



HOME

User Manual

DELMIA Process Engineer<sup>®</sup>

## Process Planning for Body-in-White



# Foreword

This manual provides an introduction to the basic operations and functions of the process planning tasks. This manual describes the process planning tasks that may be done in DPE as part of DELMIA's overall Body-in-White solution. This solution comprises both DPE and DELMIA V5 products.

While developing these functions we have made every effort to create a clearly organized, easy-to-understand program structure.

A user-friendly interface as well as a clear menu guide will enable you to quickly learn how to operate the program and to get familiar with its functions so that you can carry out your planning tasks in a quick and reliable way.

## No Liability or Guarantee

Our programs and manuals have been compiled with great care and to the best of our knowledge. They have also been tested in a production setting. However, we assume no liability and provide no guarantee that the software and related descriptions are free of error or are suitable for special purposes.

DELMIA assumes no liability for any damage that may arise from the use of this software. By using this software, the user acknowledges this exclusion from liability and shall hold DELMIA exempt from all claims.

## Copyright

The information in our documents may be copied and distributed for internal purposes provided it is done free of charge and the contents are not altered or distorted.

Any other form of usage, especially the sale on CD-ROM or in any other publication in whole or in part is only permitted after prior written consent by DELMIA.

Some parts of this software are owned by Unigraphics Solutions Inc. and are copyrighted © 2010. All rights reserved.

Some parts of this software are owned by combit® GmbH and are copyrighted. Report-/Print module List and Label® Version 8.0: Copyright combit® GmbH 1991-2010.

## Modifications

Moreover, DELMIA retains the right to make modifications and improvements to the product described in this manual at any time without prior notification.

DELMIA and the 3DS logo are registered trademarks of Dassault Systèmes or its subsidiaries, in the United States or other countries.

© 2001-2010 Dassault Systèmes - All rights reserved

# Table of Contents

<b>1. Introduction</b>	<b>1</b>
1.1 How to Use this Manual	1
1.2 Documentation Conventions and Symbols	1
1.3 New Functions in DPE Process Planning for Body-in-White	2
<b>2. Importing and Aligning Product Structures for MCM-Projects</b>	<b>3</b>
2.1 Workflow for Importing and Aligning Products	3
2.1.1 Important Terms	4
2.1.2 MCM-Project – Importing and Aligning Product Components	4
2.1.3 Body-in-White Planning – Using Imported Product Structures	5
2.2 Using Administrative Settings	6
2.2.1 Exceptions Prove the Rule	7
2.2.2 Changing Relation Settings	7
2.2.3 Table of Settings	10
2.2.4 Using the Settings for an MCM-Project	14
2.2.5 Scripts for Updating Products	18
<b>3. Extending the Fastener and Fastening Process Model</b>	<b>20</b>
3.1 Fastener Plantypes in Process Engineer	20
3.1.1 Extending the Fastener Model	20
3.1.2 Extending the Fastening Process Model	20
3.1.3 Examples of Customizing Plantypes	21
<b>List of Figures</b>	<b>26</b>
<b>List of Tables</b>	<b>27</b>
<b>Index</b>	<b>28</b>

# 1.Introduction

This manual explains how to use the Process Engineer process planning tasks required to use the DPE portion of DELMIA's Body-in-White solution.

## 1.1 How to Use this Manual

This manual enables you to get familiar with the operation and functions of the Process Engineer. This manual describes:

- Importing and aligning product structures for MCM projects

### Note

*When handling the process planning tasks, please also refer to the general introduction to Process Engineer in the General Introduction Manual.*



Click [General Introduction](#) to access the manual.

## 1.2 Documentation Conventions and Symbols

The symbols used in this manual are intended to provide you with keys to the contents in an immediately understandable manner.



This symbol is used to introduce key concepts that are covered in the sections immediately following this symbol. As a result, this symbol most frequently appears at the beginning of chapters or sections.



### Note

*This symbol is used to mark notes, which provide you with additional information you need to have for further work. You will either find the Note sign at the beginning of a chapter or in a particular text passage in the chapter. Texts bearing this sign are additionally marked with **Note**. The text is always in italics.*




### Caution

*This symbol indicates that the text that follows describes particular circumstances that you must avoid to avoid potential errors with the operation of the program or harm to data. You will either find the Caution sign at the beginning of a chapter or near a particular text passage in the chapter. Texts that are introduced by this sign are additionally marked with **Caution**. The text is always in italics.*

### Example

This symbol marks examples which serve to illustrate a certain situation.

- 1) This symbol marks the individual operational steps involved in a particular operating instruction. Operating instructions describe operational steps, for example, how to open a menu or execute a function.
- This symbol marks listed subjects. The symbol for listed subjects can be either used to structure a continuous text or to list main subject keywords.
- This symbol marks list inside a bulleted or numbered list.
-  This symbol marks cross reference information that is available in another manual.

### 1.2.1 New Functions in DPE Process Planning for Body-in-White

No new functionality has been added for this release.

## 2.Importing and Aligning Product Structures for MCM-Projects

Product structures from external database systems needed for the framework in V5 can be imported and aligned in E5 MCM-projects.

After processing in V5 has been completed with the assistance of one of the various program modules such as *Manufacturing System Definition* or *Assembly Process Planner*, the processed project can be saved in the Manufacturing Hub.



### Note

*Products are imported and aligned for DPE MCM-Projects only.*

A product can be imported and/or aligned manually in the MCM-Project (versions are created directly in the project), by way of script or with the assistance of software using an *Application Programming Interface* (API)-interface.

The MCM-Project must be configured first, to import and align product structures from external databases.

To do this, the settings described in the following are required:

This chapter consists of four main subjects:

- [Workflow for the import and alignment of products](#)
- [Using administrative settings](#)
- [Using the settings for MCM-Projects](#)
- [Script for product update](#)



### Notes

*The importing and/or aligning of products and the required setup should only be carried out by an administrator or user with equal rights.*

*To import and/or align a product on an MCM-project, a task with a corresponding area of validity must be selected in an MCM-project.*

The settings described in the following chapters support the corresponding chapter in both the [Administrative Tasks Manual](#) and the [Manufacturing Change Management Manual](#).

