pure::variants - Connector for Capella Manual

Parametric Technology GmbH

Version 7.0.0.685 for pure::variants 7.0

2025

Table of Contents

1. Introduction 1
    1.1. Software Requirements 1
    1.2. Installation 1
    1.3. About this manual 1
2. Using the Connector for Capella 1
    2.1. Starting pure::variants 1
    2.2. First Steps 2
    2.3. Capella-Specific Propagation 2
    2.4. Migrating Mapping Models 7

1. Introduction

The pure::variants Connector for Capella enables pure::variants and the pure::variants Connector for EMF Feature Mapping to work on Capella projects. For the basic documentation of how to use pure::variants with EMF-based projects, please refer to the pure::variants Connector for EMF Feature Mapping Manual. In this documentation only Capella-specific details are documented.

1.1. Software Requirements

The following software has to be present on the user's machine in order to support the pure::variants Connector for Capella:

Capella: Capella or Capella Studio of version 5.0.0 - 7.0.0 is required. Compatibility with other Capella releases is not guaranteed. In Capella the pure::variants Connector for EMF Feature Mapping needs to be installed.

The pure::variants Connector for Capella is an extension for pure::variants and is available on all supported platforms.

1.2. Installation

Please consult section **pure::variants Connectors** in the **pure::variants Setup Guide** for detailed information on how to install the connector (menu **Help** -> **Help Contents** and then **pure::variants Setup Guide** -> **pure::variants Connectors**).

1.3. About this manual

The reader is expected to have basic knowledge about and experiences with pure::variants. The pure::variants manual is available in online help as well as in printable PDF format here.

2. Using the Connector for Capella

2.1. Starting pure::variants

Depending on the installation method used either start the pure::variants-enabled Eclipse or under Windows select the **pure::variants** item from the **program** menu.

If the **Variant Management** perspective is not already activated, do so by selecting it from **Open Perspective -> Other...** in the **Window** menu.

2.2. First Steps

The Connector for EMF Feature Mapping and the Connector for Capella provide the functionality to add variability information to Capella projects, and derive variants of these Capella projects. To start adding variability information to Capella projects, please first show the **Mappings** view via **Window->Show View**. How to use this view and how to set up the transformation to derive variants is described in the pure::variants Connector for EMF Feature Mapping Manual.

2.3. Capella-Specific Propagation

When deriving a Capella variant from a Capella master project, the Capella master project is first copied to the output directory of the transformation. Then certain elements are deleted from the copied project based on the variability information of the Capella master project. In this scenario only elements are deleted that are mapped to a pure::variants condition (which evaluates to `false`). However, for some element types, also other related elements should be deleted. To avoid having to map these related elements to the same pure::variants condition, the Connector for Capella provides several propagation rules.

Basically a propagation rule ensures that if Element A is removed during transformation, also Elements B, C, and D are removed.

To understand which Condition is propagated to which element and why, you can expand the mapped elements further in the **Mappings** view. This shows the next level of propagated elements and the name of the propagation rule. See Figure 2, “Realizing Function Propagation” for an example.

