CATIA FLXML Exporter User Guide

CATIA Magic 2021x Refresh2

3DEXPERIENCE®

DASSAULT SYSTEMES | The 3DEXPERIENCE® Company

Contents

CATIA FLXML Exporter Plugin Presentations

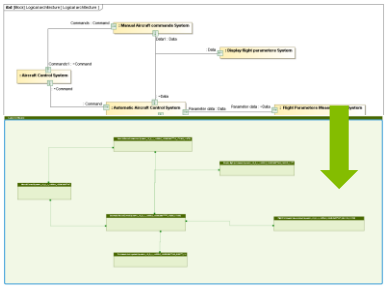
What will the customer use this for?

The CATIA FLXML Exporter Plugin will allow clients to generate FLXML from an UML/SysML/UAF model in order to allow the Digital continuity and to initiate the Design layer from System layer.

Purpose

Ensure digital continuity and master interfaces of a System architecture in CATIA Magic with sub-systems either developed in 3DEXPERIENCE Platform, CATIA Magic and other tools.

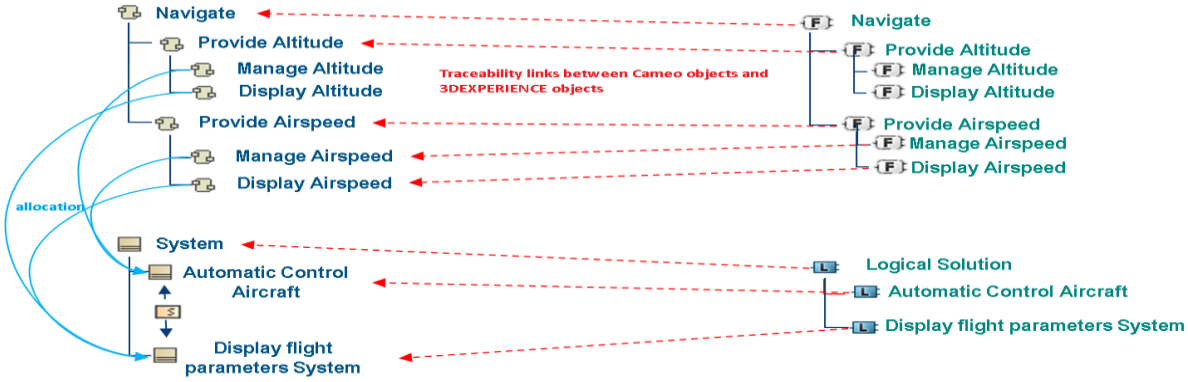
The aim of the CATIA FLXML Exporter Plugin is to add the export to FLXML generation in order to transform UML/SysML/UAF … models to Functional and Logical structures in the 3DEXPERIENCE.



Ensure digital continuity and master interfaces of a System architecture in CATIA Magic with sub-systems either developed in 3DEXPERIENCE Platform, CATIA Magic and other tools.

Initiate transition from conceptual to design layer. Mapping based conversion of CATIA Magic data format into 3DEXPERIENCE Platform data format with lifecycle management.

Example of scenario supported by default by the FLXML plugin: Creation of Logical Objects and related Functional objects, and creation of implement links between FL objects.



Other usages of the CATIA FLXML Exporter are not recommended.

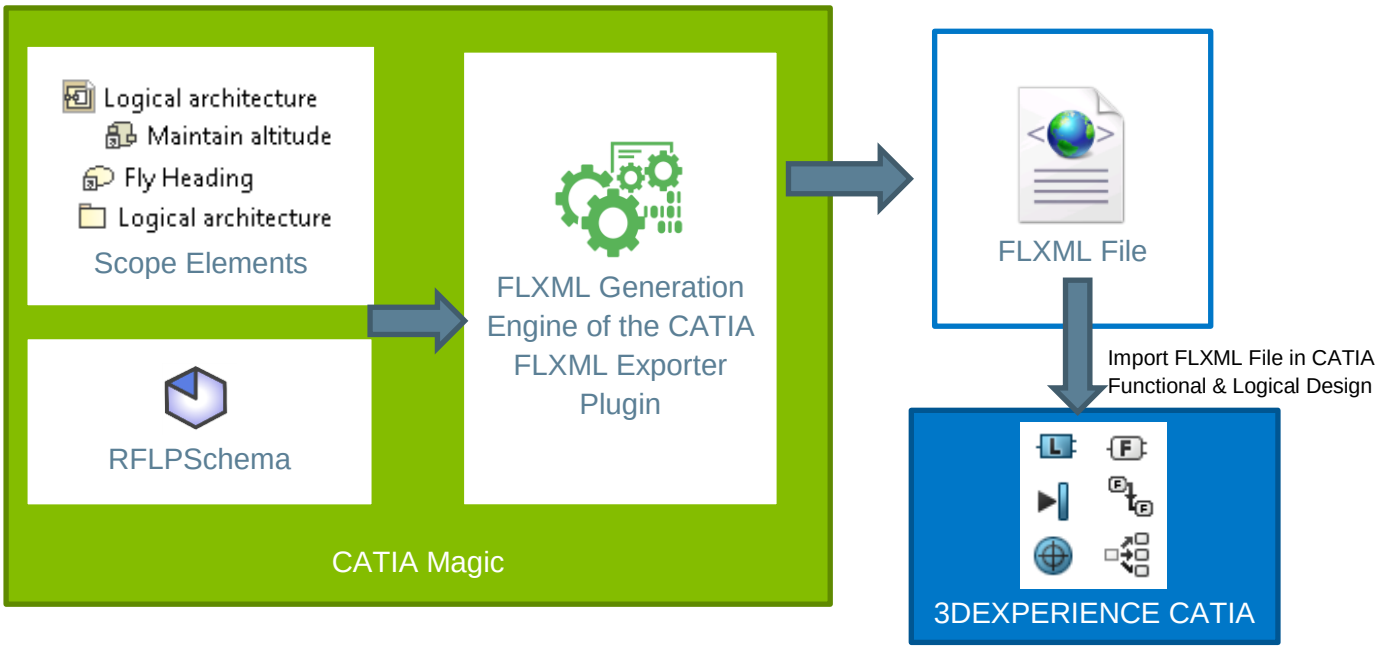
License

The CATIA FLXML Exporter Plugin is a free plugin. No check of license is done. The plugin installation directory follow the rules is described here: https://docs.nomagic.com/display/MD190SP4/Plugins+directories .

Principles

From a selection of scope elements and transformation schema, the FLXML Generation Engine of the CATIA FLXML Exporter Plugin will generate a FLXML File.

Then the user will manually import the FLXML in CATIA via the appropriate menu item in Functional Logical Design application of 3DEXPERIENCE.



A RFLPSchema is a set of UML Elements used to generate from UML/SYSML/UAF/… elements a FLXML stream that can be imported in the 3DEXPERIENCE.

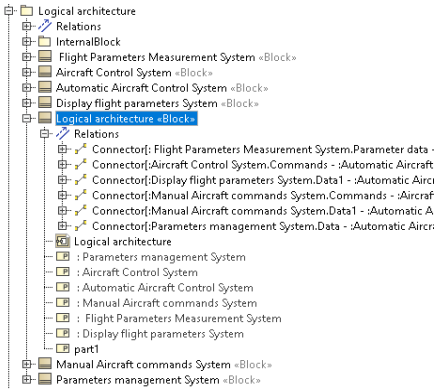
Packaging

The CATIA FLXML Exporter Plugin will be delivered only for CATIA Magic 2021 Refresh 2 version.
Only in CATIA Magic 2021 Refresh 2 version.
For previous support of CATIA Magic (19 or version 2021 Refresh 1 or before) see with R&D.
The CATIA FLXML Exporter is a free plugin.
The CATIA FLXML Exporter is not delivered with 3DEXPERIENCE Role/App.

Navigation

Navigation from a scope element (not a diagram)

A scope element is not a diagram
By default all elements under a scope element are taken into account in the FLXML generation, **if a transformation rule is compliant with the element** including relations.



All elements under the “Logical Architecture” Block will be taken into account.

With the OOTB SysML to FL schema, if the user adds an Use Case object called UC1 under the “Logical Architecture” Block, this Use Case UC1 will not have FLXML generation because there is no transformation rule for Use Case in the OOTB SysML to FL schema.
The Connector relations will have FLXML generation with the OOTB SysML to FL schema.

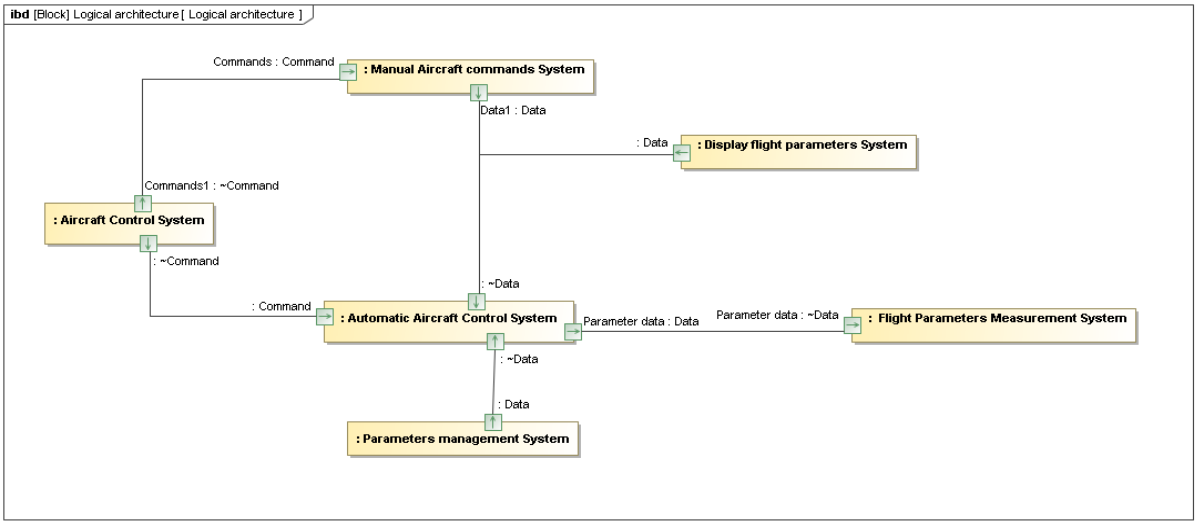
FLXML generation transforms elements of a scope element but also other elements of the model required in the transformation. For instance in the previous picture, to transform the PartProperty ":Aircraft Control System" to a Logical Instance, the property Type of this PartProperty is used to provide the Reference of this Logical Instance.In this case the Type is the Block "Aircraft Control System" which is not under the scope element "Logical Architecture".

