

# Enterprise Architect Connector for Polarion

User Manual



---

LieberLieber Software GmbH  
Philipp Kalenda  
Version 1.1, 03.08.2022

---

## Inhalt

Introduction.....	2
Installation.....	2
Troubleshooting the Installation .....	2
Supported EA Versions and Connection Types .....	2
Licensing .....	3
Trial License .....	3
Floating License .....	3
License Server.....	3
Apply a Floating License .....	4
Configuration.....	5
Polarion Server Configuration .....	5
Connector Mapping Configuration.....	5
Default Mapping Configuration.....	5
Type Mapping.....	6
Attribute Mapping.....	7
Mapping EA Connectors to Work Item Link Roles.....	9
Features.....	10
Workflow and Restrictions.....	10
Direction of Exchange.....	10
Recreation of Hierarchy.....	10
Import from Polarion.....	10
Export to Polarion.....	11
Export Diagrams .....	12
Export Trace Links.....	13

# Introduction

This manual describes how to use the EA Connector for Polarion, developed by LieberLieber Software GmbH. This document is based on the version 1.0.0-beta.27 and used Enterprise Architect (EA) 15.2.1559 as reference. The EA Connector for Polarion is currently in beta state and is still under development. We advise that a productive use of the connector is at your own risk.

If you experience any issues or if you have any feedback, don't hesitate to contact [support@lieberlieber.com](mailto:support@lieberlieber.com).

## Installation

The Polarion Connector is delivered as MSI Package and has to be installed via setup. The setup will install an Enterprise Architect addin, which will be COM registered. Proper permissions for the execution of a setup are required.

Before running the setup, make sure to close all Enterprise Architect instances. Simply click through the installation wizard and finish the setup.

## Troubleshooting the Installation

If the addin is not showing up in EA, this might be due to a failed COM registration of the addin. To confirm this, open the Specialize > Manage Addins menu in EA. If you can see the Polarion addin displayed with an "Error missing ...." code, you have to manually register the COM library.

To do this, start a command line window as admin and run the following command:

```
"C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe" "C:\Program Files  
(x86)\LieberLieber\Polarion EaConnector\EaAddin\LL.MDE.Apps.Polarion.EaConnector.Addin.dll"  
/codebase
```

If you are using EA version 16 you also have to run this command with the 64 bit version of the RegAsm.exe:

```
"C:\Windows\Microsoft.NET\Framework64\v4.0.30319\RegAsm.exe" "C:\Program Files  
(x86)\LieberLieber\Polarion EaConnector\EaAddin\LL.MDE.Apps.Polarion.EaConnector.Addin.dll"  
/codebase
```

After running these commands, the Polarion Connector addin should appear in the Specialize menu. If you are still facing issues with the installation, contact [support@lieberlieber.com](mailto:support@lieberlieber.com).

## Supported EA Versions and Connection Types

Enterprise Architect 13 and higher is supported, however Enterprise Architect 15 or 16 is recommended. Supported file formats are:

- EAP
- EAPX
- QEAX

According the Sparx Systems, QEA files should be used when a single user is accessing the model data. However, the connector is accessing the model from a different process than EA. EA is blocking the project file and no other process is able to access the project file due to this restriction. We therefore highly recommend the use of QEAX files.

If you have already used the QEA file and don't want to lose data, there is a simple step to convert from QEA to QEAX:

Simply rename the file extension from .gea to .qeax.

The following DMBS Connections are supported:

- MS SQL
- MySQL
- Oracle

Both encrypted connection strings and Cloud Server repositories **are not supported** by the connector.

# Licensing

Before using the connector, you have to provide a valid license. There are two license types: Trial and Floating. Both offer the full functionality, but a trial license is always time-wise restricted.

