



HOME

User Manual

DELMIA Process Engineer®

ENOVIA VPM V5 – DELMIA Process Engineer Integration DELMIA DELMIA – ENOVIA VPM V5



Foreword

This manual describes how to set up the DELMIA – ENOVIA VPM V5 Connection. It focuses on DELMIA Process Engineer Version 5.19, but also explains important differences between the settings for this version and those prior to and including 5.11.

To set up and customize the DELMIA – ENOVIA VPM V5 Connection, the registry keys must be altered. In addition, other information within a configuration file must also be changed.

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 © 2002. 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 1991-2001.

Copyright © 1999, International Business Machines Corporation and others. All Rights Reserved.

Copyright © 1999-2000, The Apache Software Foundation. All rights reserved.

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.

© 2001-2009 Dassault Systèmes - All rights reserved

Table of Contents

| | |
|--|-----------|
| ENOVIA VPM V5 – | 1 |
| DELMIA Process Engineer Integration | 1 |
| DELMIA DELMIA – ENOVIA VPM V5 | 1 |
| Foreword | ii |
| Table of Contents | iv |
| 1. Introduction | 1 |
| 1.1 How to Use this Manual | 1 |
| 1.2 Documentation Conventions and Symbols | 2 |
| 1.3 New Functions in DELMIA – ENOVIA VPM V5 Connection | 3 |
| 2. General | 4 |
| 2.1 Abbreviations | 5 |
| 3. Prerequisites | 6 |
| 3.1 Server Side | 6 |
| 3.2 Client Side | 6 |
| 3.3 Client and Server Sides | 6 |
| 4. Launching Applications | 7 |
| 4.1 Start PPRLoader in Batch Mode | 7 |
| 4.2 Export from ENOVIA VPM V5 | 12 |
| 5. Altering Settings | 21 |
| 5.1 General Mapping Information | 21 |
| 5.2 CAA Application Authentication | 23 |
| 5.3 Customization Settings for Planning Context | 31 |
| 5.4 PPRLoader Configuration Settings | 33 |
| 5.5 PPRLoader Customization Settings | 38 |
| 5.6 ProductDataGen Configuration Settings | 42 |
| 6. Reference Information | 43 |
| 6.1 PPRDaemon | 43 |
| 6.2 Product Structure Mapping | 45 |
| 6.3 Extended Part Filtering | 51 |
| 6.4 Multi Instancing | 54 |

| | | |
|------|---|-----|
| 6.5 | Resource Support | 58 |
| 6.6 | Computing Manufacturing Context | 58 |
| 6.7 | Customized/Extended Attribute Mapping | 62 |
| 6.8 | Substitute Parts | 68 |
| 6.9 | Support Management of In Work Design | 69 |
| 6.10 | Document-Object- and Document-Attachment Mapping | 71 |
| 6.11 | Transfer of Multi Value Attributes | 72 |
| 6.12 | Update Protocol and Management | 74 |
| 6.13 | Initial Import | 82 |
| 6.14 | Partial Product Data Transfer | 84 |
| 6.15 | Block Data Transfer | 88 |
| 6.16 | Incremental Update | 90 |
| 6.17 | Support of Product Specifications and COPS | 93 |
| 6.18 | Support Best-So-Far (BSF) Strategy in ENOVIA VPM V5 | 101 |
| 6.19 | Support Export of Multiple PRCs in One Call | 102 |
| 6.20 | Improved Support of Product Classes Export | 104 |
| 6.21 | Support Consumption of ENOVIA Work Packages | 124 |
| 6.22 | Support Product Maintenance Lifecycle | 129 |
| 6.23 | ENOVIA Filter for EAR Data | 134 |
| 6.24 | Customer Specific File Markings | 141 |
| 6.25 | Specification Alias Handling | 142 |
| 6.26 | New Binary Format for XML Product Structure Files | 144 |
| 6.27 | File Transfer via HTTPS, HTTP, or FTP | 145 |
| 6.28 | Secure Registry Settings | 149 |
| 6.29 | Reference Import | 151 |
| 6.30 | Transfer of Configuration Data | 153 |
| 6.31 | Domain Effectivity | 155 |
| 6.32 | Transfer of Multiple Effectivity Domains | 156 |
| 6.33 | Enhanced Extended Filtering on Export | 163 |
| 6.34 | Multiple Parts Filtering on Import | 165 |
| 6.35 | Filter on Effectivities | 173 |
| 6.36 | Spares Configuration Data | 176 |
| 6.37 | Version Management Support | 179 |

| | |
|------------------------|------------|
| Appendix A | 181 |
| Appendix B | 187 |
| Appendix C | 191 |
| Appendix D | 194 |
| Appendix E | 205 |
| Appendix F | 210 |
| List of Figures | 218 |
| List of Tables | 223 |
| Index | 224 |

1.Introduction

This manual provides information on running the DELMIA – ENOVIA VPM V5 Connection. It describes the functionality in DELMIA Process Engineer Version E5R15, but also explains important differences between the settings for this version and those prior to and including 5.11.

