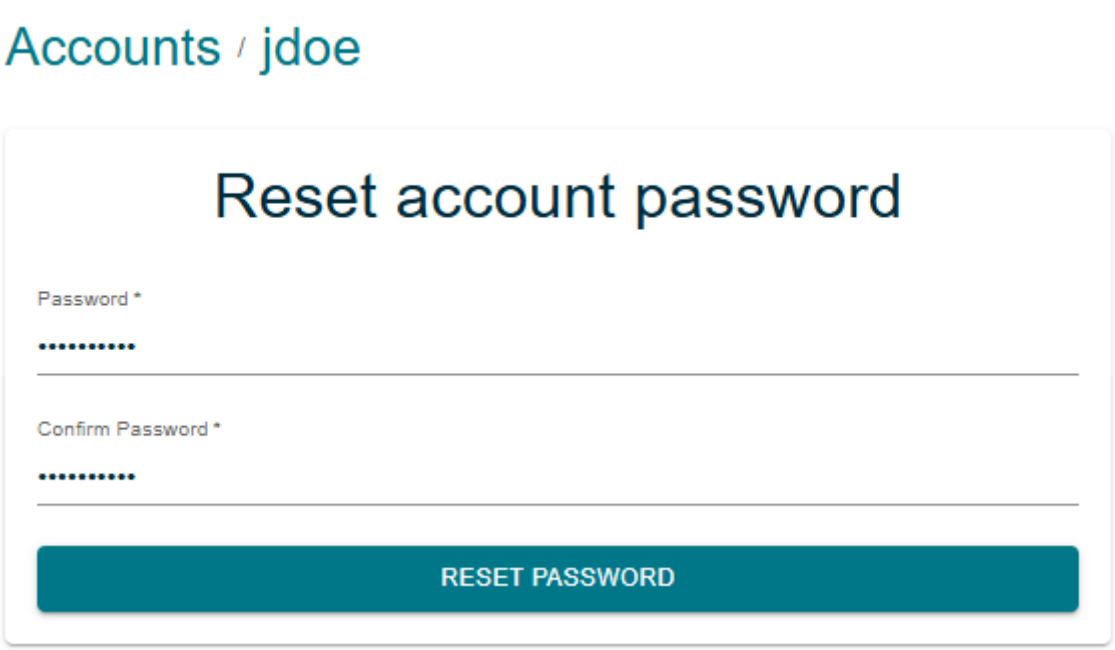


Figure 136. Reset Password

Enter the desired password, confirm it in the second text box, and click on 'Reset Password' to confirm.

The 'Reset Password' button is only active if the two password match.

The new password is active immediately and will be checked on the next login attempts of the user. However, if the user is currently logged in, they will not be required to verify their password until their session expires or they logout.

Transferring Ownership

Users can *own* projects or teams. This *ownership* can be transferred to another user, if needed.

Who's the Owner of this Project?

The owner of a project is displayed on the welcome page, as a little round icon that can be hovered with the mouse to get the owner's username.

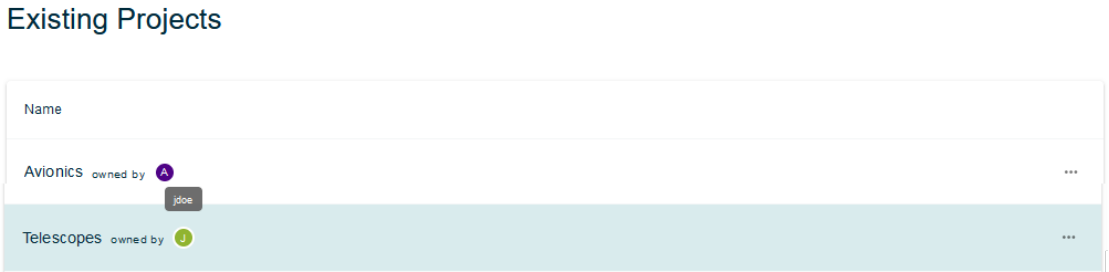


Figure 137. Owner of a Project

When a user creates a project or a team, he or she becomes its owner. The owner of a project can define a team that will grant administration rights to another user. Only the owner of a team can manage it.

The `admin` user can transfer the ownership of teams and projects of a user, to another user.

Transferring ownership form `user1` to `user2`:

- Makes `user2` the owner of all the *Projects* `user1` owned.
- Makes `user2` the owner of all the *Teams* `user1` owned.
- Removes `user2` from all these *Teams*, since as new owner they don't need to be explicitly in any of these *Teams*.

It is possible to disable the account of `user1` after transferring their ownership if `user1` must no longer access Publication for Capella.

Click on the 'Transfer Ownership' button.

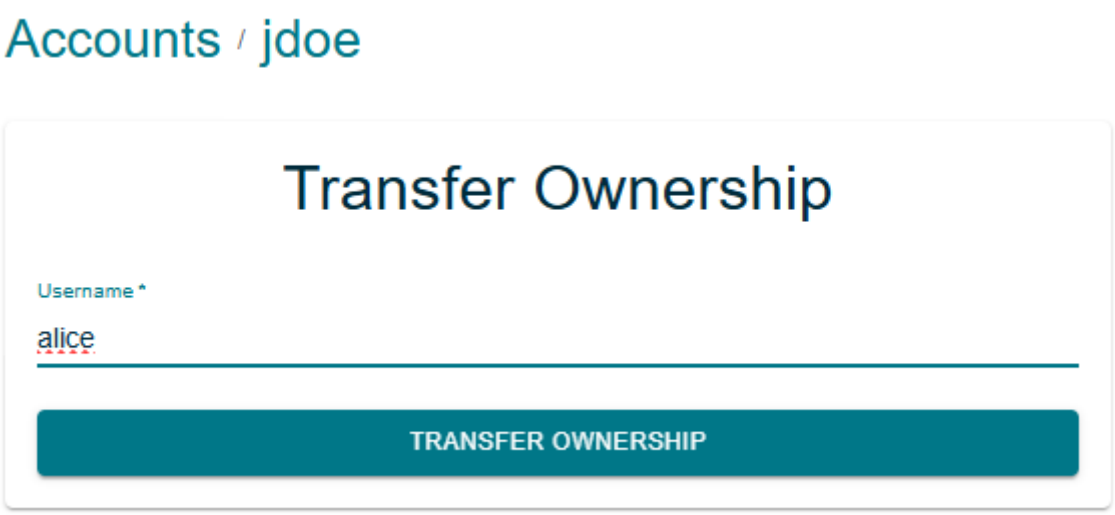


Figure 138. Transfer Ownership

Enter the *username* of the account that will become the new owner, and click on 'Transfer Ownership' to confirm.

Licenses Management

Server administrators can register licenses and grant privileges to their users. A license contains a set of privilege tokens which can be assigned to the server accounts.

The license system makes it possible to grant privileges to users. These privileges will restrict the access levels users can be granted on projects using teams (See Teams Management).

Here are the defined privileges:

- `Read`;
- `Write` (also called `Contributor`);
- `Admin`.

The effective rights granted to the user are computed according to the following table, taking into account both the Teams in which the user is, and the license assigned to the user.

For a given project:

Assigned License → / Team status ↓	`Read`	`Write`	`Admin`
in no team	do nothing	do nothing	do nothing
in team(s) with no access to this project	do nothing	do nothing	do nothing
in a team with access `Read` to this project	read	read	read
in a team with access `Write` to this project	read	read, edit	read, edit
in a team with access `Admin` to this project	read	read, edit	read, edit, administer

In other words, the access right to a specific project is computed from the Teams the user belongs to, within the limit granted by the license permissions assigned to this user.

To access the licenses management page, click on the user icon:

Figure 139. User Icon

Then in the user menu, click on 'Licenses':

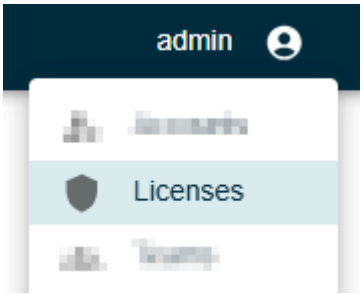


Figure 140. Licenses Menu

The Licenses page is then displayed:

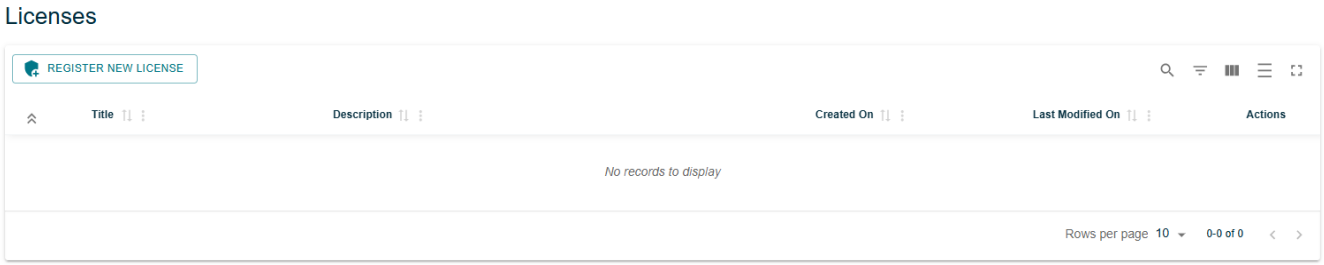


Figure 141. Licenses Page

License Registration

To register a license, click on the 'Register New License' button.

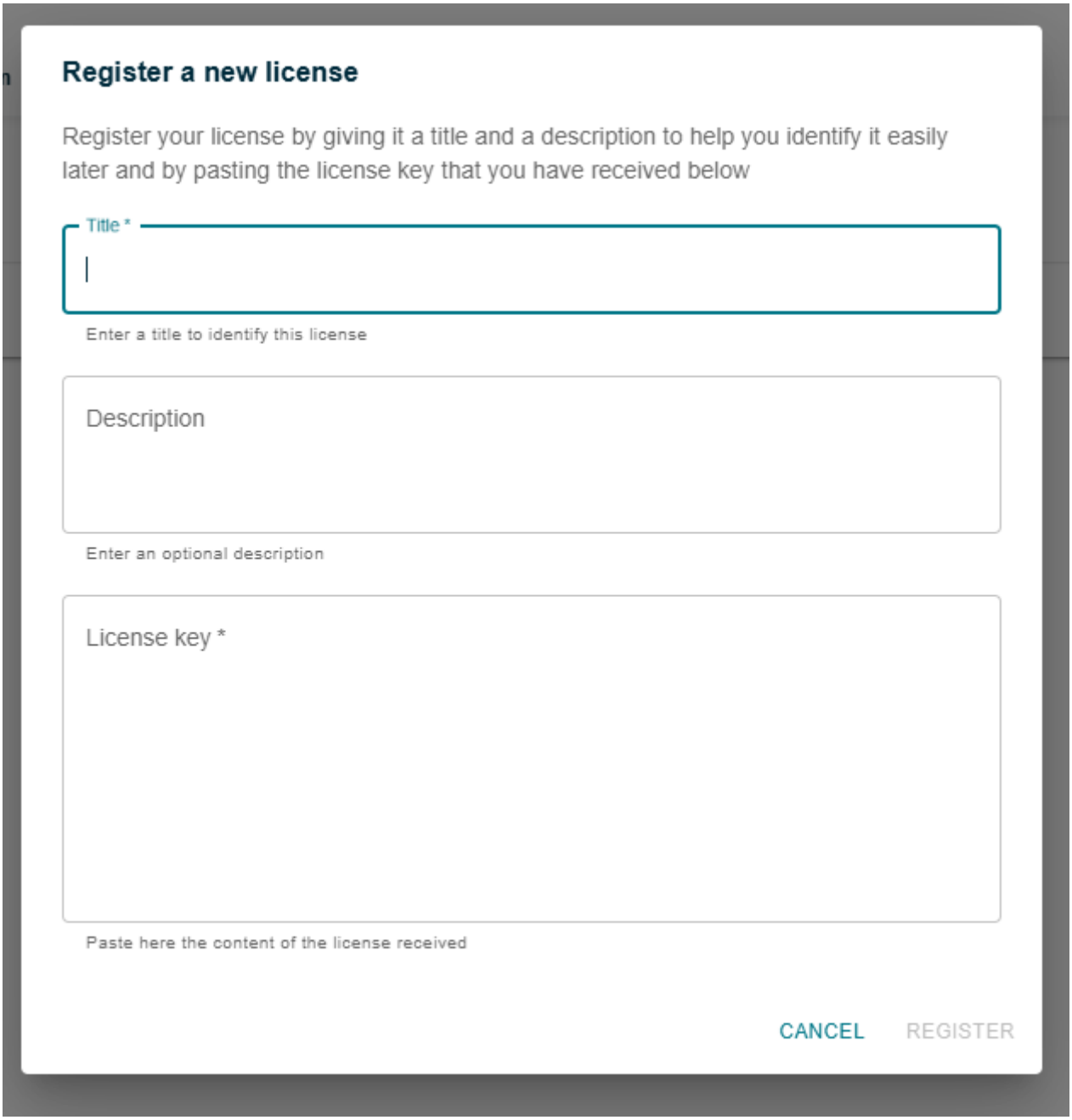


Figure 142. License Registration Dialog

Once registered, the license displays its available tokens:

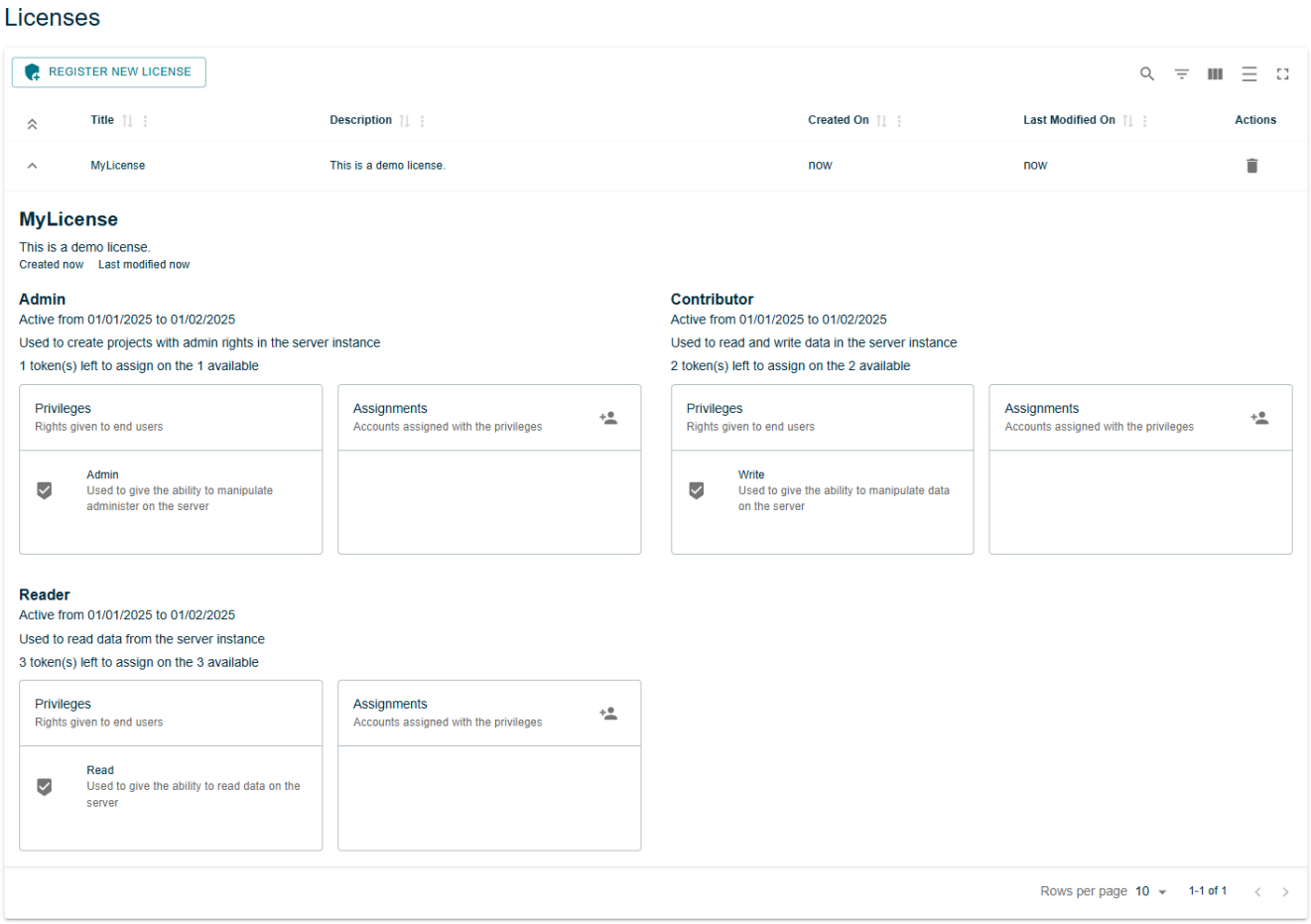


Figure 143. Installed License

Once registered, the license name and description cannot be updated. The license has to be deleted then registered again to set the desired name and description.

Token Assignment

If a user logs in and has no active license permission at all, Publication for Capella will attempt to automatically assign a license token with permission `Read`, if there is one available. This will happen if the user logs in for the first time, or if a user's assigned token is expired. A user who was previously assigned more than `Read` will not recover the permission they had, but will at least be able to access the server's contents.
This is intended to minimize the administrative overhead of managing the license tokens.
This behavior can be turned off by setting the property `perseus.license.auto.assign.read.on.login` to `false`.
If the behavior is deactivated then no automatic license token assignment will be performed, and the `admin` user must manually assign all license tokens, including `Read` tokens.

To assign a token to an account, click on the '+' button in the related 'Assignments' section.

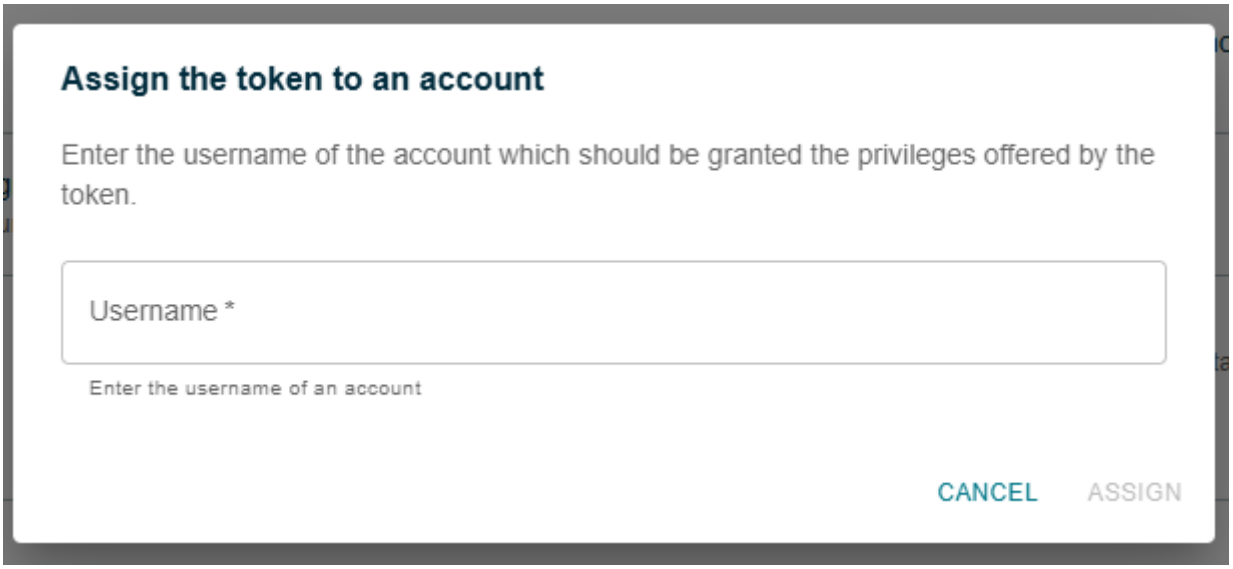


Figure 144. Token Assignment Dialog

Enter the *user name* in the field and click on the 'Assign' button.

There is no auto-completion on the user name yet. The user name to enter is the one that is displayed to users on the top right of the web UI (which could be opened in another tab of your browser).

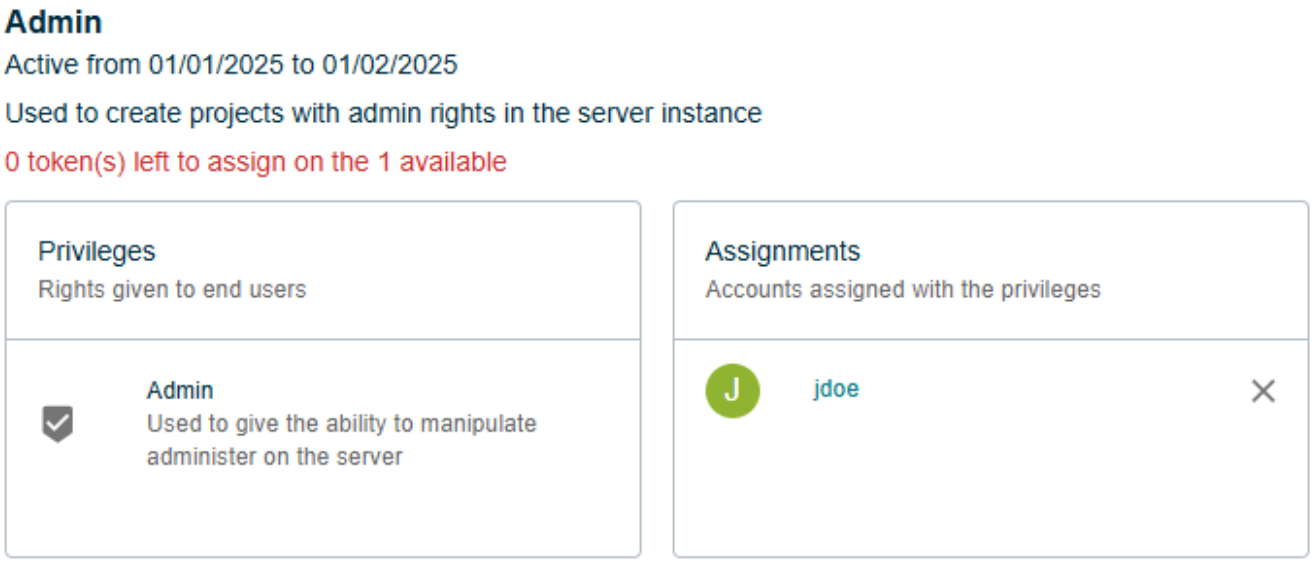


Figure 145. Assigned Token

Authentication

Publication for Capella support two end-user authentication mechanisms:

- Plain username/password authentication.
  - The account `admin` can only use this authentication mechanism.
  - Other user accounts can be created by `admin`.
- OpenID Connect authentication.
  - Setting up OpenID Connect is optional, but recommended.

- A user account is created automatically on the Publication for Capella server when a new user successfully logs in via the OpenID Connect process.

Content Security Policy White List

When answering HTTP requests, Publication for Capella automatically adds the list of friends in the `frame-ancestors` directive of the `Content-Security-Policy` header of its responses. Additional white listed URIs can be added to that header in the `config/csp-whitelist.json` file as a list of string, e.g.: `["http://some_host"]` or `["http://some_host","http://some_other_host"]`. The file location can be set using the property `perseus.csp.whitelist` in the file `config/application.properties`.

Any web site that should be able to embed Publication for Capella pages in `iframe` elements and that is not declared as an OSLC friend should be declared in this white list.

Application Properties Summary

This section lists the properties available for a server administrator to configure Publication for Capella.

These properties need to be set in the file `config/application.properties` of the server.

They have sensible default values whenever it is possible.

Properties Inherited from Underlying Frameworks

General Application Server Properties

Property key	Default value	Description
`server.port`	443	The port the server must listen to.
`server.ssl.key-store-type`		Certificate type, e.g. PKCS12
`server.ssl.key-store`		Path to the SSL keystore file, e.g. `config/keystore_file`
`server.ssl.key-store-password`		Password of the SSL Keystore file
`server.ssl.key-alias`		Alias of the SSL key in the keystore file
`server.ssl.key-password`		Password of the key in the file (generally identical to the file's password)
`spring.datasource.url`	`jdbc:postgresql://perseus-server-postgres:5432/perseus`	The JDBC URL of the Publication for Capella database.

OpenID Connect Properties

The following properties need to be set only if authentication with OpenID Connect is desired.

Table 2. Spring Security Framework - OAuth2 Properties

Property Key	Description
`spring.security.oauth2.client.provider.keycloak.issuer-uri`	The issuer URI of the OpenID Connect server, which can be used to discover other URIs (authorization URI, token URI, User Info URI, and JWK Set URI). For example, when using keycloak, `http://keycloak.example.com/realms/Publication`. If this property is set, the 4 next properties are not necessary.
`spring.security.oauth2.client.provider.keycloak.authorization-uri`	The authorization URI of the OpenID Connect server. For example, when using keycloak, `http://keycloak.example.com/realms/Publication/protocol/openid-connect/auth`.
`spring.security.oauth2.client.provider.keycloak.token-uri`	The token URI of the OpenID Connect server. For example, when using keycloak, `http://keycloak.example.com/realms/Publication/protocol/openid-connect/token`
`spring.security.oauth2.client.provider.keycloak.user-info-uri`	The user information URI of the OpenID Connect server. For example, when using keycloak, `http://keycloak.example.com/realms/Publication/protocol/openid-connect/userinfo`
`spring.security.oauth2.client.provider.keycloak.jwk-set-uri`	The JWK set URI of the OpenID Connect server. For example, when using keycloak, `http://keycloak.example.com/realms/Publication/protocol/openid-connect/certs`
`spring.security.oauth2.client.provider.keycloak.user-name-attribute`	The attribute to consider as the username within Publication for Capella, for instance `preferred_username`. The value read in this attribute must be stable over time. Do not use an attribute that would contain the readable person name such as 'Jane Doe', but rather an attribute that would contain a stable and unique login such as 'jdoe'.
`spring.security.oauth2.client.registration.keycloak.authorization-grant-type`	The authorization grant type to use by Publication for Capella, should be `authorization_code`.
`spring.security.oauth2.client.registration.keycloak.client-id`	The Publication for Capella Client-ID, as registered in the OpenID Connect server.
`spring.security.oauth2.client.registration.keycloak.client-secret`	The Publication for Capella Client secret, as registered in the OpenID Connect server.
`spring.security.oauth2.client.registration.keycloak.redirect-uri`	The OpenID Connect server's redirect URI, for example `{baseUrl}/login/oauth2/code/{registrationId}`.

Property Key	Description
`spring.security.oauth2.client.registration.keycloak.scope`	The OpenID Connect scope to use, should be `openid`.
`spring.security.oauth2.resourceserver.jwt.issuer-uri`	The JWT issuer URI, for example `http://keycloak.example.com/realms/Publication`. This is used to secure the API with access tokens when the ID provider uses JWT access tokens. This must not be used together with `spring.security.oauth2.resourceserver.opaquetoken.introspection-uri`.
`spring.security.oauth2.resourceserver.opaquetoken.introspection-uri`	The id provider introspection URI to verify opaque access tokens, for example `http://ipd.example.com/oauth2/introspect`. This is used to secure the API with access tokens when the ID provider uses opaque access tokens. This must not be used together with `spring.security.oauth2.resourceserver.jwt.issuer-uri`.
`spring.security.oauth2.resourceserver.opaquetoken.client-id`	The ID of the client that uses these access tokens. This must only be used together with `introspection-uri`.
`spring.security.oauth2.resourceserver.opaquetoken.client-secret`	The secret of the client that uses these access tokens. This must only be used together with `introspection-uri`.

Sirius-Web Properties

Property key	Description
`sirius.components.cors.allowedOriginPatterns`	A list of patterns for servers that can use the Publcation server's resources in their own web pages. For example, `,https://some-server.example.com:[]`.

Obeo Enterprise Framework Properties

Property key	Description
`ocp.license`	License key obtained from Obeo.
`obeocloudplatform.admin.password`	Default password of the `admin` account. Must be set before the first launch of the Publication server after installation. This property is used only the first time the server is launched, to create the `admin` user's password. It is then recommended that the administrator changes his or her password *via* the web UI.

Publication for Capella Properties

The following properties are specific to Publication for Capella.