## Trial License

If you contacted [sales@lieberlieber.com](mailto:sales@lieberlieber.com) regarding the evaluation of the connector, you have received a trial license. The trial license will be sent as zip archive, containing a .lic file. Copy the file and place it in the folder %appdata%\LieberLieber\Polarion.EaConnector\Licenses. After restarting EA you should see an active trial license:

License Information

Active

Apply license

Is the license valid	✓
License type	User
End of maintenance	21-aug-2022
Host ID	ANY
Customer	Philipp LieberLieber
Issuer	LieberLieber Software GmbH
Issued date	21-jul-2022

## Floating License

### License Server

To use a floating license, you have to setup and host an [RLM](#) license server. Follow the guide at <https://help.lieberlieber.com/display/LIC/Installation+of+Floating+License+Server> to setup a the license server for LieberLieber products.

## Apply a Floating License

To apply a floating license open the License menu with Specialize > Polarion Connector > License and switch to the "Apply License" tab. In the next step, enter the port and the server address of the Floating License Server into the "Server Address" text box, in the following format: port@server

License Information



Active

Apply license

Host Id

XXXXXXXXXX

License source

Floating Server

5053@rlm

Test

Apply

If the connection test with the button "Test" was successful, click "Apply".  
If the license is valid, the license details are shown in the license overview:

License Information

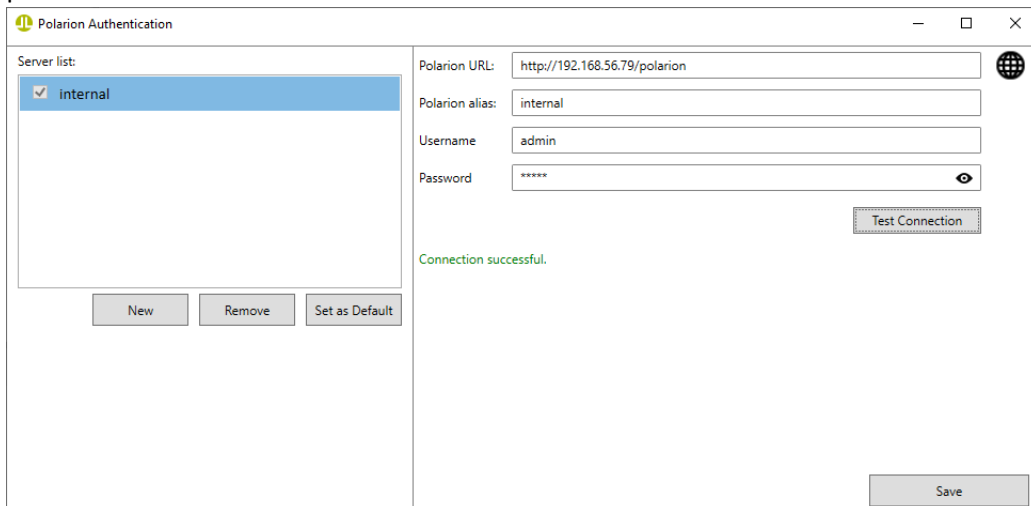


Active	Apply license
Is the license valid	✓
License type	Floating
Floating Server	rlm
End of maintenance	23-jun-2023
Customer	LieberLieber Internal
Issuer	LieberLieber Software GmbH
Issued date	23-jun-2022

# Configuration

## Polarion Server Configuration

In order to connect EA and Polarion a Polarion Server has to be configured. To do this, start the Server Configuration Dialog from Specialize > Polarion Connector > Configure Polarion Server Settings. Type in the Polarion Server (f.e. <http://PolarionBaseURL/polarion/> is enough), provide the user name and password and hit Test Connection.



If the connection was successful, save the configuration. If you want to create a new Server Configuration you can add an additional server by clicking “New”. Keep in mind that always one server will be the default configuration and you have to switch servers if you want to use another one for Import or Export operations.

