2017

InstaSchema Quick-Start Guide (Beta)

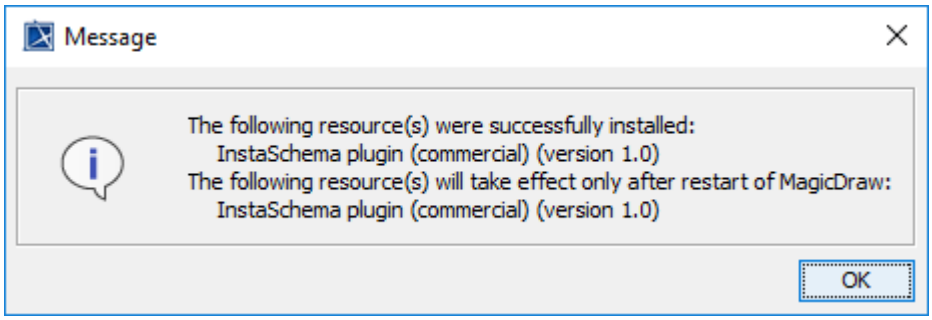
INCQUERY LABS LTD. | https://incquerylabs.com | Bocskai út 77-79. H-1113 Budapest, Hungary

Table of Contents

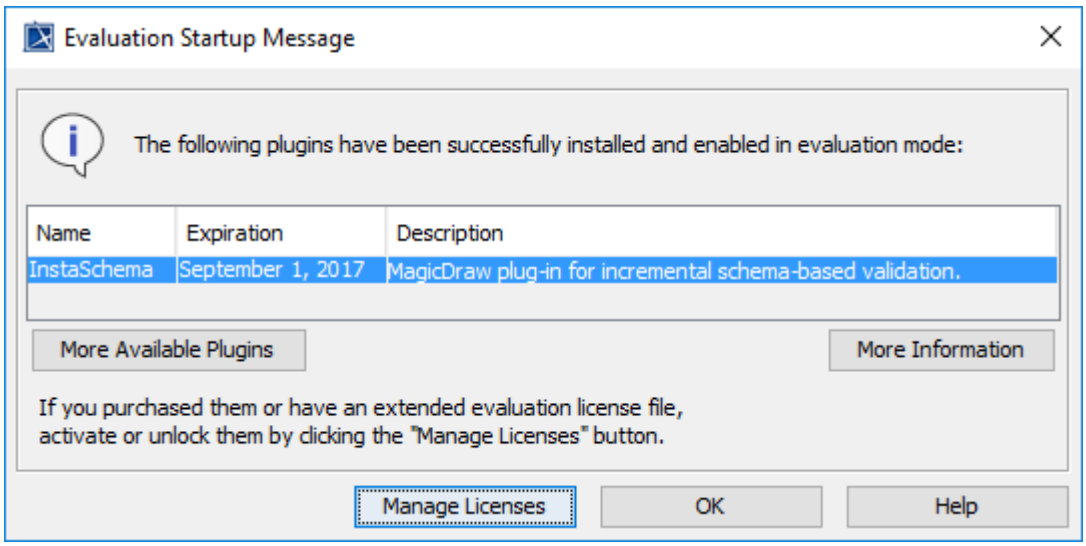
Installation Guide

Installation of the InstaSchema MagicDraw plug-in is straightforward and is done in several simple steps:

1. Install MagicDraw 18.4
2. Install MagicDraw SysML plug-in (Optional)
3. Download InstaSchema archive file
4. Open and anctivate MagicDraw
5. Open Resource/Plugin Manager (Help → Resource/Plugin Manager)
6. Select 'Import'
7. Navigate to the downloaded InstaSchema archive and select it
8. If the installation is successful, the following notification should appear:



9. Restart MagicDraw, read and accept the InstaSchema EULA. After this, the following evaluation startup message should appear.



10. At this point, the installation is complete. By default the InstaSchema evaluation version can be used for up to 90 days.

Basic Concepts

Profile Definitions in MagicDraw

In MagicDraw, it is possible to extend built-in UML and SysML profiles, through custom profile definition and DSL development. These profiles consist of sets of Stereotype objects, each associated with a meta-class. Stereotypes with a certain meta-class can be applied to instances of that given type, to extend them with domain specific information. this information is stored in tagged values. More information on profiling can be found in the MagicDraw UML Profiling and DSL Guide.