To carry out a product import and/or alignment, it is important to familiarize oneself with the configuration and change management basics. The following chapters will provide basic information on importing and aligning products.

### 2.1 Workflow for Importing and Aligning Products

Since each operation can be unique and different, this chapter gives examples of the behavior of product components during product import and alignment for an MCM-project by using a fictitious workflow. This workflow includes:

- Creating new product components after importing the product structure.

- Checking the validity of the versions following product alignment in the case of deleted, modified or newly added product components.

### 2.1.1 Important Terms

**DROP:** A DROP is used in V5 for framework planning. A DROP is the description of a defined and fixed state of planning.

Each DROP is identified by a defined area of validity.

In an MCM-project, the task and area of validity must be determined by the rightful user or administrator.

In an MCM-project, the imported or aligned product structures are opened and filtered through a DROP in V5.

**EBOM:** An EBOM is a product structure based on the design BOM.

### 2.1.2 MCM-Project – Importing and Aligning Product Components

The workflow shown in [Figure 1](#) is designed in such a way that the first step is importing a product structure (**DROP1**) and the second is to align this product structure (**DROP2**). During this alignment process, product components are being deleted, modified, and created anew.

- The import and alignment also differs depending on the selected area of validity.
- The following example shows validity for each version before and after the alignment process.
- The working status of both DROPS is set to **release**.

The following sections describe the settings needed to import and/or align products.

#### 2.1.2.1 First Step – Import Product Structure

**DROP1** shows the newly imported product structure for the **R(1-oo)** area of validity (blue box).

Following the import, the product components of version (V1) listed below are created: P0, P1, P8, and P9.

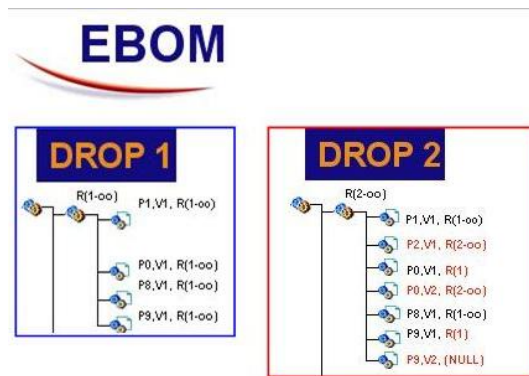


Figure 1: Example of Product Import and Alignment

### 2.1.2.2 Second Step – Product Structure is Aligned

**DROP2** shows the product alignment for the **R(2-oo)** area of validity (red box).

Resulting changes for products components following the product alignment:

- **P0** was changed: The area of validity for version 1 (V1) has been restricted and is operational in **R(1)** only. The Version (V2) in the **P0** is valid for the area **R(2-oo)**.
- **P1** and **P8** have not been changed and continue to be **valid** for the area **R(1-oo)** of version (V1).
- **P2** has been created and is valid in the area of **R(2-oo)** of version (V1), according to the area of validity selected for DROP2.
- **P9** was deleted. P9 version (V2) is invalid. The valid area for the version (V1) of P9 has been restricted to the selected area of validity for DROP2. P9 version (V1) has been restricted and is operational for the **R(1)** area of validity **only**.

### 2.1.3 Body-in-White Planning – Using Imported Product Structures

The actual planning for the product structures that have been imported and/or aligned from external databases, is carried out in the V5 body-in-white planning products, which include:

- DPM Fastening Process Planner
- Assembly Process Planner
- Manufacturing System Definition

To describe the entire body-in-white planning process is not the purpose of this section: A complete description of the body-in-white planning can be found in corresponding V5 user manuals.

This chapter highlights some of the important considerations you may want to take into account during the planning of MCM-projects.

#### 2.1.3.1 Opening an MCM-project in V5

A variety of different DROPS can be opened and edited in V5.



### 2.1.3.2 Relationships

An important aspect of planning body-in-white solutions is considering the relations between product structures and processes or resources. The process and resource relationships are planned in V5 and saved in the Manufacturing Hub.

#### Relationships to Assembly-Processes

- If relations exist between products and assembly-processes, they are automatically updated in V5 to the valid version.
- If relations exist between products and resources, they can be updated to the latest version manually in V5, following a product alignment.

## 2.2 Using Administrative Settings

The properties (**Is configured** and **Has versions**) of all types to be considered during the import or alignment of products in MCM-projects, must be set to **Yes**. The configuration can be done on type and plantype levels.

To carry out an effective import and/or alignment of a product for an MCM-project, note the following:

- It is advantageous to set at the least, the properties **Is configured** and **Has versions** on **Yes** for the types **ergocompproductdefault** and **ergocom-productfeature**, to perform a product import and/or alignment.
- For instance, if further types are to be used for product import and/or alignment, such as the **ergocompproductfeatureclampingpoint type**, the **Is configured** and **Has versions** properties of these types must be set to **Yes** to be recognized.

Proceed as Follows:

The following example shows the basic steps to change the properties:

- 1) First open the Configuration Tool to change the properties of a **Is configured** and **Has versions** type to **Yes**.
- 2) Then select the types and in the properties dialog, change the value of both attributes to **Yes**.
- 3) Save the change.