The option "Export references outside of scope elements" has an impact on all the elements taken into account for the generation and on the depth of navigation of the model.

A lot of elements outside of the scope elements can be treated by the engine if they are referenced by the scope elements or elements under the scope elements.

- Navigation from a diagram

The user may decide to generate a FLXML from an IBD diagram used as scope element. The selection can be done from Diagram Panel, the diagram itself, Search Panel, Containment Tree ….



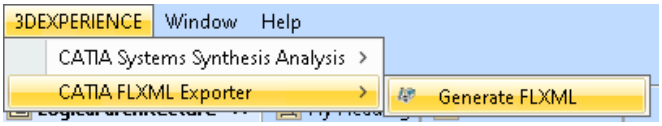
All the elements referred by the Presentation Elements in the IBD diagram will have an FLXML Generation.
In the example above, Block used as Type of PartProperty will have an FLXML Generation.

In this example, the PartProperty"part1" under the "Logical Architecture" block will not be taken into account because this part is not in the IBD Diagram used as scope element, except if the option "Export references out of scope elements" is checked.

CATIA FLXML Exporter Plugin UI

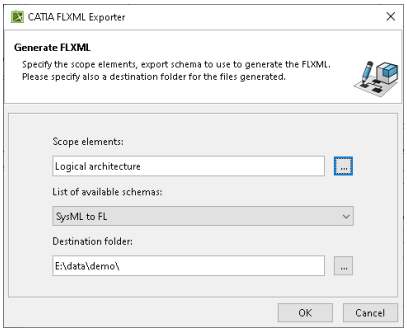
Accessibility

The CATIA FLXML Exporter is accessible via the “3DEXPERIENCE” menu as shown in the image below.



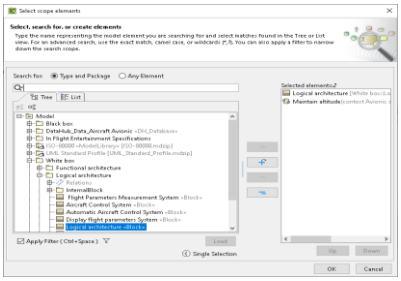
The plugin provides a contextual menu in the ContainementTree to validate the schema of generation.

Generate FLXML Dialog



Once clicked on the action “Generate FLXML“ menu, the dialog box below will be displayed. This dialog box will help the user to provide the requirement information to generate the FLXML.

- ### Scope Elements



The read-only ‘scope elements’ text box is used to enumerate the scope elements to use to generate the FLXML.

This list is initialized with any element selected in the Containment Tree first.
If a diagram is highlighted, the selection is initialized with the diagram.
Multiple elements can be selected as scope elements of the FLXML generation.
There is not control regarding the type of the selected scope element.
If there is only one element scope element and if this element is a diagram, the scope of the FLXML generation will be the elements in the diagram only. The layout of the FLXML element created will be the layout of the object in the diagram.

The browse button  close to the scope elements will give access to the Selected Elements of CATIA Magic.

- ### List of available schemas

This list contains the list of elements with the stereotype RFLPSchema in the current project and used projects and profile. See RFLP Profile § for more details on the RFLPSchema type.

A RFLP Schema is a set of rules that will allow the user to transform the elements of its models to FL objects by generating FLXML.
The RFLP Profile contains the OOTB schema SysML to FL.

- Destination Folder

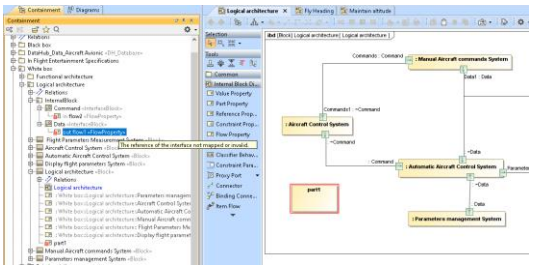
The destination folder will store the files generated by the plugin. Each FLXML generation of the plugin will create a specific destination folder in which all files created by the plugin during the generation will be copied.

Validation

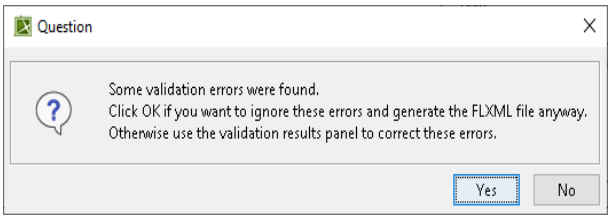
The first validation is about the schema used to generate FLXML.
An example of verifications is to check that mandatory attributes are not missing in the definition of the schema.
The second validation is done when parsing the hierarchy of the scope elements by checking mandatory information.
Validation is done on elements in the scope's hierarchy for which compliance rules of the schema are found, for instance :

- A Block is mapped by default in the OTB SysML to FL schema,
- A Slot with not mapping defined in the OTB SysML to FL schema will not be validated but not considered as error.

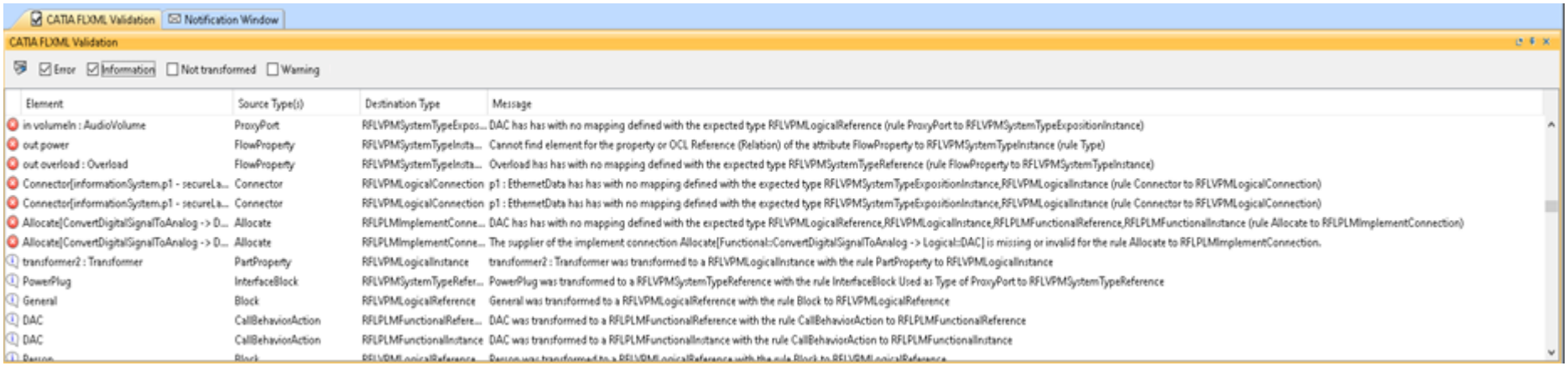
For example, a PartProperty can have a type with the System stereotype, which has not been defined in the mapping of the chosen schema, in this case there is a validation error.
At the end of The CATIA FLXML Validation panel (see next page) is displayed. Only errors occured during the validation process are displayed first. Moreover, the user can decide to view also the elements transformed, not transformed and warning messages. The error annotations will be visible in the project.



The CATIA FLXML Validation panel is different from the Validation Results Panel of CATIA Magic (https://docs.nomagic.com/display/CRMP2021x/Validation+Results+panel).
Even if there are validation errors in the generation of the FLXML, the user may decide to generate the FLXML. The question below will be asked to the user:

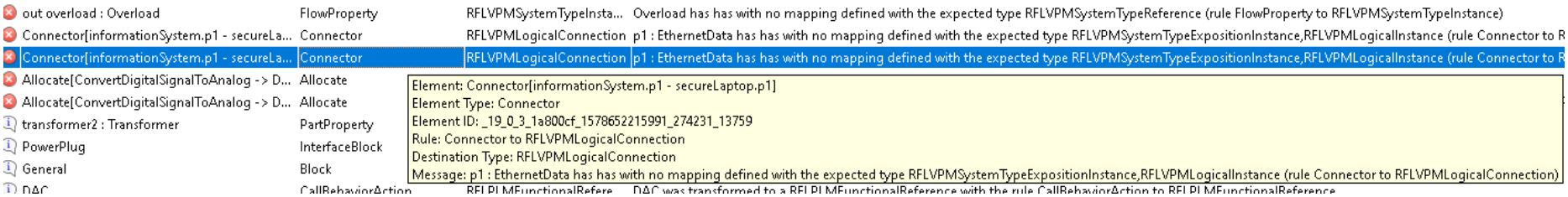


If the user clicks on yes the FLXML file will be created. In that case the user will be responsible of what he will then import in CATIA.



The plugin will create an annotation on each element where at least one error occurred.