DSL Customizations

MagicDraw profiles define sets of stereotypes that extends the UML metamodel, while customizations focus on auxiliary and tooling related information for each stereotype of a given profile. Through these customizations, UML profiles can be expanded into fully fledged domain-specific languages. these customization objects define among others, the following:

- Containment constraints
- Endpoint type constraints for edges
- Also allows the definition of smart tagged value instantiation..

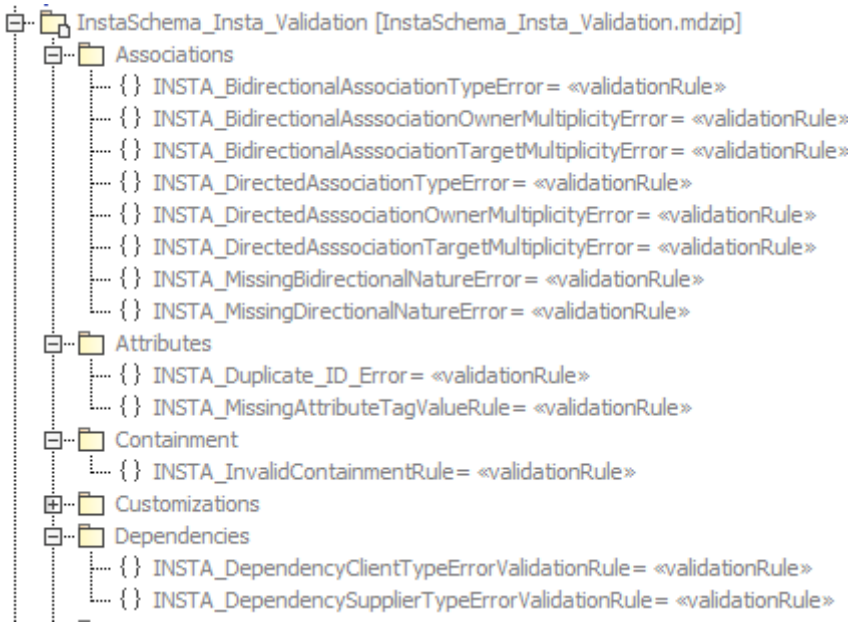
After the profile and its corresponding customizations have been defined, the profile's domain specific types can be used in custom diagram types as well. More information on domain specific modelling can be found in the MagicDraw UML Profiling and DSL Guide.

Structure-based validation in InstaSchema

As with any modeling language, custom DSLs need to be validated. DSL validation can be achieved via sets of user defined UML constraints. Large sets of these, however can be tedious to define. InstaSchema automatically defines conformance validation rules based on DSL structure.

InstaSchema introduces the *SchemaCustomization* stereotype, which can be applied to *Customization* objects to mark them for the validation engine. Elements with the given customization's target stereotype are validated based on the information stored in the profile definition and the appropriate customizations. *SchemaCustomization* stereotype instances also contain information regarding additional *Association*-related constraints.

The InstaSchema validation engine contains a set of predefined constraints, which are parameterized based on the contents of the profile definition. These rules are contained in a shared project shipped with InstaSchema.

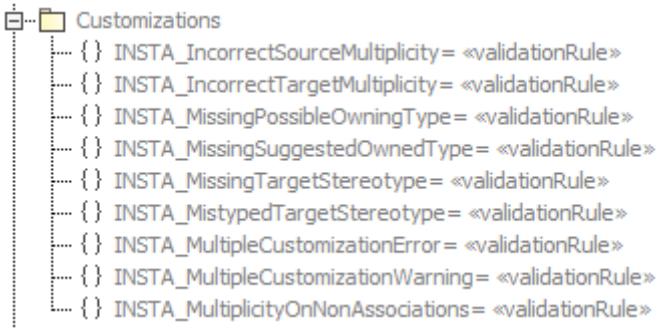


1. Figure Profile validation rules in InstaSchema

The validation rule set defines the following structural validation rules:

- Rules for edges in the DSL:
  - Stereotyped Association: Role types shall correspond to the types specified in the appropriate customization.
  - Stereotyped Dependency: Endpoint types shall correspond to the types specified in the appropriate customization.
  - Stereotyped Association: Role multiplicity shall conform with the specified tag values in the related *SchemaCustomization* instance.
  - Stereotyped Association: Role navigability shall conform with the specified tag values in the related *SchemaCustomization* instance
- Rules for DSL Properties (Tag Values)
  - There shall be no duplicate tag value of an ID Propery (isID == true).
  - There shall be no missing tag value for a mandatory stereotype Property (multiplicity > 0).
- Rules for containment relations in DSLs
  - Stereotyped element: Each stereotyped element shall be contained by an element specified in its customization's *possible Owners* property.

Naturally, during the definition of the customizations themselves possible errors might remain hidden. To abolish this issue, InstaSchema defines a set of *meta-level* rules that check whether the customization definitions contain errors.



2. Figure Meta validation rules in InstaSchema

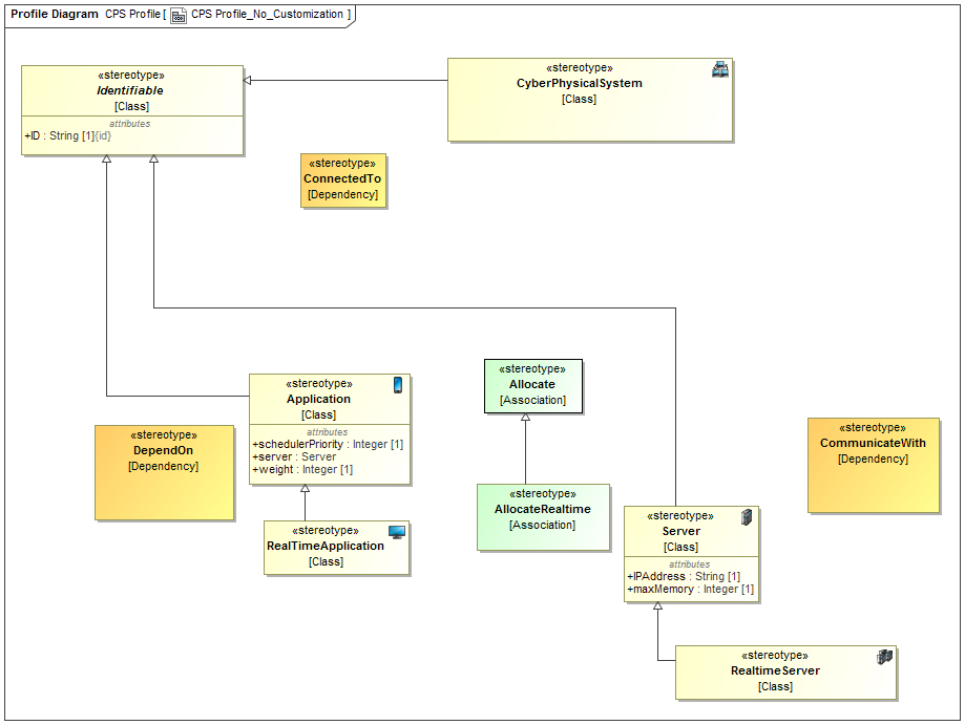
Both meta, and instance level rule implementations field quick fixes, to help the quick resolution of DSL-related validation errors. As the validation rules are only parameterized by the input profile and customization data, the rules themselves are already implemented. Based on the input paremeters (DSL elements), it is possible for the framework to provide sane quick fix suggestions for each pre-defined validation rule.

Cyber-Physical Systems Example

The following part elaborates how the InstaSchema validation is enabled on an already existing DSL. For this part, please download the InstaSchema example zip file, and open the project *SysMLSmartHouseDemo.mdzip*. Note, that the archive also contains a version of the project where InstaSchema validation has already been set up.

Example project - CPS Profile -

the example project *SysMLSmartHouseDemo* defines a custom DSL for cyber-physical systems. Cyber-physical systems are a network of servers that host different applications, which need to be able to communicate with each other. These servers and applications also possess domain specific properties. The figure below shows the UML profile used in the example project.