## Connector Mapping Configuration

To map Polarion work items to EA elements, the connector uses two separate mapping files for import and export:

- C:\Program Files (x86)\LieberLieber\Polarion  
EaConnector\EaAddin\mapping\PolarionImportMapping.xml
- C:\Program Files (x86)\LieberLieber\Polarion  
EaConnector\EaAddin\mapping\PolarionExportMapping.xml

## Default Mapping Configuration

The default import mapping has the following element and connector mappings:

- Items
  - Polarion Requirements → EA Requirements
  - Polarion Test Cases → EA Test Cases (Use Case with stereotype “testcase”)
- Link Roles
  - Polarion Relates To → EA Association
  - Polarion Depends On → EA Dependency
  - Polarion Implements → EA Realization

The default export mapping has the following element, connector and attribute mappings:

- Items
  - EA Class → custom Polarion item type “class”
  - EA Component → custom Polarion item type “component”
  - EA Interface → custom Polarion item type “interface”
- Link Roles
  - EA Realization → Polarion Implements

Each mapping consists of a set of “mappingRules” where you define how an element from EA corresponds to an element from Polarion.

## Type Mapping

To map the type, a “matchingRule” is used, for example:

```
<matchingRule>
  <externalProperty name='_objectType' value='block' />
  <myumlProperty name='_objectType' value='Class' />
</matchingRule>
```

This matching rule maps the Polarion work item type “block” (externalProperty) to the EA element type “Class” (myumlProperty).

Class is a generic element type, which would also include SysML Blocks, since they derive from UML Classes. In order to define a more specific mapping the matching rule has to be extended by a stereotype:

```
<matchingRule>
  <externalProperty name='_objectType' value='block' />
  <myumlProperty name='_objectType' value='Class' />
  <myumlProperty name='_stereotype' value='SysML1.4::block' />
</matchingRule>
```

This specific mapping only considers SysML Blocks and ignores UML Classes.

Another example of a special type mapping is the EA element type Requirement. Since requirements do not exist in the base UML definition, a special stereotype needs to be used to map requirements:

```
<matchingRule>
  <externalProperty name='_objectType' value='feature' />
  <myumlProperty name='_objectType' value='Class' />
  <myumlProperty name='_stereotype' value='EA Specifics 1.0::requirement' />
</matchingRule>
```

Following EA types are supported:

- AcceptEventAction
- ActionInputPin
- Activity
- ActivityParameterNode
- Actor
- Artifact
- CallBehaviorAction
- CallOperationAction
- Component
- Class
- DataType
- Enumeration
- InstanceSpecification
- Interaction
- Interface
- OpaqueAction
- Operation
- Package
- Property
- Port
- SendSignalAction
- Signal
- State
- StateMachine
- Trigger
- UseCase

## Attribute Mapping

To map an attribute, a “property” is used:

```
<property external='title' myuml='Name'/>
```

In this example, the Polarion attribute “title” (external) is mapped to the EA attribute “Name” (myuml).

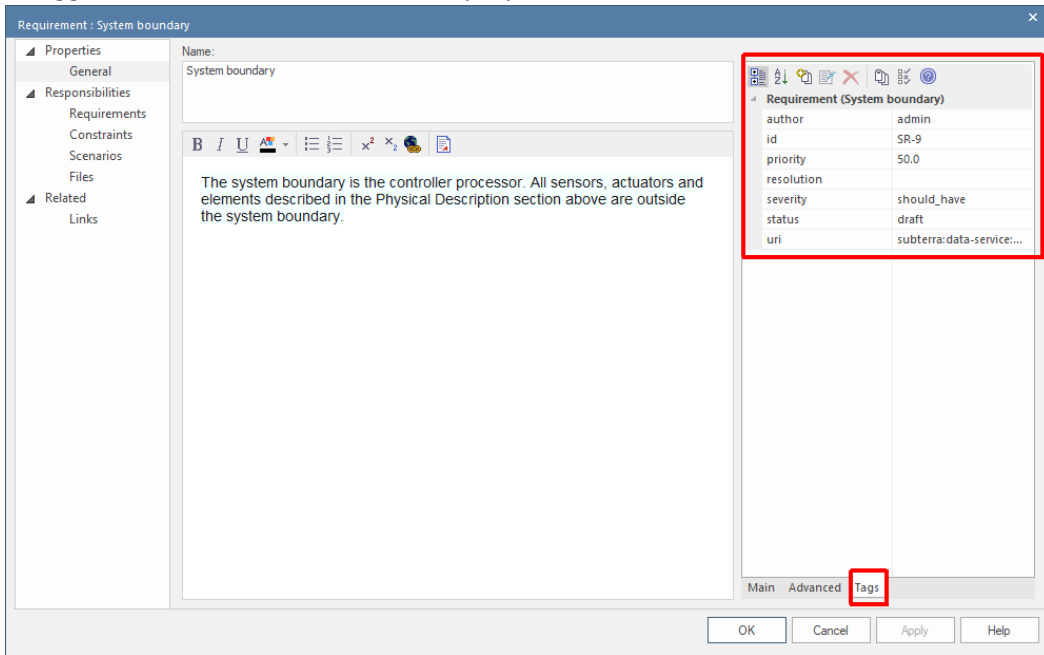
If you want to use custom EA attributes (=tagged values), you can use the same mechanism of the property mapping, but use a double “:” in front of the attribute name:

```
<property external="priority" myuml="::Priority"/>
```

