



# ***IPD – Server***



---

# Contents

<b>IPD – Server</b>	<b>1</b>
<b>Introduction</b>	<b>3</b>
How to read the symbols	4
<b>Customization</b>	<b>5</b>
How to customize Plantypes	5
How to customize Parent Child Relations	8
Example to customize the parent child relation	8
<b>Index of Figures</b>	<b>13</b>

# Introduction

The operation mode of functioning and menu structure described in this Installation of *IPD- Server User Manual* are explained to you in a simple and understandable manner. It shows how to install and set up the IPD-Server so you can use it when working with DELMIA software.

Change Management Controlled (CMC) deals with constraints regarding data consistency. The functionality is necessary to mark objects being under the CMC control and to ensure, that the data consistency according to the Change Management is never broken.

The versions can represent the evolution of an object through successive stages (revisions) or parallel alternatives (variants). The versioning services capture the history of modifications by tracking different versions as affected objects to actions managed by ENOVIA. The configuration services evaluate version selection rules over modification history, in order to control the visibility of the versioned objects.



---

**Note:**

*The Change Management Control options are enabled only when the Manufacturing Hub project is Change Management (CMC) controlled. Otherwise, these options are shown in a disabled state.*

*The availability of the Change Management Control options functionality is controlled by a separate license. You must have this license installed to open a project which is Change Management (CMC) controlled.*

---

**The configuration services will allow to:**

- label projects to be CMC versioned
- work with CMC in detached mode
- perform CMC related consistency checks
- perform filtering necessary for displaying of versioned objects

## How to read the symbols

The signs and symbols used in this manual as well as in all further manuals are nice to look at, nevertheless, they are intended to be a user guide which explains the contents in an easy-to-understand manner. Chapters and chapter sections are marked by headings. The heading font size varies according to the different usages.

The meaning of the symbols will be explained in the following section:



This symbol indicates excerpts of text which describe the functional scope you will learn about in a given chapter. Therefore, you will normally find this sign at the beginning of a chapter or a section. In addition, this sign will mark important passages.



---

**Note:** *This symbol refers to notes which provide further information on a topic that is necessary for continuing work steps. The note symbol can be found at both the beginnings of chapters as well as at certain points in the text within a chapter. The texts introduced by these symbols are additionally marked with the word **note**. The text itself always appears *italics type*.*

---



---

**Warning:** *This symbol indicates circumstances which could lead to possible errors in the operation of the program; you should therefore pay close attention to them. The warning symbol can be found at both the beginnings of chapters as well as at certain points in the text within a chapter. The texts introduced by these symbols are additionally marked with the word **warning**. The text itself always appears *italics type*.*

---

### Example

This symbol refers to examples which clarify certain topics.



This symbol indicates the individual operating steps in instructions. Operating instructions describe operating steps, for example, how to open a menu or execute a function.



This symbol refers to lists. The symbol for listed subjects can either be used to structure a continuous text or to list main subject keywords.



This symbol indicates that there is further information on this topic available in another manual.

# Customization

This chapter will provide you with information necessary to customize the Manufacturing Hub to properly support the Change Management functionality.

The entire customization is done within a plantype set. Furthermore, the paragraph will include additional information about how Default-PRO-DPM-IGRIP\_V5\_CMC has been prepared.

## How to customize Plantypes

This paragraph will provide you with information necessary to customize the Manufacturing Hub to properly support the Change Management functionality. The entire customization is done within a plantype set. Furthermore, the paragraph will include additional information about how Default-PRO-DPM-IGRIP\_V5\_CMC has been prepared.

For each plantype that is in scope of Change Management the property '**Has versions**' has to be set to 'Yes'. Only versionable plantypes are tracked by Change Management; in order to modify them, an action and a modification statement are required. Non-versionable plantypes are not in scope of Change Management, and for them the property '**Has versions**' has to be explicitly set to 'No'. It is especially important for product-plantypes not to be versionable in order to allow their import through the Bridge.



---

### Note

*For plantypes in R15 this property is usually set to 'Yes', therefore the focus should also be on those plantypes, which are not in scope of Change Management, especially products.*

---

In Default-PRO-DPM-IGRIP\_V5\_CMC only processes are versionable.

The property '**Is Configured**' describes the kind of effectivity the object obtains.