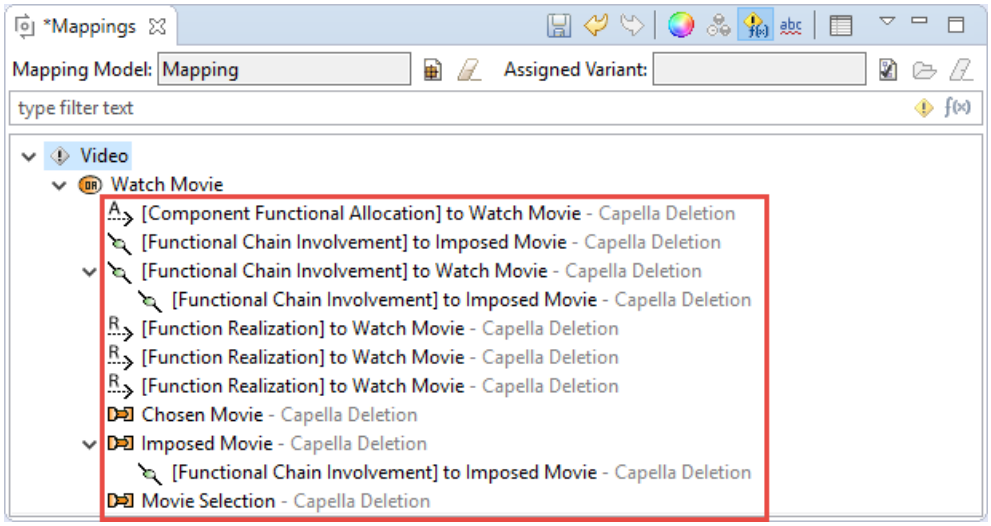
Standard Capella Propagations

The following propagation rules are supported for Capella projects.

Capella Deletion

The *Capella Deletion* propagation rule replicates the behavior of a Capella delete operation. It returns all other elements that are removed automatically by Capella when an element is deleted. Thus it is ensured that variant preview and transformation result match, even though during preview no actual deletion is done. This propagation rule is always enabled.

Figure 1. Capella Deletion Propagation

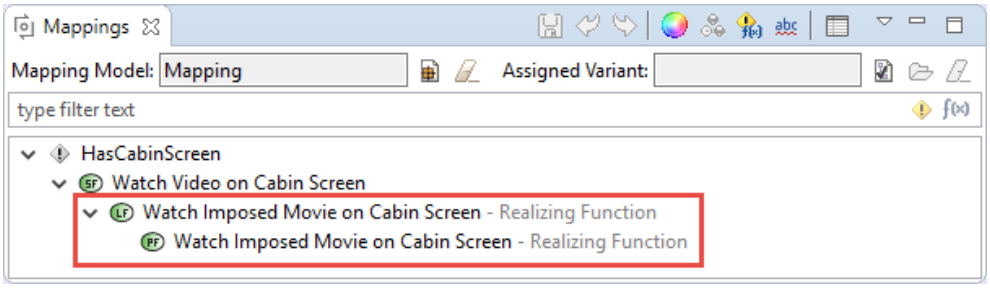


Function Realization

When a function is removed during transformation, each realizing function is also removed if the realizing function does not realize any other function. For example, system function *Watch Video On Cabin Screen* is mapped to condition `HasCabinScreen`. Therefore, its realizing logical function *Watch Imposed Movie On Cabin Screen* is indirectly mapped to the same condition. The same applies to the realizing physical function of *Watch Imposed Movie On Cabin Screen*.

If, for example, the physical function would realize another logical function and that logical function would still be in the output model after transformation, `HasCabinScreen` would not be propagated to the physical function. Please note that for the sake of simplicity all Capella Deletion propagations have been removed from the image.

Figure 2. Realizing Function Propagation

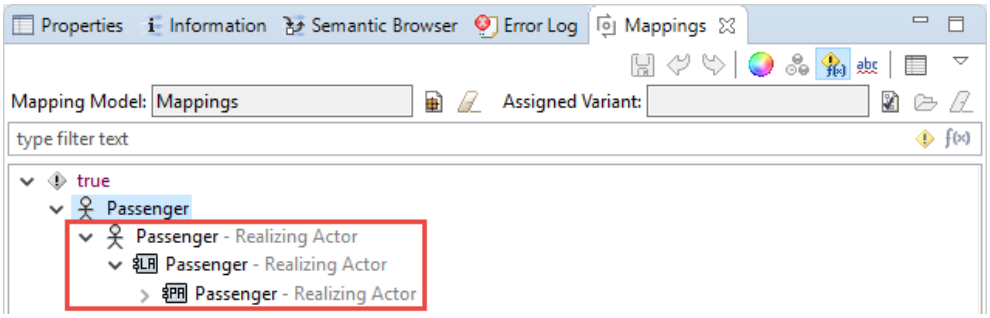


Actor Realization

The *Realizing Actor* propagation works in the same way as the Realizing Function propagation. The only difference is that it applies to actors instead of functions. When an actor is removed during transformation, each realizing actor is also removed if it does not realize any other actors.

Please note that for the sake of simplicity all Capella Deletion propagations have been removed from the image.