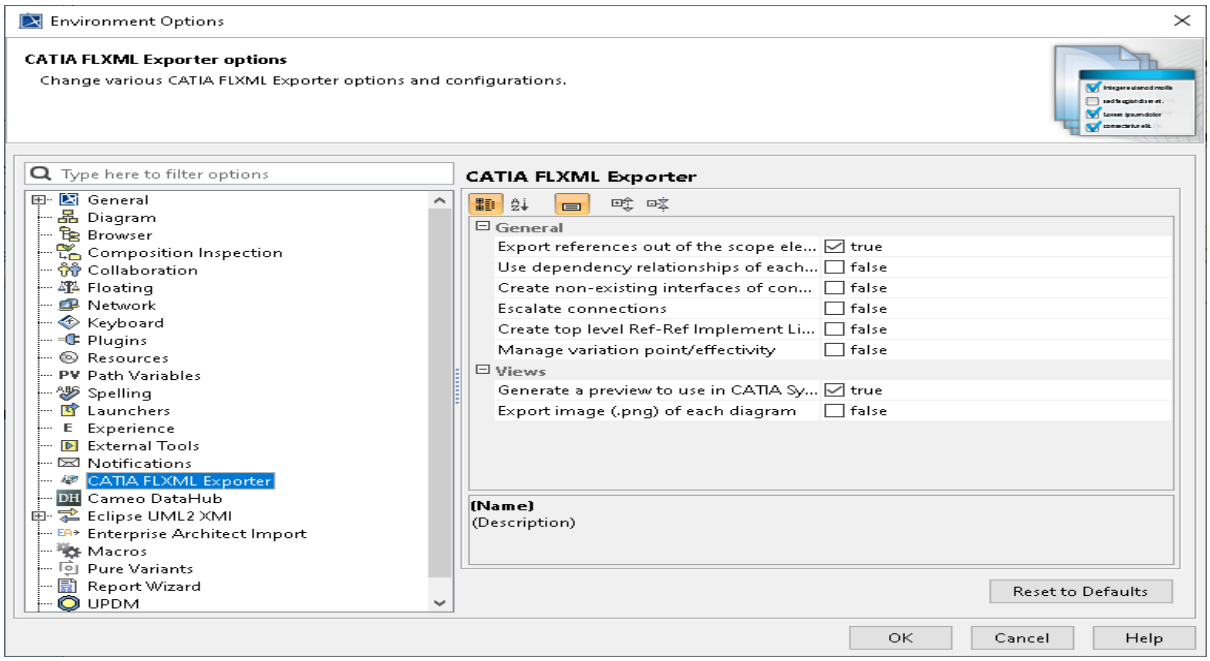
A clear button  will allow the user to clear the content of the CATIA FLXML Validation panel and the annotations on elements (a F5 can be required too to update CATIA Magic UI).
A tooltip on each line will provide information on the annotation.



The user can use check box Error/Information/Not transformed or Warning to filter the generation messages found during the generation of the FLXML.

The menu Select in Containment Tree will allow the user to select the validated element in the Containment Tree.

CATIA FLXML Exporter Plugin Options



Export references out of scope elements

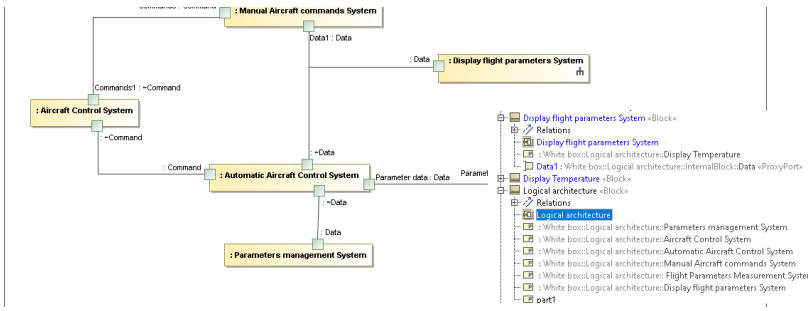
Default: true.

This option is to adapt the depth of navigation in the model.

Example: Logical architecture IBD used as scope of element for the generation.

If this option is checked, the Block “Display flight parameters Systems”’ and the elements under this block will be taken into account.

If this option is not checked, this Block will be taken into but not all the elements under this block, the PartProperty : “Display Temperature” for instance will not be taken into account.



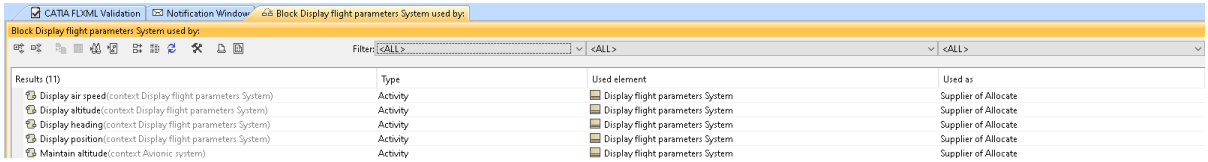
Use dependency relationships of each element

Default: false

This option behaves like the action associated to the contextual menu item “Related Elements/Used by” that displays related dependency usage of an element in CATIA Magic.

In the case of the engine of the plugin, this option is only used to take into account also dependency relationships of each element.

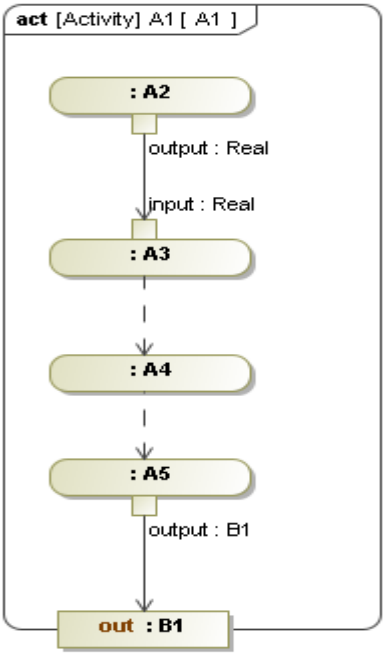
The option can be useful to retrieve automatically allocation links of an element for instance, because the allocation can be easily out of the scope elements. It is possible to add manually all this relationships as scope elements.



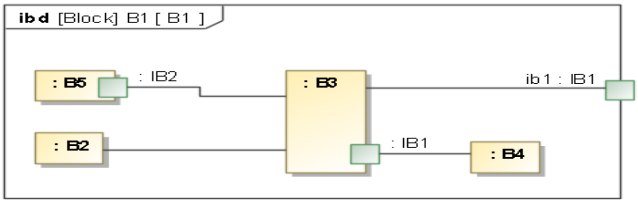
This option is false by default because retrieving all these dependencies for a lot of elements can be time consuming.

Create non-existing interfaces for connection

Default: false

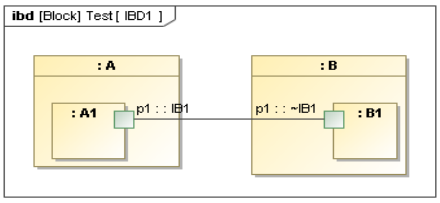


Interfaces must exist on both end of a connection in the FL Modeler.
These Activity diagram and IBD diagram are not compatible as is with the FL Modeler. This option checked to true will indicate to the engine to generate automatically missing interfaces for missing ports/pins.
Default references will be used to create this interface.
Instead of using this option, it is recommended that the user creates missing interfaces/ports/pins manually. (to keep coherent traceability between both models ?)

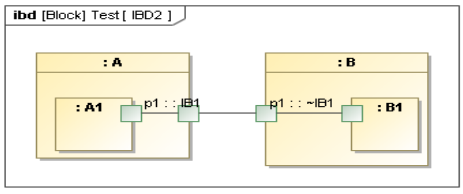


Escalate connections

Default: false



The FL Modeler does not support relations between interfaces of differenct level without having been escalated to the parent level.



With this option checked to true, the engine of the plugin will automatically escalate ports and connections as shown in the adapted IBD here.

Instead of using this option, it is recommended that the user does it manually.

Create top level ref-ref implement links

Default: false

In CATIA Magic it is possible to create an allocate relationship between a PartProperty and CallBehaviorAction. This kind of allocation relationship correspond to an OCC-OCC relation in the FL Modeler.

This OCC-OCC relation requires a parent REF-REF implement link. If no mapping to create this REF-REF implement link exists (between Activity and Block for instance), the engine **will try** to create a top level REF-REF implement link.

Manage variation point/effectivity

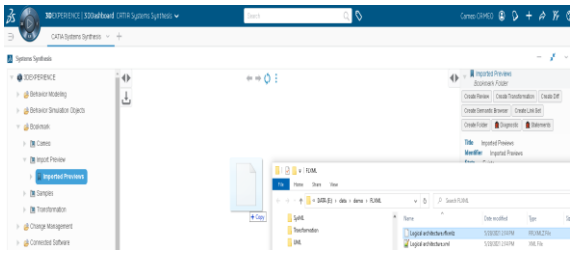
Default: false
This option set by default will allow the user to support to management of variation point/effectivity in its model.
The CATIA FLXML Exporter plugin will generate in a sub-folder called EffectivityXML in this destination folder and will place XML effectivity files in it.

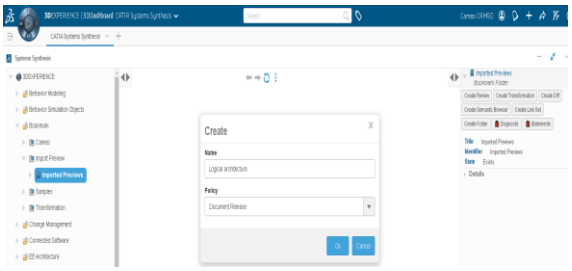
Generate a preview to use in CATIA Systems Traceability

Default: true

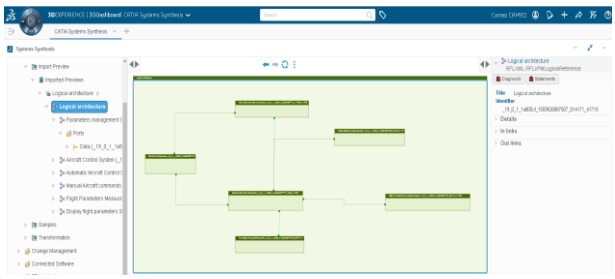
This option is used to indicate to the plugin to create a preview (a *.rflxmlz file file) in the destination folder, that can be used to preview the FLXML to import then in CATIA.