Property key	Default value	Description
perseus.jama.poll	true	Enablement of the polling of Jama Connect servers. true to turn on and false to turn off the polling.
perseus.jama.poll.delay.seconds	30	Jama server polling delay, in seconds.
perseus.sync.attachments.enable	true	Enablement of the synchronization of attachments (with Jama Connect servers). true to turn on and false to turn off the polling.
perseus.metaclass.filter.default		Path to the metaclass filter global defaults file.
perseus.versioning.branch-lock-timeout-ms	1000	Timeout for Branch locks, in milliseconds.
perseus.model-lock-timeout-ms	3000	Timeout for Model locks, in milliseconds.
perseus.model-publish-attempts	10	Number of attempts to lock the model for publications.
perseus.csp.whitelist		Path to the Content Security Policy whitelist file.
perseus.properties.mappings		Path to the file that contains the mappings of properties
perseus.license.auto.assign.read.on.login	true	Automatically assign a license token with privilege 'Read' on successful login if one is available and user has no license token.
perseus.oauth2.registration.id.for.{contributor-client}	keycloak	Registration ID of the OAuth-2.0 Client that provides JWT to the Perseus contributor client. For the time being, this value must be keycloak, even if a different OpenID Connect server is used. The segment {contributor-client} must be replaced by the client ID declared for the Publication for Capella Contributor Client in the OIDC server.
perseus.oauth2.jwt.client.id.claim	azp	Name of the claim to use to obtain the client ID from the JWT provided by the OIDC server for a contributor client authentication. By default, the client ID is read in the azp claim (Authorized Party), but tests have shown that different OIDC server provide that information in different claims, for instance client_id. If the claim is not found, Publication for Capella will attempt to read it from the audience claim.

Property key	Default value	Description
`perseus.oauth2.signing.algorithm.{registration-id}`	`RS256`	Signing Algorithm to use to verify JSON Web Tokens issued by the authentication provider corresponding to `{registration-id}`. For example, when using `keycloak` as registration ID, set `perseus.oauth2.signing.algorithm.keycloak` to `HS256` to verify tokens if they're signed with the HS256 algorithm. Valid values are: `RS256` (the default), `RS384`, `RS512`, `ES256`, `ES384`, `ES512`, `PS256`, `PS384`, `PS512`, `HS256`, `HS384`, and `HS512`.
`perseus.oauth.connection-request-timeout-ms`	`3000`	Timeout for OAuth connection requests, in milliseconds.
`perseus.oauth.connect-timeout-ms`	`5000`	Timeout for OAuth connections, in milliseconds.
`perseus.oauth.socket-timeout-ms`	`8000`	Timeout for OAuth sockets, in milliseconds.
`perseus.oauth.token.validity`		The duration for which an OAuth-1.0 token is valid, in **ISO-8601** format. For example: `P1D` for one day, `PT1H` for one hour, etc.
`perseus.oslc.host-name`		The public URL of the publication server, as it is accessed by third-party servers and end-users. For example, `https://perseus.mycompany.com:9443`.
`perseus.oslc.rootservices-cache-validity-sec`	`300` (5 minutes)	Duration for which an entry in the rootservices cache is considered valid.
`perseus.oslc.rootservices-cache-size`	`100`	Size of the rootservices cache.
`perseus.oslc.links.fetchTimeoutSec`	`10`	Timeout to fetch OSLC links, in seconds.
`perseus.oslc.preview.defaultHeight`	`400px`	Default height of OSLC previews, as a CSS String (including unit).
`perseus.oslc.preview.defaultWidth`	`600px`	Default width of OSLC previews, as a CSS String (including unit).
`perseus.oslc.thread.pool.size`	`4`	Number of Threads to use for the OSLC Thread pool.
`perseus.oslc.linking.default`		Path to the OSLC linking default config file.
`perseus.oslc.credentials.validity`	`PT8H` (8 hours)	The duration for which OSLC credentials are valid, in **ISO-8601** format. For example: `P1D` for one day, `PT1H` for one hour, etc.

Property key	Default value	Description
`perseus.http.client-headers[X-Atlassian-Token]`	`no-check`	This property must be set to make it possible for Publication for Capella to access the Atlassian (Jira or confluence) OSLC API. This is necessary to be able to connect to recent versions of Jira or Confluence. Adding this property will cause Publication for Capella to systematically add a header `X-Atlassian-Token: no-check` to all HTTP requests emitted to third-party servers.

Third-Party Specific Documentation

IBM Jazz Team Server

Publication for Capella Global Configuration with IBM Jazz Team Server Tutorial

Description

This tutorial aims at showing Publication for Capella ability to interact in a Global Configuration context. It shows step by step how to configure IBM Jazz Team Server and Publication for Capella to allow the OSLC linking between them in a Global Configuration context.

This tutorial has been made with for IBM Jazz Team Server version 7.0.2.

Setup the IBM Jazz Team Server server

A IBM Jazz Team Server server must be up and running and configured with two applications:

- A Global Configuration application
- A Requirements Management application

The Requirements application will be used for the purpose of this tutorial, in order to create links from Publication for Capella to IBM DOORS Next requirements. Other kind of applications could be used instead.

Those applications should appear on IBM Jazz Team Server dashboard.

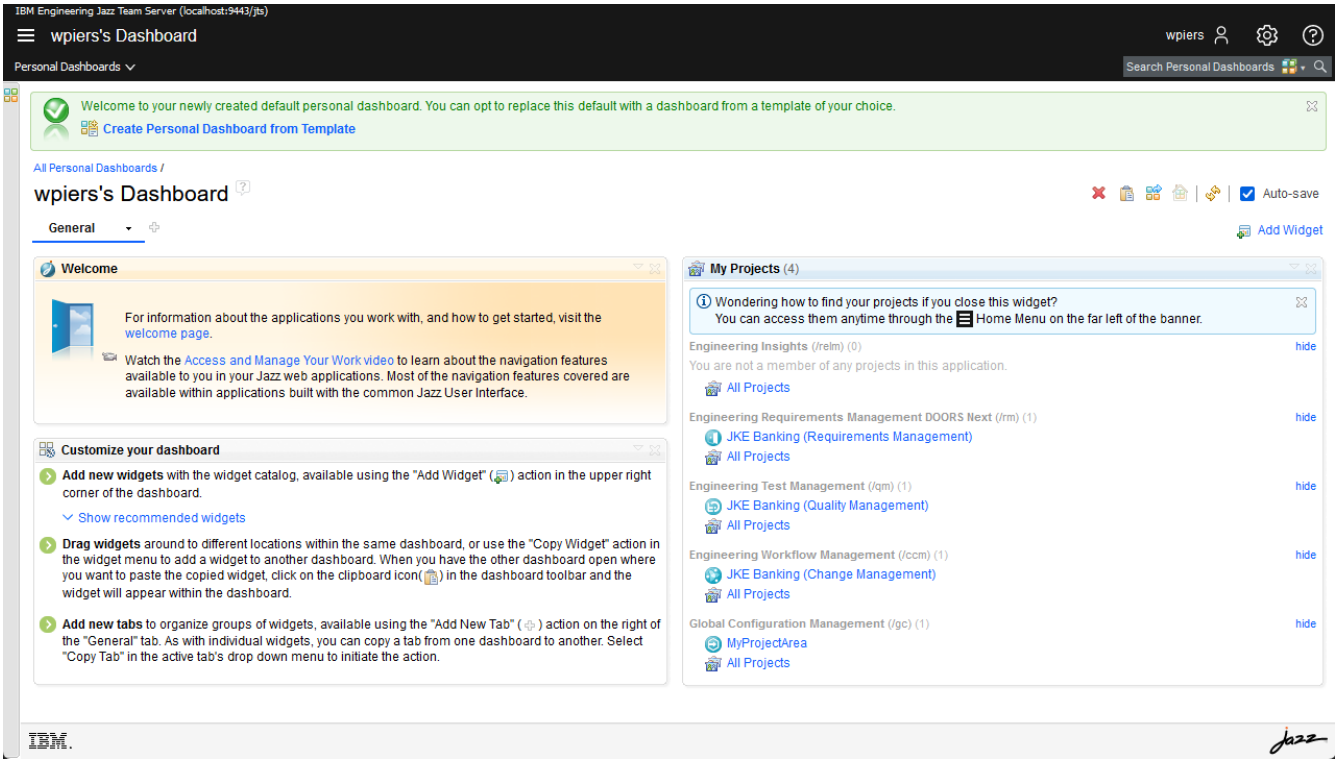


Figure 146. IBM Jazz Team Server Dashboard

This tutorial requires the Global Configuration application to contain a Project Area with at least one Component and the associated Stream. The Requirements Management application is linked to this Project Area and can switch from a Global Configuration to another.

Here are the key steps to setup that environment in IBM Jazz Team Server:

1. Register a Global Configuration license key in IBM Jazz Team Server

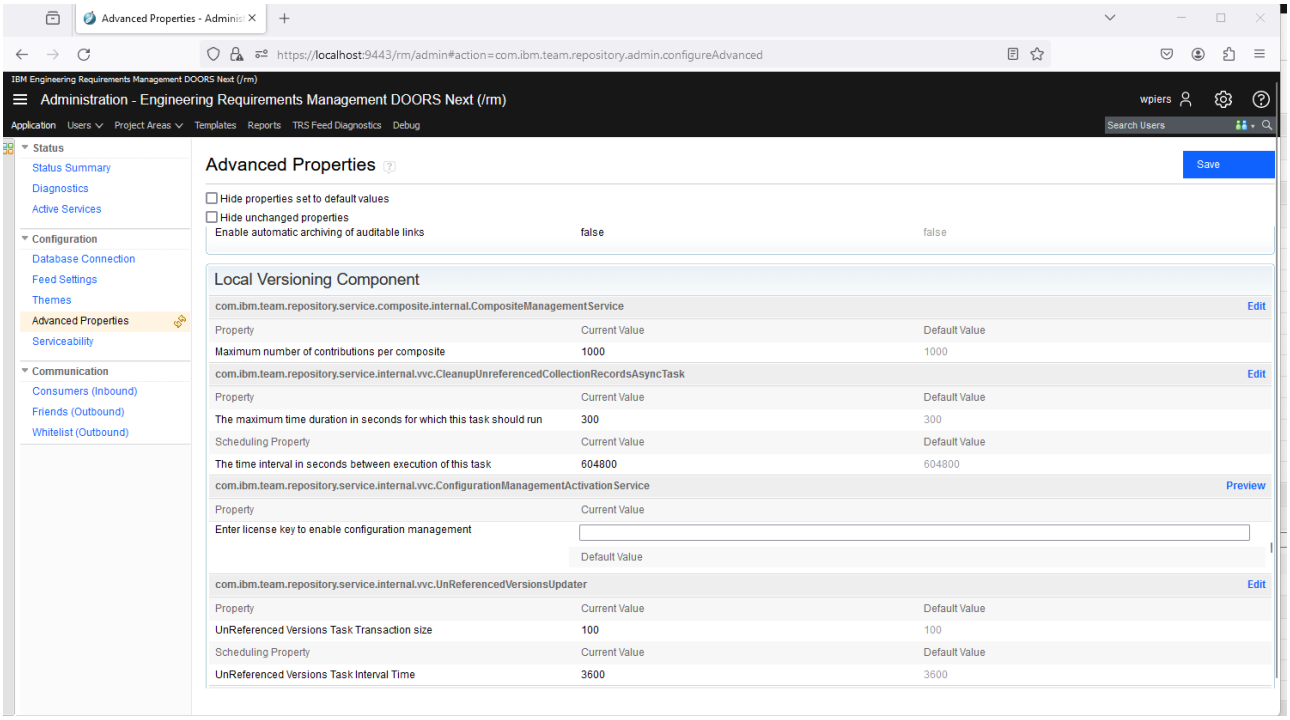


Figure 147. Register License Key

2. In the Global Configuration application (*<jazz_server_url>/gc*), create a **Project Area**

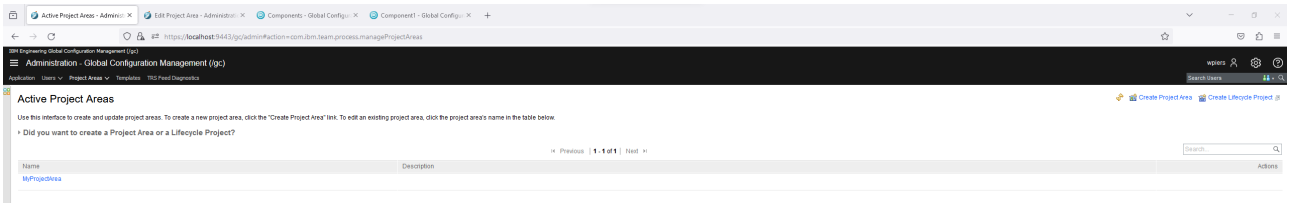


Figure 148. Create a Project Area

3. Add yourself as a *member* of this Project Area with the *Administrator* process role

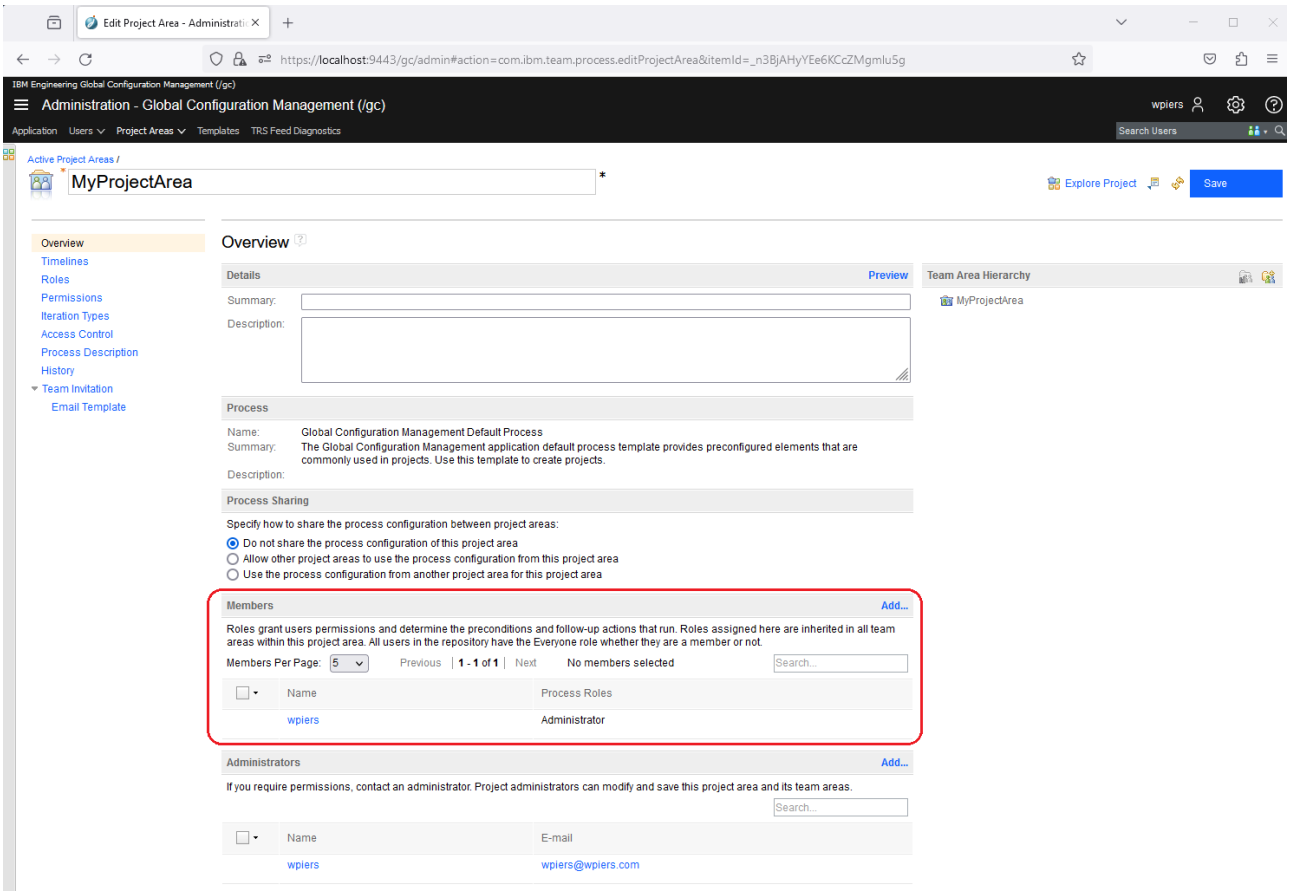


Figure 149. Add Project Area Member

4. Create at least one **Component** in this Project Area

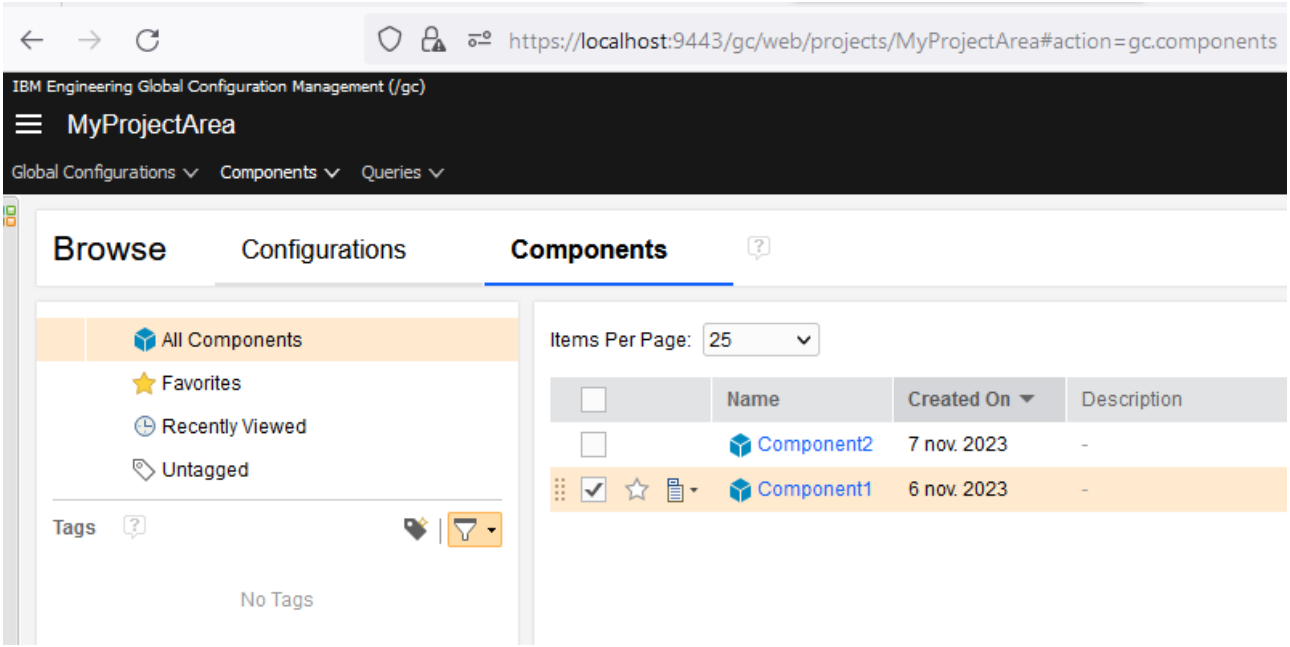


Figure 150. Create Components

5. An *Initial Development* **Stream** is automatically created

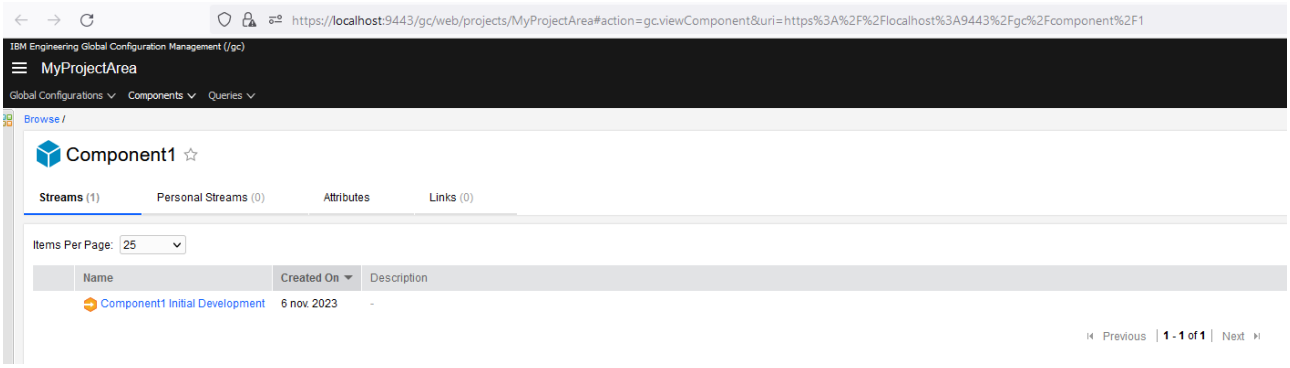


Figure 151. Creates Stream

6. In the Requirements Management application (*<jazz_server_url>/rm*), create a **Project** (or reuse a demo project from IBM Jazz Team Server)