Figure 3. Realizing Actor Propagation

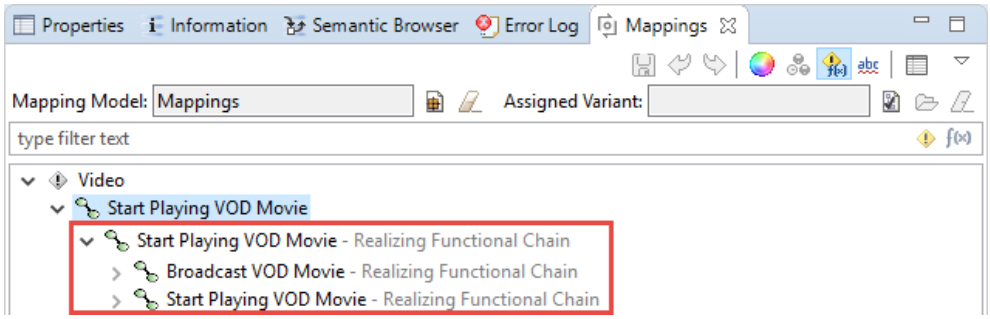


Functional Chain Realization

The *Realizing Functional Chain* propagation works in the same way as the Realizing Actor and Function propagations. The only difference is that it applies to functional chains instead of actors or functions. When a functional chain is removed during transformation, each realizing functional chain is also removed if it does not realize any other functional chains.

Please note that for the sake of simplicity all Capella Deletion propagations have been removed from the image.

Figure 4. Realizing Functional Chain Propagation



Function Ports and their Connectors

When a function is removed during transformation all its Function Ports and the connectors to other ports (each port's Functional Exchanges, allocated Component Ports and Component Exchanges) are also removed. For example, condition `HasCabinScreen` is propagated to functional exchange *Audio Signal* and *Displayed Video*, as well as to the connected ports. Please note that for the sake of simplicity all Capella Deletion propagations have been removed from the image.

Figure 5. Function Ports and their Connectors Propagation

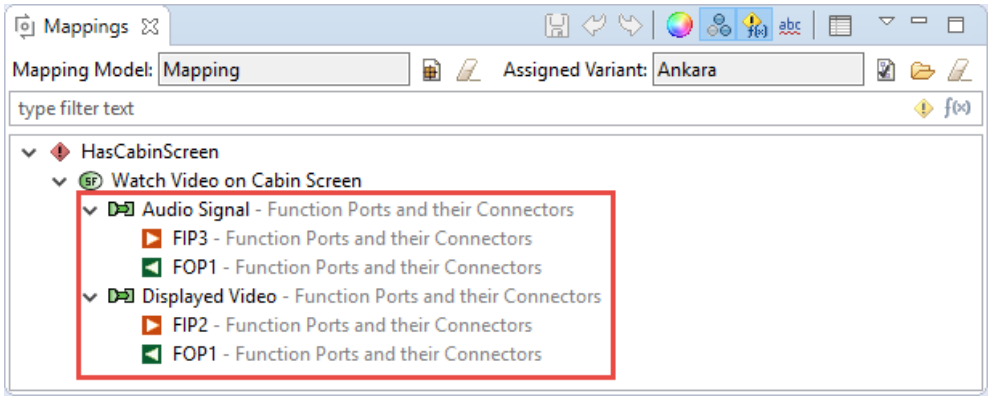
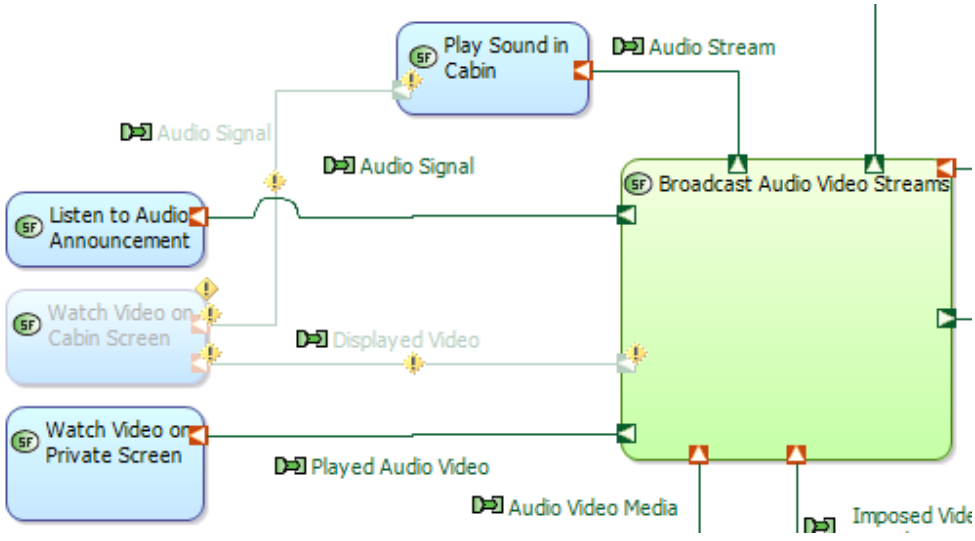