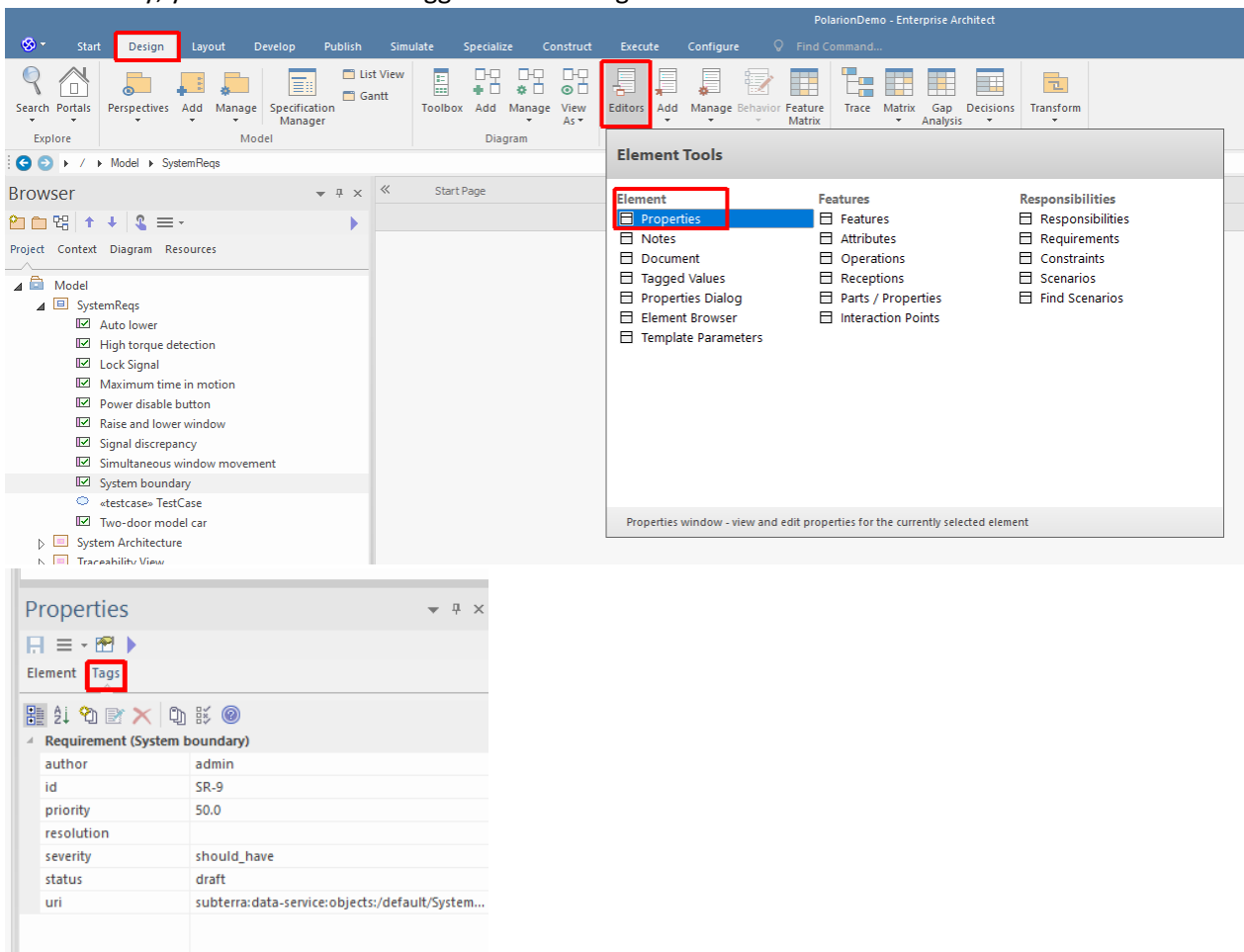
In this example, the Polarion attribute “priority” is mapped to an EA tagged value “Priority”. A tagged value will be created by the import if it does not exist. Tagged values don’t have to be created upfront.