By Category   Alphabetical	
<input type="checkbox"/> <b>Basics</b>	
Base class type name	Process Component (ergocompprocessdefault)
Caption singular	Operation
Caption plural	
Plantype UUID	2793bcb4-ac27-4d19-9d95-23b00f5f3f4f
Description	
<input type="checkbox"/> <b>Display</b>	
Bitmap	process
Attributes per row	1
Dialog Type	Property Sheet
Special attribute	
Depends on values	No
<input type="checkbox"/> <b>Flags</b>	
Has versions	Yes
Is searchable	No
Is abstract	No
Has attachments	No
Is relevant	No
Own rights	Inherited from parent type
Display parent/child relations in dbeditor	Inherited from parent type
Is configured	Max. Configured
Defined by	Customer

Figure 1: Configured

We distinguish

- No (Not configured)
- Yes (Standard configuration mode)
- Max. Configured (maximum configuration mode)

## Notes to Configurable



### Note

*In Default-PRO-DPM-IGRIP\_V5\_CMC processes are Max. Configured and the plantype Manufacturing Assembly (non-versionable) applies the standard configuration.*

- If plantypes are not configurable, no effectivity is applied on them. In case of standard configuration we distinguish whether plantypes are versionable or not.
- For versionable plantypes, the newly created objects or checked-out versions obtain the effectivity from the modification statement.
- While checking-in a version, the effectivity of any version with a planning state equal to that is cut with the effectivity from the selected modification

statement. There is a special handling for non-versionable plantypes that are configurable. This requires a modification statement to be selected when these objects are created. Their effectivity is then again obtained from the selected modification statement.

- Maximum configuration means plantypes hold either “*always*” or “*never valid*” effectivity.
- When these objects are created or checked-out no effectivity is applied on them; when they are checked-in, the effectivity of any predecessor version, which has a planning state equal to that, is set to ‘*never valid*’. The maximum configured plantypes don’t support parallel versioning.

## How to customize Parent Child Relations

Only the parent child relations that have an owner, which is a versionable plantype, are in scope of Change Management.

Before the parent child relation is customized for Change Management, it should be first overwritten in the plantype set. This has to be done for each plantype which participates in the relation. It also has to be done for the so called *reverse* relation.

### Example to customize the parent child relation

An example on how to customize the parent child relation

⇒ “process\_implements\_requirements” is described here.

The relation is customized such that the operation is the owner.

First step is to overwrite the relation, which is described in the figures below. For this purpose, the ‘overwrite’ on the source plantype has to be selected in the context menu, and then the target plantype is selected.

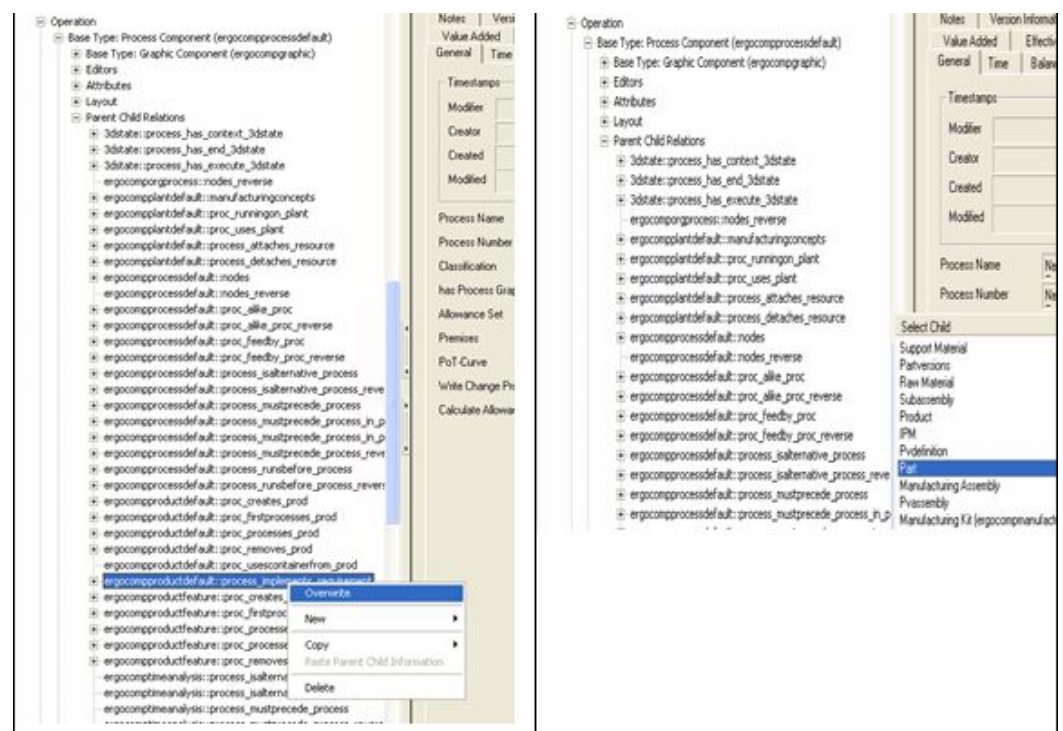


Figure 2: Example - process\_implements\_requirements



We now have to customize the relation for Change Management. Properties marked red are important. The first three of them turn off some functionalities that conflict with Change Management. We have to set them always to these values; otherwise Change Management won't work properly.