Figure 6. Variant Mode with Function Ports and their Connectors Propagation Enabled

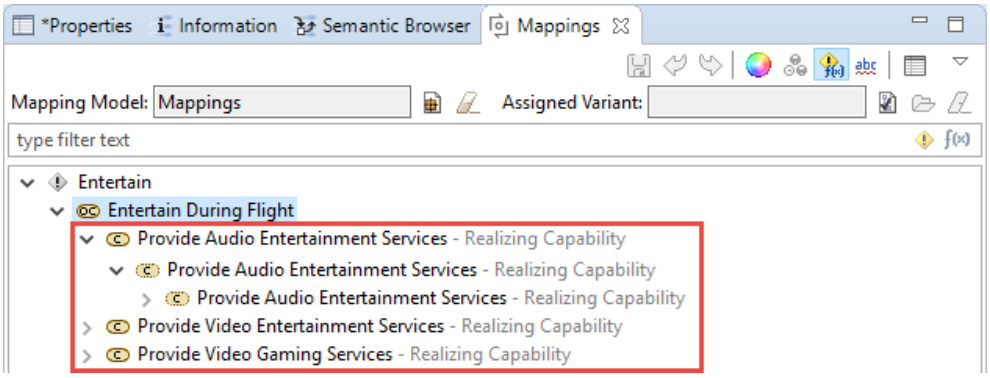


Capability Realization

The *Realizing Capability* propagation works in the same way as the Realizing Function, Actor and Functional Chain propagation. The only difference is that it applies to capabilities. When a capability is removed during transformation, each realizing capability is also removed if it does not realize any other capabilities.

Please note that for the sake of simplicity all Capella Deletion propagations have been removed from the image.

Figure 7. Realizing Capability Propagation

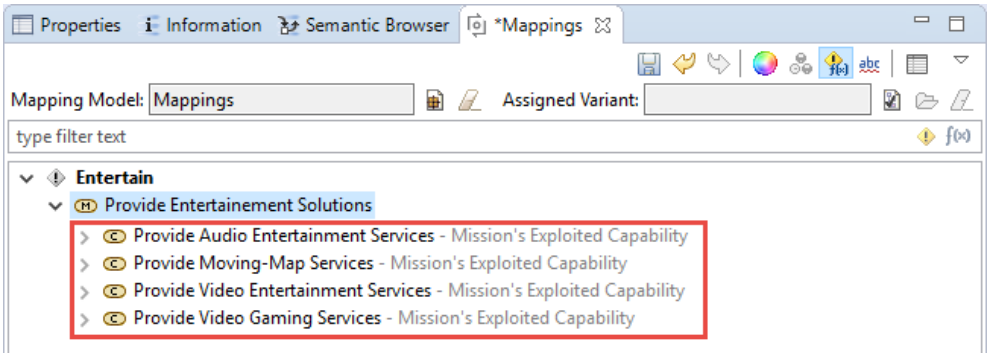


Mission's Exploited Capabilities

When a mission is removed during transformation, each exploited capability is also removed if it is not exploited by any other mission.

Please note that for the sake of simplicity all Capella Deletion propagations have been removed from the image.