A tagged value can be found in the EA properties on an element:



Alternatively, you can enable the tagged value dialog:



Following EA attributes are supported:

- Alias
- Author
- Classifier
- Complexity
- CreatedDate
- LowerBound
- ModifiedDate
- Multiplicity
- Name
- Notes
- Phase
- PrimitiveType
- Status
- Stereotype
- UpperBound
- Version

## Mapping EA Connectors to Work Item Link Roles

To map EA connectors to Polarion Work Item Link Roles, the matching rule is also used:

```
<matchingRule>
  <externalProperty name='_linkField' value='implements' />
  <myumlProperty name="_objectType" value="Realization" />
</matchingRule>
```

In this example the link role “implements” from Polarion (externalProperty) is mapped to the EA connector type “Realization” (myumlProperty). In order to use Polarion Link Roles, you have to use the ID of the defined link role in the mapping file.

Supported link types are:

- Association
- Dependency
- Generalization
- Realization

# Features

## Workflow and Restrictions

### Direction of Exchange

The base workflow that is enforced by the connector is defined as follows:

1. Import elements from Polarion to EA
2. Link elements imported from Polarion to existing EA elements
3. Export the EA elements along with their diagrams and links from EA to Polarion.

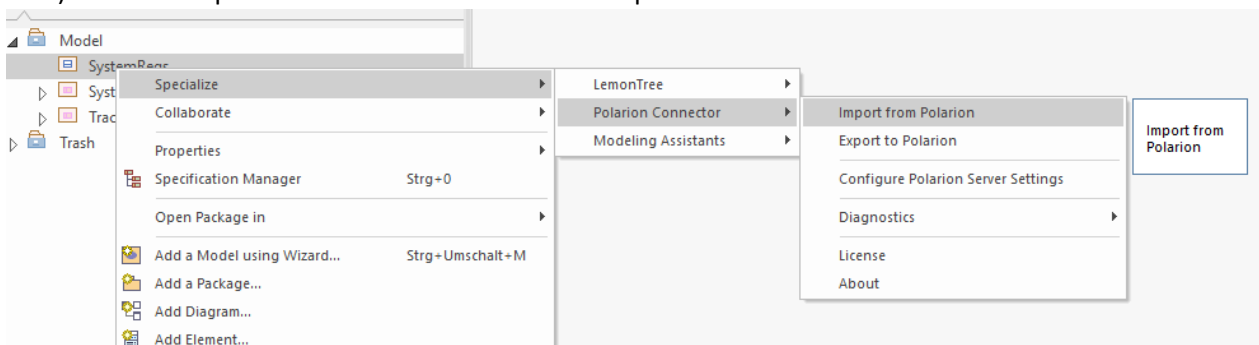
Although the connector supports writing data in both directions, **the same elements cannot be exchanged into both directions**. This restriction has been proven as the best practice when exchanging data between modelling and ALM tools and reduces to complexity of parallel editing and conflict solving.