- The property '**Copy link to child (Change Management Controlled)**' has to be set to 'Yes' for all relations that are copied when a new version is created.
- For relations that are in scope of Change Management, 'Yes' is mandatory.

The property '**Owner type**' specifies how the owner plantype is related to the relation.

**We distinguish among**

- No Owner
  - Source Owner
  - Target Owner
  - Explicit Owner
  - Common Parent Owner
- 
- ⇒ If the relation doesn't have an owner, it's not in scope of Change Management.
  - ⇒ 'Source owner' and 'Target owner' declare source or target plantype respectively to be owner of the relation. This information can be deduced at the runtime, and the owner doesn't have to be specified explicitly, if the relation is created using API.
  - ⇒ Explicit owner can be an object of any valid plantype. DPE is not able to deduce this information, so the reference to the explicit owner has to be passed to API, when the relation is created.
  - ⇒ 'Common Parent owner' refers to the direct common parent of source and target according to the relation *nodes* (BOM). DPE is not able to deduce this information, so the reference to the common parent owner has to be passed to API, when the relation is created.

## Copy link to child

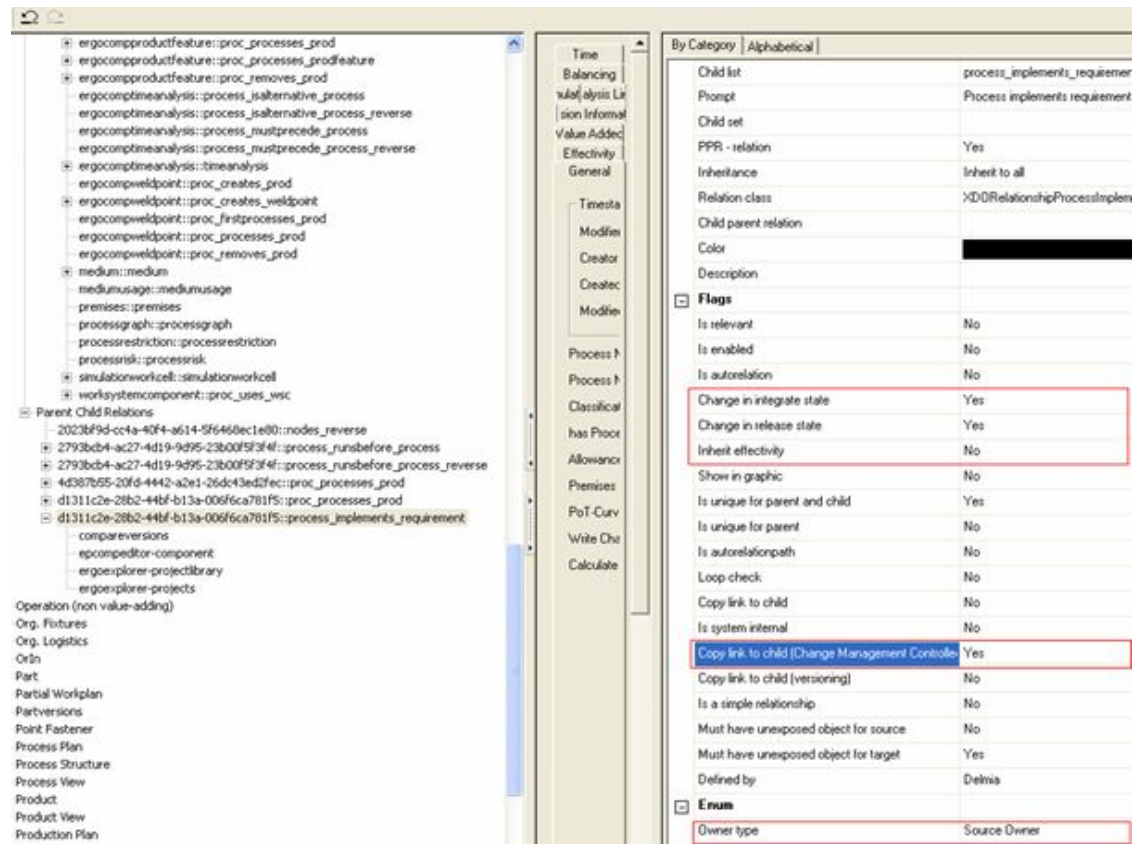


Figure 3: Example copy link to child

We have to repeat this procedure for the reverse relation.