Drag & Drop the *.rflxmlz file in a folder of a Bookmark in the CATIA Systems Traceability Application of the 3DEXPERIENCE





Then you can preview and navigate in the FL model generated in the FLXML. It is only a preview displayed by the RFLXML connector in CATIA Systems Traceability. No FL objects are created at this stage, it is only a preview.

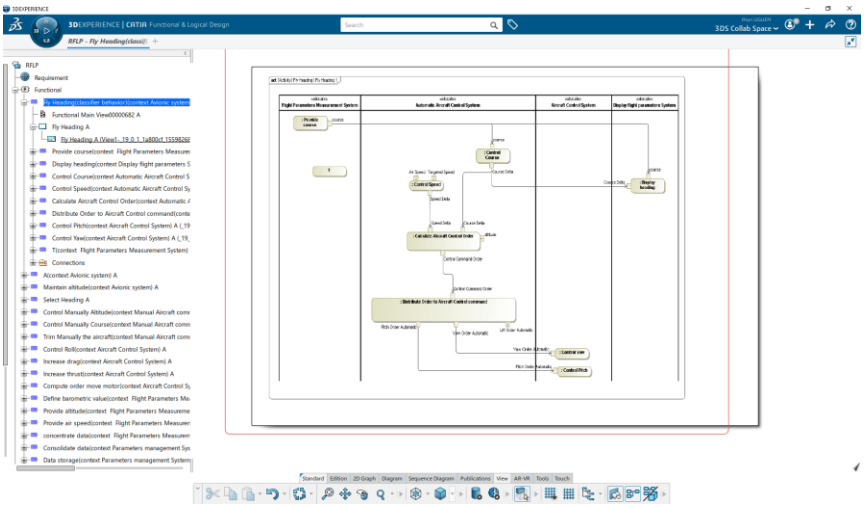


Export image (*.png) of each diagram

Default: false

Require CATIA Variables: RFLP_SCH_IMPORT=1 and RFLP_IMPORT_EXPORT=1
Not supported in the FL Connector and in the RFLXML connector in CATIA System Traceability (preview generated by the plugin).

This option will allow to view image of each diagram of CATIA Magic inside CATIA.
Tabular/Matrix and Map diagram in CATIA Magic are not supported by this option.
Only symbol diagrams.
With this option the user will have the possibility to view each CATIA Magic diagrams see new page for an example.

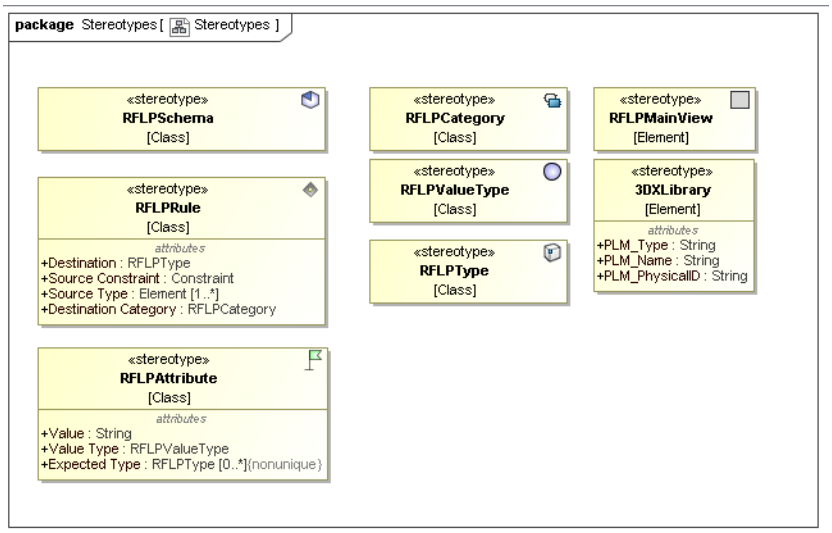


Activity Diagram as png image in a view of the Functional Reference

CATIA FLXML Exporter Plugin RFLP Profile & Schema

The RFLP Profile contains all the structures to help the user to define its mapping and also the OOTB schema SysML to FL.

The diagram below shows the list of stereotypes available to define its schema of transformation (or reuse the OOTB SysML to FL) schema.



RFLPSchema

A RFLPSchema contains 1..n to RFLPRule, inner elements of the RFLPSchema.
A schema is identified by its name.

RFLPRule

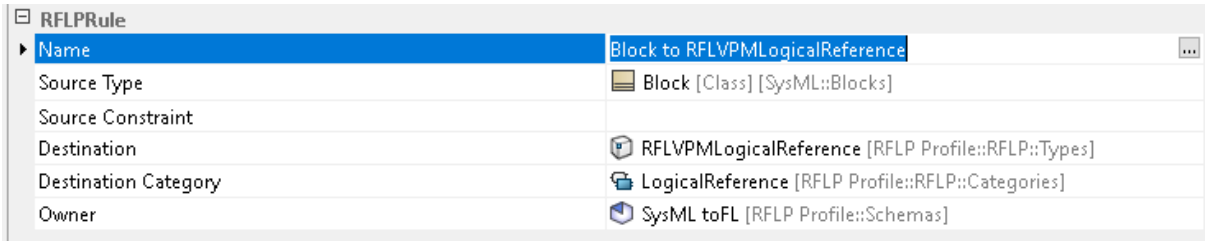
A RFLPRule contains 0..n to RFLPAttribute, inner elements of the RFLPRule.

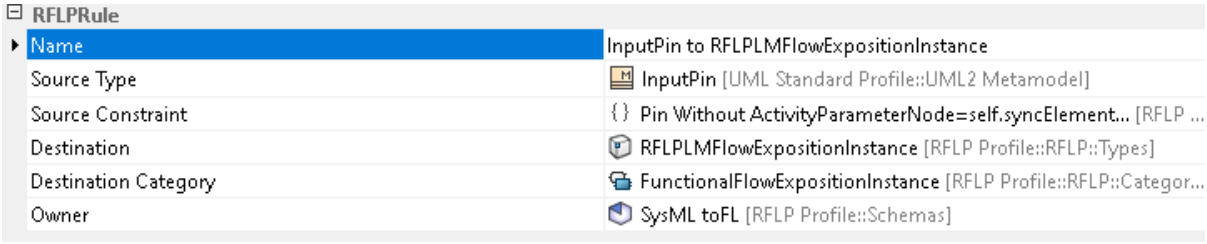
A RFLPRule defines the source type of the transformation, defined by the selection of 1..n type of element or stereotype(s).
A RFLPRule defines the source constraint of the transformation. Sometimes the selection of a type/stereotype is not sufficientto indicate what is the source of the transformation. In this case the user can define constraint to apply to filter more the source of the transformation. Most of the time the constraint will be an Object Constraint Language for UML expression.

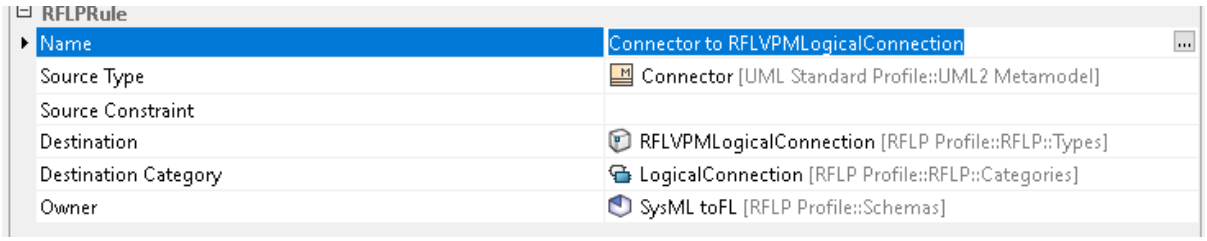
A RFLPRule defines the Destination of the transformation, this destination must be a RFLPType element. This RFLP Type element can be either the default OOTB RFLPType provides in the RFLP Profile or any custom RFLPType defined by a client in its projets.

A RFLPRule defines the Destination Category of the transformation. The Destination Category is the category where the FLXML generated by the plugin must be placed. The FLXML stream is defined by a XSD with a set of categories. Not all the categories of the FLXML are supported in the first version of the plugin.

Below some examples of RFLPRules extracted from the OOTB RFLPSchema SysML to FL:



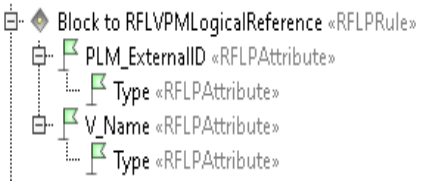




RFLPAttribute

A RFLPAttribute contains 0..n to RFLPAttribute, inner elements of the RFLPAttribute.
A small example of a FLXML stream showing the FLXML for a RFLVPMLogicalReference is described hereafter :
In this example the attributes are PLM_ExternalID, BoudingBox, V_Name.
the following attributes are mandatory : PLM_ExternalID and BoudingBox.
V_Name is not a mandatory attribute.
~~The RFLPAttribute mandatory or not depends of the category/type of the RFL object to create.~~
Position attributes (like BoundingBox) are automatically generated by the plugin and the user will not have to specify these attributes.