Figure 8. Mission's Exploited Capabilities Propagation

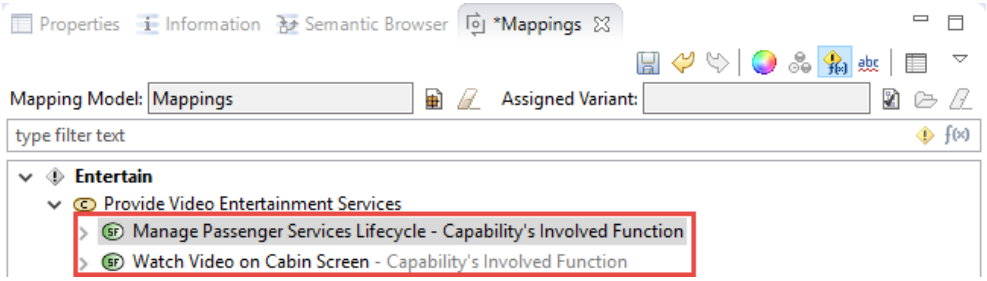


Capability's Involved Function

When a capability is removed during transformation, each of its involved functions is also removed if it is not involved in any other capability.

Please note that for the sake of simplicity all Capella Deletion propagations have been removed from the image.

Figure 9. Capability's Involved Function Propagation

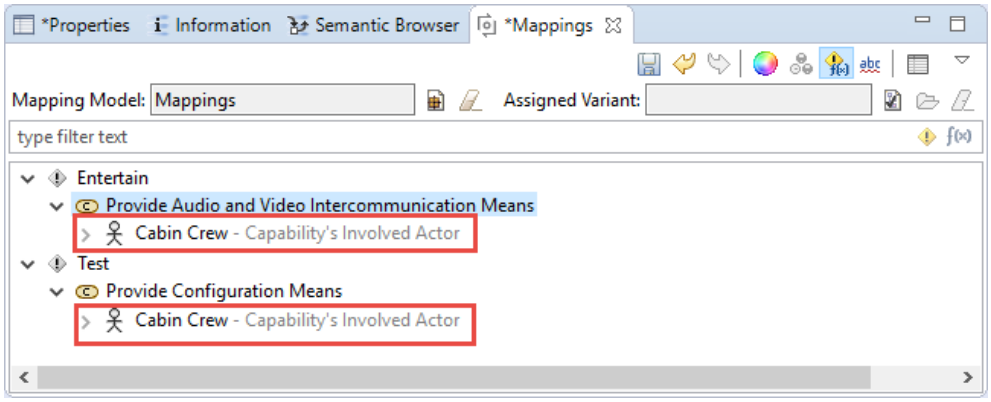


Capability's Involved Actor

When a capability is removed during transformation, each of its involved actors is also removed if it is not involved in any other capability.

Please note that for the sake of simplicity all Capella Deletion propagations have been removed from the image.

Figure 10. Capability's Involved Actor Propagation

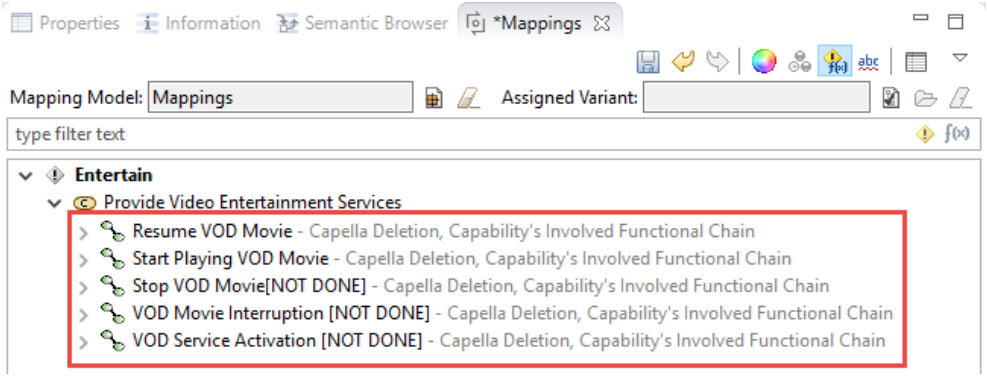


Capability's Involved Functional Chain

When a capability is removed during transformation, each of its involved functional chains is also removed if it is not involved in any other capability.

Please note that for the sake of simplicity all Capella Deletion propagations have been removed from the image.