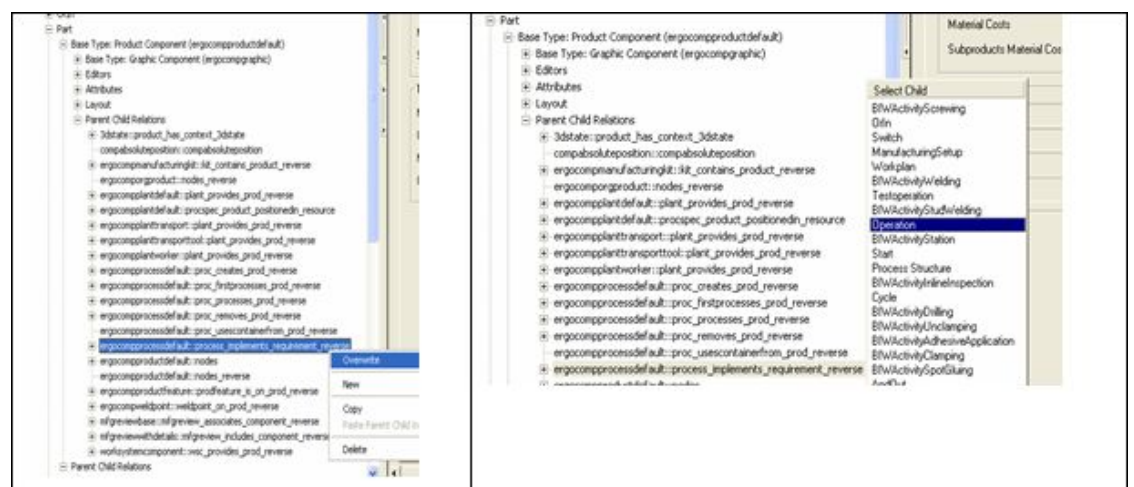


Figure 4: Example reverse relation

We use the same settings except for '**Owner type**' in the case that the owner is the source or target plantype. If so, we have to use the opposite value, as the figure below shows.

By Category   Alphabetical	
Child list	process_implements_requirement_reverse
Prompt	Requirement is implemented by process
Child set	
PPR - relation	PPR Reverse Relation
Inheritance	Inherit to all
Relation class	XDRRelationshipProcessImplementsRequirement
Child parent relation	
Color	
Description	
<b>Flags</b>	
Is relevant	No
Is enabled	Yes
Is autorelation	No
Change in integrate state	Yes
Change in release state	Yes
Inherit effectivity	No
Show in graphic	No
Is unique for parent and child	Yes
Is unique for parent	No
Is autorelationpath	No
Loop check	No
Copy link to child	No
Is system internal	No
Copy link to child (Change Management Controlled)	Yes
Copy link to child (versioning)	No
Is a simple relationship	No
Must have unexposed object for source	No
Must have unexposed object for target	Yes
Defined by	Delmia
<b>Enum</b>	
Owner type	Target Owner

Figure 5: Example Owner type

In Default-PRO-DPM-IGRIP\_V5\_CMC following relations are customized to be in scope of Change Management:

Parent Child Relation	Involved Plantypes	Owner
<i>nodes<sup>3</sup></i>	<i>Process View – Process Plan</i>	<i>No Owner</i>
<i>Nodes</i>	<i>Process Plan – Workplan</i>	<i>Target Owner</i>
<i>Nodes</i>	<i>Workplan – Operation</i>	<i>Target Owner</i>
<i>process_runsbefore_process</i>	<i>Operation – Operation</i>	<i>CommonParent Owner</i>
<i>proc_processes_prod</i>	<i>Operation – Part</i>	<i>Source Owner</i>
<i>proc_processes_prod</i>	<i>Operation – Subassembly</i>	<i>Source Owner</i>
<i>process_implements_requirement</i>	<i>Operation – Part</i>	<i>Source Owner</i>
<i>process_implements_requirement</i>	<i>Operation – Subassembly</i>	<i>Source Owner</i>
<i>process_implements_requirement</i>	<i>Workplan – Part</i>	<i>Source Owner</i>
<i>process_implements_requirement</i>	<i>Workplan – Subassembly</i>	<i>Source Owner</i>

Figure 6: Table of Change Management

**Limitation 1**

Mixed Mode (Interaction of objects created under Detached Mode and Non-Detached Mode) is not supported.

**Limitation 2**

Versioning of Ergoitems is not supported by CMC.

**Limitation**

Only non-versionable components (products and resources) can be transferred through the Bridge.

---

# Index of Figures

Figure 1:Configured	6
Figure 2: Example - process_implements_requirements .....	8
Figure 3: Example copy link to child.....	10
Figure 4: Example reverse relation .....	10
Figure 5: Example Owner type.....	11
Figure 6: Table of Change Management.....	11