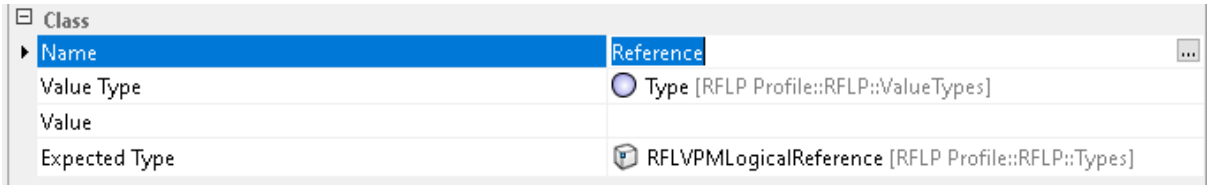
```
<ID Type="RFLVPMLogicalReference" Value="1">
  DestinationType
   <Mandatory>                    Attributes
     <PLM_ExternalID Type="String">_19_0_3_31c012d_1586227029551_415403_47000</PLM_ExternalID>
     <BoundingBox XMax="3248" XMin="5" YMax="959" YMin="5"/>
   </Mandatory>
   <V_Name Type="String">Injection System</V_Name>
</ID>     Attributes
```



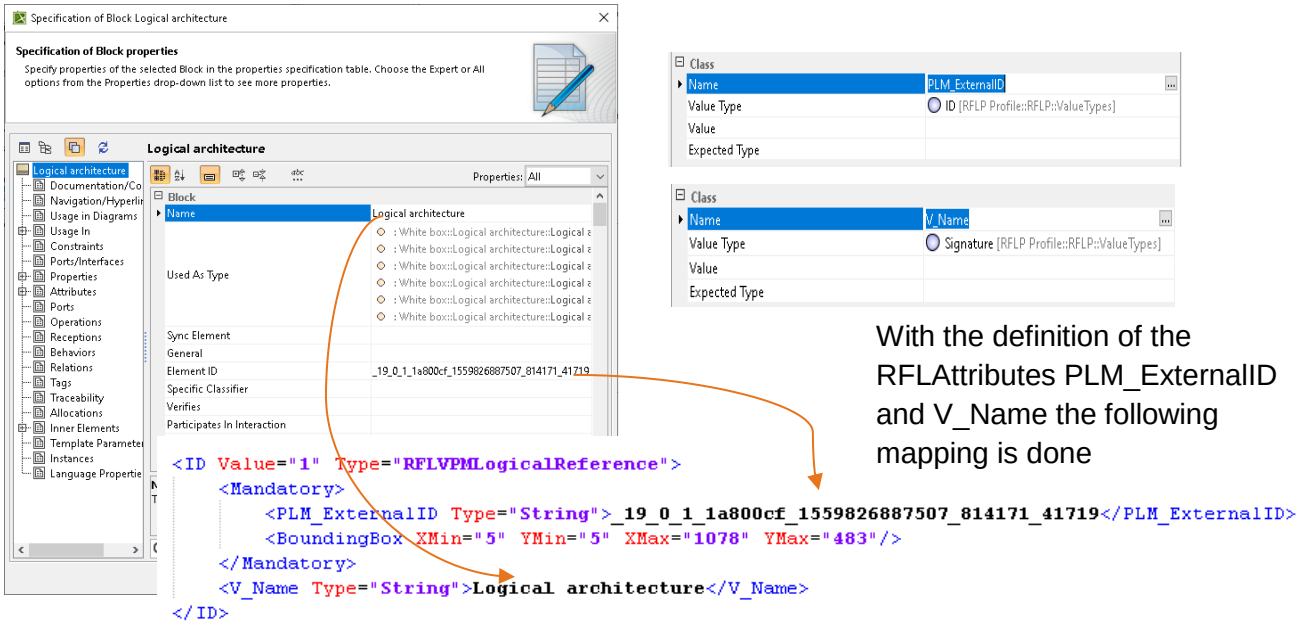
This RFLPRule generated the previous FLXML. You can see that only the PLM_ExternalID or V_Name attributes are generated.
The V_Name attribute contains an attribute Type so RFLPAttribute Type have been defined for the RFLPAttribute V_Name.
A RFLPAttribute defines only one RFLPValueType. A RFLPValueType is an enumeration of predefined mapping with element property and other possible types for the Value.

A RFLPAttribute can define a Value, this value can be of different type regarding the RFLPValueType previously selected.

A RFLPAttribute can define 0.n Expected Types. Expected Type are used when a same element under the scope elements can have mutilple RFLPRule, this property is used to specify which destination type is expected for this property.
For instance a Block can be transformed in multiple FL types : Logical Reference, System Type Reference ... below for the “Reference” RFLPAttribute of the RFLPAttribute “Relation” of the RFLPRule “PartProperty to RFLVPMLogicalInstance”, it is explicitely specified that it is a Logical Reference that it is expected, so probably will concern the RFLPRule “Block to RFLVPMLogicalReference”

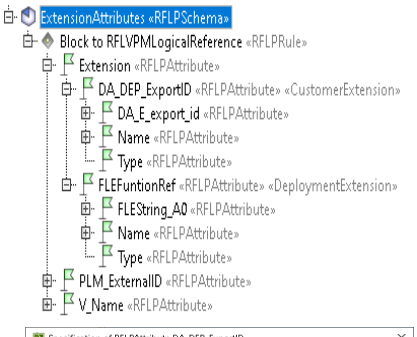


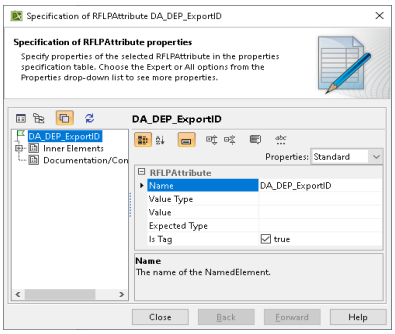
With the definition of a RFLAttribute the user defines a mapping between property of an element/tagged value of an element/… and attributes in the FLXML.



With the definition of the RFLAttributes PLM_ExternalID and V_Name the following mapping is done

It is possible to assign tagged value to RFLPAttribute via an OCL Expression oclAsType(<stereotype>).<tag name>, example oclAsType(SysML::Requirement).Text





See the OOTB RFLP Schema “SysML to FL” for other examples of definition of RFLPAttributes.
Customer Extensions and Deployment Extensions are supported by the plugin. Below an example of usage. The stereotypes CustomerExtension and DeplomentExtension can help the user to specify multiple CustomerExtension or DeplomentExtension under the Extension attribute.

The Is Tag on RFLPAttribute is used to indicate that the sub attribute is a child FLXML tag element (an not an attribute of the current FLXML tag)

RFLPValueType

Each RFLPAttribute must have a ValueType specified. The diagram below shows the list of possible ValueType.

ValueType	Description
Literal	The value provided will be a literal
OCLExpression	The value or the element will be provided by an OCL Expression, the result can be an Element or a literal
Name	Predefined property that corresponds to the name of the Element
Signature	Predefined property that corresponds to the signature of the Element for instance for a PartProperty the name can be empty but the signature is “: Block1” where Block 1 is the name of the type of property
Owner	Predefined property that corresponds to the owner of the element.
Type	Predefined property that corresponds to the type of the element. Useful only for element which have the type property of course
ID	Predefined property that corresponds to the unique ID of the Element. Server ID for TWC objects. LocalID for mdzip.

Stereotype RFLPMainView

When a Block has multiple IBD diagrams, the user can set the Stereotype MainView to the IBD Diagram that he wants to be used as RFLPMainView in the FLXML. Consequently all positions of PresentationElements will refer to the IBD diagram set as MainView.

The RFLPMainView can be used for any types of diagrams in CATIA Magic.

If the RFLPMainView stereotype has not been set, **by default the plugin will use the first diagram found**. If two MainView diagrams have been set, the first MainView is taken into account.

If multiple diagrams of an element has the RFLPMainView stereotype only the first diagram with the stereotype is taken into account.

The user can create an RFLPAttribute called Layout with an OCL expression to indicate which diagram to use as layout.

Stereotype 3DXLibrary

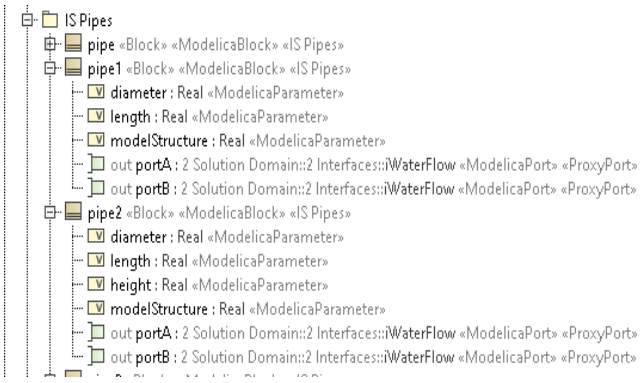
This stereotype will be used to indicate to the plugin if a Block, an Activity,a Logical Reference or Functional Reference refers to an existing library or component in the 3DEXPERIENCE.

If the user wants to specify a 3DX library or component, he should indicate the PLM_ExternalID and the PLM_Type of this library/component in the 3DEXPERIENCE. But he can also use the PLM_PhysicalID to select a specific revision of a library.

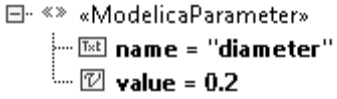
In the FLXML generated only the Type and PLM_ExternalID will be filled, in any case the library/component in the 3DEXPERIENCE cannot be updated by a FLXML generated by the CATIA FLXML exporter plugin.