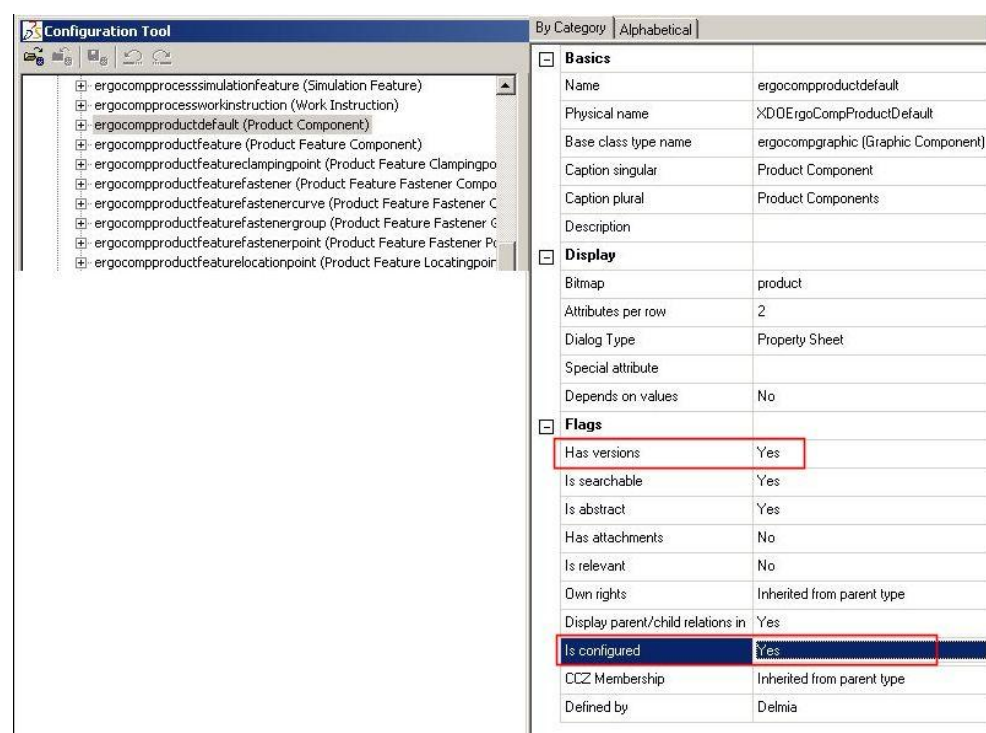


Figure 2: Attribute is Configured – Set Value to Yes

## 2.2.1 Exceptions Prove the Rule

As always exceptions prove the rule. In general, the attribute **Has Versions** must usually be set to **Yes** for types and plantypes. However, for the following plantypes:

Assembly Operation, Manufacturing Assembly, Spot Weld, and Line you must set the attribute **Has Versions** on **No**.

## 2.2.2 Changing Relation Settings

To carry out the product update, the relation settings for the following types can be changed in the Configuration Tool:

ergocompproductdefault, ergocompproduct, ergocompproductfeature, ergocompproductdefault, and ergocompproductmanufacturingassembly

What setting changes are needed for individual relations are listed in the tables in the [Ergocompproductdefault Table Settings](#) section.

### Setting Changes in the Configuration Tool

The **ergocompproductdefault::nodes** and **ergocompproduct::nodes\_reverse** relations can be used to show the basic steps of changing the relation settings in the Configuration Tool.

Both relations can be found in the directory **ergocompproductdefault > parent-child relation** of the Configuration Tools.



For more information about configuration, please refer to the [Administrative Tasks Manual](#).

### Example – ergocompproductdefault::nodes

- 1) Open the context menu.

#### Example:

Change settings for the **ergocompproductdefault::nodes** relation.

- 2) Select the **ergocompproductdefault** type and open the **parent-child relation** directory.
- 3) Select the **ergocompproductdefault::nodes** relation from the **parent-child relation** directory.



**Figure 3: Select the ergocompproductdefault::nodes Relation**

- 4) Select the properties for the **ergocompproductdefault::nodes** relation and change the settings accordingly.
- The picture shows the settings that need to be changed. The settings are highlighted in red.

By Category	Alphabetical
Change in integrate state	Yes
Change in release state	Yes
Child list	nodes
Child parent relation	
Child set	
Child type	ergocompproductdefault
Color	
Consider for Activation by Link	No
Copy link to child	No
Copy link to child (Change Management Controlled)	Yes
Copy link to child (versioning)	No
Defined by	Delmia
Description	
Exclude from Manufacturing Change Management ar	No
Inherit effectivity	No
Inheritance	Inherit to all
Is a simple relationship	No
Is autorelation	No
Is autorelationpath	No
Is configured	No
Is enabled	Yes
Is relevant	No
Is system internal	No
Is unique for parent	No
Is unique for parent and child	No
Loopcheck group	No loopcheck
Must have unexposed object for source	No
Must have unexposed object for target	No
Owner type	Source Owner
Parent type	ergocompproductdefault

Figure 4: Settings - Change the ergocompproductdefault::nodes Relation

**Example:**

Change the settings for the ergocompproduct::nodes\_reverse relation.

**Example - ergocompproduct::nodes\_reverse**

- 1) Repeat the foregoing action by once again selecting **ergocompproductdefault** type and open the **parent-child relation** directory.
- 2) Select the **ergocompproduct::nodes\_reverse** relation from the parent-child relation directory.

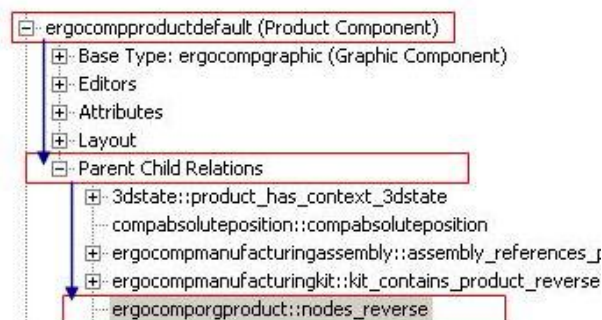


Figure 5: Select the ergocompproduct::nodes\_reverse Relation

- 3) Select the properties for the **ergocomporgproduct::nodes\_reverse** relation and change the settings accordingly.
- The image shows the settings that need to be changed. The settings can easily be recognized by the red highlighting.