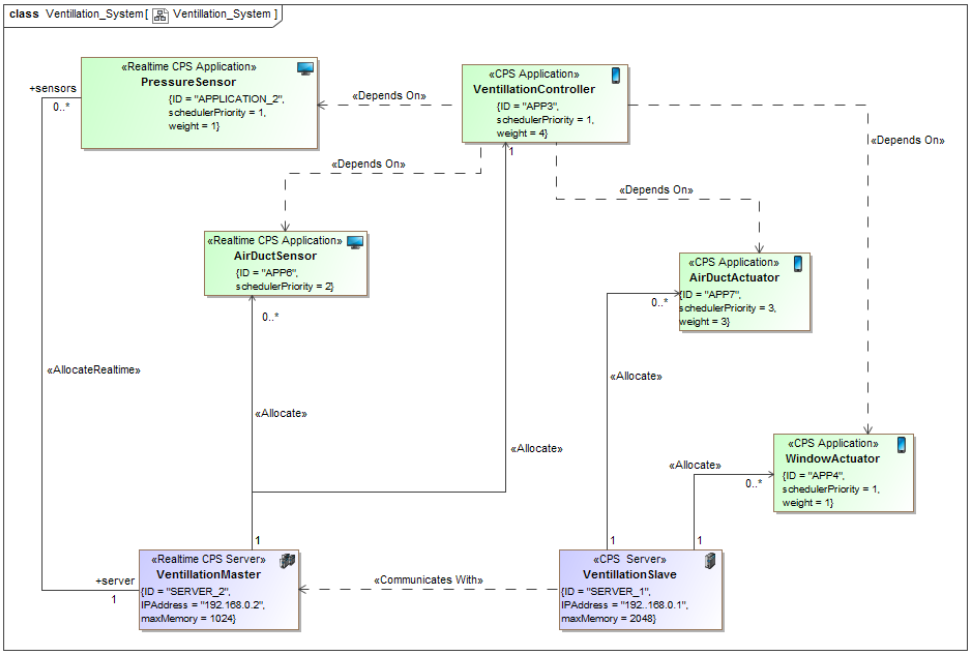
3. Figure: The CPS profile

The profile consists of the following stereotypes:

- ***Identifiable***: Abstract supertype of every CPS type, defines a mandatory String ID property.
  - metaClass: Class
- **CyberPhysicalSystem**: Main unit of containment, can contain variaous servers and applications.
  - metaClass: Class
- **ConnectedTo**: Connection relation defined between two CyberPhysicalSystem elements.
  - metaClass: Dependency
- **Application**: Base application type

    - metaClass: Class
    - Defined properties:
        - schedulerPriority : Integer [1]: Mandatory integer attribute that defines the priority of each application in the execution queue.
        - server : Server: Optional property that stores the server allocating the Application
        - weigth: Integer [1]: Mandatory integer property that defines the server resources required for running the given application.
- **RealTimeApplication**: Subtype of Application, defines an application that can only be run on real time servers.
- **DependOn**: Relation between two Application elements, meaning that one application relies on the output of another.
    - metaClass: Dependency
- **Server**: Base server type
    - metaClass: Class
    - Defined properties:
        - IPAddress: String [1]: Mandatory String attribute that contains the server's IP range
        - maxMemory: Integer [1]: Mandatory Integer attribute that defines the maximum available memory for the given server.
- **RealTimeServer**: Server that can also host RealTimeApplications
- **CommunicateWith**: Defines communication capabilities between two server objects.
    - metaClass: Dependency
- **Allocate**: Stereotyped association that specifies which server hosts which application
    - metaClass: Association
- **AllocateRealtime**: Subtype of Allocate, defines which RealtimeServer hosts which application.

The Example project also defines a DSL for this profile via a set of Customization objects and domain specific diagrams. the project also contains an instance CPS model describing parts of a simple Smart Home system.