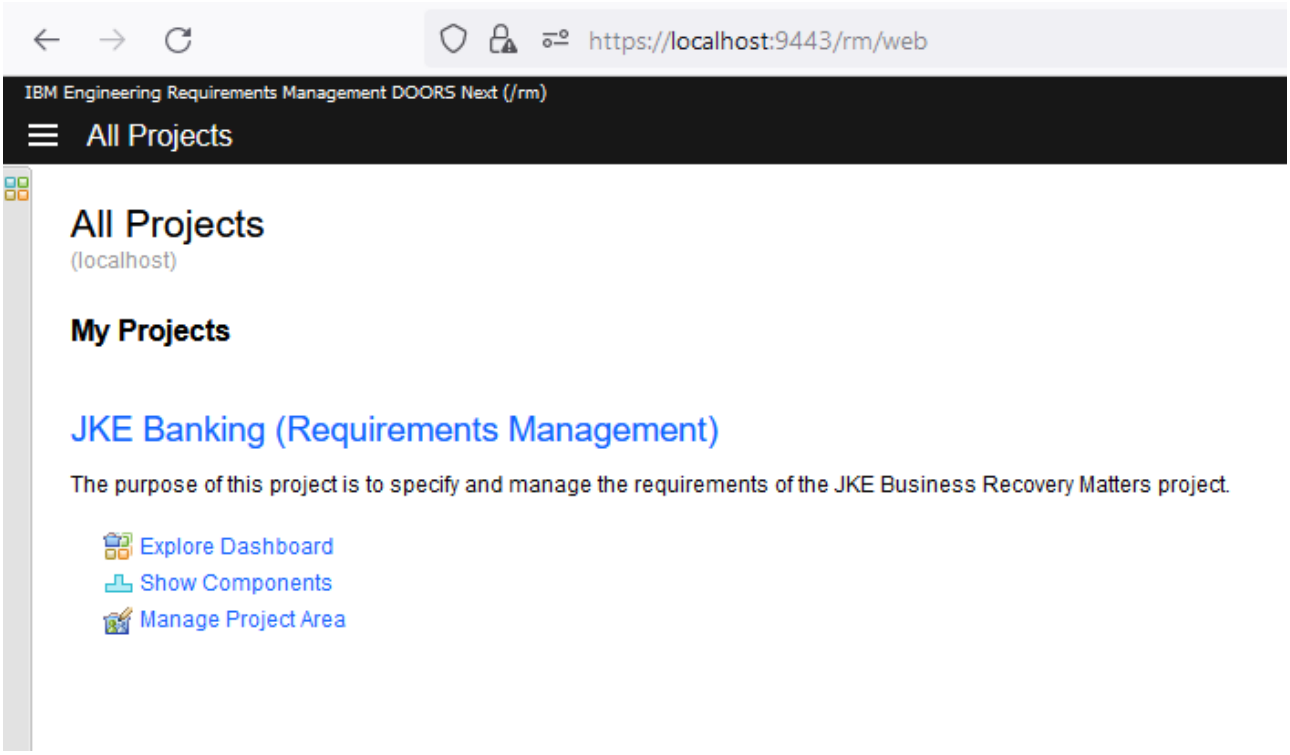


Figure 152. Requirements Project

7. Activate Global Configuration for this project

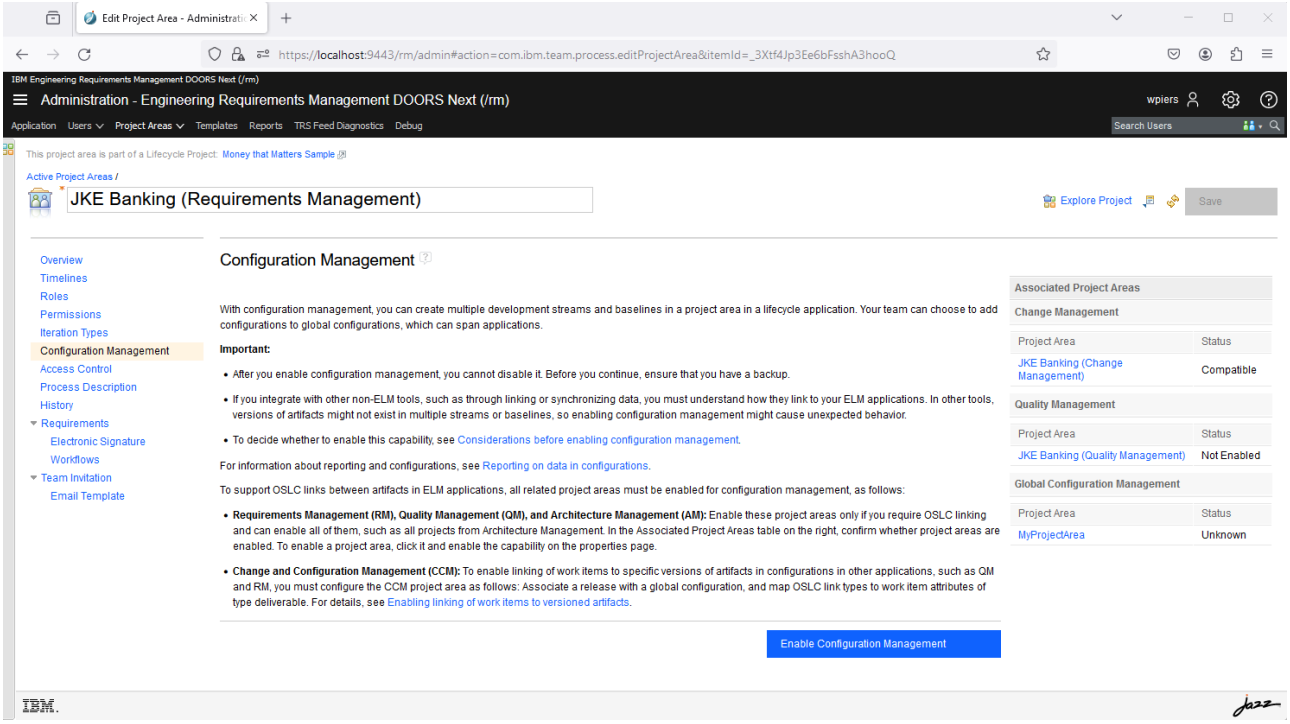


Figure 153. Activate Global Configuration

8. In the Global Configuration application, associate the Project Area with that project

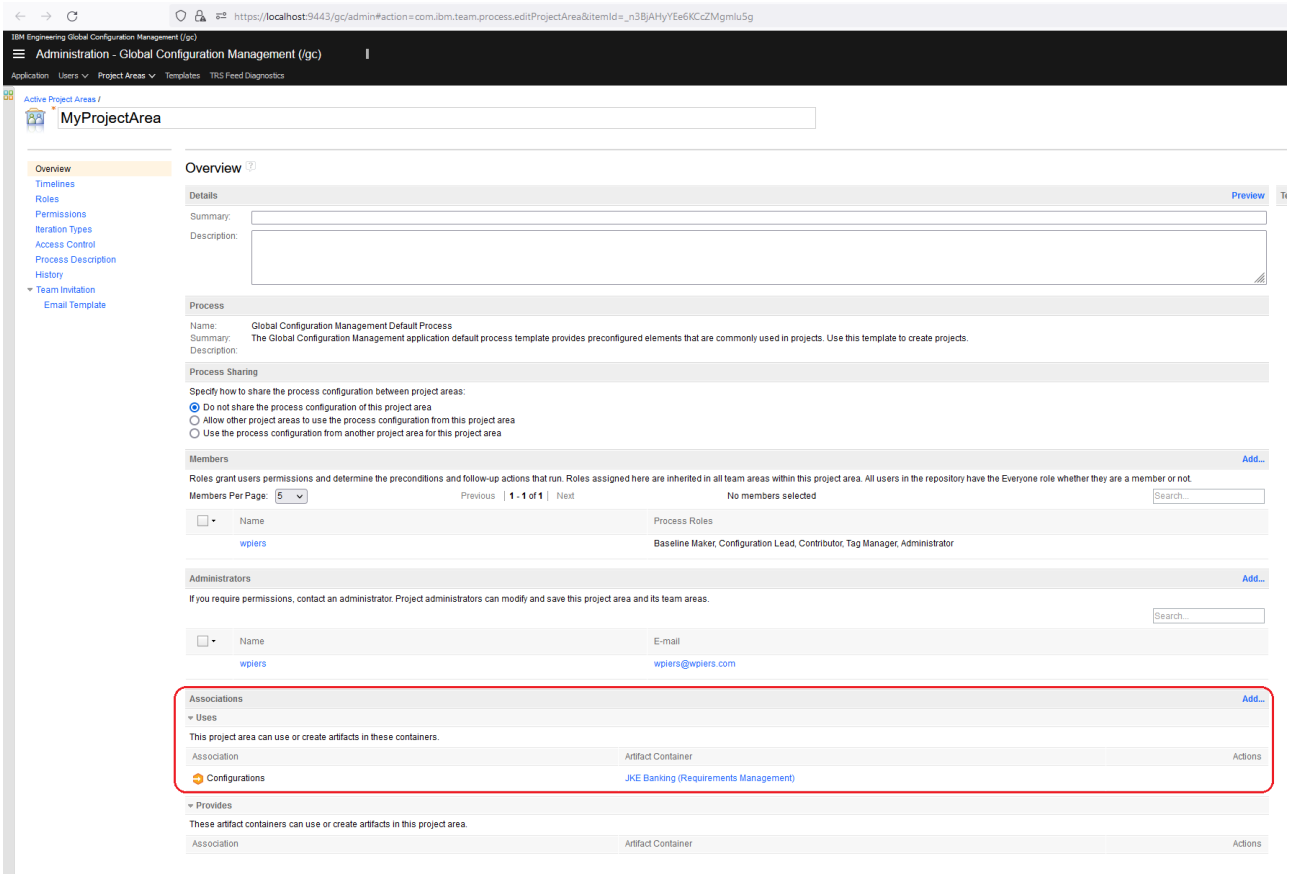


Figure 154. Project Area association

9. In the Requirements Management Project, create a **Stream**

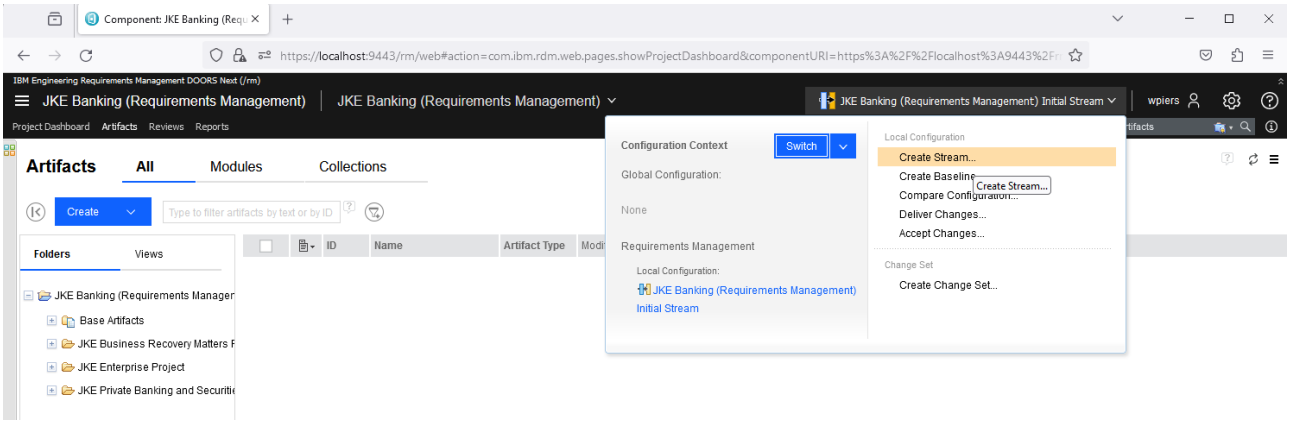


Figure 155. Create Stream

10. In the Global Configuration application, add this Stream to the Component you created before

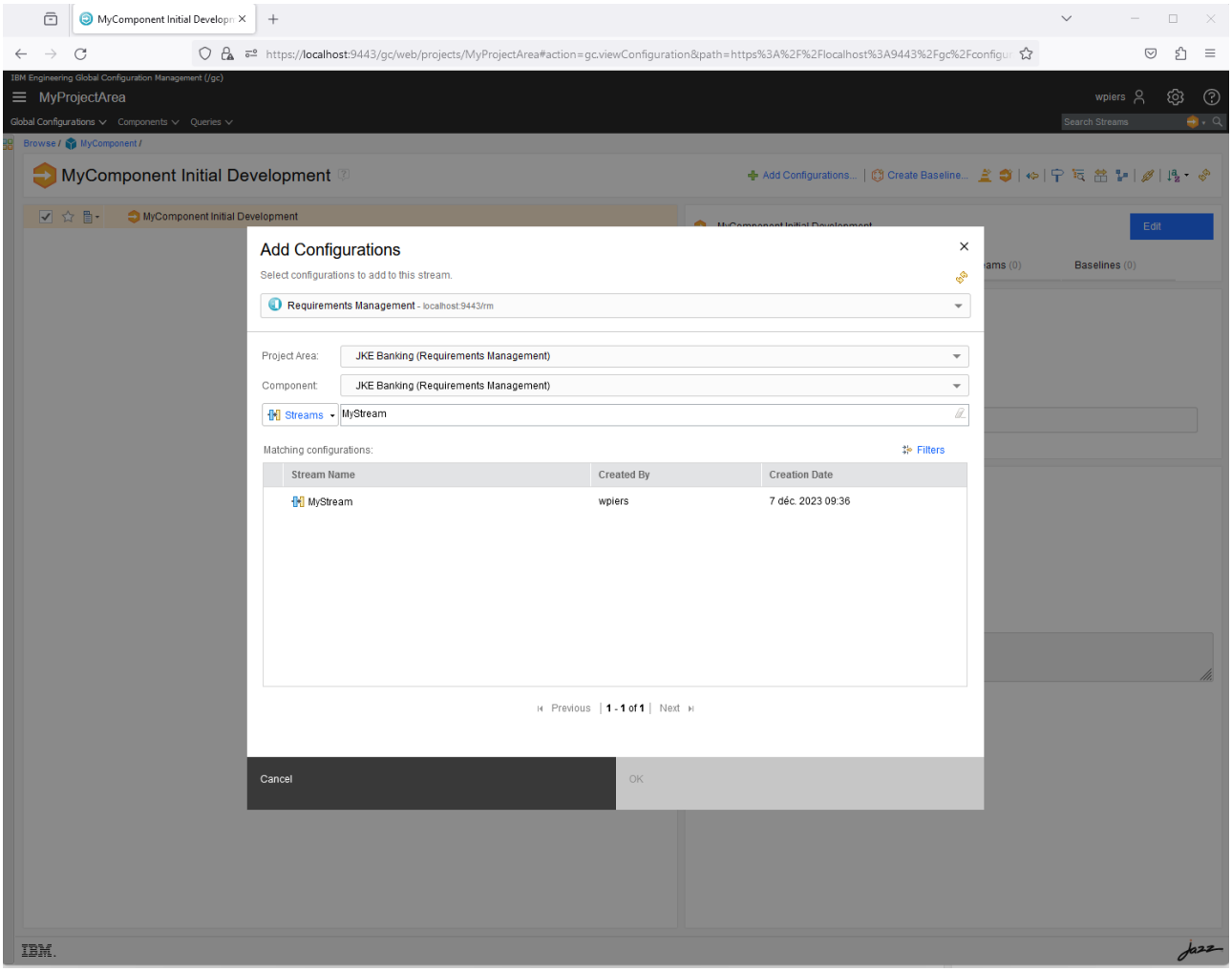


Figure 156. Link Streams

You should now be able to switch between configurations from the Requirements Project:

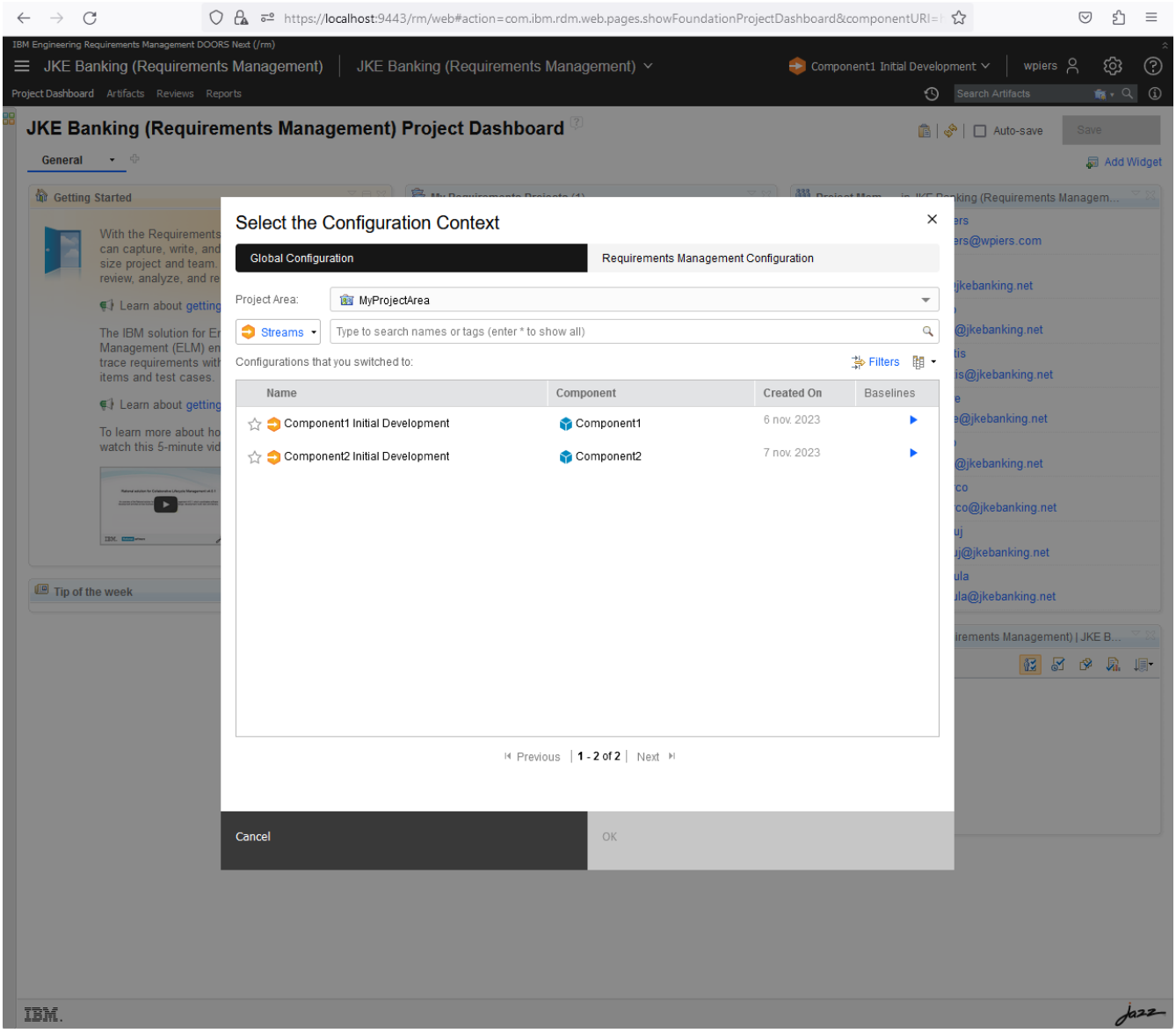


Figure 157. Switch Configuration

The tutorial focuses on the use of Global Configuration in Publication for Capella, please rely on IBM Jazz Team Server documentation for more details.

If you use Google Chrome, an extra step is required (at least with IBM Jazz Team Server 7.0.2) to allow OSLC delegated dialogs. It is described at https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/lifecycle-management/7.0.2?topic=server-customize-websphere-liberty-samesite.

Setup the Publication for Capella Server

A Publication for Capella server (minimal version 2023.12) must be up and running. A server administrator will be needed to register OAuth friends.

1. Create a New Publication for Capella Project

Figure 158. New Project Creation

2. In the project settings, click on *Activate Global Configuration*, confirm

Figure 159. Global Configuration activation

3. Publish a Capella model on this project

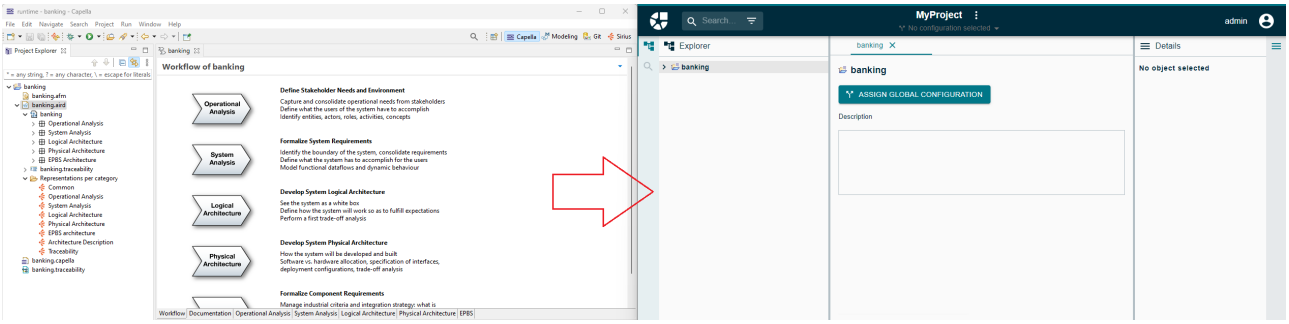


Figure 160. Published model

Register IBM Jazz Team Server in Publication for Capella

To establish the communication with IBM Jazz Team Server we must first register Publication for Capella as a friend in IBM Jazz Team Server, in the `<jazz_server_url>/jts/admin#action=com.ibm.team.repository.admin.friends` administration page.

Then we can create two friends:

- The Global Configuration application (`<jazz_server_url>/gc`)

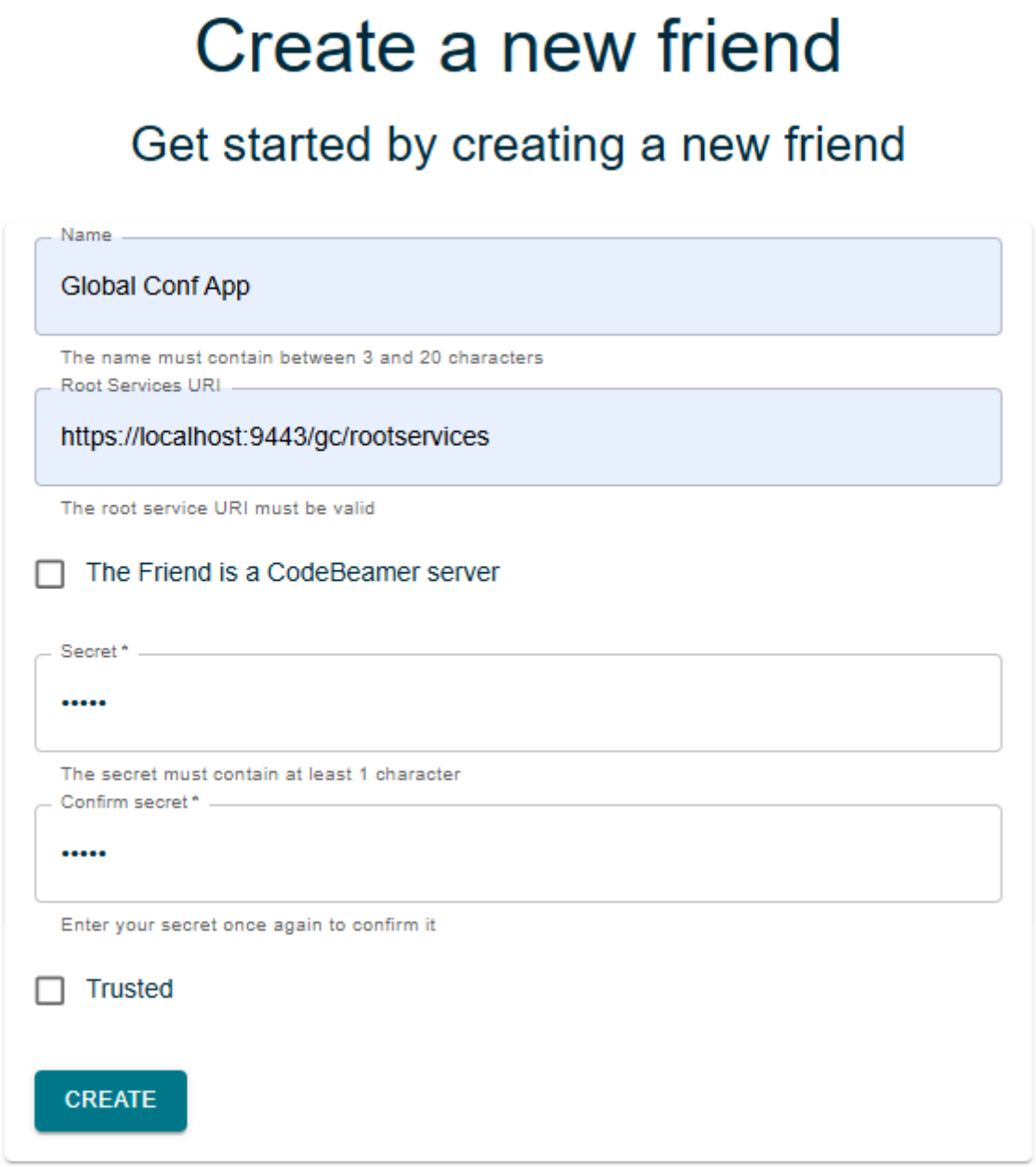


Figure 161. Global Configuration Application Friend Creation

- The Requirements Management application (`<jazz_server_url>/rm`)

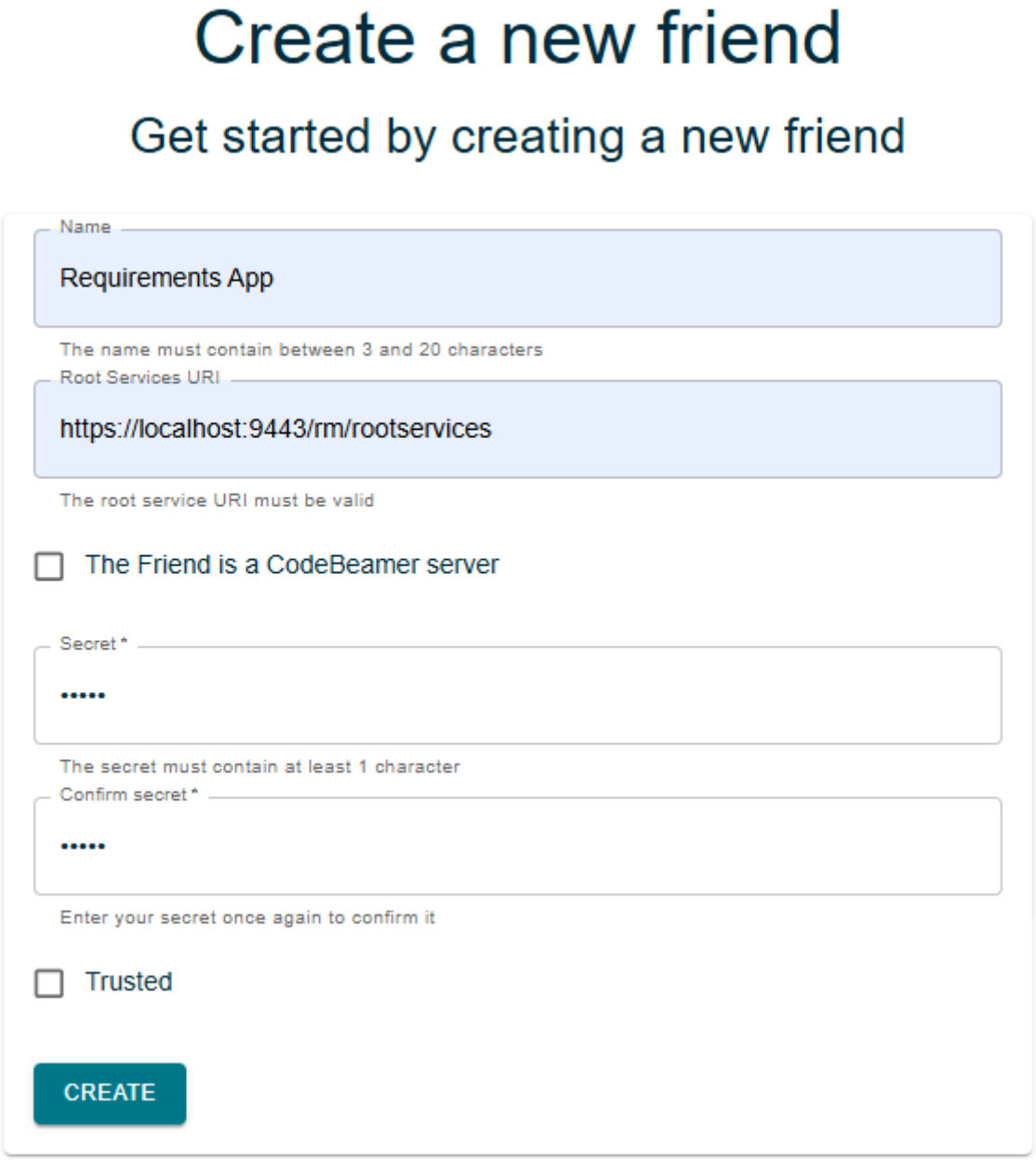


Figure 162. Requirements Management Application Friend Creation

If IBM Jazz Team Server uses a self-signed certificate, it must be exported from IBM Jazz Team Server website then registered in Publication for Capella JVM.

Approve IBM Jazz Team Server Consumers

The new Publication for Capella friends are viewed as consumers on the IBM Jazz Team Server side and must be approved to allow the connection.

1. Open the IBM Jazz Team Server admin interface (*<jazz_server_url>/jts/admin*) and click on the *Manage Server* setting.

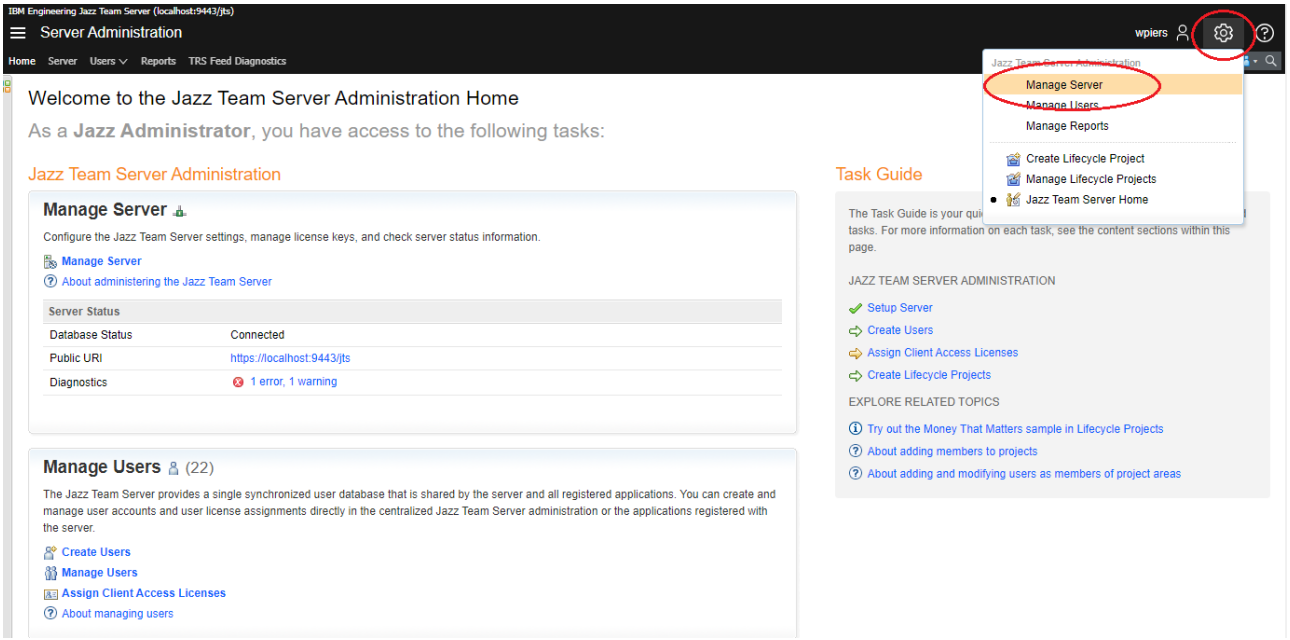


Figure 163. IBM Jazz Team Server Administration

2. On this page there is a section dedicated to consumers management with a *Consumers (Inbound)* link.

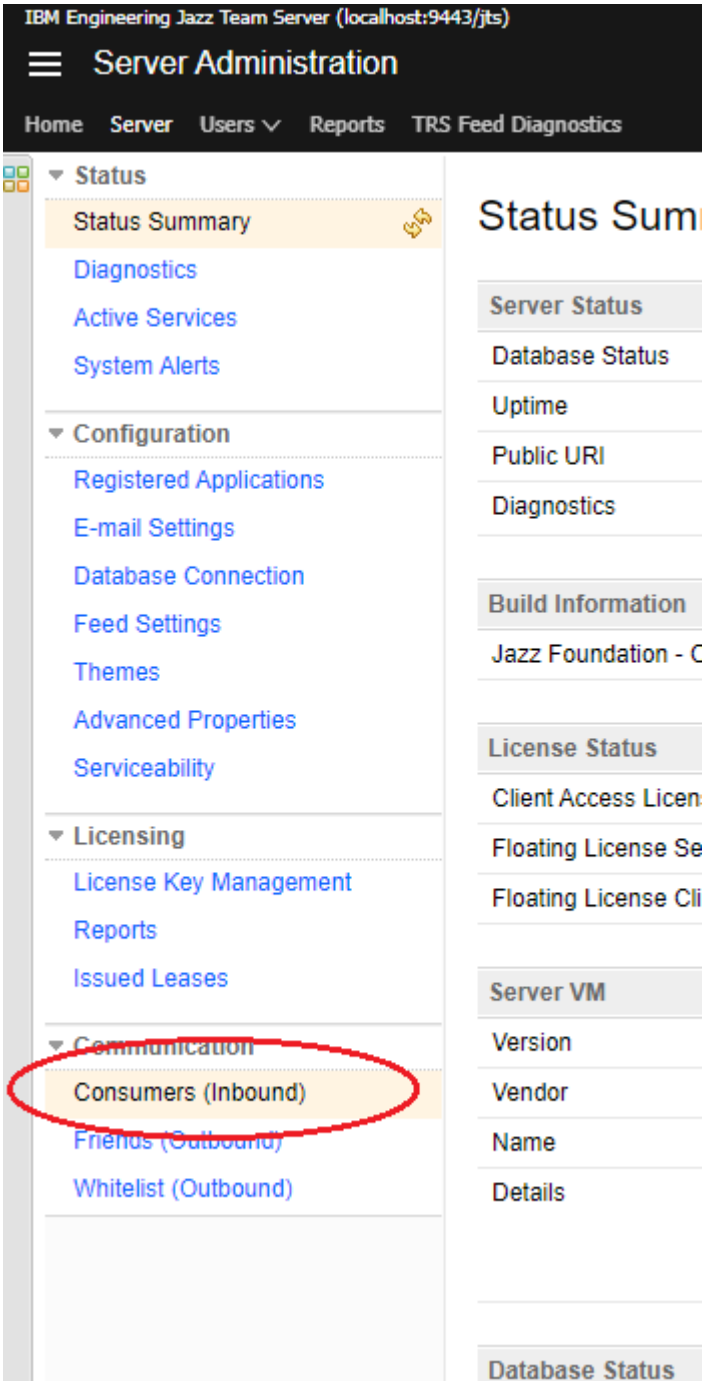


Figure 164. IBM Jazz Team Server Consumers

3. This link leads to a page where consumers can be approved. Change their status then click on the *Save* button.

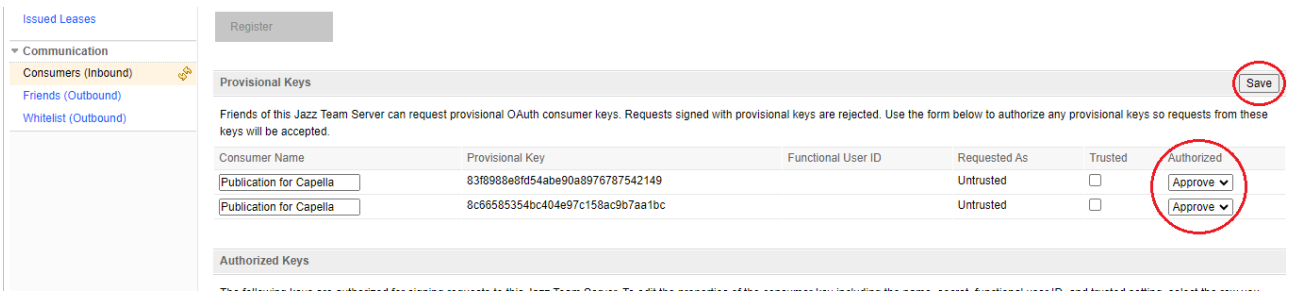


Figure 165. Approve IBM Jazz Team Server Consumers

Associate the Publication for Capella Project to IBM Jazz Team Server Service Providers

At this point we can associate the Publication for Capella project with the needed service providers from those two friends:

- A Global Configuration provider
- A Requirements provider

In the project settings, open the *Associations* tab and create those associations.