Figure 11. Capability's Involved Functional Chain Propagation

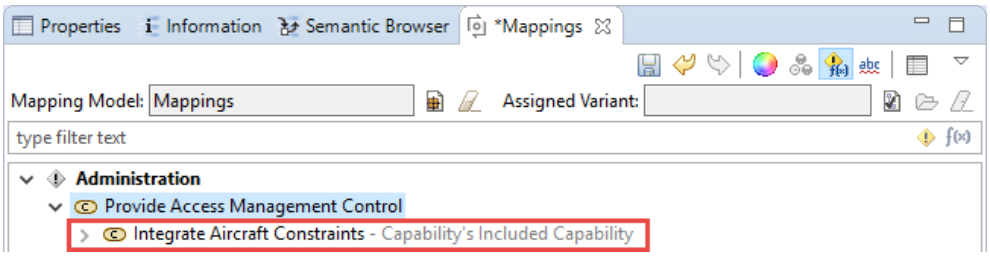


Capability's Included Capability

When a capability is removed during transformation, each of its included capabilities is also removed if it is not included by any other capability.

Please note that for the sake of simplicity all Capella Deletion propagations have been removed from the image.

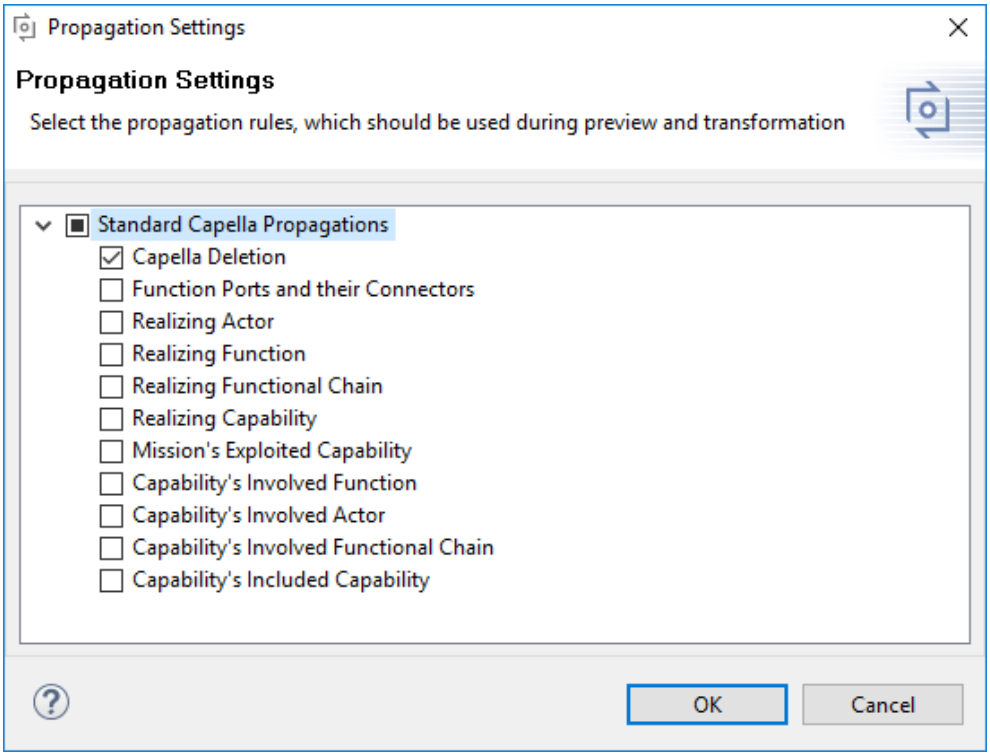
Figure 12. Capability's Included Capability Propagation



Propagation Settings

Propagation rules can be enabled or disabled in the propagation settings ( ) of the **Mappings** view. These settings are used during transformation of Capella variants, and when variant mode or decoration mode is enabled.

Figure 13. Propagation Settings



2.4. Migrating Mapping Models

If you are using Capella of version >= 5.0, the old mapping models should be migrated to work with the newer capella versions. While loading a mapping model into the **Mappings** View, you will see a message dialog that allows you to migrate your model. The same dialog will be seen during the transformation. See Figure 14, “Mapping model Migration”.

Figure 14. Mapping model Migration