By Category	Alphabetical
Change in integrate state	Yes
Change in release state	Yes
Child list	nodes_reverse
Child parent relation	
Child set	
Child type	ergocompproductdefault
Color	
Consider for Activation by Link	No
Copy link to child	No
Copy link to child (Change Management Controlled)	Yes
Copy link to child (versioning)	No
Defined by	Delmia
Description	
Exclude from Manufacturing Change Management ar	No
Inherit effectivity	No
Inheritance	Inherit to all
Is a simple relationship	No
Is autorelation	No
Is autorelationpath	No
Is configured	No
Is enabled	Yes
Is relevant	No
Is system internal	No
Is unique for parent	No
Is unique for parent and child	No
Loopcheck group	No loopcheck
Must have unexposed object for source	No
Must have unexposed object for target	No
Owner type	Target Owner
Parent type	ergocompproductdefault

Figure 6: Settings - Change the ergocomporgproduct::nodes\_reverse Relation

### 2.2.3 Ergocompproductdefault Table Settings

The relations are listed separately for each type. Along with the settings described in the table, all the property values of the relations such as **Is configured** and **Has versions** have been set to **Yes**. Set the corresponding values for the particular types that are to be used for product import and/or alignment.

- Relations, options, and values are all listed separately in individual columns.
- All relations have the same options (column options), but the values (column values) for each are different.
- The two examples on the preceding page show how the values can be changed (*Please refer to the [Changing Relation Settings](#)*).

Table 1: Settings for ergocompporductdefault

Relation	Options	Value
<b>Type: ergocompporductdefault</b>		
Ergocompporduct: nodes_reverse	Change in integrate state	Yes
	Change in release state	Yes
	Copy link to the child object (change management)	Yes
	Owner type	No Owner
Ergocompporductdefault: nodes	Change in integrate state	Yes
	Change in release state	Yes
	Copy link to the child object (change management)	Yes
	Owner type	Source Owner
Ergocompporductdefault: nodes_reverse	Change in integrate state	Yes
	Change in release state	Yes
	Copy link to the child object (change management)	Yes
	Owner type	Target Owner
Ergocompprocesdefault: proc_firstprocesses_prod_reverse	Change in integrate state	Yes
	Change in release state	Yes
	Copy links to the child object (change management).	Yes
	Owner type	No Owner
Ergocompmanufacturingassembly: assembly_references_product_reverse	Change in integrate state	Yes
	Change in release state	Yes
	Copy link to the child object (change management)	Yes
	Owner type	Target Owner
Ergocompproductfeature: prodfeature_is_on_prod_reverse	Change in integrate state	Yes

Relation	Options	Value
	Change in release state	Yes
	Copy link to the child object (change management)	Yes
	Owner type	Target Owner
Ergocompprocessdefault: proc_processes_prod_reverse	Change in integrate state	Yes
	Change in release state	Yes
	Copy link to the child object (change management)	On Demand
	Owner type	No Owner

Table 2: Settings for ergocompproduct and ergocompproductfeature

Relation	Options	Value
Type: ergocompproduct		
Ergocompproductdefault: nodes	Change in integrate state	Yes
	change in release state	Yes
	Copy link to the child object (change management)	Yes
	Owner type	No Owner
Type: ergocompproductfeature		
Ergocompproductdefault: prodfeature_is_on_prod	change in integrate state	Yes
	change in release state	Yes
	Copy link to the child object (change management)	Yes
	Owner type	SourceOwner
Ergocompprocessdefault: proc_processes_prod_reverse	Change in integrate state	Yes
	Change in release state	Yes



Relation	Options	Value
	Copy link to the child object (change management)	On Demand
	Owner type	No Owner
Ergocompprocessdefault: proc_processes_prodfeature_reverse	Change in integrate state	Yes
	Change in release state	Yes
	Copy link to the child object (change management)	Yes
	Owner type	No Owner

Table 3: Settings for ergocompprocessdefault

Relation	Options	Value
Type: ergocompprocessdefault		
Ergocompproductdefault: proc_firstprocesses_prod	Change in integrate state	Yes
	Change in release state	Yes
	Copy link to the child object (change management)	Yes
	Owner type	No Owner
Ergocompproductdefault: proc_processes_prod	Change in integrate state	Yes
	Change in release state	Yes
	Copy link to the child object (change management)	On Demand
	Owner type	No Owner
Ergocompproductfeature: proc_processes_prod	Change in integrate state	Yes
	Change in release state	Yes
	Copy link to the child object (change management)	On Demand
	Owner type	No Owner
Ergocompproductfeature: proc_processes_prodfeature	Change in integrate state	Yes
	Change in release state	Yes



Relation	Options	Value
	Copy link to the child object (change management)	Yes
	Owner type	No Owner

**Table 4: Settings for ergocompmanufacturingassembly and ergocompproductfeature fastener**

Relation	Options	Value
Type: ergocompmanufacturingassembly		
Ergocompproductdefault: assembly_references_product	Change in integrate state	Yes
	Change in release state	Yes
	Copy link to the child object (change management)	Yes
	Owner type	Source Owner
Type: ergocompproductfeaturefastener		
Ergocompproductfeaturefastener: fastener_overrides_fastener	Change in integrate state	Yes
	Change in release state	Yes
	Copy link to the child object (change management)	Yes
	Owner type	Source Owner
Ergocompproductfeaturefastener: fastener_overrides_fastener_reverse	Change in integrate state	Yes
	Change in release state	Yes
	Copy link to the child object (change management)	Yes
	Owner type	Target Owner

## 2.2.4 Using the Settings for an MCM-Project

This chapter discusses the settings that are imperative in carrying out the import and/or alignment of products from external database systems on an MCM-project.



For more information on configuration, please refer to the [Manufacturing Change Management Manual](#).

- 1) To mark a project as an MCM-project check for the **MCM-Project** entry when creating a new MCM-project.