Dynamic attribute creation

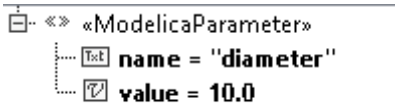
The following exampleexplains the usage of dynamic attribute creation based on two Blocks called pipe1 & pipe2. These two pipes own ValueProperty with Stereotype ModelicaParameter. Each diameter ValueProperty has a corresponding tagged value with the diameter and a specific value for each pipe (0.2 for the pipe1 and 10.0 for pipe2)



pipe1 ‘s diameter ValueProperty and associate tagged value 0.2

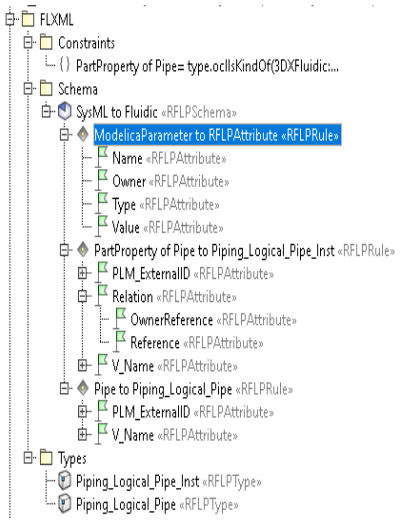


pipe2 ‘s diameter ValueProperty and associate tagged value 10.0



As a user of the plugin:

- I want create a Pipe instance in the FLXML called “:pipe 1” that has the Block pipe 1 as Type
- I want create a Pipe instance in the FLXML called “:pipe 2” that has the Block pipe 2 as Type
- I want create a FLXML attribute called C_diameter for the Pipe instance “ : pipe1” mapped to the ValueProperty diameter of the Block pipe1 and provide the 0.2 value
- I want create a FLXML attribute called C_diameter for the Pipe instance “ : pipe2” mapped to the ValueProperty diameter of the Block pipe2 and provide the 10.0 value



To resolve this case, a mapping between the ValueProperty and an Attribute DestinationType must be defined. The ValueProperty must have the Stereotype Attribute of the FLXML Profile. Below is a representation of this RFLPSchema to support this scenario:

To define a dynamic attribute the user will have:

- To create a RFLPRule to indicate to generate a RFLPAttribute from an Element source Type defined by a stereotype or some other criterias
- For this RFLPRule mandatory define these 4 RFLPAttributes:
  - Name : to indicate the name of the attribute to create
  - Owner : to indicate to which owner add this RFLPAttribute created dynamically
  - Type : to indicate the type of the value RFLPAttribute to create
  - Value : to indicate how to get the value of this attribute, can be an OCL expression or a literal or …
- For this RFLRule an optinal attribute, “Instance” with a value to true can be created in order to indicate that the RFLAttribute must be affected to all instances of the owner of this RFLPAttribute if this attribute is a Reference.

RFLPProfile

The OOTB RFLPSchema SysML to FL is stored in the RFLP Profile since the RFLPSchema is an Uml Element that contains other inner elements (RFLPRule/RFLPAttribute). Consequently, it is CATIA Magic project (Teamwork Cloud or mdzip project) that store the definition of the RFLPSchema used to do the generation of FLXML.

A client can decide to create a dedicated profile or used project to define its own RFLPSchema that will allow to generate FLXML from the elements of its projects.

OOTB RFLPSchema SysML to FL

Below CATIA Magic table that list some of RFLPRule defined in the OOTB RFLPSchema “SysML to FL” located in the RFLP Profile.

Criteria

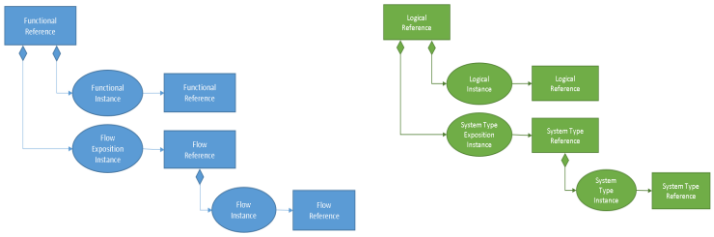
Element Type: RFLPRule,RFLPType ... Scope (optional): SysML to FL ... Filter:

#	Name	Source Type	Source Constraint	Destination	Destination Category
1	AcceptEventAction to RFLPLMFunctionalInstance	AcceptEventAction		RFLPLMFunctionalInstance	FunctionalInstance
2	AcceptEventAction to RFLPLMFunctionalReference	AcceptEventAction		RFLPLMFunctionalReference	FunctionalReference
3	Activity agregated by a Block to RFLPLMImplementConn	Activity	self.owner.oclIsTypeOf(SysML::Block)	RFLPLMImplementConnection	ImplementConnection
4	Activity to RFLPLMFunctionalReference	Activity		RFLPLMFunctionalReference	FunctionalReference
5	ActivityFinalNode to RFLPLMFunctionalInstance	ActivityFinalNode		RFLPLMFunctionalInstance	FunctionalInstance
6	ActivityFinalNode to RFLPLMFunctionalReference	ActivityFinalNode		RFLPLMFunctionalReference	FunctionalReference
7	ActivityParameterNode to RFLPLMFlowExpositionInstanc	ActivityParameterNode		RFLPLMFlowExpositionInstance	FunctionalFlowExpositionInstance
8	Allocate to RFLPLMImplementConnection	Allocate [Abstraction]		RFLPLMImplementConnection	ImplementConnection
9	Block Owner of ValueProperty/FlowProperty to RFLVPMS	Block [Class]	(self.ownedElement->select(pe|pe.oclIsKindOf("MD Customization fo...	RFLVPMSystemTypeReference	LogicalFlowReference
10	Block to RFLVPMLogicalReference	Block [Class]		RFLVPMLogicalReference	LogicalReference
11	Block Used as Type of ActivityParameterNode to RFLPLM	Block [Class]	self._typedElementOfType->select(pe|pe.oclIsKindOf(ActivityParamet...	RFLPLMFlowReference	FunctionalFlowReference
12	Block Used as Type of FlowProperty to RFLVPMSystemTy	Block [Class]	self._typedElementOfType->select(pe|pe.oclIsKindOf(SysML::"Ports&...	RFLVPMSystemTypeReference	LogicalFlowReference
13	Block Used as Type of Pin to RFLPLMFlowReference	Block [Class]	self._typedElementOfType->select(pe|pe.oclIsKindOf(Pin))->size() > 0	RFLPLMFlowReference	FunctionalFlowReference
14	Block Used as Type of ProxyPort to RFLVPMSystemTypeR	Block [Class]	self._typedElementOfType->select(pe|pe.oclIsKindOf(SysML::"Ports&...	RFLVPMSystemTypeReference	LogicalFlowReference
15	CallBehaviorAction to RFLPLMFunctionalInstance	CallBehaviorAction		RFLPLMFunctionalInstance	FunctionalInstance
16	CallBehaviorAction to RFLPLMFunctionalReference	CallBehaviorAction	self.behavior.oclIsInvalid() or self.behavior.oclIsUndefined()	RFLPLMFunctionalReference	FunctionalReference
17	Connector to RFLVPMLogicalConnection	Connector		RFLVPMLogicalConnection	LogicalConnection
18	ControlFlow to RFLPLMFunctionalConnection	ControlFlow		RFLPLMFunctionalConnection	FunctionalConnection
19	DecisionNode to RFLPLMFunctionalInstance	DecisionNode		RFLPLMFunctionalInstance	FunctionalInstance
20	DecisionNode to RFLPLMFunctionalReference	DecisionNode		RFLPLMFunctionalReference	FunctionalReference
21	FlowProperty to RFLVPMLogicalInstance	FlowProperty [Property]	self.owner.oclIsTypeOf(SysML::Block)	RFLVPMLogicalInstance	LogicalInstance
22	FlowProperty to RFLVPMSystemTypeInstance	FlowProperty [Property]	self.owner.oclIsKindOf(SysML::"Ports&Flows"::InterfaceBlock)	RFLVPMSystemTypeInstance	LogicalFlowInstance
23	ForkNode to RFLPLMFunctionalInstance	ForkNode		RFLPLMFunctionalInstance	FunctionalInstance
24	ForkNode to RFLPLMFunctionalReference	ForkNode		RFLPLMFunctionalReference	FunctionalReference
25	InitialNode to RFLPLMFunctionalInstance	InitialNode		RFLPLMFunctionalInstance	FunctionalInstance
26	InitialNode to RFLPLMFunctionalReference	InitialNode		RFLPLMFunctionalReference	FunctionalReference
27	InputPin to RFLPLMFlowExpositionInstance	InputPin	self.syncElement.oclIsInvalid() or self.syncElement.oclIsUndefined()	RFLPLMFlowExpositionInstance	FunctionalFlowExpositionInstance
28	InterfaceBlock Owner of ValueProperty/FlowProperty to F	InterfaceBlock [Class]	(self.ownedElement->select(pe|pe.oclIsKindOf("MD Customization fo...	RFLVPMSystemTypeReference	LogicalFlowReference
29	InterfaceBlock Used as Type of ActivityParameterNode or	InterfaceBlock [Class]	self._typedElementOfType->select(pe|pe.oclIsKindOf(ActivityParamet...	RFLPLMFlowReference	FunctionalFlowReference
30	InterfaceBlock Used as Type of FlowProperty to RFLVPMS	InterfaceBlock [Class]	self._typedElementOfType->select(pe|pe.oclIsKindOf(SysML::"Ports&...	RFLVPMSystemTypeReference	LogicalFlowReference
31	InterfaceBlock Used as Type of ProxyPort to RFLVPMSyste	InterfaceBlock [Class]	self._typedElementOfType->select(pe|pe.oclIsKindOf(SysML::"Ports&...	RFLVPMSystemTypeReference	LogicalFlowReference
32	JoinNode to RFLPLMFunctionalInstance	JoinNode		RFLPLMFunctionalInstance	FunctionalInstance
33	JoinNode to RFLPLMFunctionalReference	JoinNode		RFLPLMFunctionalReference	FunctionalReference
34	MergeNode to RFLPLMFunctionalInstance	MergeNode		RFLPLMFunctionalInstance	FunctionalInstance
35	MergeNode to RFLPLMFunctionalReference	MergeNode		RFLPLMFunctionalReference	FunctionalReference
36	ObjectFlow to RFLPLMFunctionalConnection	ObjectFlow		RFLPLMFunctionalConnection	FunctionalConnection
37	OutputPin to RFLPLMFlowExpositionInstance	OutputPin	self.syncElement.oclIsInvalid() or self.syncElement.oclIsUndefined()	RFLPLMFlowExpositionInstance	FunctionalFlowExpositionInstance
38	PartProperty to RFLVPMLogicalInstance	PartProperty [Property]		RFLVPMLogicalInstance	LogicalInstance
39	ProxyPort to RFLVPMSystemTypeExpositionInstance	ProxyPort [Port]		RFLVPMSystemTypeExpositionInstance	LogicalFlowExpositionInstance
40	SendSignalAction to RFLPLMFunctionalInstance	SendSignalAction		RFLPLMFunctionalInstance	FunctionalInstance
41	SendSignalAction to RFLPLMFunctionalReference	SendSignalAction		RFLPLMFunctionalReference	FunctionalReference
42	Signal As Type of ActivityParameterNode to RFLPLMFlow	Signal	self._typedElementOfType->select(pe|pe.oclIsKindOf(ActivityParamet...	RFLPLMFlowReference	FunctionalFlowReference
43	Signal As Type of Pin to RFLPLMFlowReference	Signal	self._typedElementOfType->select(pe|pe.oclIsKindOf(Pin))->size() > 0	RFLPLMFlowReference	FunctionalFlowReference
44	Signal to RFLPLMFunctionalReference	Signal	(self._typedElementOfType->select(pe|pe.oclIsKindOf(Pin))->size() =0...	RFLPLMFunctionalReference	FunctionalReference
45	Signal Used as Type of a FlowProperty (owner:Block) to R	Signal	self._typedElementOfType->select(pe|pe.oclIsKindOf(SysML::"Ports&...	RFLVPMLogicalReference	LogicalReference
46	Signal Used as Type of a FlowProperty (owner:InterfaceBl	Signal	self._typedElementOfType->select(pe|pe.oclIsKindOf(SysML::"Ports&...	RFLVPMSystemTypeReference	LogicalFlowReference
47	ValueProperty to RFLVPMSystemTypeInstance	ValueProperty [Property]		RFLVPMSystemTypeInstance	LogicalFlowInstance
48	ValueType Used as Type of a Property to RFLVPMSystemT	ValueType [DataType]	self._typedElementOfType->select(pe|pe.oclIsKindOf(Property))->size...	RFLVPMSystemTypeReference	LogicalFlowReference
49	ValueType Used as Type of Pin to RFLPLMFlowReference	ValueType [DataType]	self._typedElementOfType->select(pe|pe.oclIsKindOf(Pin))->size() > 0	RFLPLMFlowReference	FunctionalFlowReference