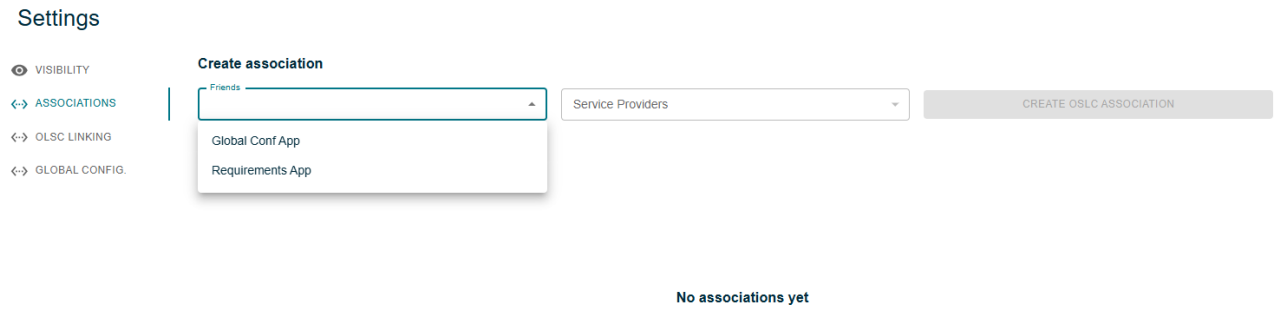


Figure 166. Create Associations

When selecting a friend in the associations settings the first time, IBM Jazz Team Server will prompt for an OAuth approval. Click *Yes* and close this dialog.

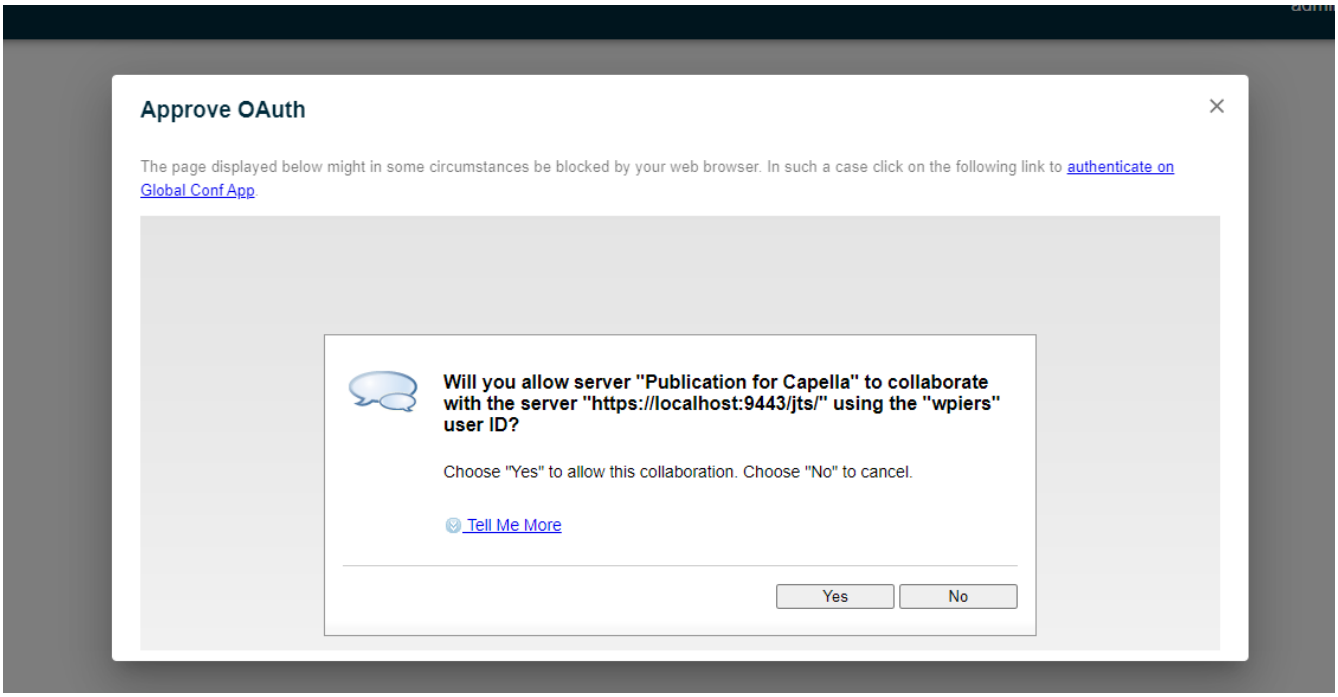


Figure 167. Publication for Capella Approve Access

The Publication for Capella project is now associated with two service providers.

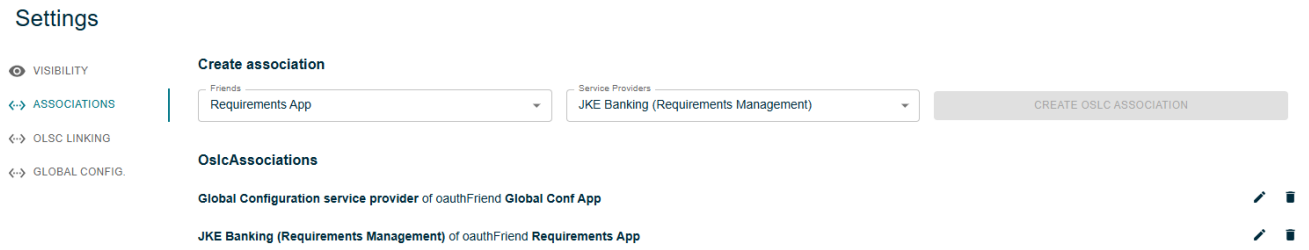


Figure 168. Publication for Capella Project Associations

Assign a Global Configuration to the Model

We can now assign a Global Configuration to our model.

1. Click on the model in the Explorer and click on the *Assign model configuration* button. A dialog opens and purpose configurations available on IBM Jazz Team Server.

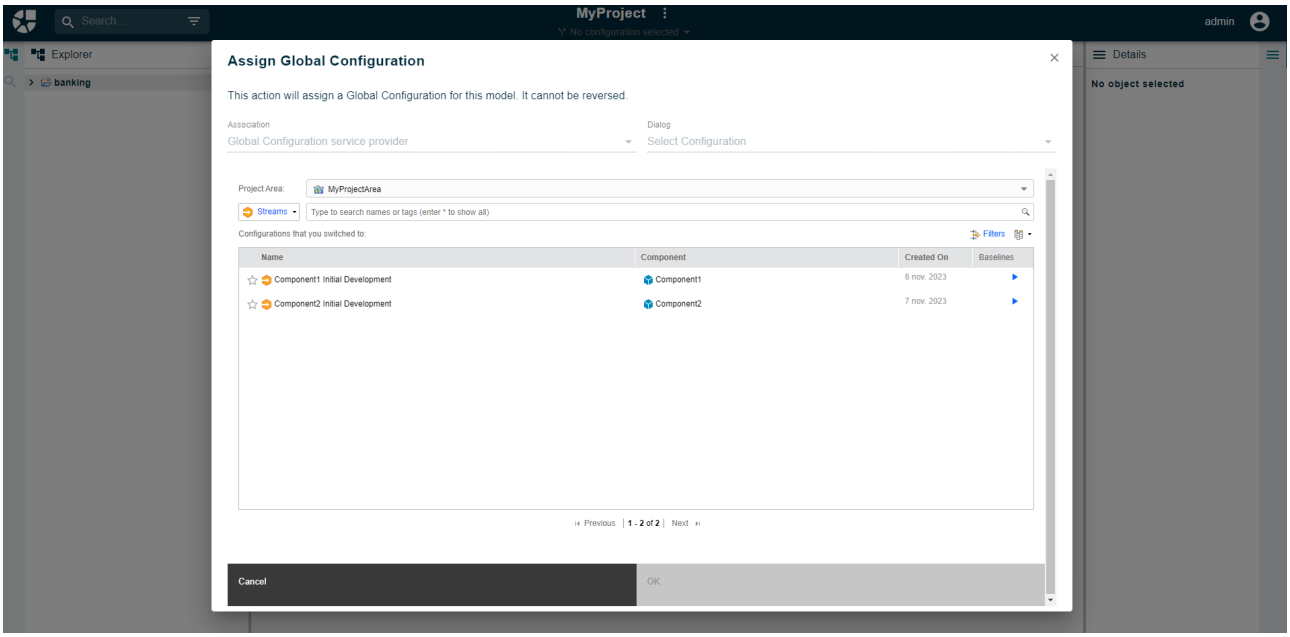


Figure 169. Assign Model Configuration

2. Select a configuration and click *OK*, then *Confirm* on the warning dialog.

Filter the Project by Global Configuration

In this tutorial there is only one model in the project but if there are several models, assigned with various configurations, it is possible to filter models by Global Configuration to display only the models that are assigned a given Global Configuration.

1. Click on the *No configuration selected* label under the project name. A dialog opens and purpose configurations available on IBM Jazz Team Server.

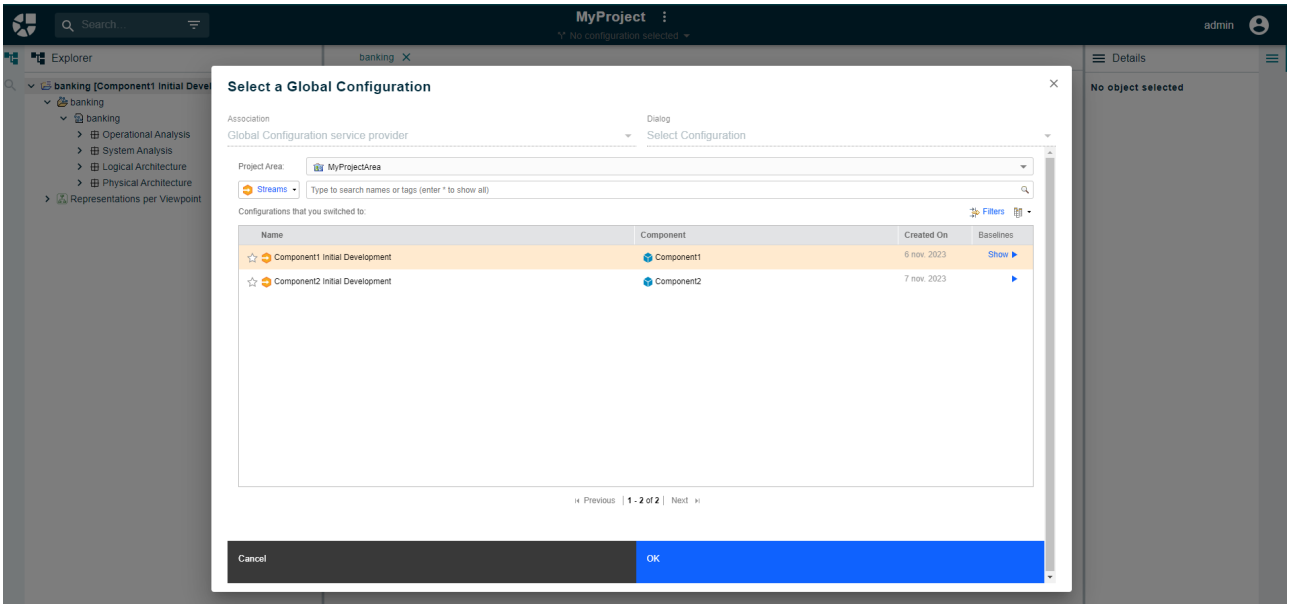


Figure 170. Select Project Configuration

2. Select a configuration, click *OK*. The label now shows the configuration title.

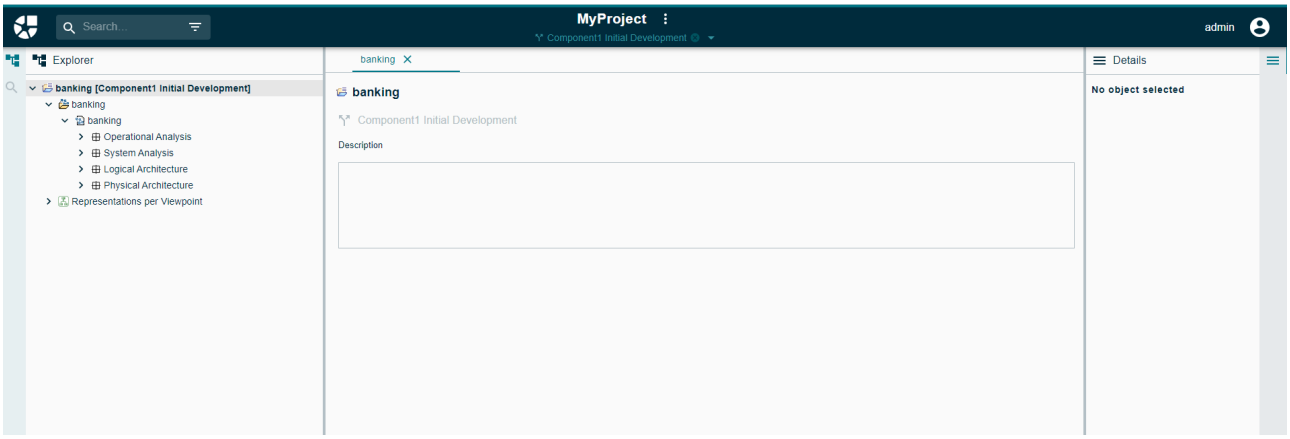


Figure 171. Selected Project Configuration

You can simply reset this filter by clicking on the cross icon next to the configuration label.

Create an OSLC Link

To demonstrate that the model is now tied to the assigned Global Configuration we are going to create a link from Publication for Capella to some element in DOORS Next.

1. Select the Root Logical Function in the Explorer then click on the link creation button.

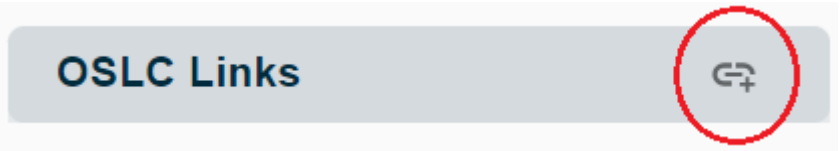


Figure 172. Create Link Button

2. Select a requirement. The dialog already only shows requirements available within the assigned configuration.

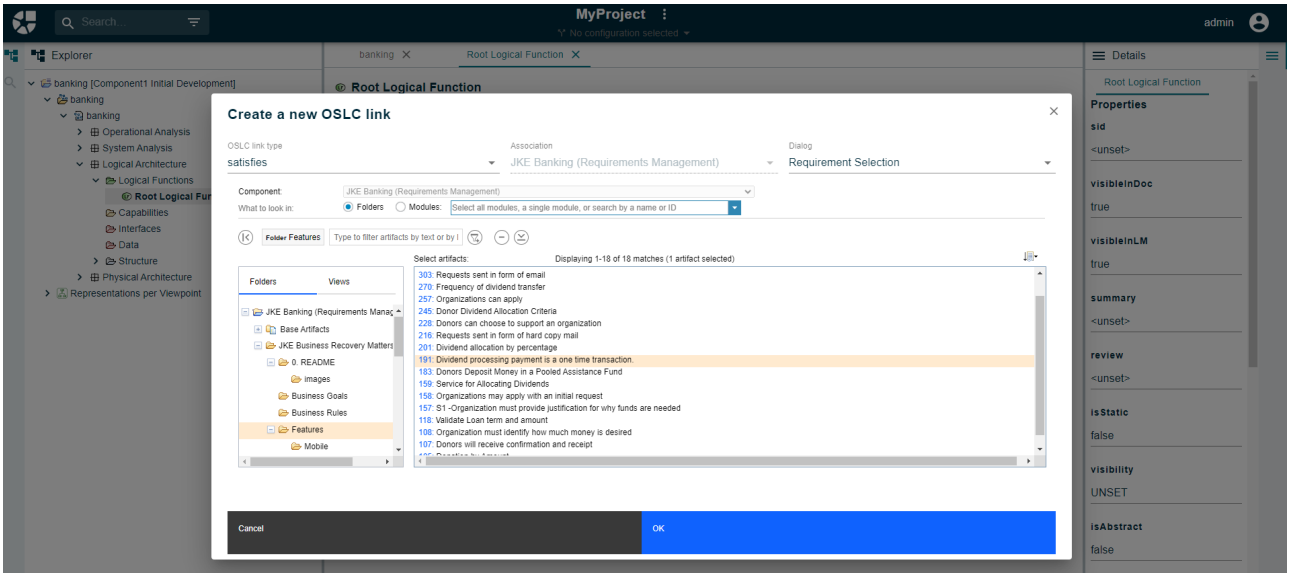


Figure 173. Create Link

3. The link is created on Publication for Capella.

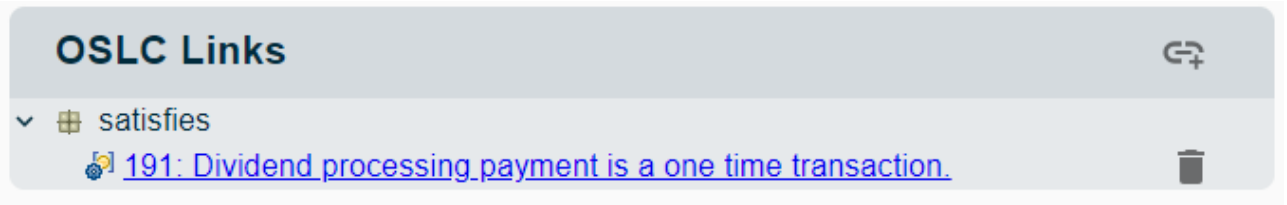


Figure 174. Link Created

4. Clicking on the link will lead to the requirement page on DOORS Next, within the right configuration.

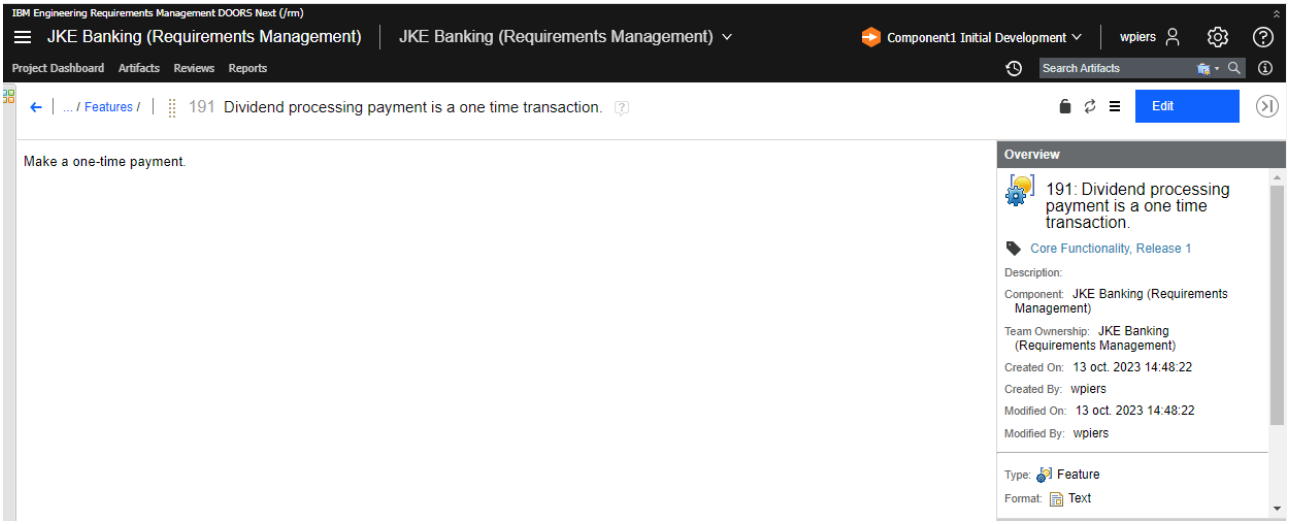


Figure 175. Requirement in DOORS Next

Update the Capella Model

Now we are going to update the Capella model in order to fetch the new link from Publication for Capella.

1. First we need to Initialize the traceability model in order to allow storage of the configuration. Right-click on the *aird* file then click on *Initialize traceability*.

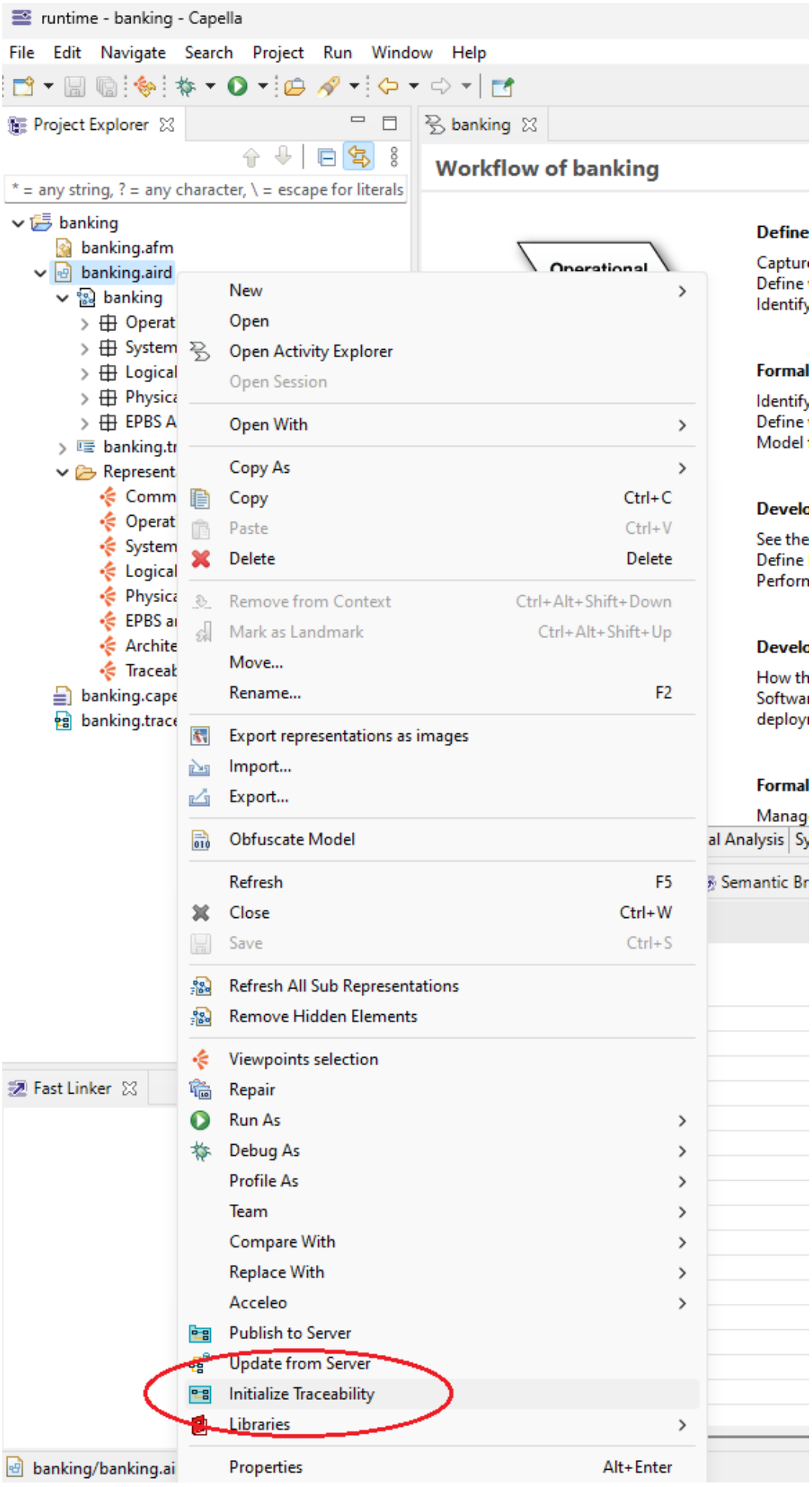


Figure 176. Initialize Traceability Action

2. In the dialog, select Requirement & Configuration Management Domains, and the two available repositories. Click *OK*.

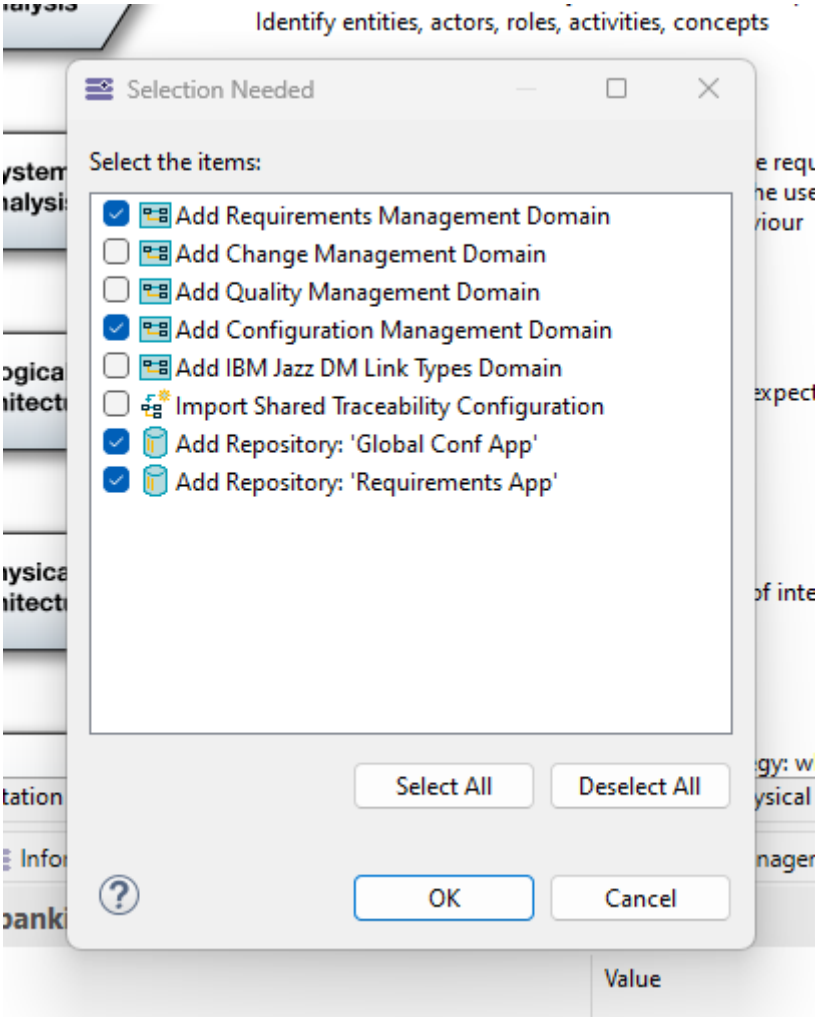


Figure 177. Initialize Traceability Dialog

3. Right-click on the *aird* file again but this time click on *Update from server*.
4. A dialog shows the new link, accept it.