### Recreation of Hierarchy

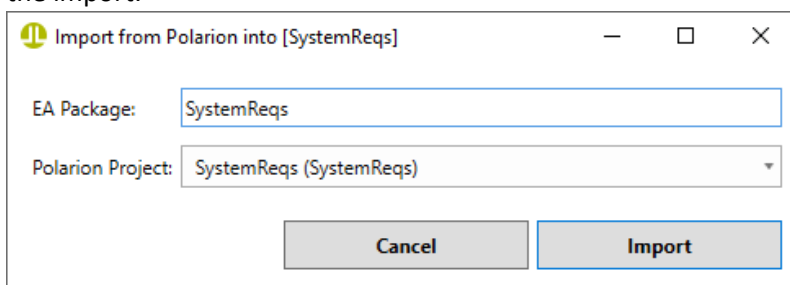
Enterprise Architect supports folders (=packages) and creating an arbitrary number of hierarchy levels for elements and packages. Since Polarion does not support hierarchical elements, the structure of an exported EA package will not be recreated. All EA elements and packages will be exported as flat list to Polarion. Also, embedded Elements will not be exported. Generally, only the first level of elements will be considered during export. The only exception for this are diagrams inside of elements.

### Import from Polarion

To import from a Polarion Project simply right-click an EA package in the Project Browser (not a model root) and select Specialize > Polarion Connector > Import from Polarion.



If you start an import for the first time on a package, a small configuration dialog will appear. In this dialog, select from which project and which tracker you want to import. If the settings are filled in, start the import.

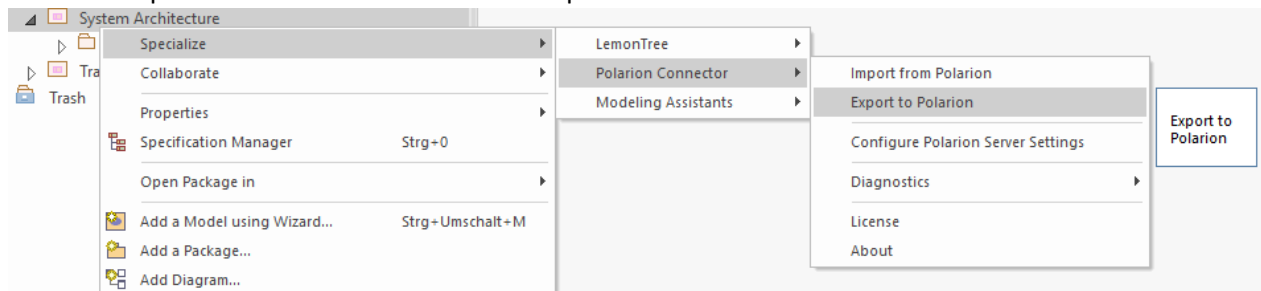


After the import was finished, a success message has to be confirmed in order to trigger a reload of the EA project. After the reload you will be able to see the changes done to the EA project file.

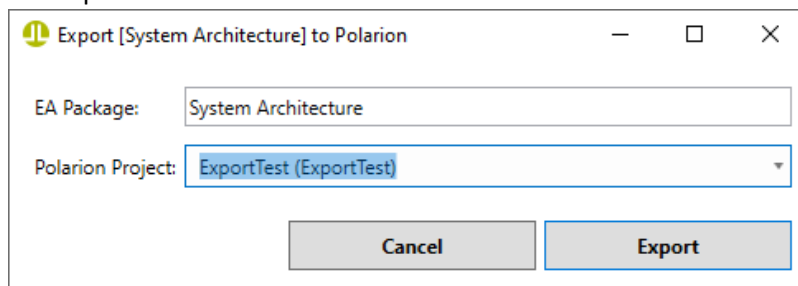
If you want to import an update simply start another import on the same package (or on one of its contained packages). This will automatically import from the same tracker without the need to configure the settings again. You will also see the “Export to Polarion” menu being disabled, which is the restriction of the workflow mentioned above.