FLXML

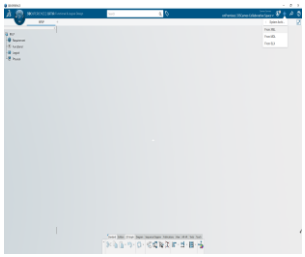
The FLXML format is the basic import / export format for Functional and Logical objects in the 3DEXPERIENCE. The FLXML is an XML file compatible with the FLImportExportXMLValidation.xsd schema.

The purpose of this document is not to explain the FLXML format. More explanations can be found at: (ref in the CAA doc). Examples of 3DEXPERIENCE FL structures that can be created via FLXML.

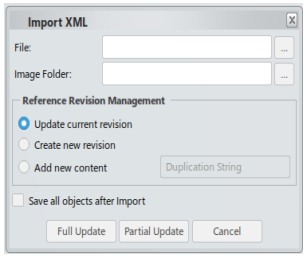


How to import FLXML in CATIA

FLXML file import is accessible only in CATIA and with the Functional and Logical Design application opened.



- Open CATIA
- Open Apps Functional & Logical Design
- Click on + button + in the toolbar
- Click on « System Architecture » menu
- Click on the "From XML …" menu
- The import dialog box is displayed, see next :



Below the official documentation of the FLXML import in CATIA: https://help.3ds.com/2021x/English/DSDoc/FleUserMap/fle-t-XMLImport.htm?ContextScope=onpremise

The documentation regarding the Functional & Logical Design Apps is available here https://help.3ds.com/2021x/English/DSDoc/FleUserMap/fle-c-ov.htm?ContextScope=onpremise&id=b5908c3113a74ed0a9946ae541c10508#Pg0

Restrictions

Some verifications done in CATIA in the import will not exist in the CATIA FLXML Exporter Plugin. For instance:

- The plugin will not verify that the owner reference of a RFLVPMLogicalSystemInstance is a RFLVPMLogicalSystemReference or a sub type of RFLVPMLogicalSystemReference.
- The plugin will not verify that the reference of a RFLPLMFunctionalMissionInstance is a RFLPLMFunctionalMissionReference
- …

All this kind of verifications cannot be done in the plugin, only in the import of the FLXML in CATIA.

FLXML Constraints & CATIA FLXML Exporter Plugin Usages

FLXML object unicity

The unicity of an object in the 3DEXPERIENCE and in the FLXML is guaranteed in two ways:

- TNR : Type Name and Revision
- Physical Id : unique identifier

The PLM_ExternalID attribute is the Name in the TNR

```
<RFLP>
  <LogicalReference>
    <ID Value="100001" Type="RFLVPMLogicalReference">
      <Mandatory>
        <PLM_ExternalID Type="String">log-40368474-00000001</PLM_ExternalID>
        <BoundingBox XMin="0" YMin="0" XMax="504" YMax="336"></BoundingBox>
      </Mandatory>
      <V_Name Type="String">Logical Reference00000001</V_Name>
      <revision ID="2" Type="String">A.1</revision>
    </ID>
  </LogicalReference>
</RFLP>
```

In this FLXML example, the Type is RFLVPMLogicalReference, the Name is log-40368474-00000001 and the revision is A.1.

In the Plugin, by default in the OOTB mapping:

Attribute	Description
PLM_ExternalID	Initialize with the unique identifier of the CATIA Magic Element. For Teamwork Cloud, the unique identifier on the server.
V_Name	Initialize the signature of the CATIA Magic Element, name provided by Java API RepresentationTextCreator.getRepresentedText(element)
Type	The type is provided automatically with the rule mapping

The user can defined its own way to identify the unicity of objects.

Traceability

FLXML Object unicity will allow to keep the traceability between CATIA Magic objects and FL Objects. **A CATIA Systems Traceability pattern** will help to create links between CATIA Magic objects and FL Objects and keep the traceability.

The mechanism of branching in the 3DEXPERIENCE will modify the PLM_ExternalID, consequently the user will have to have to find a mechanism to keep the traceability between CATIA Magic Objects and FL Objects.

Revision

In the FLXML definition, there is a revision attribute for objects.

The import of the FLXML in CATIA does not take into account this revision attribute in the import process. Only the import of the FLXML in CATIA manages the revision of objects.
The CATIA FLXML Exporter plugin does not manage the revision of objects.

FLXML Update

The FLXML unicity of objects guarantees an update scenario of objects. For example this scenario can be managed:

- Generate a FLXML for a simple IBD
- Import the FLXML in the 3DEXPERIENCE
- Add a PartProperty related to a Block in your IBD
- Generate a new FLXML for your simple IBD, this FLXML will be used to update FL objects in the 3DEXPERIENCE
- Import the FLXML in the 3DEXPERIENCE and the FL objects are created with the new Logical Instance → the unicity of FLXML objects guarantees good update of FL objects.

This scenario works in full update or incremental mode in the import of FLXML CATIA in CATIA.
See PES or Specification of the import in CATIA for more explanations no the different modes or possibilities of import FLXML.

FLXML Delete

The deletion is automatically managed by the import FLXML in CATIA with the Full update scenario in the import of CATIA. In incremental update in the import of FLXML in CATIA, there is no deletion of objects.

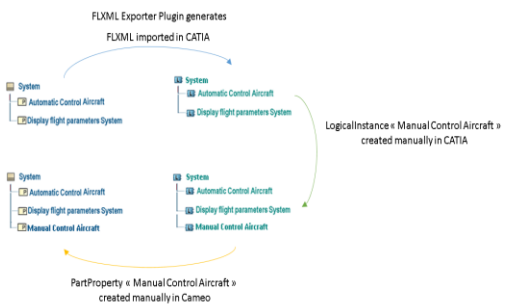
Below the scenario in a full update mode in CATIA.

- Generate a FLXML for a simple IBD with 4 PartProperty
- Import the FLXML in the 3DEXPERIENCE
- Remove a PartProperty related to a Block in your IBD and in the Containment Tree
- Generate a new FLXML for your simple IBD, this FLXML will be used to update FL objects in the 3DEXPERIENCE
- Import the FLXML in the 3DEXPERIENCE and verify that the PartProperty transformed to a LogicalInstance is removed from the MainView and from the hierarchy.

Only the import of the FLXML in CATIA manages the deletion of objects. The CATIA FLXML Exporter plugin does not manage the deletion of objects directly.

Synchronization between FL and CATIA Magic

The FLXML Exporter Plugin will not synchronize CATIA Magic Models from a FLXML stream.



Consequently, any changes done on the FL structures in CATIA after a first import in CATIA of the FLXML, must be reported manually **if needed** in the CATIA Magic models too.

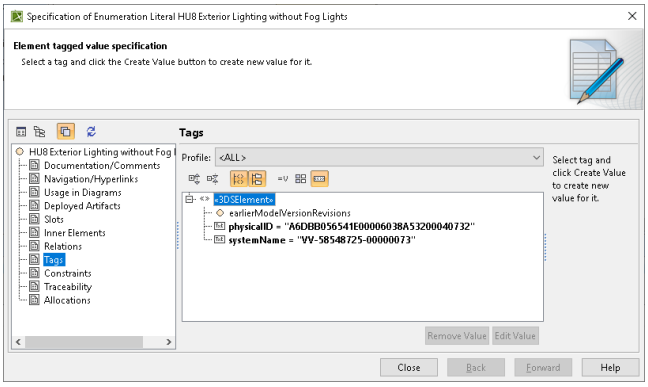
The synchronization is done only one-way CATIA Magic → 3DEXPERIENCE.

The FLXML Exporter Plugin will not allow synchronization both ways.

Effectivity/ Variation Point

The plugin supports the effectivity on instances with the option Manage effectivity set to true. In this case the plugin works in combination of the 3DEXPERIENCE ENOVIA Model Definition Integration plugin.
There are some considerations in order that the effectivity is supported by the plugin:

- **Only configuration models imported by the 3DEXPERIENCE ENOVIA Model Definition Integration plugin version CATIA Magic 2021x Refresh2 or above are supported.**
- Only Existence Variation Point are supported
- Only Existence Variation Point assigned to elements transformed to instance are supported
- NOT condition are not supported by the plugin
- All feature (or type of feature) must have the 3DSElement stereotype
- All feature (or type of feature) must have the systemName filled with the corresponding name in the 3DEXPERIENCE. This taggedvalue is initialized by the 3DEXPERIENCE ENOVIA Model Definition Integration plugin. If they are not initialize, the user must initialized them manually.