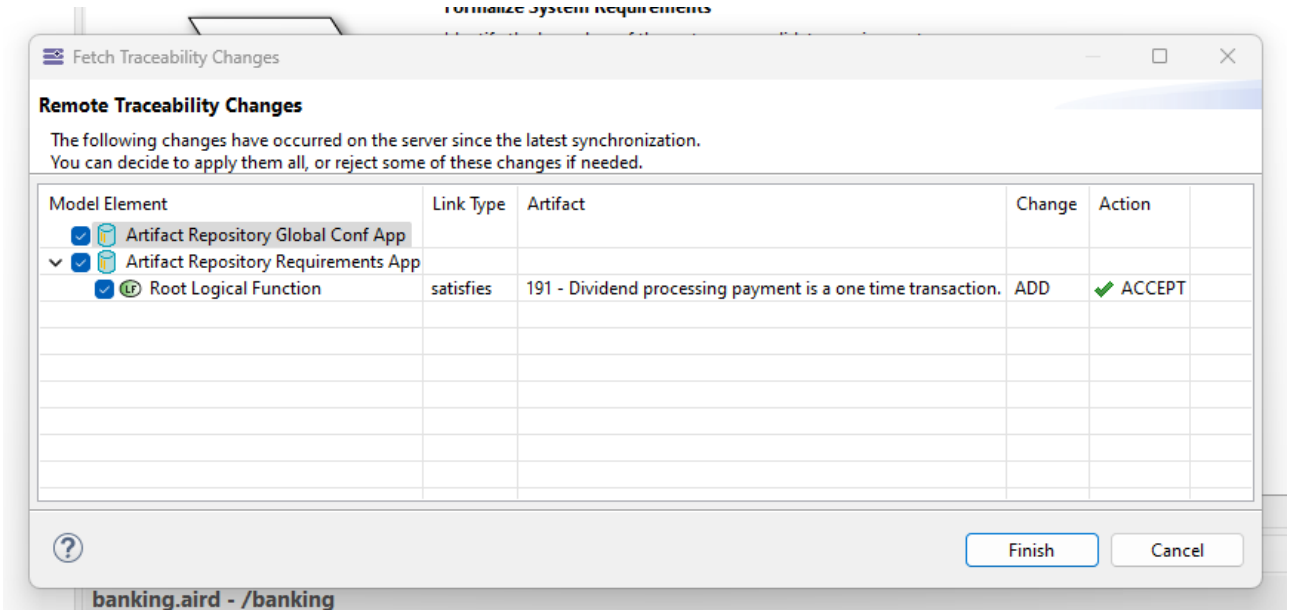


Figure 178. Update Changes

Polarion

Polarion Configuration for Publication for Capella

Publication for Capella exposes model data according to the OSLC AM (Architecture Management) domain.

Polarion can be configured to make it possible to link Polarion items to model objects exposed by Publication for Capella.

This document describes the necessary configuration that must be done in Polarion.

Global Administration Settings

- Declare the Publication for Capella server as a friend:
  - In Polarion, connect as an administrator;
  - Go to 'Global Administration > Linked Data > Linked Data Friends';
    - This configuration is made at `http://<polarion.mycompany.com>/polarion/#/administration/linkeddata/friends`;
  - Click on 'Add Friend' and enter the following information:
    - URI: `"https://<publication.mycompany.com>/oslc/rootservices"`;
      - e.g. `https://publication.mycompany.com:9090/oslc/rootservices` for a Publication for Capella server deployed on port `9090` at `publication.mycompany.com`;
    - Name: `"Publication for Capella"` (or any name to recognize the Publication for Capella server instance);
    - OAuth Secret: a password - Use a random string of characters, no need to remember it or save it anywhere else;
  - Click on Next. A dialog appears to validate the provisional key that will be used to authenticate communications between the two servers:

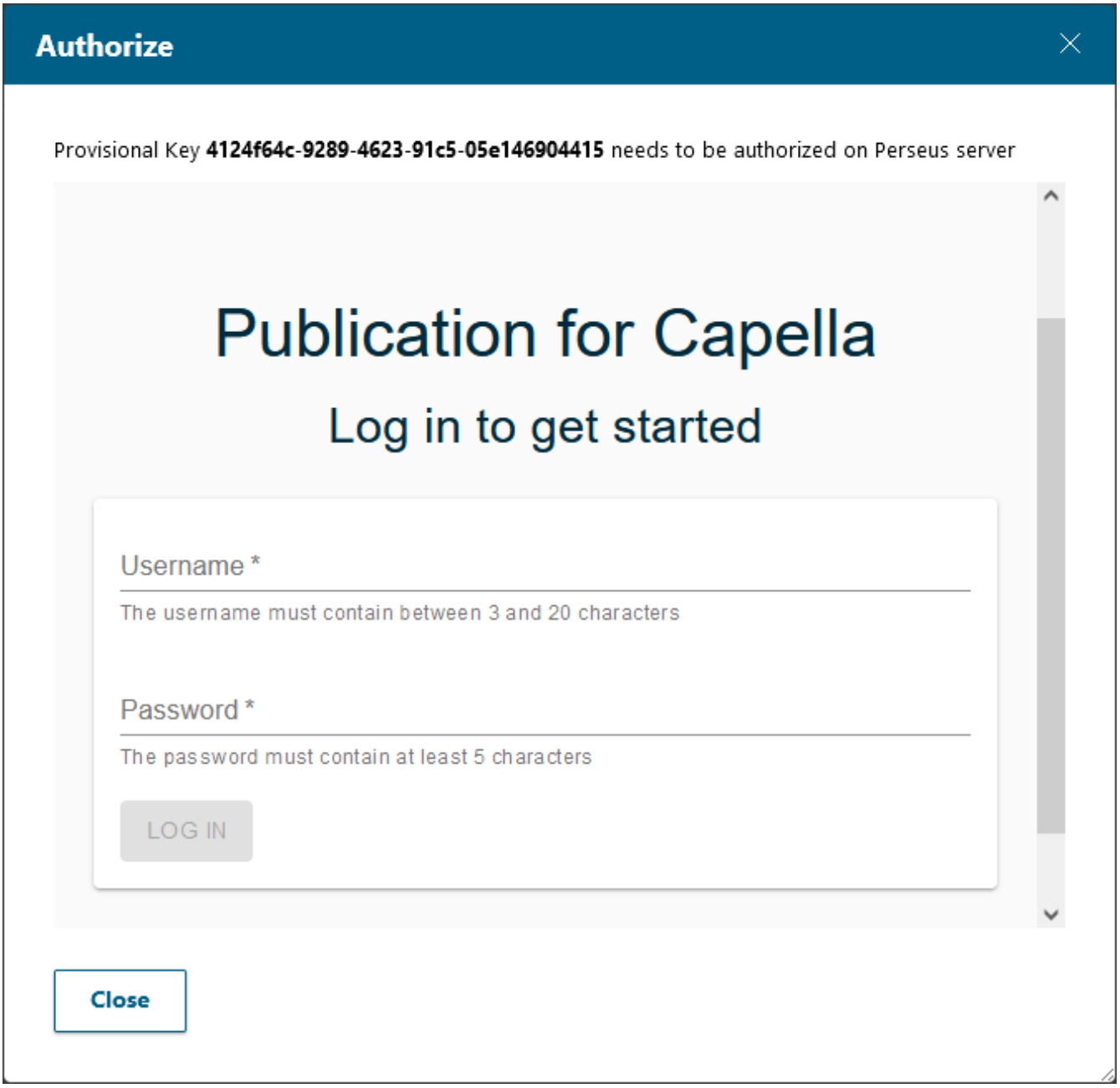


Figure 179. Publication for Capella Admin Login

- Use the *Server Administrator* credentials to connect in this dialog (by default, "admin"/"odweb"):

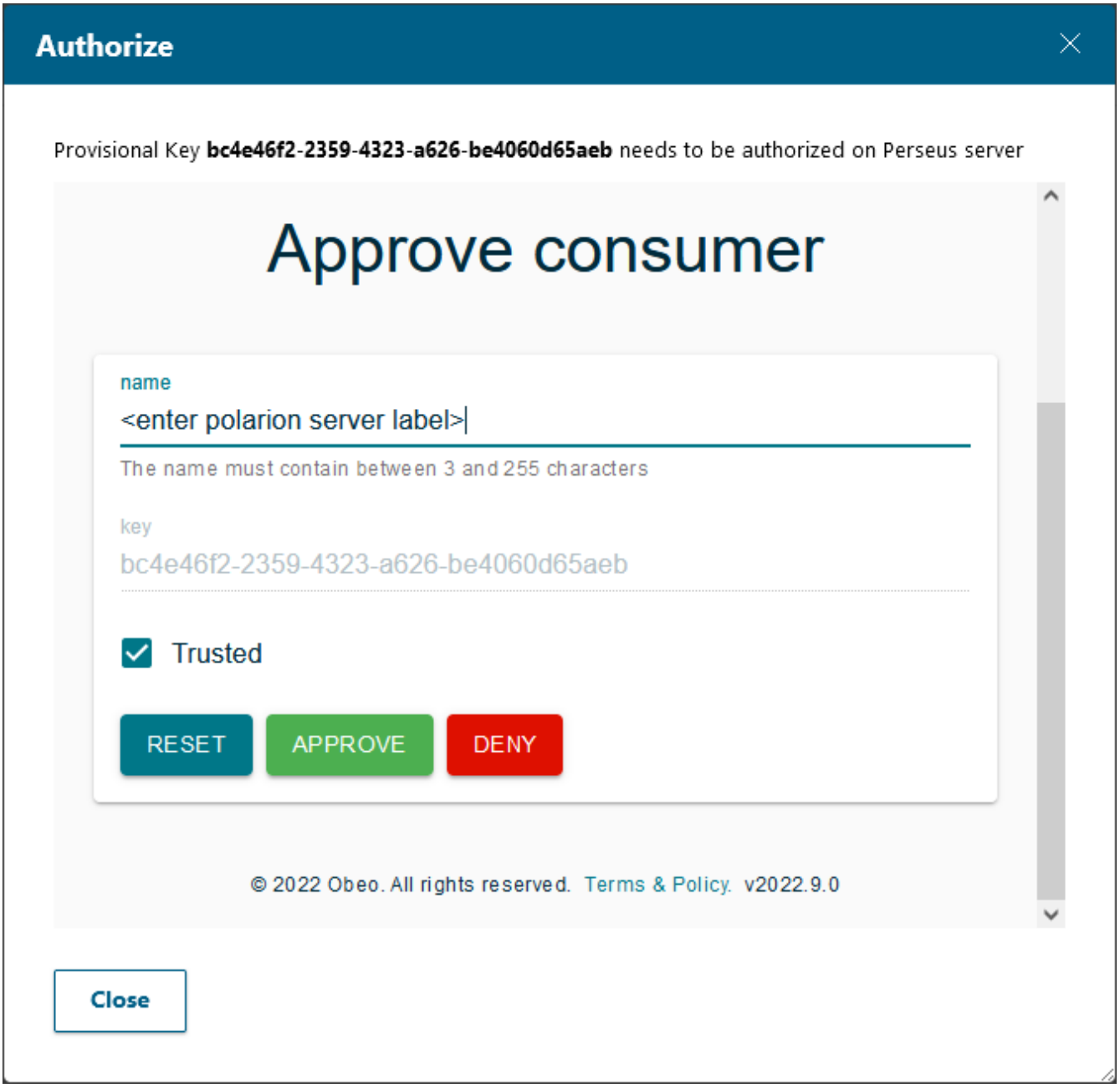


Figure 180. Validate Provisional Key

- Enter a name for the Polarion server, as seen from Publication for Capella;
- Leave `"Trust Level"` to `"Trusted"`;
- Click on 'Approve' - You can still edit the server name, and access that edition page later from Publication for Capella.

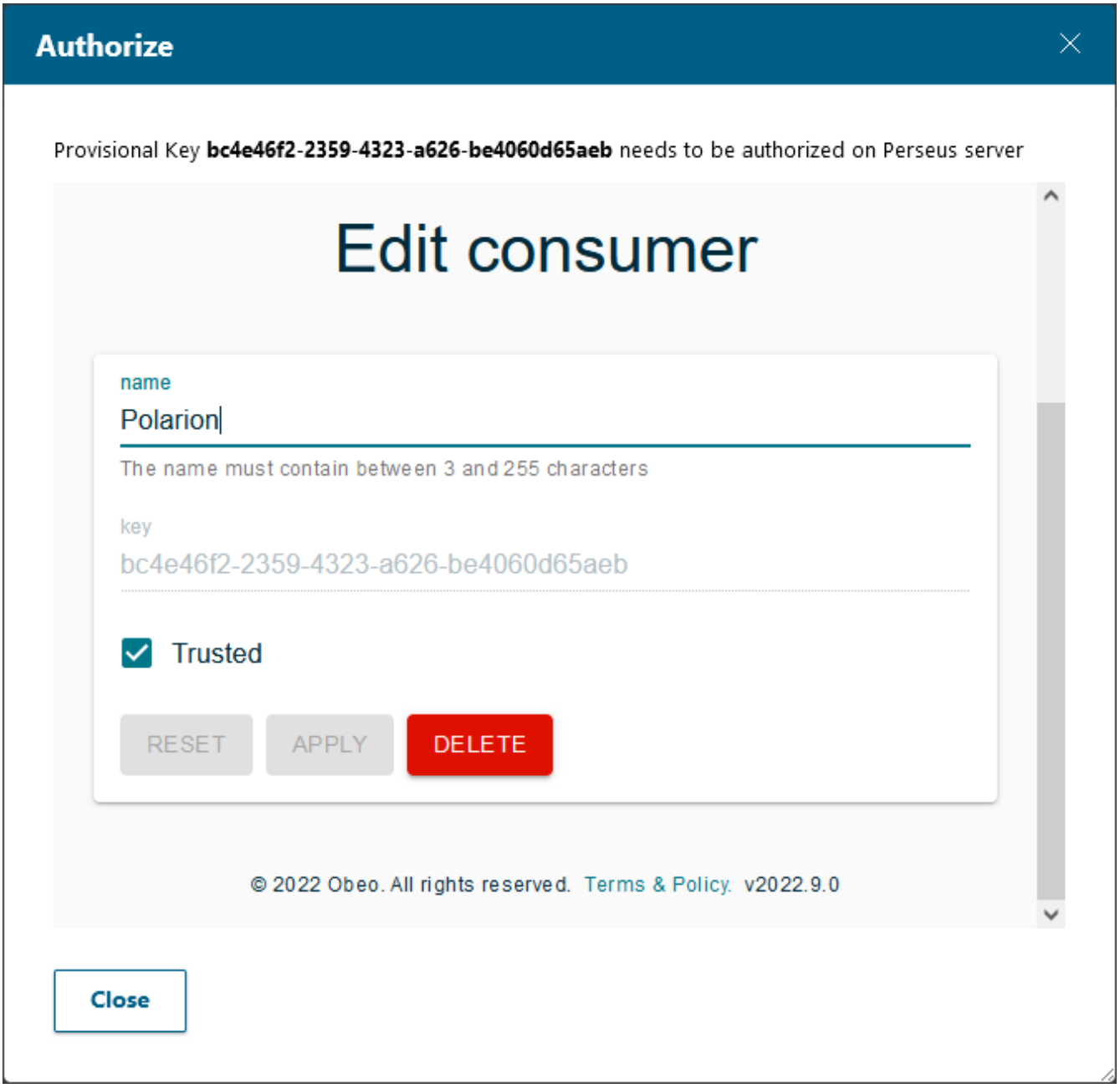


Figure 181. Provisional Key Approved

- Close the dialog with the Close button;
  - If you don't have these credentials, ask the Publication for Capella *Server Administrtor* to connect to the Publication for Capella server and navigate to `https://<publication.mycompany.com/admin/oauth/consumers` to validate the provisional key;
  - The Publication for Capella friend appears in the list of OSLC friends;

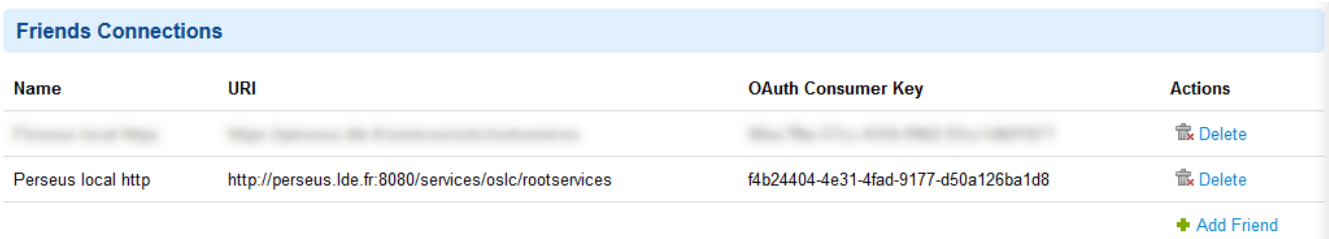


Figure 182. OSLC Friend Added

- Configure the Linked Data Semantics:

- Go to 'Global Administration > Linked Data > Linked Data Semantics';
  - This configuration is made at `http://<polarion.mycompany.com>/polarion/#/administration/linkeddata/semantics`;
- In the `Configuration` editor area:
  - In the `<namespaces>` element, add a namespace for OSLC AM with `prefix="oslc_am"` and `url="http://open-services.net/ns/am#"`;
  - In the `<domains>` element, add domain with `prefix="oslc_am"` and resourceType `label="Architecture Resource", uri="oslc_am:Resource"`;
  - In the `<linking>` element, add the relevant links according to the kinds of links you want to make available. An example is provided below;
- Save the configuration.

```
<namespaces>
  (...)
  <namespace prefix="oslc_am" url="http://open-services.net/ns/am#"/>
</namespaces>
<domains>
  (...)
  <!-- OCP -->
  <domain prefix="oslc_am">
    <resourceType label="Architecture Resource" uri="oslc_am:Resource"/>
  </domain>
</domains>
<linking>
  (...)
  <!-- OCP -->
  <link name="oslc_rm:elaboratedBy" reverse="oslc_rm:elaborates">
    <from type="oslc_rm:Requirement"/>
    <to type="oslc_am:Resource"/>
  </link>
  <link name="oslc_rm:satisfiedBy" reverse="oslc_rm:satisfies">
    <from type="oslc_rm:Requirement"/>
    <to type="oslc_am:Resource"/>
  </link>
  <link name="oslc_rm:specifiedBy" reverse="oslc_rm:specifies">
    <from type="oslc_rm:Requirement"/>
    <to type="oslc_am:Resource"/>
  </link>
</linking>
```

Project Configuration

Each Polarion project should be associated to one or several Publication for Capella projects to be able to create OSLC links between each other.

- Go to the configuration page of the Polarion Project;

    - Go to '<polarion_project> > Linked Data > Linked Data Associations';
    - This configuration is made at `http://<polarion.mycompany.com>/polarion/#/project/<polarion_project>/administration/linkeddata/serviceproviders`;
- Declare the Publication for Capella server as a friend:
    - Click on 'Add Association';
        - In the dialog that opens, select the relevant Publication for Capella friend server (that has been configured in section Global Configuration):

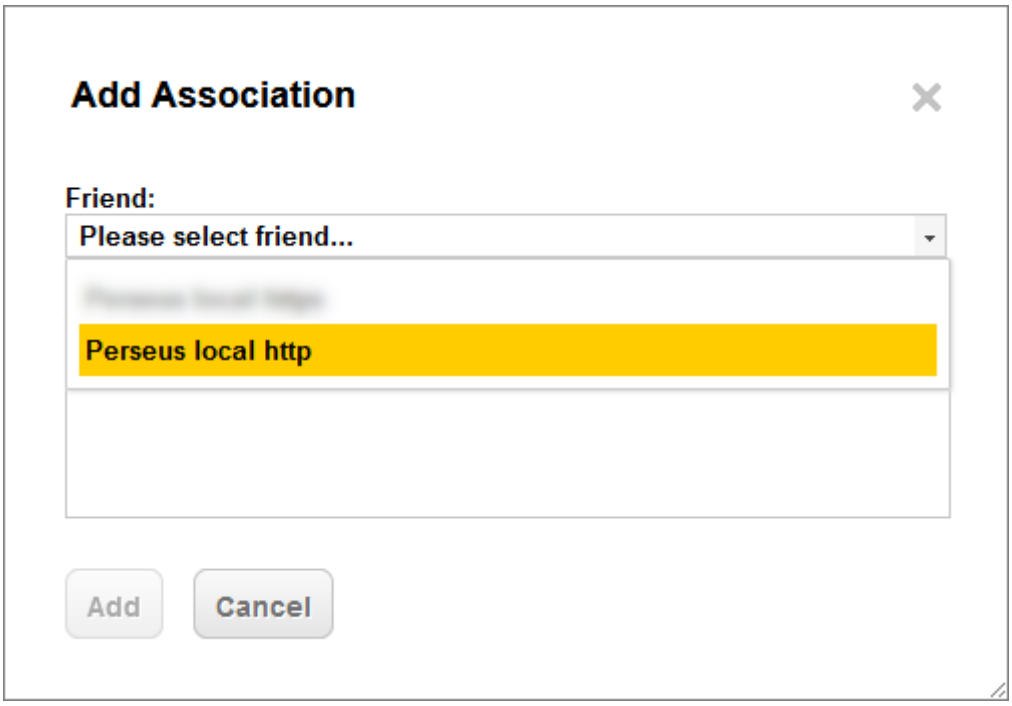


Figure 183. Select Friend Publication for Capella Server

- A pop-up window opens

check the web browser pop-up settings if the pop-up doesn't show up.

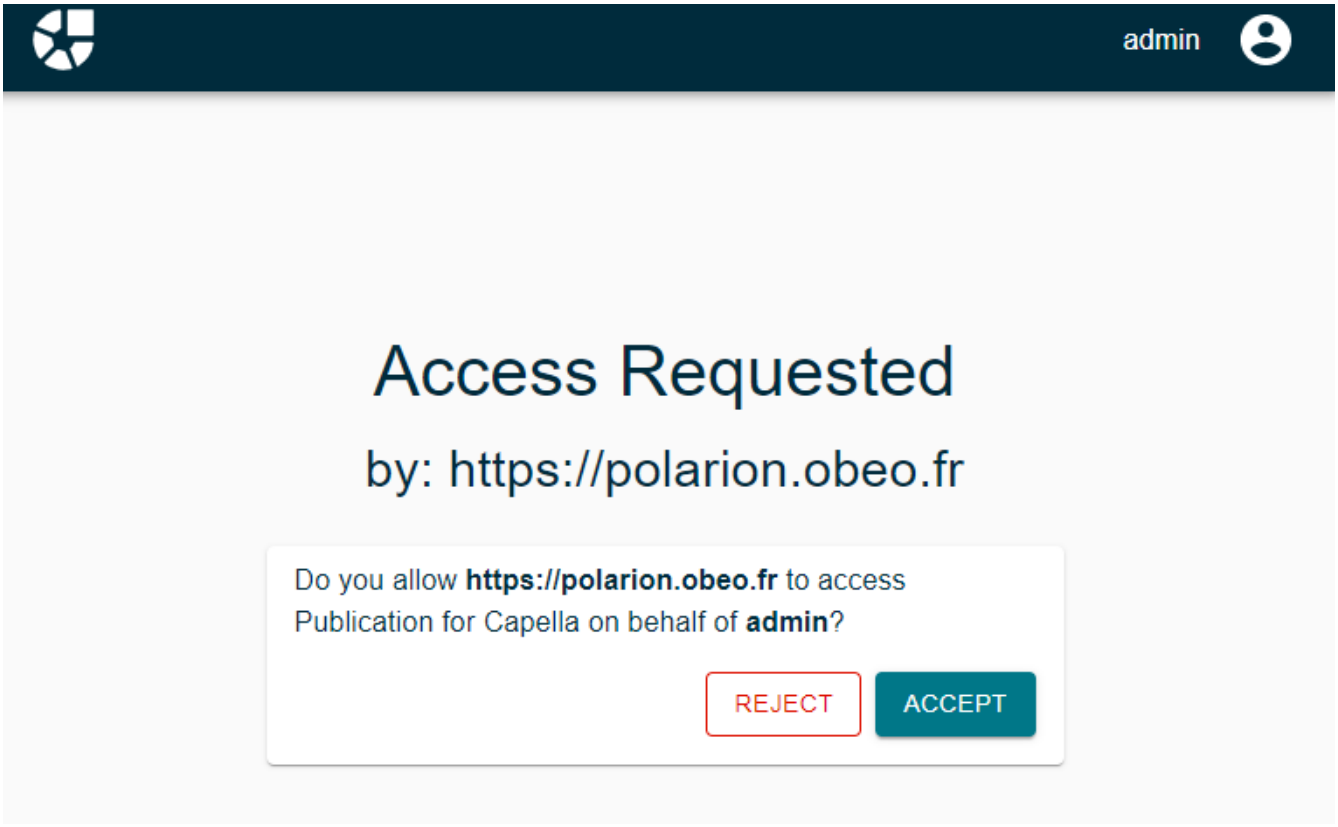


Figure 184. Credentials for Project Association

Sometimes, Polarion's authentication is rejected because the authentication token is too old. In such a case, you will get an error message indicating that the connection failed. Close the pop-up window and reopen it and Polarion should authenticate properly.

- Enter valid credentials to connect to Publication for Capella;
- Select project(s) you want to link to then click on 'Add':

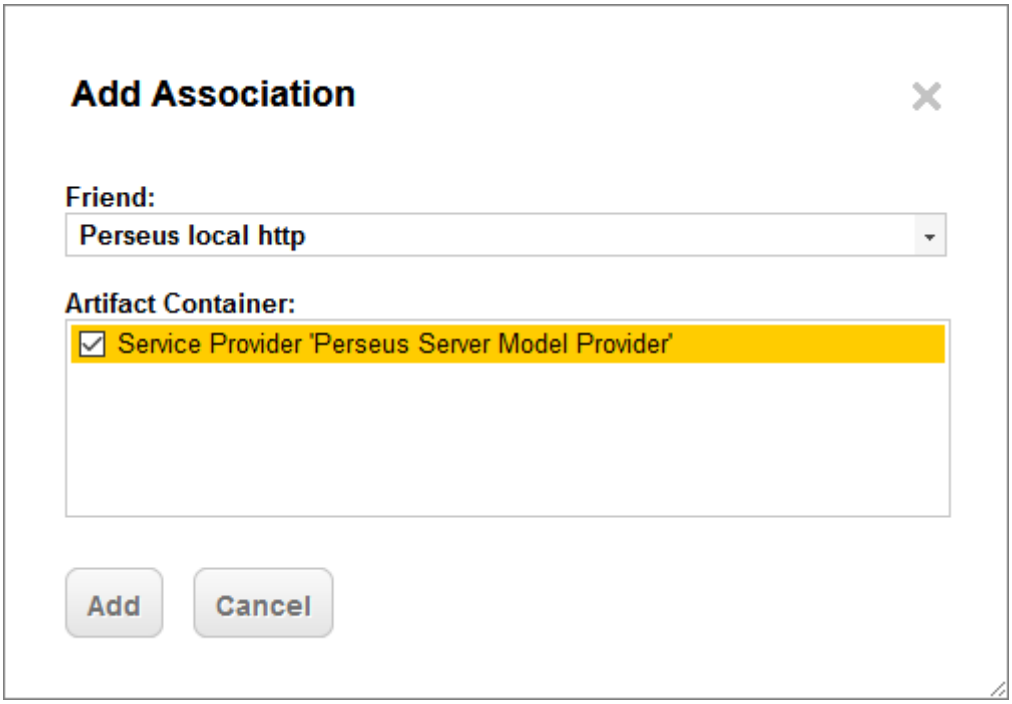


Figure 185. Select Publication for Capella Service Provider

- The association appears in the list of the Polarion project's linked data associations;

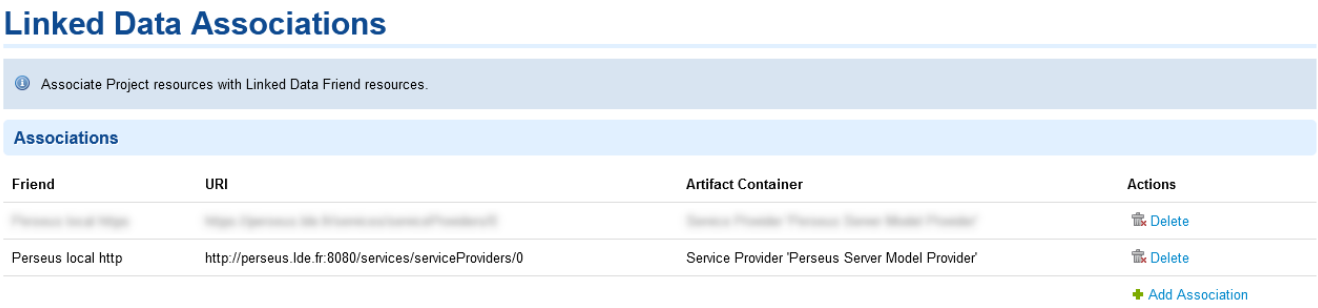


Figure 186. New Project Association with Publication for Capella Added

- Go to the Linked Data Mapping page of the Polarion Project;
  - This configuration is made at `http://<polarion.mycompany.com>/polarion/#/project/<polarion_project>/administration/linkeddata/mapping`;
  - In the `<link-role-mappings>` section, add mappings for the roles you want to use to link polarion objects to Publication for Capella model elements;
  - Save the configuration;
    - Example:

```
<link-role-mappings>
  (...)
  <link-role-mapping linkRole="elaborated_by"
oslcLinkProperty="oslc_rm:elaboratedBy"/>
  <link-role-mapping linkRole="satisfied_by" oslcLinkProperty="oslc_rm:satisfiedBy"/>
  <link-role-mapping linkRole="specified_by" oslcLinkProperty="oslc_rm:specifiedBy"/>
</link-role-mappings>
```

- Go to the configuration management page of the Work item enumerations;
  - This configuration is made at `http://<polarion server>/polarion/#/project/drivepilot/administration/workitems/enumerations`;
- Edit the `workitem-link-role-enum.xml` by clicking on the 'Edit' link in the column 'Actions';
  - Make sure there is a line for each role defined in the previous step, and that these roles are associated to the right type of Polarion element. If not then create them.
    - Example:

Figure 187. Workitem Link Role Item Configuration

- Click on 'Save' to save the configuration.