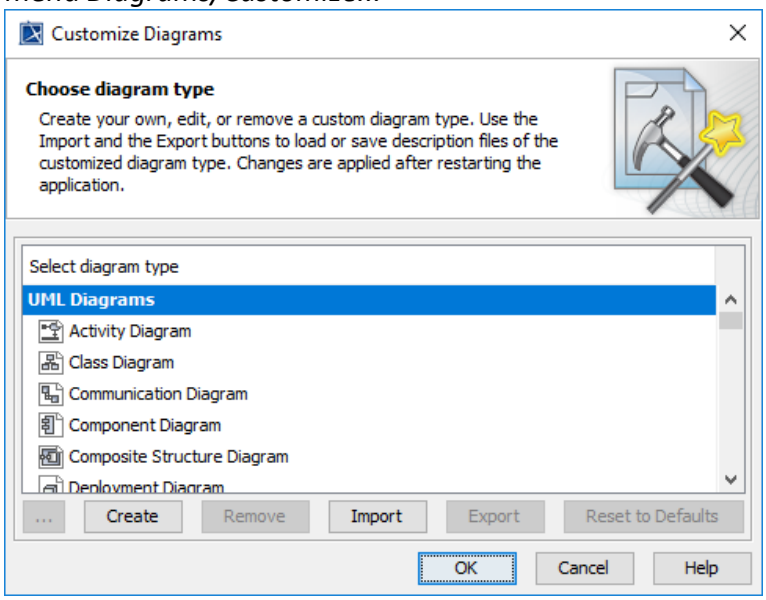


4. Figure: ventillation sub-system of the Smart House model

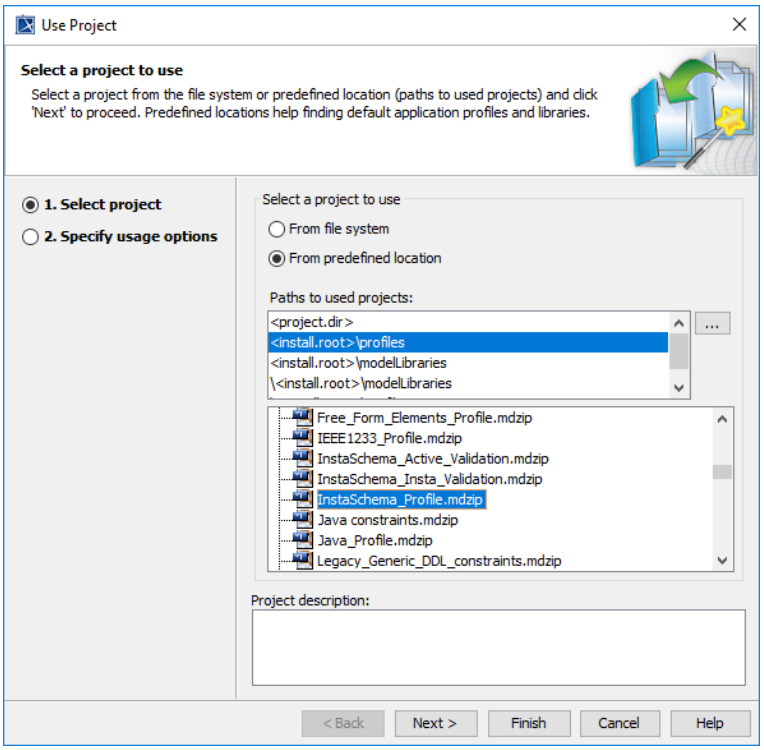
Enable Structure-based validation for the CPS DSL

To enable structure-based validation for the CPS DSL, the following steps need to be undertaken:

1. Install InstaSchema, and open the SysMLSmartHouseDemo project.
2. If not imported automatically, import the diagrams specified by the CPS DSL, via the menu *Diagrams/Customize...*