## Export to Polarion

To export to a Polarion Project simply right-click an EA package in the Project Browser (not a model root) and select **Specialize > Polarion Connector > Export to Polarion**.



If you start an export for the first time on a package, a small configuration dialog will appear. In this dialog, select to which project and which tracker you want to export to. If the settings are filled in, start the export.



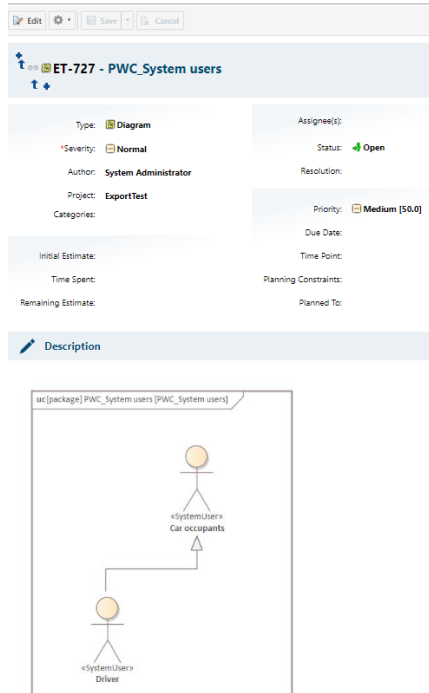
If you want to export an update simply start another export on the same package (or on one of its contained packages). This will automatically export from the package to the same tracker without the need to configure the settings again. You will also see the “Import from Polarion” menu being disabled, which is the restriction of the workflow mentioned above.

## Export Diagrams

Diagrams are exported automatically with each export from EA. There are two types of diagrams, which are exported to Polarion:

### 1. Diagram elements in packages

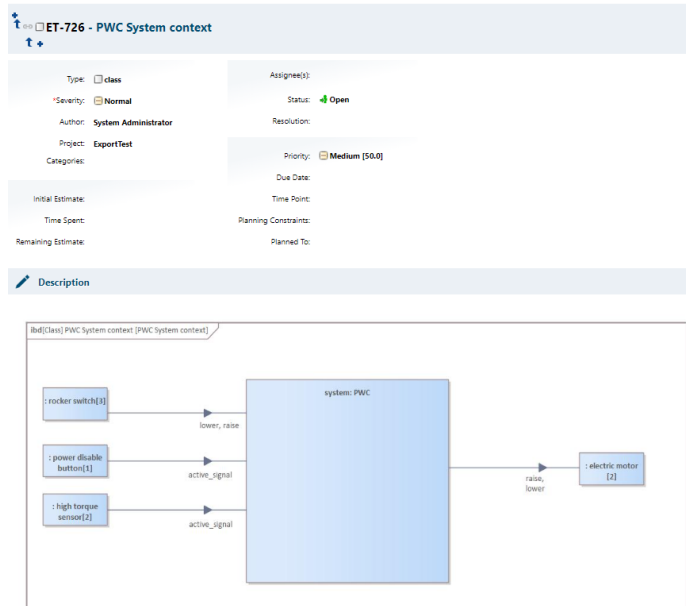
Diagrams in packages will be created as separate work items in Polarion. Those work items will always be created with the work item type “Diagram” and will contain an attachment of the diagram image, which is also embedded into the work item description.



The screenshot shows a Polarion work item for ET-727 - PWC\_System users. The work item is of type "Diagram", severity "Normal", author "System Administrator", project "ExportTest", and category "Categories". It has a status of "Open", priority "Medium [50.0]", and is assigned to "Assignee(s)". The description section contains a UML diagram titled "ac[package] PWC\_System users [PWC\_System users]". The diagram shows two actors: "Driver" and "Car occupants", both labeled as "«SystemUser»". A directed association arrow points from "Driver" to "Car occupants".

### 2. Diagrams contained in elements

Diagrams inside of elements are not created as separate work items in Polarion. This is because of the lack of hierarchy capabilities in Polarion. To maintain the relationship between an element and its child diagram, those diagrams will be created as attachment of the parent element only. The diagram image is also embedded into the work item description.