Configuration is over. It should now be possible to link Polarion work items to Publication for

Capella model elements.

If you want to create links from Publication for Capella to Polarion, then you need to made same kind of configuration but into Publication for Capella. Refer to the Publication for Capella user manual, chapter `OAuth Consumers Management`.

Linking from Polarion to Publication for Capella

Glossary

Model

A model is a set of objects with properties and links between each other. These properties and links define a semantic that can be understood by knowledgeable users or software. In general, these properties and links conform to a set of rules defined by a meta-model.

Account

A user account that makes it possible to a given person or software to connect to the Publication for Capella server.

Team

A team groups a number of users to give them specific access rights to a specific project.

Project

A project groups any number of models stored by the Publication for Capella server. It is the unit of exposure of data to the OSLC API of the Publication for Capella server.

OSLC

Open Services for Lifecycle Collaboration. OSLC is a standard defining how distinct repositories can link their data and keep them synchronized.

Overview of Polarion - Publication for Capella Connectivity

Publication for Capella exposes model data according to the OSLC-AM (Architecture Management) domain.

Polarion can be configured to make it possible to link Polarion items to model objects exposed by Publication for Capella.

This document describes how links can be created from Polarion tickets to Capella model elements exposed by Publication for Capella.

Pre-requisites

- The Publication for Capella server must be running. For the examples used in this document, we will assumed it is deployed at the URL `http://perseus.mycompany.com:9443`
- An administrator must have configured the association between Polarion and Publication for Capella, as described in the Publication for Capella Deployment guide.

Creating a link to a Model Element

- Navigate to the Polarion work item that you want to link to a Publication for Capella model object.
- In the 'Linked Work Items' section, click on the pencil to edit the list of related work items.

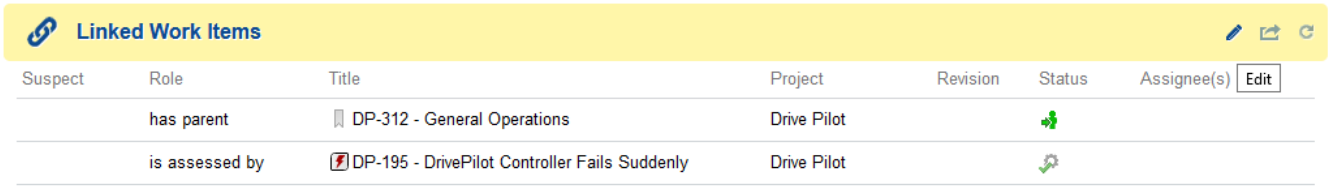


Figure 188. Linked Work Items

- In the 'Role' combo-box select the kind of link that you want to use. It must be configured to be compatible with objects exposed by Publication for Capella.

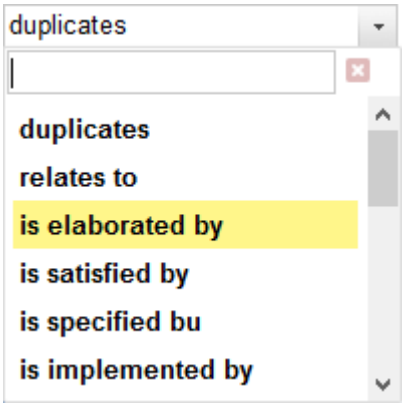


Figure 189. Polarion Link Types

- The little icon to link to external items become active.
  - Click on the icon to link to a remote item.
  - In the pop-up that appears, select the Publication for Capella friend server that contains the model elements you want to link to.

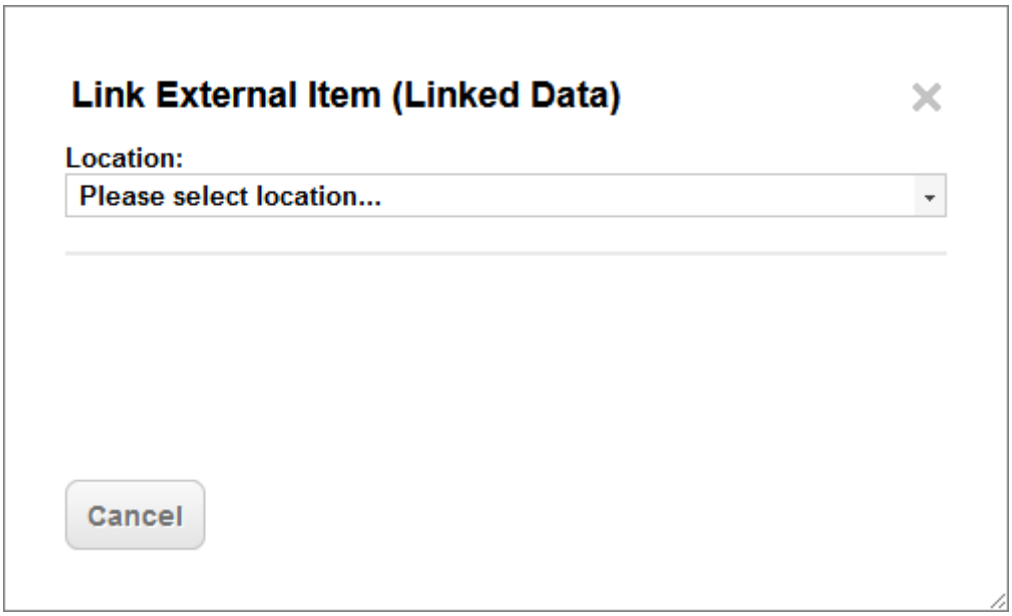


Figure 190. Select Publication for Capella Friend Server

- Then, click on 'Select an Existing Item'.
  - NOTE: Publication for Capella does not support creating new model elements from Polarion. It is only possible to link to existing model elements. This is not a limitation but rather a design decision, and this is not expected to change in the future.

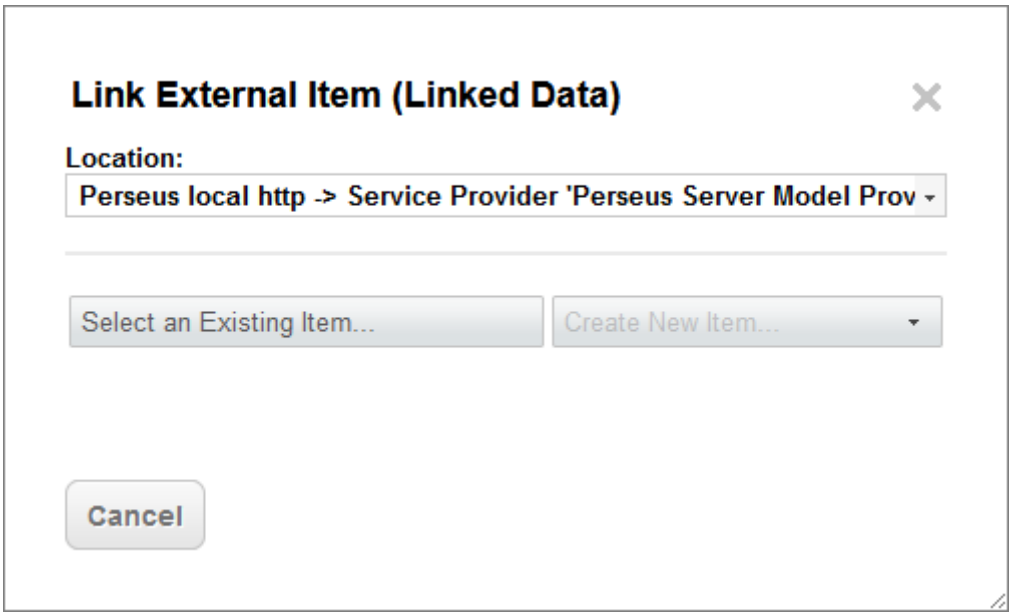


Figure 191. Select Existing Item

Figure 192. Start Adding New Link

- If needed - Enter valid credentials if prompted for credentials.
- If needed - Allow Polarion to browse your data on Publication for Capella if prompted to.
- A dialog is displayed to search for model elements in the Publication for Capella server. Elements to link can be retrieved a model explorer or searched using Publication for Capella filtered search UI.

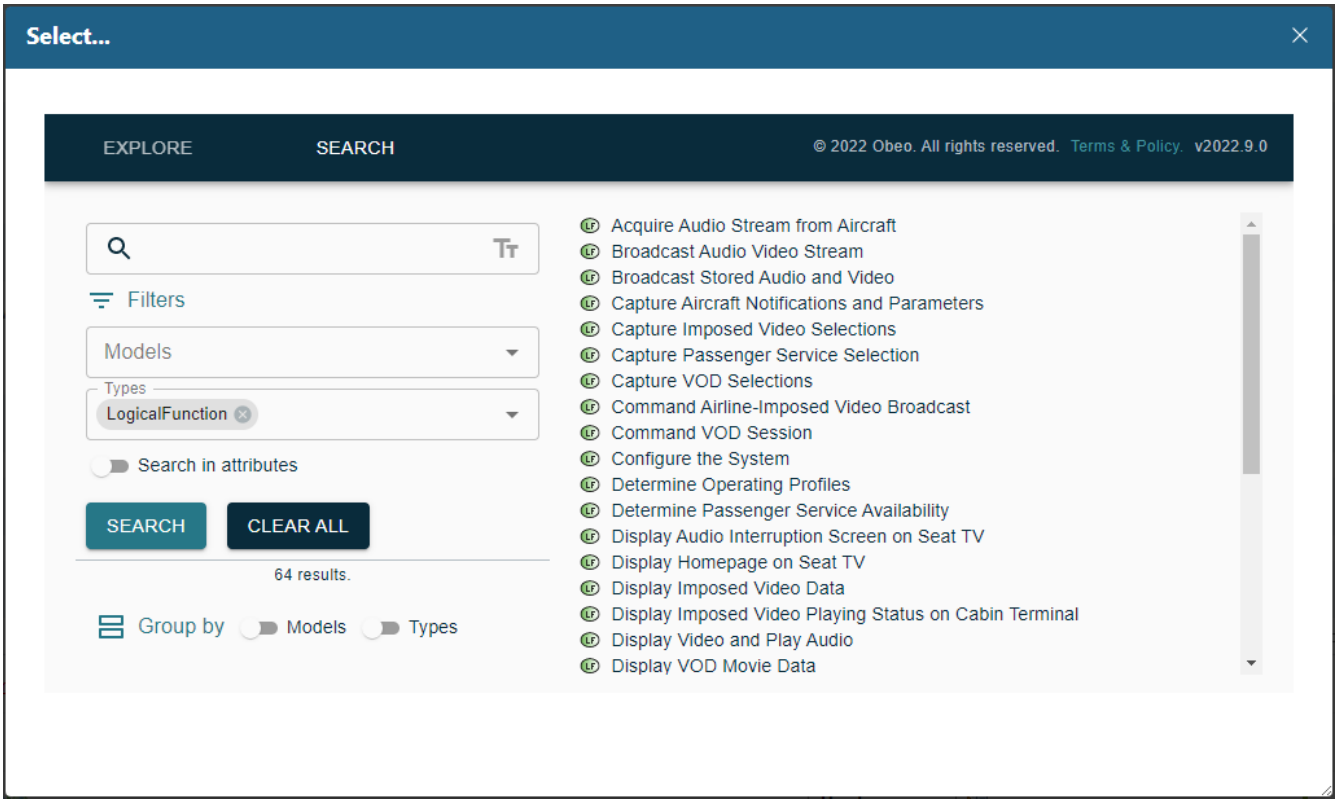


Figure 193. Search Dialog

- Select the element you want to link to by clicking on it.
- The selected element gets added to the Polarion.

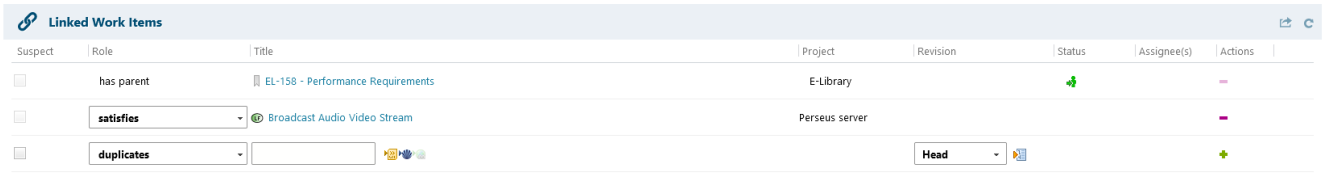


Figure 194. Remote Item Added

- Save the modifications of the work item.

Removing a link

- In the 'Linked Work Items' section, click on the pencil to edit the list of related work items.
- Click on the minus sign to remove a link
- Save the changes

Known Limitations - Polarion integration

- Publication for Capella will sometimes return an error 'Expired OAuth token' - Just reattempt to link the element, you will have to allow Polarion to access your data again.
- Search limitations:
  - Filtering elements that are sub-types of a given type is not supported.

Publication for Capella Integration with Jama Connect

Introduction

Publication for Capella version 2025.10.0 introduces compatibility with Jama Connect.

This document describes:

- The specifics of this integration within Publication for Capella;
- The effect of this integration within the Jama Connect web UI.

Pre-requisites to the Publication for Capella Integration with Jama Connect

This integration does not require any additional plug-in or add-on installed in Jama Connect. Even though Publication for Capella is advertised as an OSLC compliant application, it is **not** necessary (and not even useful) to activate the OSLC option of Jama Connect.

Setting Up Authentication with Jama Connect

As usual in Publication for Capella, all data exchanges between Publication for Capella and another server are done on behalf of a specific user. In general, this is the logged-in user, but regarding the Jama Connect integration there is a special case to deal with the polling of data.

Anyway, it is necessary that any Publication for Capella user who needs to use the Jama Connect integration also has a personal account on the connected Jama Connect server.

Creating API Credentials in Jama Connect

Each user must create an API Key dedicated to Publication for Capella in the Jama Connect server to enable communication between Publication for Capella and Jama Connect. This requires a *creator license* in Jama Connect (at the time this documentation is written).

Creating an API key is achieved by going to the user's Profile page:

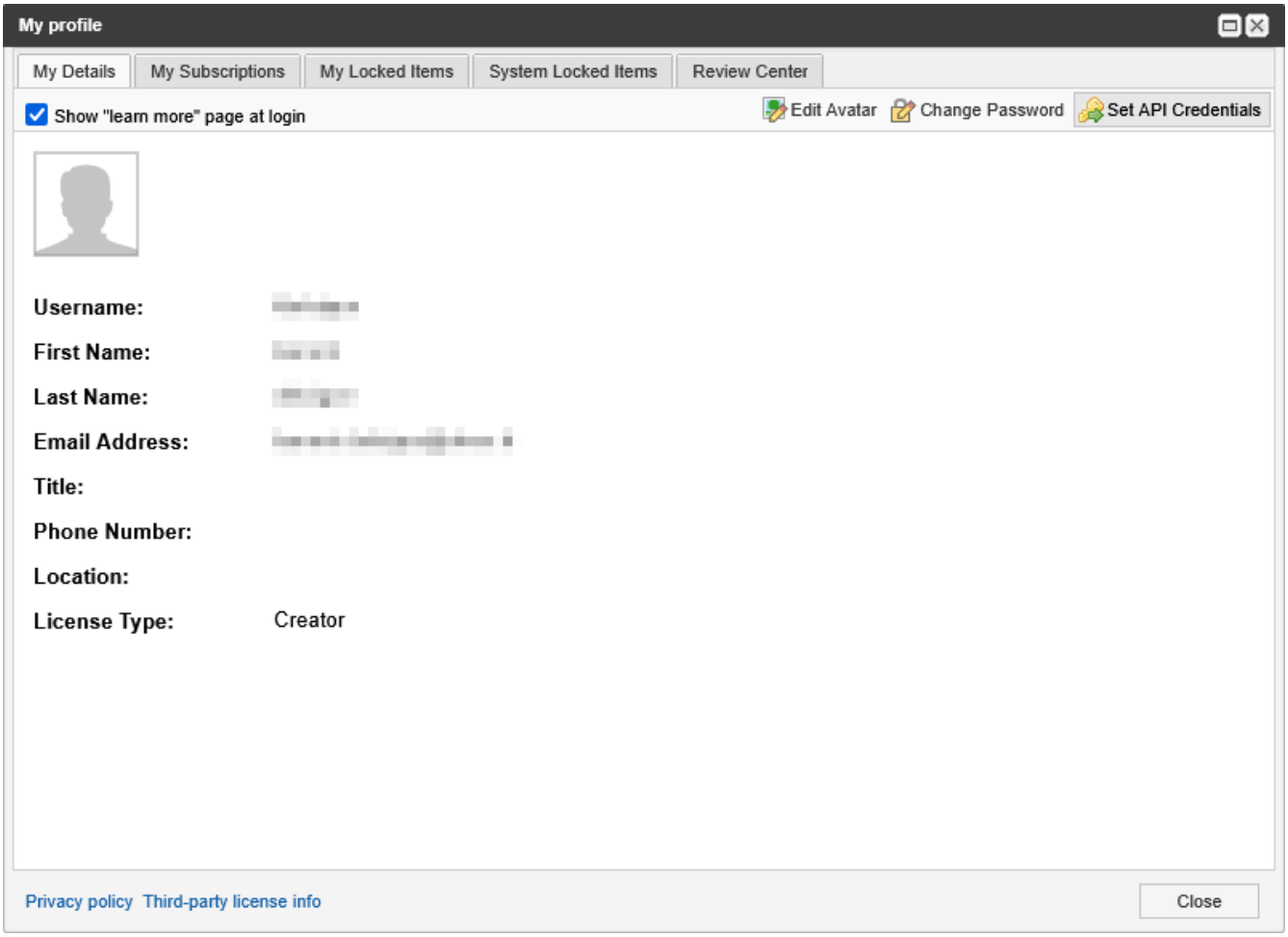


Figure 195. Jama Connect User Profile Page

Then, click on 'Set API Credentials'

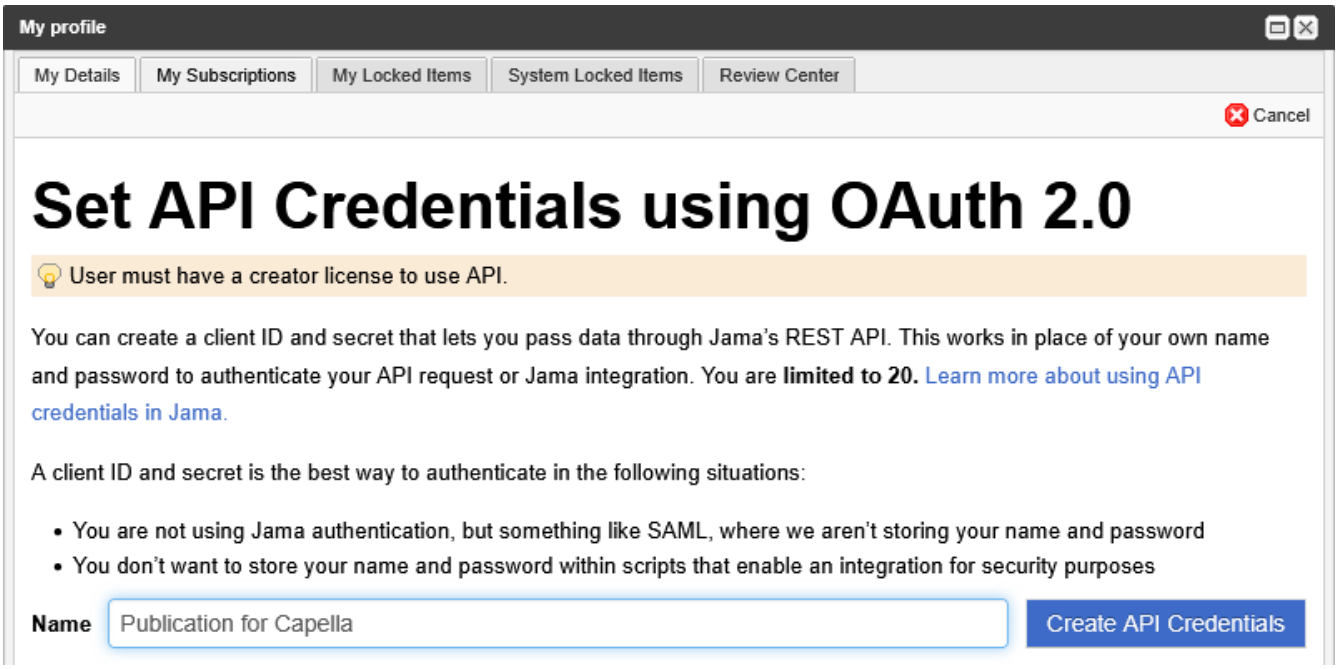


Figure 196. Jama Connect Create API Key - Step 1

Set the label to 'Publication for Capella' or any meaningful value and click on 'Create API Credentials'.

The page will then display the newly created credentials.

Storing the Jama Connect API Credentials

The user must copy/paste these credentials and store them securely; they will be needed within Publication for Capella to link the user's Publication for Capella account with the user's Jama Connect account.

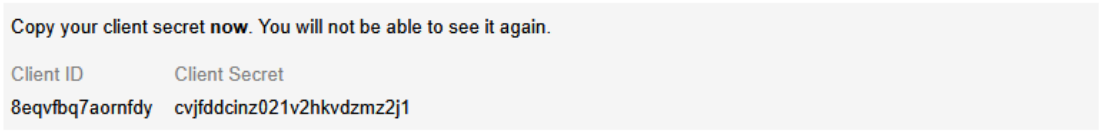


Figure 197. Jama Connect Create API Key - Step 2

Later on, displaying the Jama Connect Profile page will only show the API Key, but never the associated secret, that's why it is important that the user stores this password securely somewhere, at least until it is set in Publication for Capella.

Setting Jama Connect API Credentials in Publication for Capella

The first time the Publication for Capella server will need to access Jama Connect on behalf of a user, it will prompt for the user's API Key Credentials.

The user must then enter their Jama Connect API Key and secret, which will be stored by Publication for Capella. These credentials will make it possible for Publication for Capella to authenticate with Jama Connect on behalf of this user whenever it is necessary, using a state-of-the-art authentication mechanism known as OAuth-2.0.

Revoking Jama Connect API Credentials

The user can revoke their Jama Connect credentials at any time by deleting them from its user profile page on Jama Connect.

This will prevent any further communication on behalf of this user between Publication for Capella and Jama Connect.

Such credentials can be deleted and recreated at will, if the Jama Connect limit of number of API Keys per user is not exceeded.

Specifics of the Publication for Capella Integration with Jama Connect

This section describes how the Jama Connect integration differs from other integrations provided by Publication for Capella.

Creation of Links in Publication for Capella

Links to Jama Connect items can only be created in Capella for now.

This limitation may be removed in a later version, but for now links cannot be created from a model object page in the Publication for Capella web UI.

Management of Diagrams as Attachments

The Jama Connect integration provides a specific feature that is not available for other integrations: When a diagram is linked to a Jama Connect Item, the diagram image gets attached as a JPEG image to the items it is linked to. This attachment's image of all modified diagrams is updated each time the model is published.

Management of Attachment Synchronization Issues

Like issues with link synchronization, there can be issues dealing with attachment synchronization. Publication for Capella provides mechanisms to safely recover from such issues without incurring data loss or data corruption.

Such issues will cause a warning icon to appear on the involved model in the model explorer:

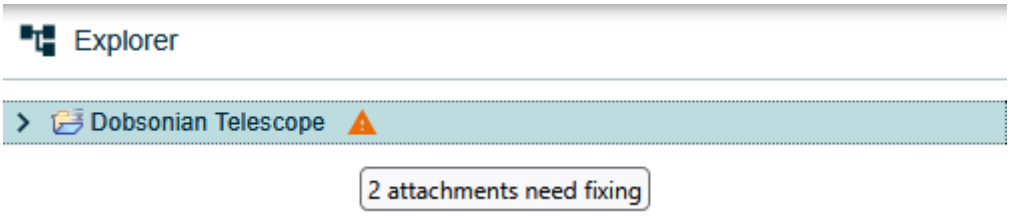


Figure 198. Attachment Issues in Model Explorer

Such issues can be examined and addressed on the model page:

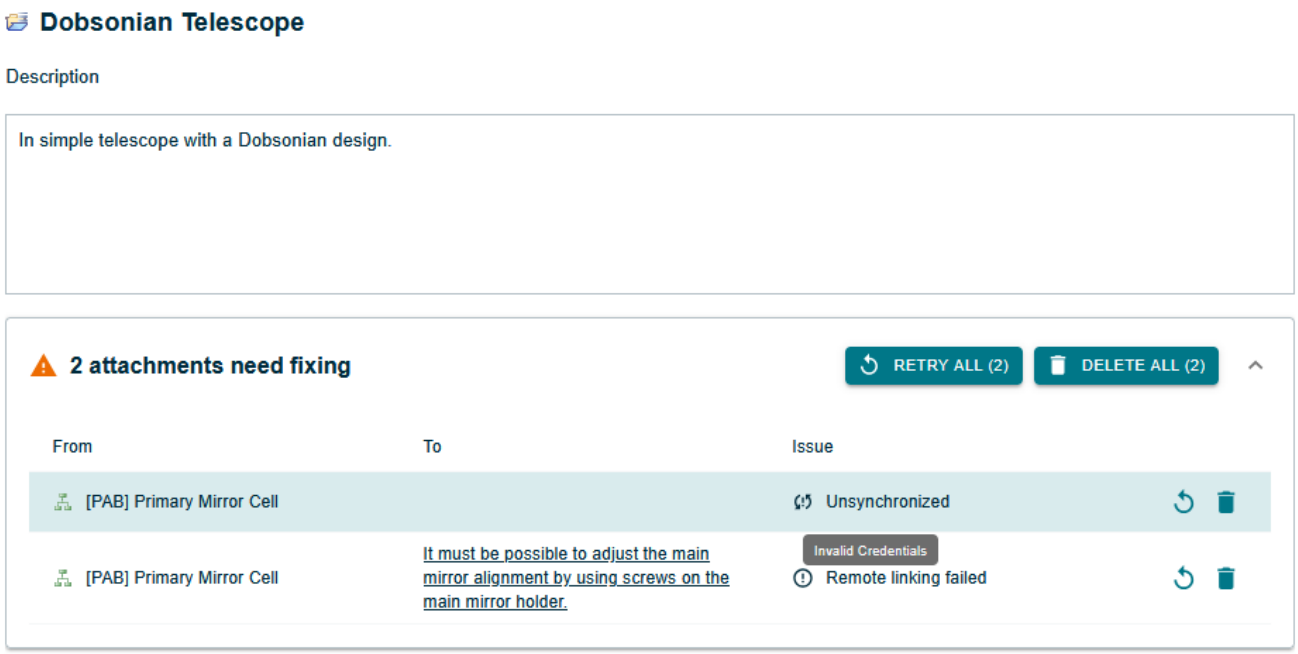


Figure 199. Attachment Issues in Model Page

There are 2 kinds of attachment issues:

- Issue with attachment data (e.g. the image data could not be updated because of an authentication issue).
- Issue with attachment links with Jama Connect items:
  - Failure to *add* an attachment to an item.

    - Failure to *remove* an attachment from an item.

In the screenshot above, the first line (with the empty column 'To') is an attachment data synchronization issue, whereas the second line is an attachment link synchronization issue.

For each issue, two actions are available to the right of the corresponding row:

- Retry the action that failed. This will solve authentication issues by re-asking the current user's API Key credentials if that is needed.
- Delete the failing element:
    - If the issue is on attachment data, this action will delete the attachment **in Publication for Capella only**, without performing any action in Jama Connect. The user should then go remove the corresponding data in Jama Connect, if there are any.
    - If the issue is on an attachment link, this action will delete the attachment link **in Publication for Capella only**, without performing any action in Jama Connect. The user should then remove the corresponding attachment link in Jama Connect, if there are any.