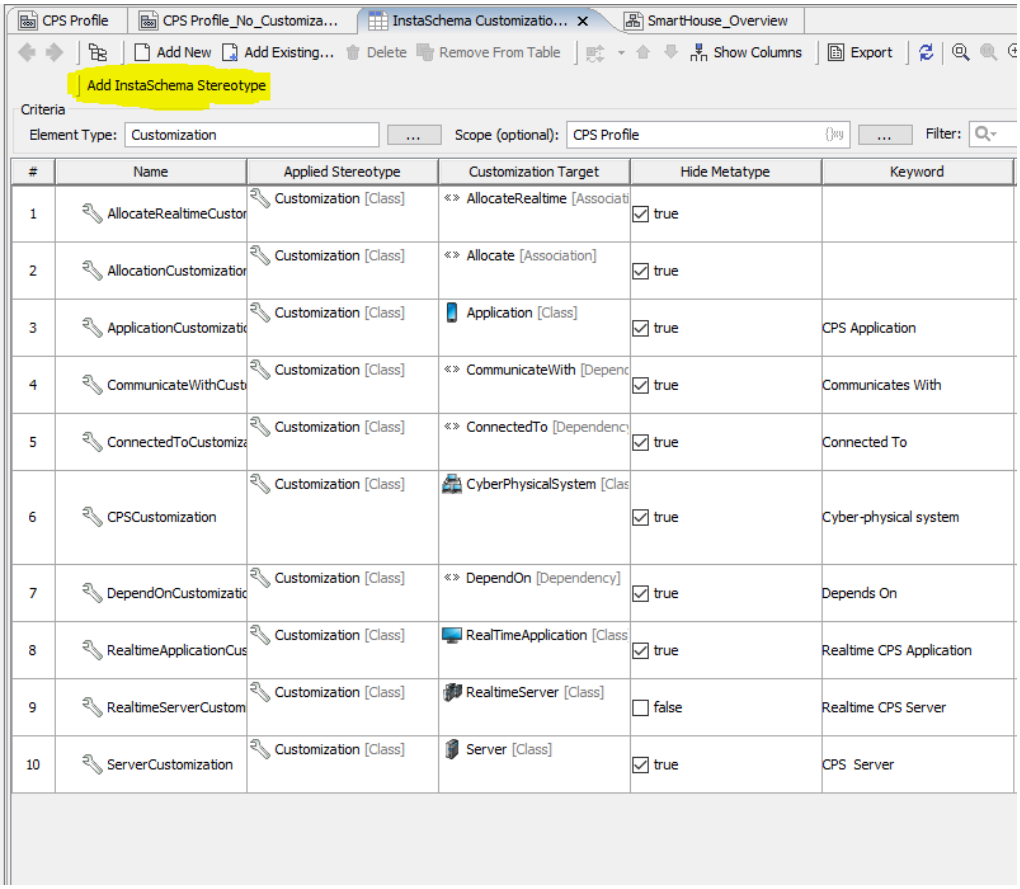


   a. Select *Import* and navigate to *CPS_Detail_Diagram_descriptor.xml* and *CPS_Overview_Diagram_descriptor.xml*
3. Select the InstaSchema Profile as external resource, via File/Use Project/Use Local Project
   a. Here Select InstaSchemaProfile as shown on the figure below

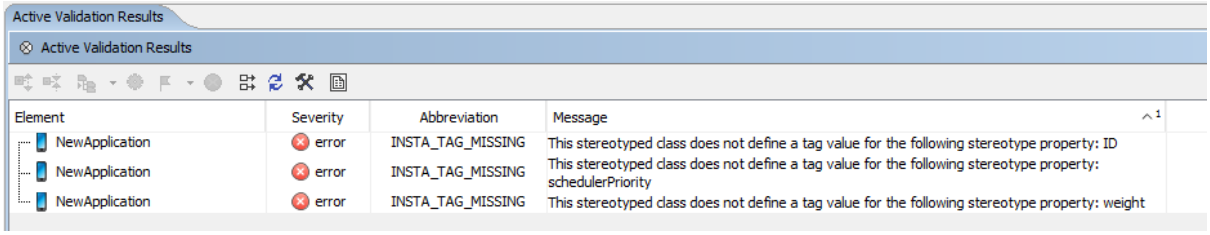


   b. Also select InstaSchema_Insta_Validation in the similar fashion.

4. Navigate to the *CPS Profile* package, and select 'Define InstaSchema Customization Model' from the right click context menu
5. At this point a generic table opens that summarizes the already implemented customizations for the CPS DSL.
    a. Add *SchemaCustomization* stereotype to the customizations, to enable InstaSchema validation on them. This can be done via the button *'Add InstaSchema Stereotype'*



    b. From this point on the validation and quick-fix features of InstaSchema are available.
6. At this point the validation is enabled, Open the CPS detail diagram *Ventillation_System*
    a. Add a new Application element to it
    b. The following errors should appear:



Thank you for trying InstaSchema. Keep in mind however, that the tool itself is still experimental. If you encounter any issues please report them on the following forum topic.