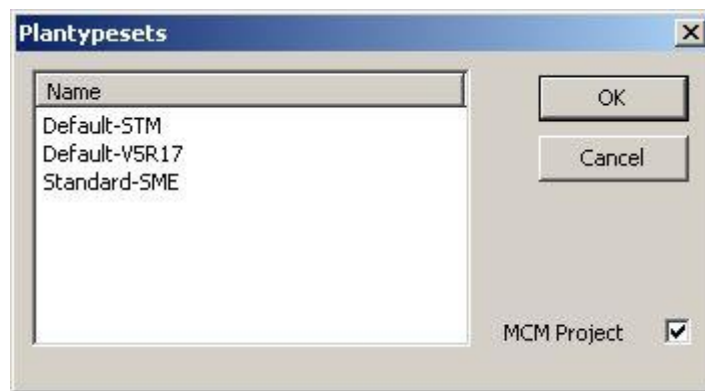


Figure 7: Mark MCM Project

### 2.2.4.1 Filter Tab in the MCM-Project

To create, edit, or delete versionable PPR components without having to select a task (action) and an area of validity (mode statement), activate the **MCM Out of Scope Creation of Versionable Objects** option.

This option is required to edit PPR components without having to select a task (action) and an area of validity (mode statement) following the import and/or alignment of a product that needed a task and a valid scope.

- 1) These setting changes can be carried out in the properties dialog of the project by way of the Filter tab.

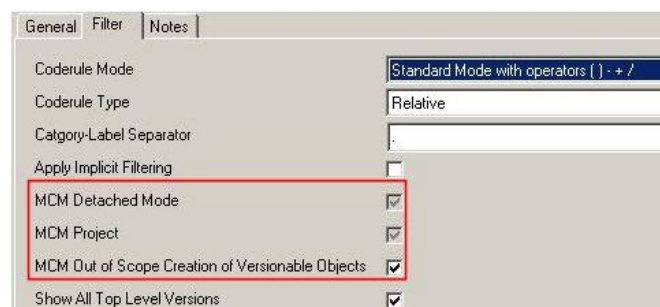


Figure 8: Properties Menu – Filter Tab

- 2) The project must be closed in order to activate the **MCM Detached Mode** option.
- 3) Select the **Regular Project <- -> MCM-Project** entry from the **Tools** menu.

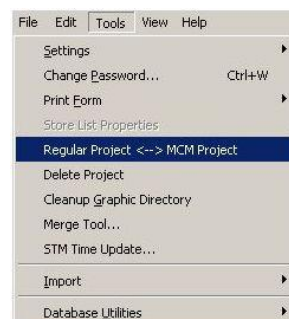


Figure 9: Tools Menu

- 4) To create tasks (actions) and an area of validity (mode statements) in the MCM-project activate the **MCM Detached Mode** option in the dialog.
- 5) Confirm the entries with **OK**.

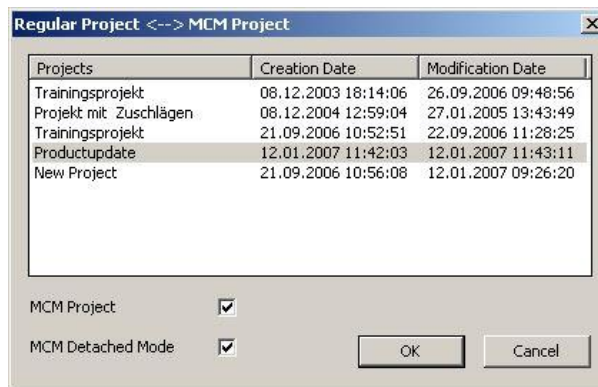


Figure 10: Activate MCM Detached Mode

### 2.2.4.2 Planning Status for the Product Import

At least two planning states are required to edit the imported and/or aligned products:

#### Editing the Planning Status before Importing

In the case of MCM-projects the planning state is controlled by the corresponding action state.

- The selected task (action) must be set to the action state of **working** prior to importing and/or aligning of products. It is under these circumstances only, that product structures from external databases can be successfully imported and/or aligned. These product components also receive the **working** planning status after import and/or alignment.

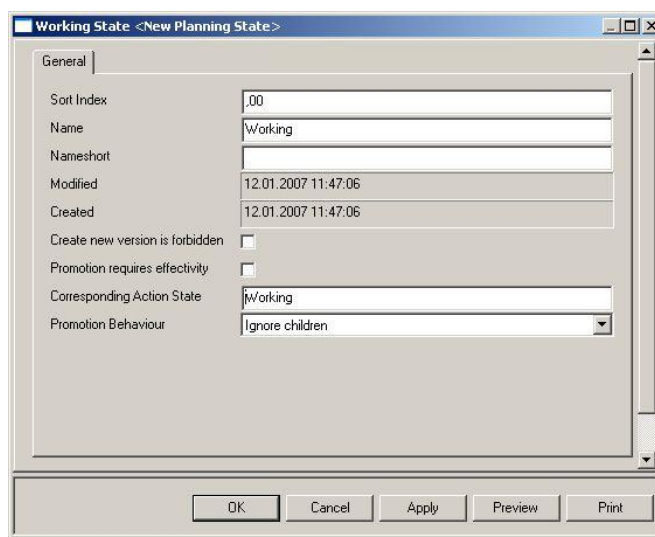


Figure 11: Edit Planning State

#### Planning Status – Released

To complete the editing after product import and/or alignment, raise the action state to **Released**. Once the action state has been raised, all product components that are assigned to this task (action) automatically receive the *Released* planning state.



#### Note

*Only product components with the released planning status are available for further use in V5 body-in-white planning products.*

The screenshot shows a software dialog box titled "Released State <New Planning State>". It has a "General" tab selected. The fields and their values are as follows:

Field	Value
Sort Index	.00
Name	Released
Nameshort	
Modified	12.01.2007 11:48:19
Created	12.01.2007 11:48:19
Create new version is forbidden	<input type="checkbox"/>
Promotion requires effectivity	<input type="checkbox"/>
Corresponding Action State	Released
Promotion Behaviour	Ignore children

At the bottom of the dialog, there are five buttons: OK, Cancel, Apply, Preview, and Print.