Finally, two global actions are available at the top of the table, to apply the same action to all attachment issues:

- Retry all.
- Delete all.

Deactivating Attachment Management

If, for any reason, pushing Capella diagrams as attachments to Jama Connect is not desirable, it is possible to deactivate attachments completely. The *system administrator* must set the following property in the file `config/application.properties` to `false` (its default value is `true`):

```
perseus.sync.attachments.enable = false
```

Polling of Associated Jama Projects

Whenever links are created online during a publication, Publication for Capella will update the relevant Jama Connect Items by adding or removing the modified links and managing the Diagram attachments if there is any. This will work if the 'OSLC Linking Configuration' is properly set up.

To listen to link changes that may happen in Jama Connect, the project owner must decide which Jama Connect projects to listen to and must declare one association for each of the Jama Connect projects that needs to be followed.

For each of these associations, Publication for Capella will regularly 'poll' the friend Jama Connect server to verify whether traceability links (created from Publication for Capella) have been deleted (or added, even if this is unlikely).

This polling mimics the usual OSLC mechanism that makes it possible to delete links from any server while keeping the data synchronized on both servers. The main difference is that there will generally be a delay for deletions to be considered, whereas with the standard OSLC mechanism

deletions are immediately considered. Another difference is that all the link deletions detected by this polling mechanism will be part of a single new commit, whereas the standard OSLC mechanism will cause one commit per link deletion.

Whenever traceability link deletions or additions are detected, Publication for Capella will automatically make the corresponding change on the associated links within the Publication for Capella server.

The polling delay is set by default to 30 seconds, but this delay can be configured by the *Server Administrator*.

Authentication for Polling

Polling is performed on behalf of the **project owner**.

Consequently, the project owner must create Jama Connect API Credentials and set them in Publication for Capella.

The recommended way to do this for the project owner is to declare the relevant associations, which will ask for the project owner's Jama Connect API Credentials, if needed.

If the ownership of the project is transferred to a different user, this user must in turn ensure they create API Credentials in Jama Connect and set them in Publication for Capella.

Polling Warnings

If anything goes wrong during polling, a warning marker will be displayed to the right of the project name. Hovering this marker will display an explanatory tooltip.

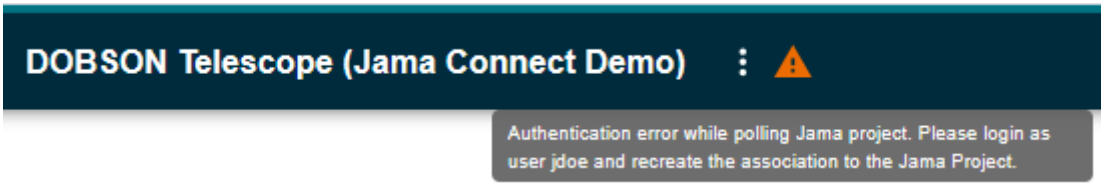


Figure 200. Jama Connect Polling Warning

Polling Settings

The system administrator can deactivate polling completely or set the polling interval to any desired value.

Here are the relevant properties:

```
# Set the following to false to deactivate polling
perseus.jama.poll = true
# set the following to an integer value which will be used a the polling interval, in
seconds.
# It is recommended not to use a value lower than 10 seconds.
```

```
perseus.jama.poll.delay.seconds = 30
```

Known Limitations Specific to the Jama Connect Integration

- Previews of linked Jama Connect items are not available in the Publication for Capella Web UI.
- Creation of links to Jama Connect items is not possible in the Publication for Capella Web UI. The only way to create links is by drag & drop or copy/paste in Capella.
- It is technically possible to create links from the Jama Connect Web UI, but it is complicated and error-prone, so it is recommended not to try to create traceability links from Jama Connect. The procedure to create such links is not documented, and it is not considered as a supported feature.

Effects of this Integration in the Jama Connect Web UI

The integration of Publication for Capella with Jama Connect has two main visible impacts:

- Links to Capella model elements exposed by Publication for Capella can be present in Jama Connect item pages. Clicking on such links from the Jama Connect UI will navigate to the corresponding page in Publication for Capella.
- Capella Diagram images can be visible as *Attachments* in Jama Connect item pages.

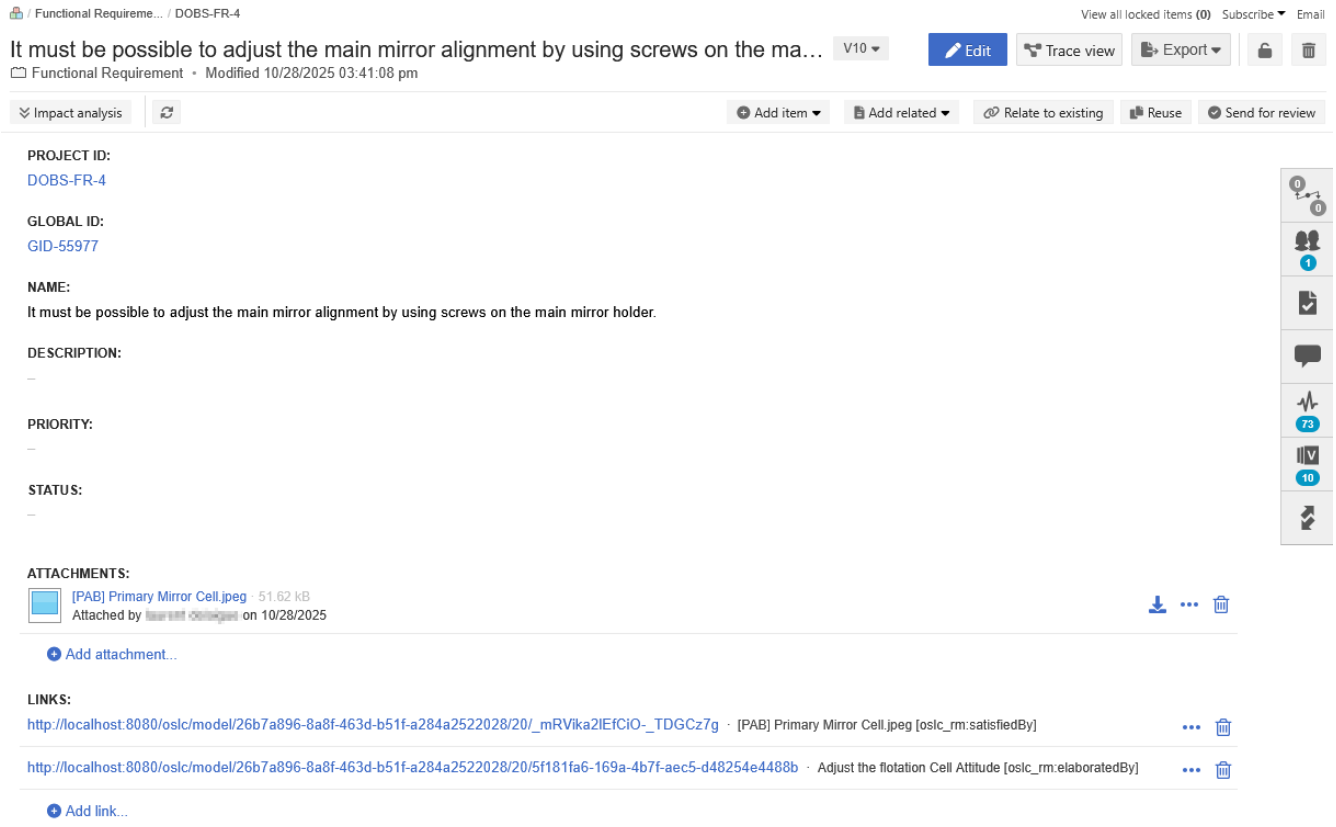


Figure 201. Impacts on a Jama Connect Item Page

It is possible to delete such traceability links directly within Jama Connect, and the polling mechanism (See Polling of Associated Jama Projects) will apply the corresponding deletion in Publication for Capella after a short delay.

If the link that is deleted is a link between a Capella diagram and a Jama Connect item, then the deletion of this link in Publication for Capella will cause the attachment to be detached from the corresponding item.

Release Notes

Version 2025.10.0

Main Server Changes

- PERS-816 - Publication for Capella is now compatible with Jama Connect™ by Jama Software.
- PERS-800 - The End-User License Agreement (EULA) has been updated.
- PERS-795 - The default property mappings configuration has been improved to support more types of Capella elements. Make sure you use the latest version of the file `config/prop-mappings.json` in your deployment to benefit from these improvements.
- PERS-783 - It is now possible to create **baselines** on any version of published models.
- PERS-784 - Any version or baseline of published models can now be browsed online. In addition, hovering the history view on any commit displays a preview of the currently selected object, as it was in this commit.

Contributor Client Changes

- PERS-781 - URL patterns are now automatically created for each recognized friend server. This automation works for most standard configurations of: IBM DOORS Next, IBM Engineering Workflow Management, Polarion, Jira, Jama Connect. It is still possible to edit these configurations manually should the need arise.

Minor Changes and Bug-Fixes

- PERS-847 - The online semantic browser panel 'OSLC Links' has been renamed to 'External Links'.
- PERS-829 - A bug has been fixed that prevented the publication result from being fetched by a user different from the one who initiated the publication.
- PERS-796 - A bug that prevented the semantic browser from displaying properly online has been fixed.
- PERS-794 - A bug that prevented moving artifacts in folders in the traceability model has been fixed. This makes it possible to easily reorganize the structure of artifacts imported in models by drag & drop or copy/paste.
- PERS-780 - Associations created to Polarion projects are now automatically configured to keep the dialogs open after confirmation.
- PERS-777 - A bug that caused 'Update publication result' to loop forever in case of invalid credentials has been fixed.

Version 2025.6.0

Server Changes

- PERS-579 - Publication for Capella is now officially compatible with Jira, *via* the add-on OSLC Connect for Jira by Sodius.
- PERS-743 - Publication for Capella is now officially compatible with Confluence, *via* the add-on OSLC Connect for Confluence by Sodius.
- PERS-748 - Rebase on latest Sirius-Web and Obeo Enterprise for Sirius.
- PERS-746 - A leak of JDBC connections has been fixed. This bug could cause a freeze of the Publication website for up to 30 seconds.
- PERS-750 - Add edge label in image maps. It is now possible to click on the labels of semantic edges (such as Functional or Component exchanges) in published diagrams to navigate to the associated semantic element. Diagrams must be republished, former diagrams will not exhibit this behavior. Edges that are not tied to a specific semantic element won't have a clickable label. The labels of edge ends are not clickable either.
- PERS-605 - Add previews in diagram images. It is now possible to preview the elements of a diagram, either internal or external, by hovering them with the mouse.

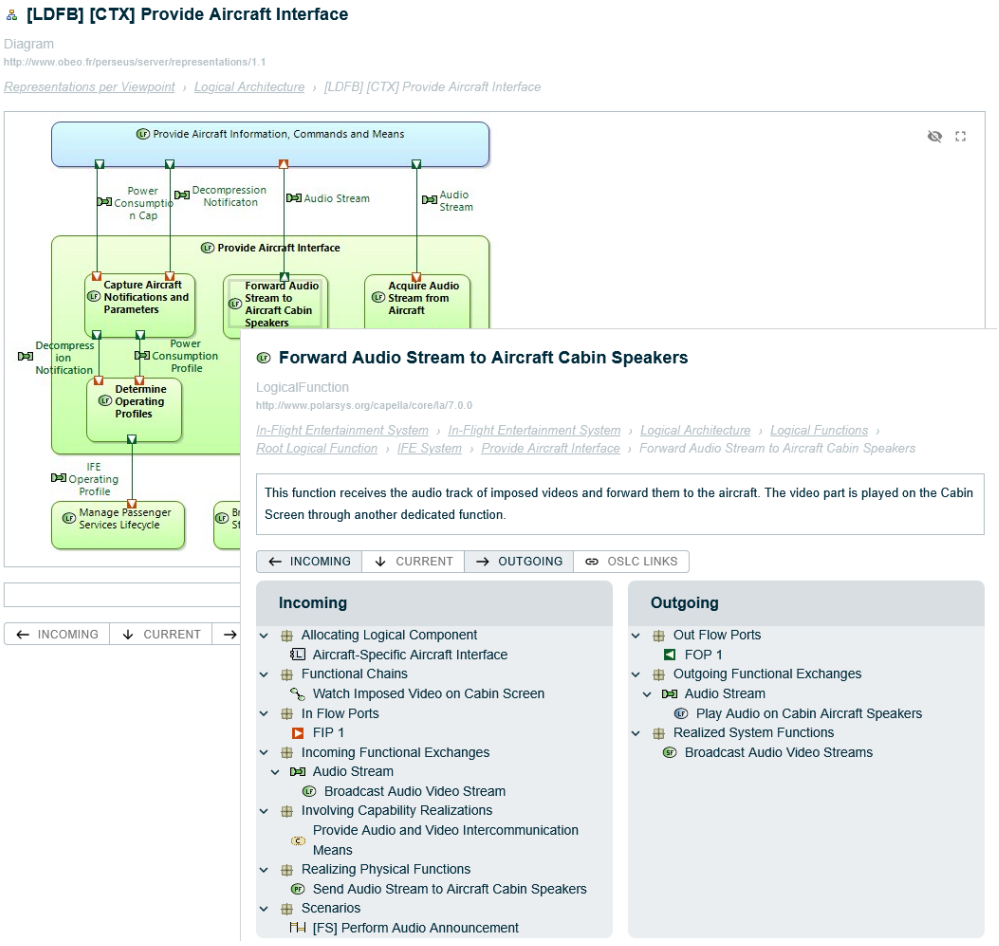


Figure 202. Previews are now available in published diagrams.

- PERS-760 - Add previews in rich text descriptions. It is now possible to preview the internal links to diagrams or model elements in the published rich text description of model elements.

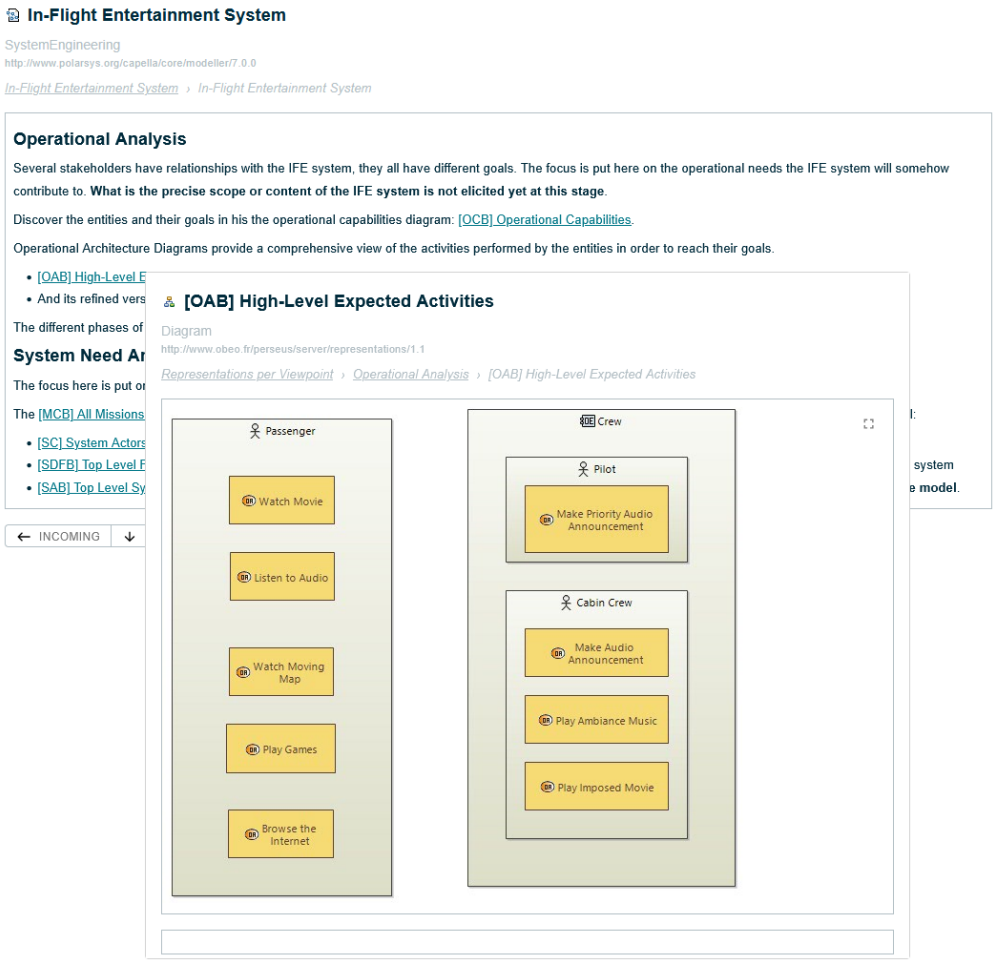


Figure 203. Previews are now available in the Rich Text documentation.

- PERS-770 - The default meta-class filter is now better aligned with the default Capella filters so that the model tree are, by default, identical.
- PERS-771 - Some diagrams were missing in the semantic model tree, this bug has been fixed.

Contributor Client Changes

- PERS-754 - Fix undesirable side-effect when traceability is used with Requirements viewpoint. An involuntary interaction between the popular Capella Requirements add-on and Publication for Capella was preventing the display of the artifact to diagram compartment (See PERS-541 below).
- PERS-752 - Move URL Mappings from preferences to traceability model. This change makes it easier to share the settings for drag&drop or copy/paste of URLs between team members.
- PERS-747 - Fix features metadata.
- PERS-719 - Improve traceability semantic browser integration.

Figure 204. Links Grouped By Type in Semantic Browser

- PERS-624 - Display ID of artifacts in labels.

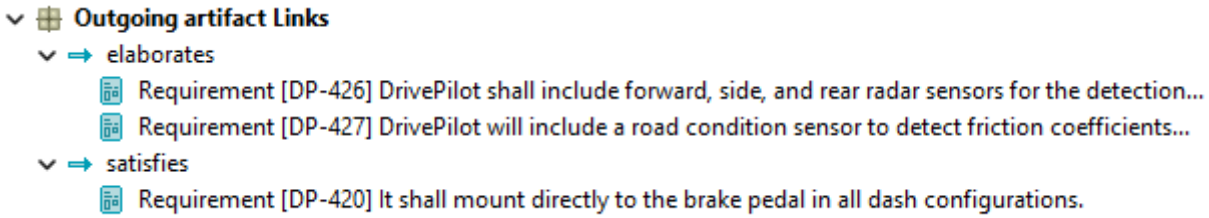


Figure 205. Artifact labels now include the artifact ID between brackets.

Version 2025.2.2

Hot-fix version to improve OpenID Connect and OAuth-2.0 support especially for opaque access tokens.

Server Changes

- PERS-751 Support configuration for opaque access tokens and not only JWT access tokens.

Contributor Client Changes

- PERS-751 Improve use of refresh tokens to avoid too many user interactions for authentication.

Version 2025.2.1

Hot-fix version to resolve a bug on the creation of associations with Friends that expose many service providers.

Server Changes

- PERS-745 Fix bug that prevent the creation of an association with a service provider of a Friend that exposes many service providers.
- PERS-579 Add documentation for connecting Publication for Capella with Atlassian Jira.
- PERS-744 Make it possible to support several signing algorithms for OpenID Connect.

Version 2025.2.0

Publication for Capella is now based on *Sirius-Web 2025.1.0* and *Obeo Enterprise for Sirius 2024.11.2*.

Here are the most interesting improvements:

- A simple mechanism to prevent some diagrams from being published by simply prefixing their name with `[PRIVATE]`;
- The possibility to easily create trace links to diagrams in Capella, which used to be quite difficult;
- Publication settings have been moved from preferences to the traceability model so that it's easily shared by all the engineers who work on a model.

Below is a full list of changes brought by this release.

Server Changes

- PERS-706 - The Publication for Capella server has more options to support more OpenID Connect authentication providers.
- PERS-709 - Unauthorized update of consumers has been fixed.
- PERS-710 - Unauthorized access to administrator pages has been fixed.
- PERS-722 - Updating an existing consumer has been fixed.

Contributor Client Changes

- PERS-706 - The Publication for Capella contributor client has more options to support more OpenID Connect authentication providers.
- PERS-541 - It is now easy to link diagrams to artifacts, either by drag and drop, or by copy/paste.
  - Artifacts can be dropped in a diagram background to create a link to this diagram.
  - Artifact URL can be pasted on a diagram background to create a link to this diagram.
  - Artifacts linked to the diagram in which they are displayed show the link in a dedicated compartment.
  - Links to diagrams that were created using previous versions of Publication for Capella will be automatically migrated when updating from the server.
  - This resolves performance issues that may arise with *Team for Capella* in certain cases.
- PERS-598 - It is now possible to filter diagrams on publish, based on a simple naming rule:
  - By default, diagrams whose names are prefixed with "[PRIVATE]" will not be published.
  - This prefix is configurable.
- PERS-720 - Actions requiring an OAuth2 login can now be cancelled.
- PERS-726 - Artifacts can now be dropped in the "Representations per category" folder in the Capella Model Explorer.

Version 2025.1.0

Publication for Capella is now based on Sirius-Web 2025.1.0 and Obeo Cloud Platform 2024.11.1. This version brings support for the OpenID Connect protocol to support user authentication.

It also brings an experimental support of CodeBeamer integration through OSLC.

Server Changes

The Publication for Capella back-end server has undergone many changes to align better with the desired architecture and take advantage of Sirius-Web and Obeo Cloud Platform.

- PERS-511 - It is now possible to log in using OpenID Connect, in addition to the legacy authentication system based on user/password credentials.
- PERS-546 - A validation of the publication data is now made by the server before triggering a job. This validation can reject the publication if some link types have been defined in the contributor Client but not in the server. In such a case, the server's configuration needs to be aligned with the contributor client configuration by creating the missing link types in the OSLC linking configuration.
- PERS-412 - Links that point to a delete artifact can now be detected and dealt with.
- PERS-577 - Publication for Capella is now compatible with CodeBeamer. The integration with CodeBeamer is achieved based on OSLC and is considered as *experimental* in this release.

Web Client Changes

- PERS-539 New options are available on OSLC associations.
  - Dialogs can be filtered in the delegated link creation dialog by type (selection or creation).
  - Another option makes it possible to decide whether the delegated link creation dialog must remain open after link creation or not.
- PERS-540 Diagrams are now displayed in the semantic model tree. They are still displayed in the 'Representations per Viewpoint', as before.
- PERS-557 Properties displayed in the Web client have been improved for many types of Capella objects.
- PERS-576 A preview is now displayed when a link to a third-party artifact or a published model element is hovered with the mouse.
- PERS-544 A warning is now displayed on model tree nodes when the model contains invalid links

Contributor Client Changes

- PERS-518 The contributor client can now use OpenID Connect to authenticate with the Publication for Capella server. This change improves security by removing the need to store the user's credentials on the local machine's secure store.
- PERS-550 The command 'Show online' now opens the diagram page when invoked by right-clicking on a diagram's background. In previous versions, it used to open the page of the semantic target of the diagram.
- PERS-564 An option to refresh diagrams before publishing has been added.
- PERS-565 A new way to create traceability links has been added, using copy/paste. First, copy the URL of the third-party artifact in a web browser, then select one model element in a diagram

and use Crtl+V to paste it. This will create a traceability link.

- PERS-575 The outcome of a Publication is now displayed in Capella to the Capella user when the publication job is over. The user is thus notified if there is a problem or if the publication was OK.

Version 2024.05.1

This is a hot fix version released to resolve a bug that prevents large models from begin published. Only the server is impacted, there is no new version of the *Contributor Client*.

Server Changes

- PERS-517 A bug preventing large models from being published has been fixed. This bug only impacts release 2024.5.0. Every customer using version 2024.5.0 is invited to upgrade to version 2024.5.1.

The *Contributor Client* version 2024.5.0 remains unchanged and is compatible with the Publication for Capella server version 2024.5.1. There is no need to upgrade the *Contributor Client* when upgrading a Publication for Capella server from 2024.5.0 to 2024.5.1.

Version 2024.05.0

Server Changes

- PERS-515 Publication for Capella now sets the `Security-Content-Policy` in its response headers so that OSLC friend web sites and white-listed web sites can embed its pages in `iframe` elements. Refer to Content Security Policy White List for more details.
- PERS-516 Publication for Capella now properly displays the OSLC links title even when they contain HTML elements.

Contributor Client Changes

- PERS-503 Messages displayed when a drag and drop takes place on unsupported element have been improved to better explain why no link is created.
- PERS-504 The configuration of the types of links that can be created between Capella elements and third-party artifacts has been improved to make it more simple and more powerful. It is now possible to use super-type names. Refer to Artifact Repositories for more details.
- PERS-505 An option `fr.obeo.perseus.ssl.trust` has been added to facilitate the use of the contributor client with self-signed SSL certificates. Refer to Trusting the Publication Server Certificate for more details.
- PERS-512 A bug that prevented publishing models with broken links has been fixed. Such links (that have no source or no target) are now ignored but do not block the publication.
- PERS-514 A category has been added in the semantic browser to display all the applied property values of an element in a way that is compliant with the PVMT Capella Add-on. When a model is

published, this category is also displayed online so that users can easily see which property values are applied to an element.

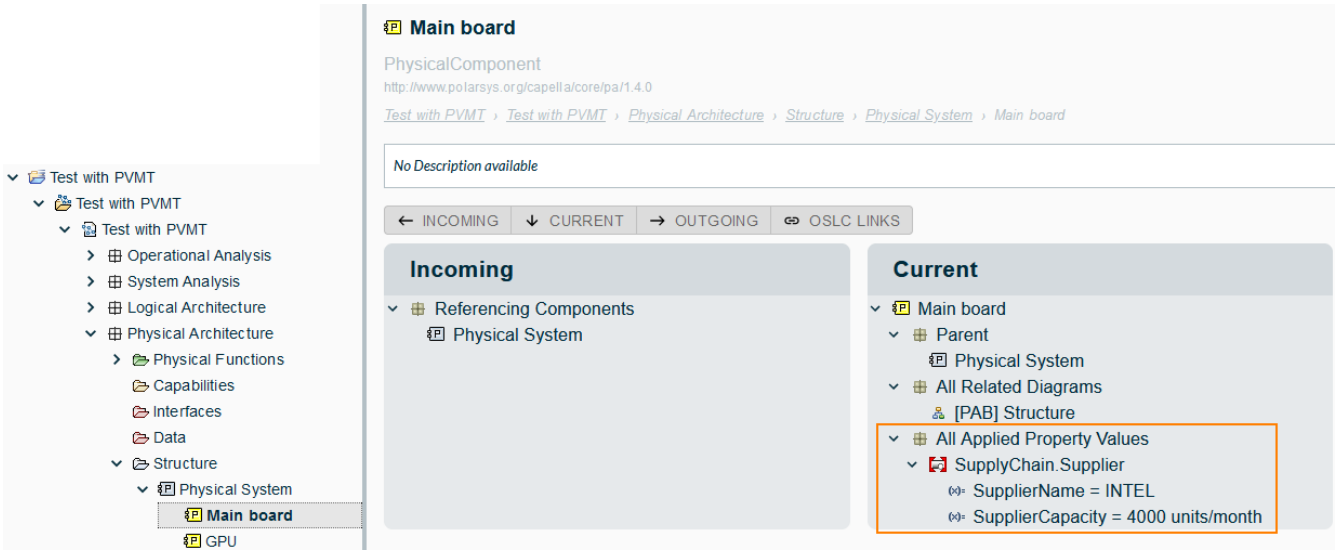


Figure 206. New Category in the Semantic Browser

Version 2024.03.0

Version 2024.03.0 brings support for versioning of the model data exposed on the Publication server.

Server Changes

- PERS-445 A new component that supports storage of model and model objects versions has been introduced.
- PERS-473 The default labels for the four IBM DOORS Next link types (trace, satisfy, specify, derives) have been fixed. It is recommended to reflect these changes in server environments in the `oslc-linking.json` file.
- PERS-476 A dependency to Spring's `UriComponentsBuilder` has been removed to fix two recent CVE:
  - CVE-2024-22243;
  - CVE-2024-22259.

Contributor Client Changes

- PERS-445 The publication wizard now requires the user to enter a mandatory message explaining the reason of the publication.
  - This message is used as a commit message that is displayed online in the model history view.
- PERS-450 The credentials to connect to the Publication server will always be requested if they are not properly set in the preferences.