The plugin will create a sub directory of the destination folder called EffectivityXML. In this folder, all effectivity XML files will be created.
Example of effectivity files:

```
<?xml version="1.0" ?>
<CfgEffectivityExpression
   xmlns="urn:com:dassault_systemes:config"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:com:dassault_systemes:config CfgEffectivityExpression.xsd">
   <Expression Domain="ConfigChange_Variant">
      <Context HolderType="Model" HolderName="MV-58548725-00000012">
         <Feature Type="ConfigFeature" Name="VAR-58548725-00000021">
            <Feature Type="ConfigFeature" Name="VV-58548725-00000073"/></Feature>
      </Context>
   </Expression>
</CfgEffectivityExpression>
```

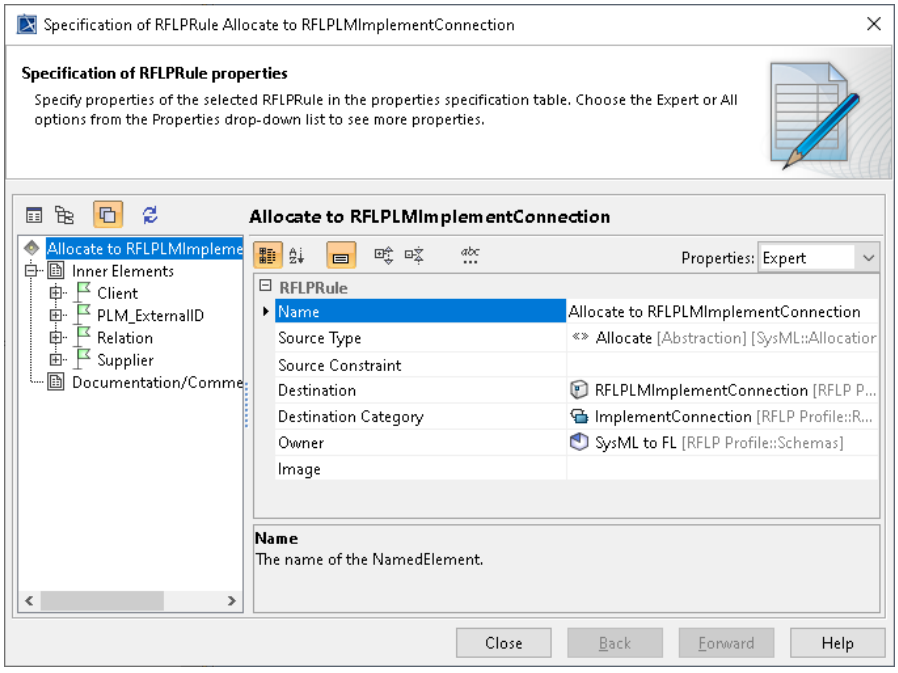
I there is some missing information to generate these effectivity files, the plugin will generate errors.
This PES will not explain how to use/manage effectivity in CATIA Magic.

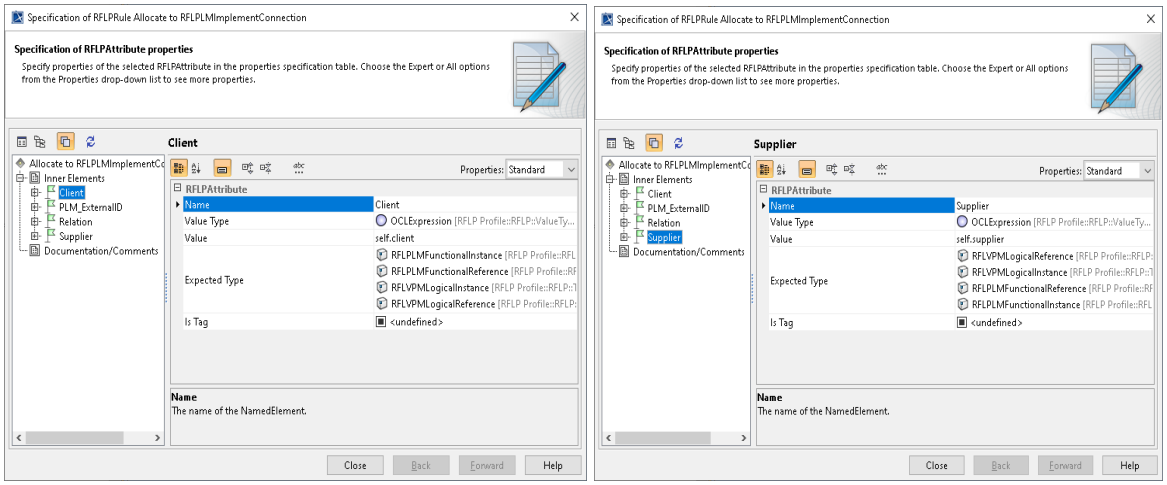
Implementation Link

There is a particular behavior to manage implementation link.
The user will have to define the Client and Supplier RFPAttribute as source and destination of its implementation connection.

The value of Client and Supplier RFPAttribute should be the element that will be transformed to a FLXML reference in the case of REF/REF implement link creation.

The value of Client and Supplier RFPAttribute can be also the element that will be transformed to a FLXML instance in the case of OCC/ OCC implement link creation. In this case the linked REF/REF implementation link of this OCC/OCC implementation link must be exist previously It is possible to use also the option “Create top level ref-ref implement links” see the options §.



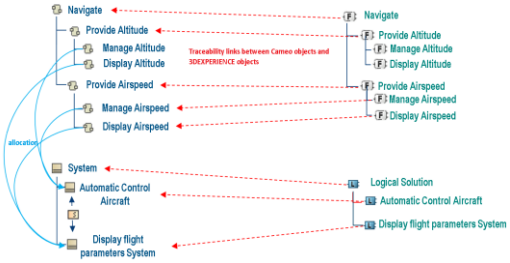


CATIA FLXML Plugin Limitations

Below the limitation of the plugin in CATIA Magic2021x Refresh 2

Some OOTB Schemas

In CATIA Magic 2021 Refresh 2, the plugin will not provide OOTB schemas for Electrical/Fluidic/EEW … Only a **non-exhaustive schema** mapping for SysML to FL will be supported. This schema is provided
in order to support the scenario below :



Enhancement Request should be provided for next releases to support other OOTB configurations.

Layout /Diagrams

Support of images inside FL objects are not supported in the first release of the plugin.

Lifecycle

The lifecycle of object will be managed in the 3DEXPERIENCE, not with the plugin. The assignation of the revision of an object in the FLXML will be managed by the user when importing the file in 3DEXPERIENCE.

Synchronization one-way CATIA Magic-FL

The CATIA FLXML Exporter plugin in the first version supports only one-way synchronization from CATIA Magic to FL. See update/delete scenarios for some synchronization details.

Schematic Views

The CATIA FLXML Exporter plugin will not support in the first version the Schematic Views in the FLXML.

Connection between System Type Instance (or Flow Instance)

The CATIA FLXML Exporter plugin will not support in the first version the FL connection between System Type Instance (or Flow Instance), only with System Type Exposition Instance (or Flow Exposition Instance). The plugin supports only connection to simple ports. Will be supported in next release.

Requirement

The CATIA FLXML Exporter plugin will not support in its first version the creation of implements links that correspond to traceability. For instance, if a Satisfy link exists between a Block and a CATIA Magic requirement, the corresponding implement link will not be created between the Logical Reference created by the plugin and the TRM Requirement synchronized with the CATIA Magic requirement via the Datahub.
Moreover support this, means have a dependency in the CATIA FLXML Exporter plugin with the Datahub plugin.
Note To achieve link traceability between requirements and logical references (both coming from CAMEO respectively using DataHub and CATIA FLXML exporter (and FLXML import)), patterns in System Traceability can be used.

Logical & Functional Parameters

The CATIA FLXML Exporter plugin will not support in its first version the creation of Logical and Functional Parameters.

FLXML Category support

Below the exhaustive list of FLXML category supported by the CATIA FLXML Exporter plugin. Other Categories are not supported by default.

#	Name	#	Name
1	FunctionalConnection	10	LogicalConnection
2	FunctionalFlowExpositionInstance	11	LogicalFlowExpositionInstance
3	FunctionalFlowInstance	12	LogicalFlowInstance
6	FunctionalFlowReference	13	LogicalFlowReference
7	FunctionalImplementConnection	14	LogicalImplementConnection
8	FunctionalInstance	15	LogicalInstance
9	FunctionalReference	16	LogicalReference

RFLP Types

Below the exhaustive list of RFLP types supported by the CATIA FLXML Exporter plugin.

#	Name	#	Name
1	RFLPLMFlowExpositionInstance	11	RFLVPMLogicalInstance
2	RFLPLMFlowInstance	12	RFLVPMLogicalReference
3	RFLPLMFlowReference	13	RFLVPMSystemTypeExpositionInstance
4	RFLPLMFunctionalMissionReference	14	RFLVPMSystemTypeInstance
5	RFLPLMFunctionalMissionInstance	15	RFLVPMSystemTypeReference
6	RFLPLMFunctionalConnection	16	RFLVPMLogicalConnection
7	RFLPLMFunctionalInstance	17	RFLVPMLogicalSystemReference
8	RFLPLMFunctionalReference	18	RFLVPMLogicalSystemInstance
9	RFLPLMFunctionalServiceReference	19	RFLPLMImplementConnection
10	RFLPLMFunctionalServiceInstance		

The CATIA FLXML Exporter plugin will support FL type data model customization and consequently supports all the sub-types of RFLP types enumerated in the table above.
All attributes supported by the import of FLXML in CATIA (including standard customization of attributes and extension attributes) are supported.