Figure 12: Planning Status - Released

### 2.2.4.3 Tasks (Actions) for the Import and/or Alignment of Products

To carry out a product import and/or alignment from an external database, a task (action) with an area of validity (mode statement) must have been created in the MCM-Project.

- For product import and/or alignment the task (action) selected, must have an area of validity (mode statement). Or the API-interface can also be used to determine the scope.
- All of the product components that have been edited by way of the product import and/or alignment, receive the same mode statement (area of validity) as the selected action (task) – e.g. #DROP(R-(1-00) area of validity.
- The selected action must be set to **working** the action state prior to importing and/or aligning of products.
- After alignment and editing is complete, raise the action state to **Released**.
- Once the action state has been elevated, all product components that have edited with the assistance of this action, automatically receive the **Released** planning state.

The screenshot shows a Windows-style dialog box titled "Action <New Action Proxy>". It has a "General" tab selected. Inside the tab, there are several input fields and checkboxes. The "Name" field contains "A1". The "Action Type" field is empty. The "Is CMC Detached Mode" checkbox is unchecked. The "Group Number" field is empty. The "Action State" field contains "Released". The "Is Usable" checkbox is checked. The "Enforce Configuration" checkbox is unchecked. At the bottom of the dialog, there are five buttons: "OK", "Cancel", "Apply", "Preview", and "Print".

**Figure 13: Work Status Release**



For more detailed information on subjects such as creating actions and mode statements (areas of validity), changing the planning and action statuses, please refer to the [Manufacturing Change Management Manual](#).

## 2.2.5 Scripts for Updating Products

These two scripts enable importing and/or alignment of a product. Both of these scripts present the basic pattern structure, to be modified and used according to the product structure to be imported and/or aligned.

### 2.2.5.1 Script - Creating a Product Structure

With the assistance of this script, a new product structure can be created.

```
Sub main(id)
    product_view_id = CreateComp(id, "", "Product View", "Product View ")
    production_view_id = CreateComp(product_view_id, "nodes", "Production View", "Production View")
    subassembly_id = CreateComp(production_view_id, "nodes", "Subassembly", "DN 1")
    CreateComp(subassembly_id, "nodes", "Part", "P0")
    CreateComp(subassembly_id, "nodes", "Part", "P1")
    CreateComp(subassembly_id, "nodes", "Part", "P2")
    CreateComp(subassembly_id, "nodes", "Part", "P3")
End Sub

Function CreateComp(parent_id, childlistname, childname, namecomp)
    comp_sci = Data.CreateComponent(parent_id, childlistname, childname)
    data.SetAttributebyId comp_sci, "name", namecomp
    comp = Data.GetAttributebyId (comp_sci, "ergocompbase")
    Data.SetAttributebyId comp, "name", namecomp
    CreateComp = comp
End Function
```

### 2.2.5.2 Script – Updating the Product Structure

This script enables an existing product structure to be updated.

```

Sub main(id)
    subassembly_id = GetLatestVersion("DN1", "name", id)
    p0_id = GetLatestVersion("P0", "name", id)
    p0_id_v2 = Version.Create(p0_id)
    p3_id = GetLatestVersion("P3", "name", id)
    p3_id_v2 = Version.Create(p3_id)
    Data.DeleteComponent(p3_id_v2, 0)
    CreateComp(subassembly_id, "nodes", "Part", "P4")
End Sub

Function GetLatestVersion(strName, criterion, id)
    project_id = Data.getAttributeById(id, "ergoproject")
    comp_id = GetComponentID(strName, criterion, project_id)
    If comp_id <> "" Then
        comp_id = GetLatest(comp_id)
    End If
    GetLatestVersion = comp_id
End Function

Function GetComponentID(strName, criterion ,project_id)
    Call Query.ResetSearch
    Call Query.SetQuery("ergocompprocessdefault", "ergoproject", "=", project_id)
    Call Query.SetConcatenator("AND")
    Call Query.SetQuery("ergocompbase", criterion, "=", strName)
    nrResult = Query.GetResultCount
    comp_id = ""
    If nrResult > 0 Then
        result_id = Query.GetFirstResult
        If result_id <> "" Then
            comp_id = result_id
        End If
    End If
    GetComponentID = comp_id
End Function

Function CreateComp(parent_id, childlistname, childname, namecomp)
    comp_sci = Data.CreateComponent(parent_id, childlistname, childname)
    data.SetAttributebyId comp_sci, "name", namecomp
    comp = Data.GetAttributebyId (comp_sci, "ergocompbase")
    Data.SetAttributebyId comp, "name", namecomp
    CreateComp = comp
End function

```

## 3. Extending the Fastener and Fastening Process Model

DPM Fastening Process Planner (in V5) provides planners with the ability to model new types of fasteners and fastening processes. This enables planners to meet the requirement of different fastening technologies being used in their fastening process plans. In order to exploit fully these V5 capabilities, you need to use fastener plantypes in DPE.

### 3.1 Fastener Plantypes in Process Engineer

There are two main categories of fastener plantypes:

#### Point Fastener Plantypes

- Sealant Point, Spot Weld, Stud, Glue Drop, Rivet, Screw, Clinch, Drill, and Adhesive Point

#### Curve Fastener Plantypes

- Adhesive Curve, Arc Welding, Sealant Curve, and Glue Bead

#### 3.1.1 Extending the Fastener Model

You can create new fastener plantype by deriving the new fastener types as follows:

- “product feature fastener point” for point type fasteners
- “product feature fastener curve” for curve/line type fasteners

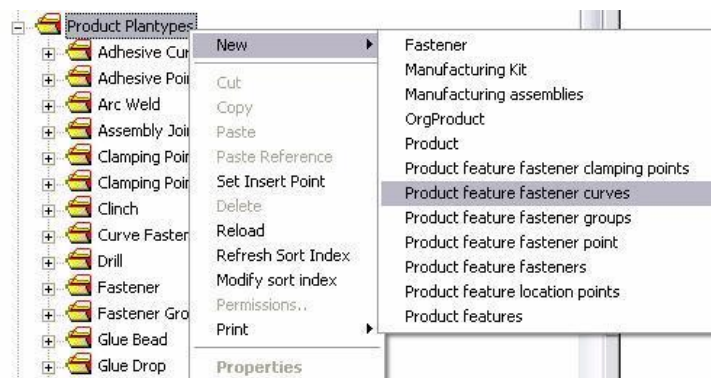


Figure 14: Deriving a New Fastener Plantype

#### 3.1.2 Extending the Fastening Process Model

You can create a new fastening process plantype by deriving from a process plantype.