Web Client Changes

- PERS-479 The web client now displays the history of publications and online link modifications of models. This history is displayed in a dedicated right-hand panel.

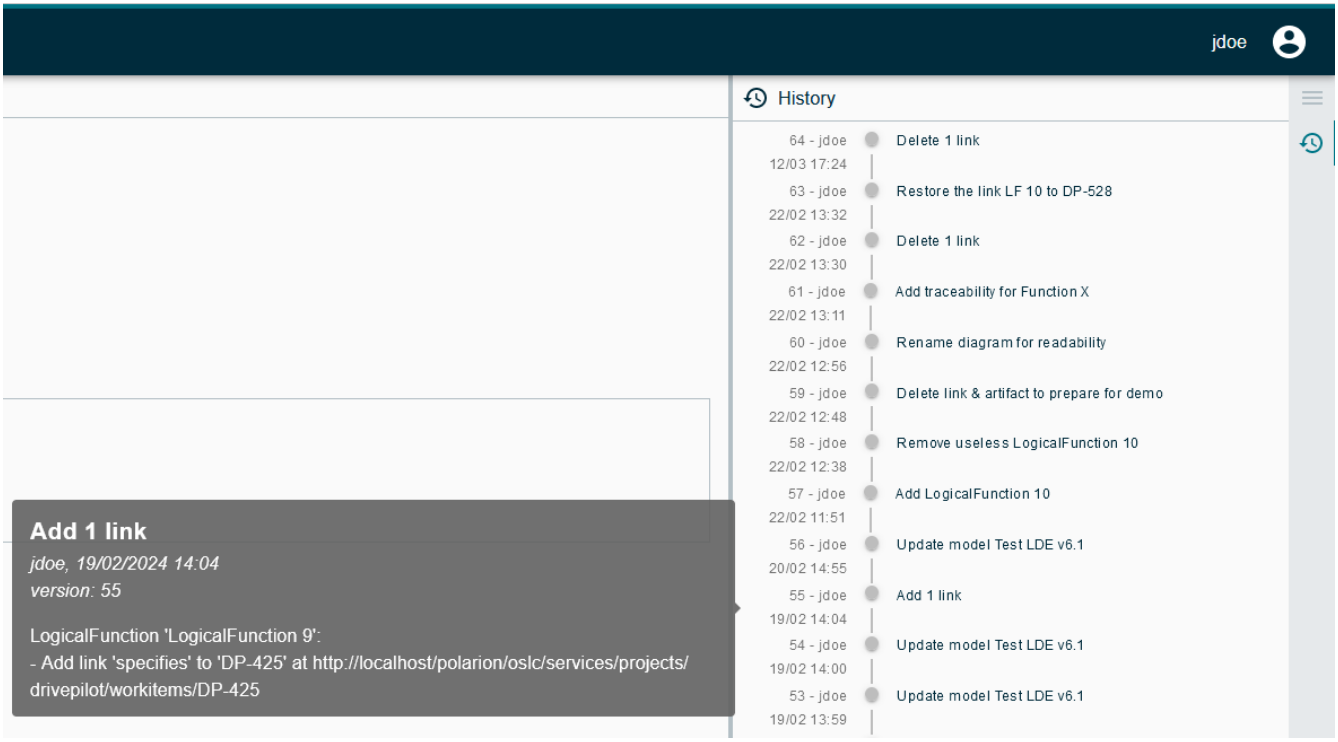


Figure 207. New History Panel in the Web Client

- PERS-222 The quick search no longer reloads the page when a search result is selected.
- PERS-227 The quick search now supports navigating in search results using the keyboard arrows.

Documentation Changes

- PERS-472 Documentation for the configuration of dropped URL mappings in the *Contributor Client* has been improved with more robust patterns. These patterns make it possible to use the drag and drop feature of Publication for Capella within System Modeling Workbench.
- PERS-477 The process to install Publication for Capella in a *Team for Capella* context has been documented in the installation guide, section Installation in a Team for Capella Context.
- PERS-497 A known issue with *Team for Capella* has been documented and will be fixed in the upcoming version 7.0. See Known Limitations in the Contributor Client for more details.
- PERS-486 The URL of Polarion's web page to validate an OSLC Friend request issued from Publication for Capella has been documented in section Create a new Friend.
- PERS-434 A known limitation with High-DPI displays has been documented in section Contributor Client Known Limitations.

Version 2023.12.0

Server Changes

- The property `perseus.oslc.linking.global` has been renamed to `perseus.oslc.linking.default`. Make sure to change this property's name in your own `application.properties` file accordingly.

Contributor Client Changes

- `PERS-425` Facilitate the reuse of a Traceability Configuration: It is now possible to export and import a traceability configuration in Capella, making it easier to configure a new Capella project for traceability with third-party repositories. Refer to Import / Export Traceability settings for more details.
- `PERS-438` The publication URL of a given Capella model is now stored in the `*.traceability` file instead of the `*.aird` file. In some cases, users will be prompted for a choice when they open their Capella model after upgrading to the latest version of the Publication for Capella *Contributor Client* add-on.
- `PERS-429` Support Global Configuration Management
  - There is now an initializer to create the OSLC Configuration Management domain. This initializer must be run if the model needs to use a *Global Configuration*.
  - Whenever a model is configured for a specific *Global Configuration*, then only artifacts compliant with the *Global Configuration* can be dragged and dropped from third-party repositories.

Web Client Changes

- `PERS-429` Support Global Configuration Management:
  - A project can now be set-up to use *Global Configuration* if connected to an IBM Jazz Team Server.
  - Within such a project, all models must be associated to exactly one *Global Configuration* to support traceability.
  - Refer to section Global Configuration Support in the user guide for more details.

Version 2023.10.0

Contributor Client Improvements

- `PERS-413` The traceability initializer now provides an initializer for the JTS domain 'LinkTypes'.

API Improvements

- `PERS-394` OSLC queries now support "where *=<perseus_url>".
- `PERS-395` OSLC queries now support "in [<perseus_url>]".
- `PERS-407` Model object data retrieval via OSLC API has been optimized.

Bug Fixes

- PERS-199 OSLC links created from another tool to an unnamed object in Publication for Capella are now navigable.
- PERS-410 It is now possible to create OSLC links from an unnamed object in Publication for Capella.
- PERS-398 Fix incorrectly encoded characters in OSLC query results.
- PERS-399 Limit access to OSLC Catalogs according to the user's permissions.
- PERS-357 - Contributor Client - Fix counting of model elements when publishing.
- PERS-409 - Contributor Client - Fix missing links issue in some Capella diagrams.
- PERS-423 - Contributor Client - Fix NullPointerException when trying to publish a model with broken representation descriptions.

Version 2023.8.0

Server Improvements

- PERS-377 - Proxies are now supported in the server. Refer to the installation guide for more details.

Web Client Improvements

- PERS-371 - The labels of OSLC Link types are now properly displayed in the OSLC Links panel, taking into account the OSLC Linking Configuration.

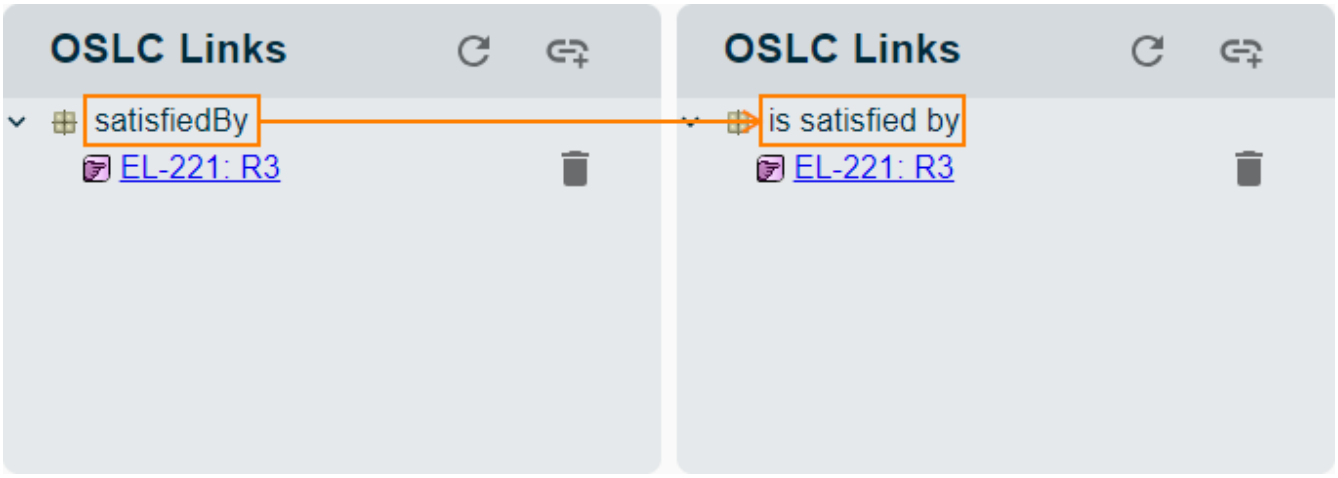


Figure 208. Improved OSLC Links Panel

- PERS-364 - Web Client - The 'refresh all artifacts' action is now accessible to all the project contributors.
  - Previously, this action was only available to the server administrator. This was wrong because most of the time the server administrator has no account on the registered OSLC Friend servers, especially when P4C is deployed in a cloud environment.

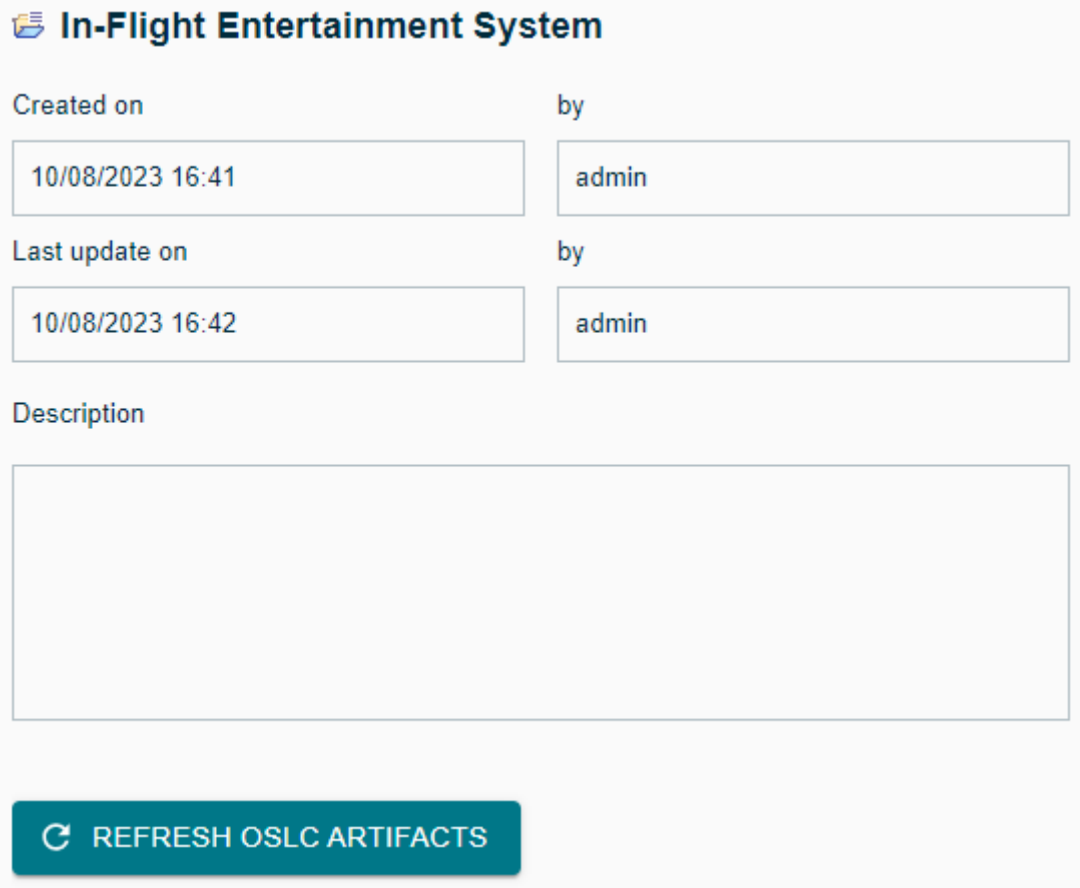


Figure 209. Refresh All Artifact available to Project contributors

- PERS-347 - Miscellaneous UI Improvements:
  - A link "back to consumer's list" has been added on all OSLC Consumer pages.
  - A link "back to project" has been added to the project settings page.
  - It is now possible to rename OSLC Friends.

Contributor Client Improvements

- PERS-377 - Proxies are now supported in the Contributor Client. Refer to the installation guide for more details.
- PERS-367 - The traceability layer is now automatically activated whenever an artifact is dropped in a diagram.
- PERS-359 - Links created from remote consumer servers are now synchronized with Capella, as long as there is a correct configuration making it possible for the Publication server to access this consumer as an OSLC Friend.

API Improvements

- PERS-383 - Miscellaneous OSLC API Improvements (Refer to OSLC API for more details):
  - The amount of information available in the RDF representation of model elements is now richer. It includes all the properties and references of the model elements.
  - The OSLC Query API now supports more types of queries.

Bug Fixes

- PERS-381 - Web Client - Links with no reverse link type can be properly created, which is absolutely necessary for compatibility with DOORS Next.
  - Previous version had a regression that caused such links to be wrongly marked as problematic.
- PERS-376 - Using the Capella `templatepatterns` meta-model is now supported.
  - Previously, trying to publish models that depend on this meta-model was prevented by a 'Constraint Violation' exception.
- PERS-373 - The order of viewpoints in 'Representations per viewpoint' in the *Web Client* is now identical to what it is in Capella, even when additional viewpoints are used.
- PERS-372 - Minor differences regarding the labels of Viewpoint Categories in the *Web Client* model tree have been fixed.
- PERS-368 - Web Client - The licensing enforcement mechanism will now properly ignore inactive accounts.
- PERS-367 - Contributor Client - Artifacts are now properly displayed in unsynchronized diagrams.
  - Previously, a bug was causing artifacts to be displayed without their type and text in unsynchronized diagrams.
- PERS-360 - Web Client - The web client will no longer fail if access to the browser's local storage is denied.
- PERS-378 - Contributor Client - Closing the Publication for Capella preferences dialogs no longer causes an exception to appear in the error log.

Version 2023.4.0

The focus of this version is on user experience improvements.

Java version

The server now requires Java 17.

Web Client Improvements

- The search treats '%' and '_' as any other characters, making it possible to search for the an exact chain containing one of those.
- OSLC link targets and their OSLC identifier are included in the search scope.
  - Consequently, it's now possible to locate model elements linked to an artifact 'ART-ID' by simply searching for `ART-ID`.

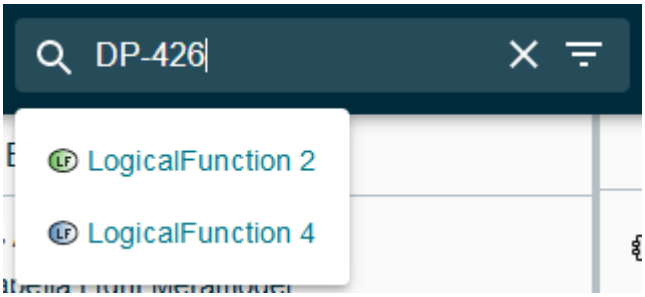


Figure 210. Quick Search showing model elements linked to an artifact

- Capella models and model elements are displayed with the same icons as within Capella.
- The labels of the Viewpoints and their ordering is now identical to what they are in Capella.
- Representations are grouped by type in the explorer tree, just like in the Capella explorer.
- The Model Tree nodes are now properly aligned, no matter whether they have children or not.
- Diagrams are scaled to the browser size. More specifically, diagram images are scaled down if they exceed the available width of the available area.
- An action makes it possible to display diagrams in full screen.
    - When in full screen, it is possible to grab the whole diagram with the middle mouse button to move it easily.

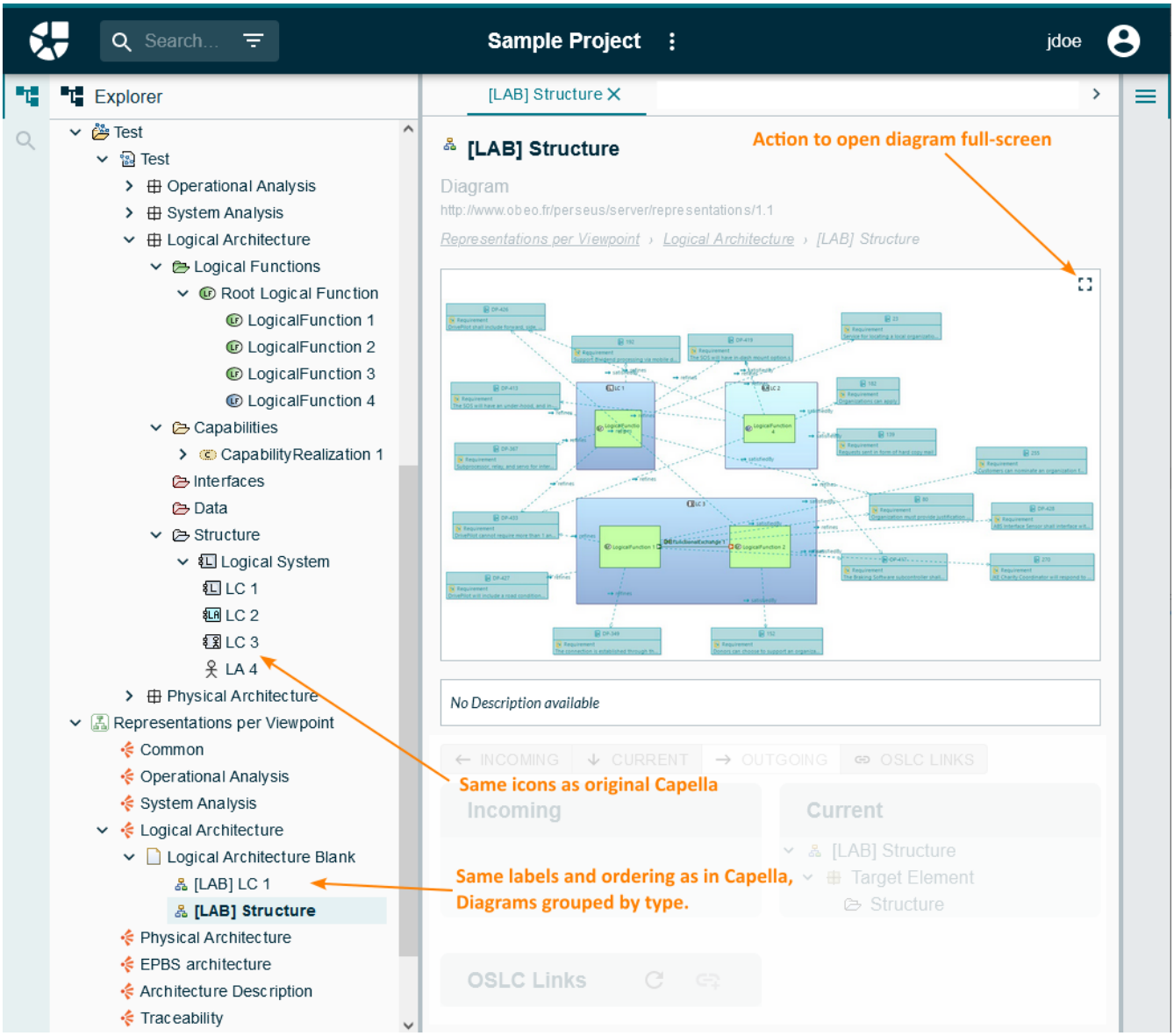


Figure 211. Miscellaneous Improvements in the web UI

- Representation Notes in diagrams can now be clicked to navigate to their target diagram.
- Clicking on objects in diagrams that have other representations attached makes it possible to access those representations.
- The jobs page now refreshes automatically.
- OSLC 'small' preview is now supported, which consequently makes it possible to have the OSLC preview being displayed by Polarion for links targeting Capella model elements in Publication for Capella.

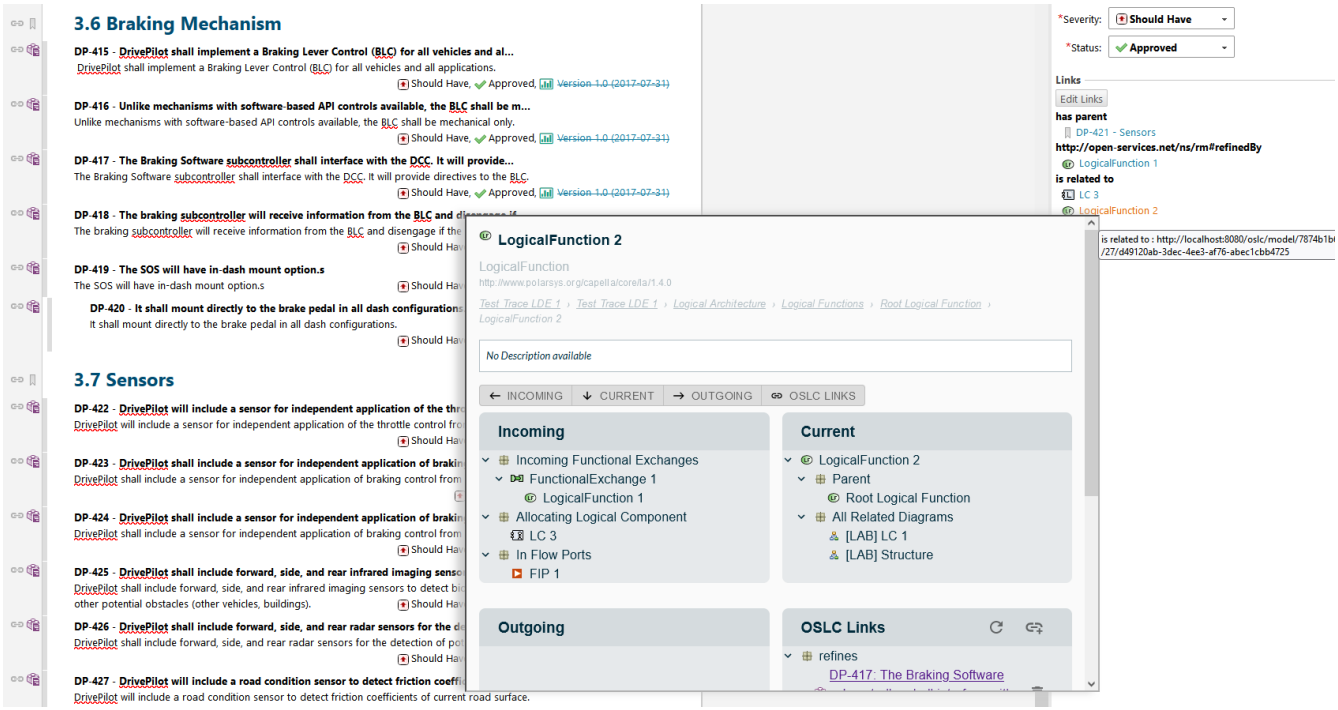


Figure 212. Small Preview working in Polarion

Contributor Client Improvements

- At the end of a publication the jobs page is no longer automatically opened. A pop-up with a hyperlink leading to the jobs page is now displayed instead.
- Updating from the server only considers friends that have been configured as Artifact Repositories within the Traceability model.
- The "Init traceability" action now only displays domains that are **not** already configured in the model.
- The "Init traceability" action now displays icons for each initializer.
- A new kind of initializer is available to facilitate the creation of Artifact Repositories in the model by retrieving available friends from the Publication server.

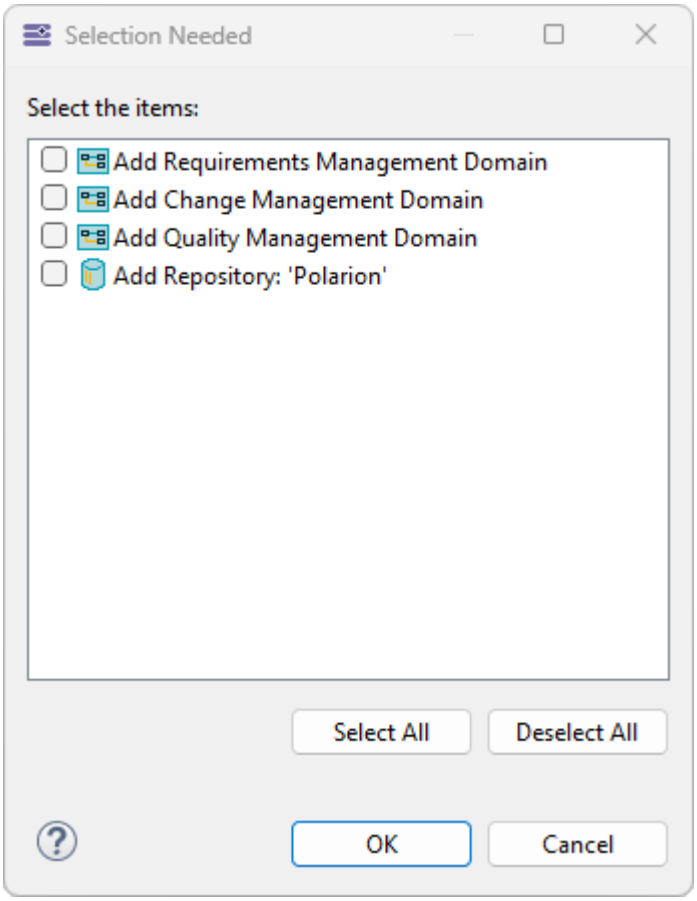


Figure 213. Artifact Repositories initialization

Bug Fixes

- A bug prevented very large models from being published (when the amount of bytes to publish exceeded the largest Integer value in java). This has been resolved, making it possible to publish very large models. However, the RAM and disk space must be sufficient to hold the published data in memory and in the temporary folder used by java, which is system dependent.

Version 2023.3.1

The version is mainly a bug-fix release, with a few improvements related to the IBM Jazz suite support.

Improvements

- Connectivity from Publication for Capella to Jazz EWM and Jazz Quality Manager is now supported.
  - In previous versions, only DOORS Next could be connected from Publication for Capella.
- Consequently, drag & drop from Jazz EWM is now supported.

Drag & drop from Jazz Quality Manager is still not supported.

Documentation Improvements

- Examples of configurations for Jazz EWM and Jazz Quality Manager have been added, both for the server OSLC Linking configuration and for the Contributor Client configuration.

Contributor Client Bug Fixes

- Diagrams are now properly refreshed when artifacts are dropped, even in *Team for Capella*.
- The Contributor Client no longer hides meta-model registration errors.
  - There was a bug in previous versions that could hide a meta-model registration error and let the publication proceed, resulting in an error message 'meta-model with hash xxxx is not registered'.
- Models with invalid views can now be published.
  - There was a bug in previous versions that prevented some models from being published. Specifically, the models with views with no viewpoint set could not be published.

Server Bug Fixes

- The regression that caused deleted links to still appear in DOORS Next is now fixed. This regression had been introduced in version 2023.3.0.

Version 2023.3.0

The main novelty of this version is the capability to manage traceability links directly within Capella, while keeping these links synchronized with the Publication server and the connected third-party repositories.

Managing Traceability in Capella

This new capability makes it possible for *System Engineers* to easily create traceability links between their model elements and third-party artifacts such as requirements, tests cases, or change requests.

The *System Engineer* can create such links simply by dragging and dropping URLs of artifacts from their website into Capella.

The drag and drop can take place in a diagram, or in the model tree. A link gets created between the dropped artifact and the model element on which it was dropped.

It is possible to configure different types of links to be created between different types of model elements and different types of artifacts. If several types of link are configured as being allowed between the dropped artifact and the model element, then the user must chose the one they want to create.

These links are available within the Capella model, but they also get to the Publication server each time the model is published. These links can also be created or deleted on the Publication for Capella website, and *System Engineers* can synchronize their Capella models to retrieve these link

changes in the Capella model.

Please refer to the User Guide for a complete description of this novelty and the related features.

Other Changes

- Models must be saved before publishing.
  - A pop-up will be displayed if the user tries to publish an unsaved model to make it possible to save the model and proceed, or cancel the publish.
- Models are automatically saved after publishing.
  - Publishing a model will modify some technical information in the model. For this reason, the model is saved automatically after publishing, whereas it appeared as dirty in previous releases.

Bug Fixes

- The display of the rich text documentation is now correct.
  - There was a bug in previous versions that caused the rich text to be incorrectly formatted after the 1st link in the documentation.
- The version number displayed in the home page footer is now correct;
  - In version 2022.11.0, the footer was incorrect and stated '2022.9.0'.