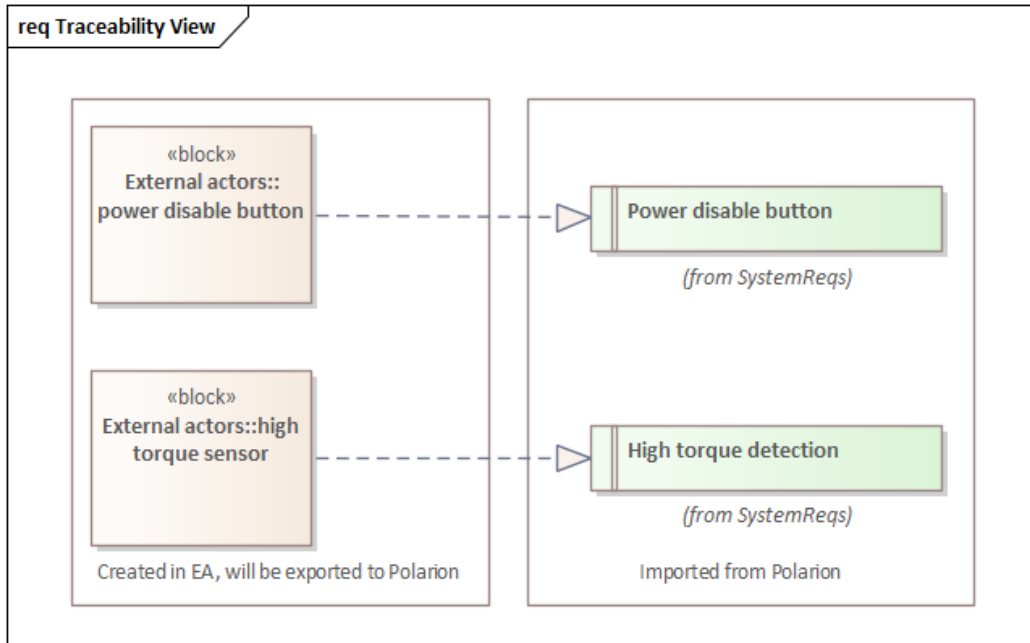


The screenshot shows a Polarion work item for ET-726 - PWC System context. The work item is of type "class", severity "Normal", author "System Administrator", project "ExportTest", and category "Categories". It has a status of "Open", priority "Medium [50.0]", and is assigned to "Assignee(s)". The description section contains a diagram titled "Bdd[Class] PWC System context [PWC System context]". The diagram shows a central box labeled "system: PWC". To its left are three boxes: "rocker switch[1]", "power disable button[1]", and "high torque sensor[2]". Arrows point from these boxes to the "system: PWC" box. The arrow from "rocker switch[1]" is labeled "lower, raise". The arrow from "power disable button[1]" is labeled "active\_signal". The arrow from "high torque sensor[2]" is labeled "active\_signal". To the right of the "system: PWC" box is a box labeled "electric motor [2]". An arrow points from "system: PWC" to "electric motor [2]", labeled "raise, lower".

## Export Trace Links

If existing and mapped, any type of link (=connector) is exported from EA to Polarion. When exporting elements to Polarion, only outgoing connectors are considered during the export. The connector will be created as Work Item Link Role in the Polarion project at the source element coming from EA.

The typical use case is to import a requirement from Polarion to EA, link an architectural element (like a SysML Block) to the imported requirement (Source = Block, Target = Requirement) and export the block to Polarion.



Exporting the blocks that were created in EA along with the Realization connector, will result in the following in Polarion:

The screenshot shows the Polarion user interface for a work item titled "power disable button" (ID: ET-709, Version: 50.0). The interface includes a top toolbar with "Edit", "Save", and "Cancel" buttons. Below the toolbar are four tabs: "Work Records", "Approvals", "Linked Revisions", and "Linked Work Items". The "Linked Work Items" tab is selected and highlighted with a red box. It displays a table with the following data:

Suspect	Role	Title
	implements	SR-2 - Power disable button