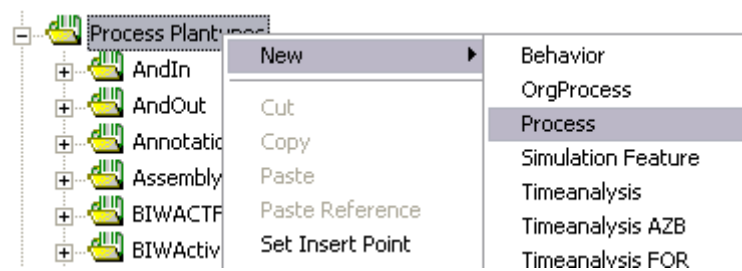


Figure 15: Deriving New Fastening Process Plantype

### 3.1.2.1 Customizing a New Fastening Process Plantype for DPM Fastening Process Planner

Newly created process plantype should be derived from following base classes:

- **BIWActivityWelding**  
Fastening processes that use fasteners of the point type.
- **BIWActivityCurveFastening**  
Fastening processes that use fasteners of the curve or line type.

### 3.1.2.2 Customizing to Bind the Extended Fastener to the Extended Fastening Process

V5 uses an XML document to determine the fastening process and the fastening relations for the fastening purpose. This XML file (ActivityFastenerRelationConguration.xml) is located in:

*ProgramInstallationDirectory\intel\_a\startup\*

This file also determines which process should be created for the fastener or validates the existing process for fasteners.

The contents of this XML file is as follows:

```
<ActivityFastenerRelation ActivityType = "Activity Plantype">
  <FastenerType>Fastener Plantype</FastenerType>
</ActivityFastenerRelation>
```

## 3.1.3 Examples of Customizing Plantypes

### 3.1.3.1 Creating a New Plantype of a Point Fastener Type

1) Create new point fastener plantype using relation

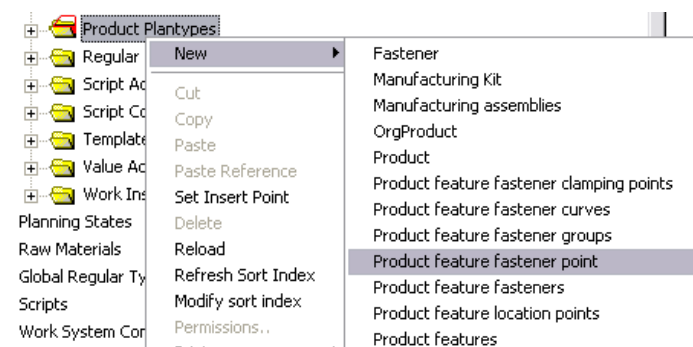


Figure 16: Creating New Point Fastener Plantype



**Figure 17: Product Feature Fastener Point**

2) Create new fastening process plantype for a Projection Welding fastener:

**Figure 18: Point Fastener Activity**

3) Derive a Projection Welding Activity from a pre-defined base class. Since it is going to be used for point type of fastener, it should be derived from **BIWActivityWelding**

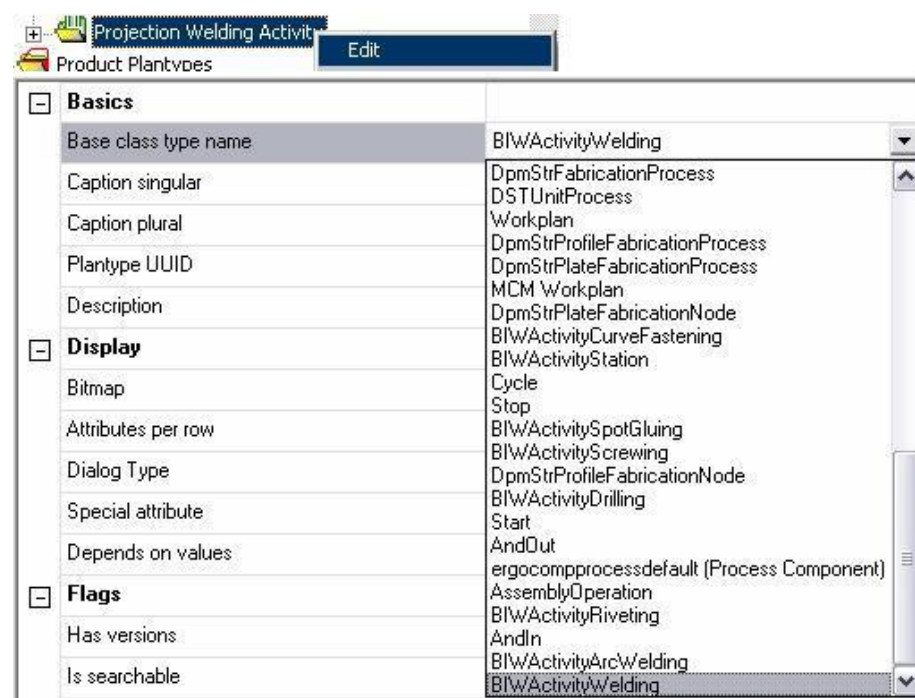


Figure 19: BIWActivityWelding

- 4) Modify the ActivityFastenerRelationConguration.xml file; add the lines below to the front of the file.

```
<ActivityFastenerRelation ActivityType = "Projection Welding Activity">
  <FastenerType>Projection Welding</FastenerType>
</ActivityFastenerRelation>
```

If the fastening process needs to be instantiated when the Projection Welding fastener is assigned to a resource, then one more customization is also needed. The new plantype should be drag/dropped under the Behavior plantype:



Figure 20: Dragging New Plantype

### 3.1.3.2 Creating a New Plantype of a Curve Fastener Type

- 1) Create a new curve fastener plantype using relation.

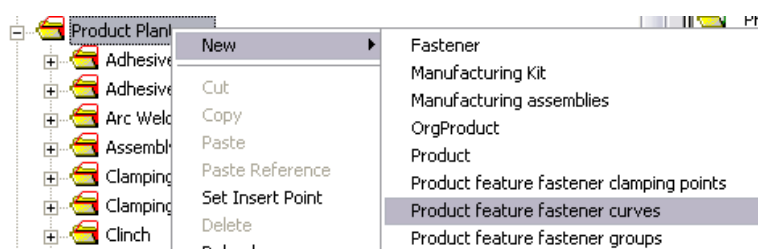


Figure 21: Creating New Curve Fastener

**Figure 22: Product Fastener Curve**

2) Create a new fastening process plantype for the MIG Welding fastener:

**Figure 23: Fastening Process Plantype**

3) Derive the MIG Welding activity from a pre-defined base class.  
Since it is going to be used for curve fastener it should be derived from **BIWActivityCurveFastening**.

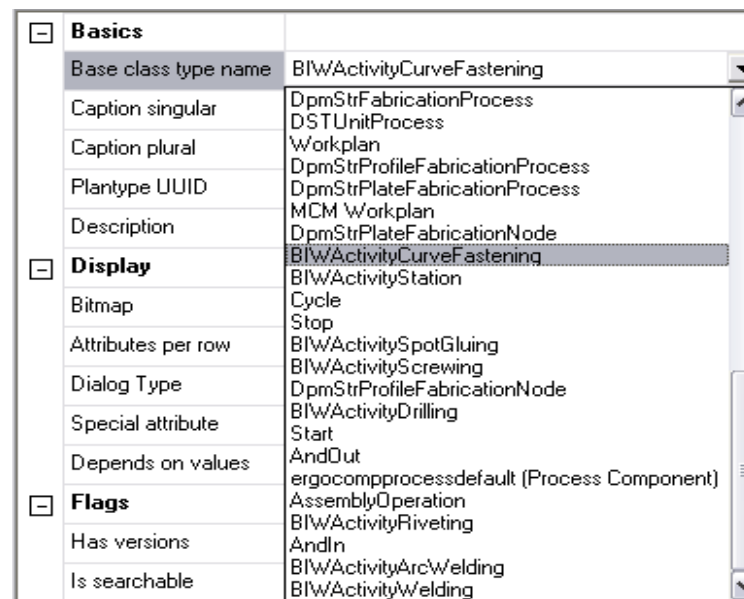


Figure 24: BIWActivityCurveFastening

- 4) Modify the ActivityFastenerRelationConguration.xml file; add the lines below to the front of the file.

```
<ActivityFastenerRelation ActivityType = "MIG Welding Activity">
  <FastenerType> MIG Welding </FastenerType>
</ActivityFastenerRelation>
```

If the fastening process needs to be instantiated when above fastener is assigned to a resource, then following customization is also needed. The new process plantype should be drag/dropped under Behavior plantype:



Figure 25: Dragging New Process Plantype

# List of Figures

Figure 1: Example of Product Import and Alignment .....	5
Figure 2: Attribute is Configured – Set Value to Yes .....	7
Figure 3: Select the ergocompproductdefault::nodes Relation .....	8
Figure 4: Settings - Change the ergocompproductdefault::nodes Relation .....	9
Figure 5: Select the ergocomporproduct::nodes_reverse Relation .....	9
Figure 6: Settings - Change the ergocomporproductt::nodes_reverse Relation .....	10
Figure 7: Mark MCM Project .....	15
Figure 8: Properties Menu – Filter Tab .....	15
Figure 9: Tools Menu .....	15
Figure 10: Activate MCM Detached Mode .....	16
Figure 11: Edit Planning State .....	16
Figure 12: Planning Status - Released .....	17
Figure 13: Work Status Release .....	18
Figure 14: Deriving a New Fastener Plantype .....	20
Figure 15: Deriving New Fastening Process Plantype .....	21
Figure 16: Creating New Point Fastener Plantype .....	21
Figure 17: Product Feature Fastener Point .....	22
Figure 18: Point Fastener Activity .....	22
Figure 19: BIWActivityWelding .....	23
Figure 20: Dragging New Plantype .....	23
Figure 21: Creating New Curve Fastener .....	23
Figure 22: Product Fastener Curve .....	24
Figure 23: Fastening Process Plantype .....	24
Figure 24: BIWActivityCurveFastening .....	25
Figure 25: Dragging New Process Plantype .....	25

## List of Tables

Table 1: Settings for ergocompporductdefault.....	11
Table 2: Settings for ergocomporgproduct and ergocompproductfeature .....	12
Table 3: Settings for ergocompprocessdefault .....	13
Table 4: Settings for ergocompmanufacturingassembly and ergocompproductfeature fastener.....	14

# Index

## G

### General Information

Introduction.....	3
Main Subjects .....	3

## N

Nonliability .....	ii
--------------------	----

## P

### Plantypes

Base Classes.....	22
Create New.....	21
Curve Fastener .....	21
Point Fastener .....	21

## S

### Scripts

New Product Structure.....	18
Updating the Product Structure .....	19

### Settings

Administrative Settings .....	6
Planning and Working State.....	16
Relations: .....	7
Table.....	10

## W

### Workflow

Definition of Terms .....	3
Examples: .....	4