The customization and setup of a working Engineering Hub to Manufacturing Hub Connection is controlled by a set of registry keys along with some additional information inside a configuration file.

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 briefly describes:

- DELMIA – ENOVIA VPM V5 Connection scenario



Note

When handling the Engineering Hub to Manufacturing Hub Connection, please also refer to the general introduction to Process Engineer in the General Introduction Manual.



Click [General Introduction](#) and [Administrator Manual](#) to access the manual for additional information.

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.

1.3 New Functions in DELMIA – ENOVIA VPM V5 Connection

If you have already worked with the Server tools you should take a careful look at this section. The following sections provide information about new features and architecture in the DELMIA – ENOVIA VPM V5 Connection:

- [Computing Manufacturing Context](#)

The Manufacturing and planning contexts of a process can be calculated by new server interface that offers the possibility to get set of products assembled before the selected process.

- [CAA Application Authentication](#)

Client has to authenticate itself at Manufacturing Hub by a unique authentication identifier. The authentication will ensure that only those clients can connect to the Manufacturing Hub that the Manufacturing Hub can authenticate as 'known' client applications. Manufacturing Hub will grant access only to those clients that have proper credentials.

Manufacturing Hub administrator can authorize selected third-party clients.

- [Customization Settings for Planning Context](#)

Customization settings for creating, deleting, and modifying Planning Context in E5.

- [Version Management Support](#)

This feature makes use of the origin of part versions and assembly relations that are stored on the Manufacturing hub as 'historical id' to identify and handle the versions correctly.

- [Boolean Attribute Filtering](#)

An additional filtering based on Boolean attribute value on the Part Master, Part Version, and Part Instance is possible, similar to the filters based on string attributes.

Multiple filters may be defined and can be joined via the logical operation "AND".

This feature is supported for all transfer modes, i.e. full/partial/incremental update.

2. General

The DELMIA – ENOVIA VPM V5 Connection closes the current gap between the engineering world (represented by ENOVIA VPM V5/CATIA V5, also referred to as the Engineering Hub) and the manufacturing side (represented by the Manufacturing Hub/DELMIA Process Engineer and DELMIA DPM V5). Setting up the Connection allows users to transfer the following kinds of information from the ENOVIA VPM V5 to DELMIA Process Engineer:

- Product Structure Information and Attributes
- Configuration Data
- Transformation Matrices and Geometry

There are two ways for the data transfer:

- The Interactive Mode
- The Batch Mode

As part of explaining how to configure, set up, and test the connection, this document describes the required adaptation of the underlying applications PPRDaemon and PPRLoader on the established customer environment and scenario.

This document has both procedural and reference information. It is intended primarily for the administrator setting up the connection, but also contains user information in the testing and appendices sections.

2.1 Abbreviations

| S.NO. | Terms | Description |
|-------|------------------------------|--|
| 1 | AR | Assembly Relation |
| 2 | Attribute Configuration file | This file contains the information about the attribute mapping to PTS. Also known as "cfg" file. |
| 3 | BSF | Best So Far |
| 4 | CM | Calculation Model (MH) |
| 5 | CO | Change Order |
| 6 | COPS | Carry Over Product Specifications |
| 7 | DPE | DELMIA Process Engineer |
| 8 | EH | Engineering Hub |
| 9 | Export Configuration file | This is an xml file containing customization of the Engineering Hub export settings |
| 10 | Filter | Filtering in the context of this document means filter out the data of a specific part during update/import completely, i.e. this part is considered as NOT existent in the XML export file. |
| 11 | Filter config file | XML file needed to define/customize available import filter options |
| 12 | II | Item Instance |
| 13 | LCA | Life Cycle Application, former name of ENOVIA VPM V5. You can find this abbreviation in some environment variables or registry settings |
| 14 | MH | Manufacturing Hub |
| 15 | MVA | Multi-value attribute |
| 16 | PC | Product Class |
| 17 | PM | Part Master |
| 18 | PRC | Product Root Class |
| 19 | PRC Class | PRC Class |
| 20 | PS | Product Specification (EH) |
| 21 | PT | Plantype. A customizable data type in DPE |
| 22 | PTS | PlanTypeSet: A set of PTs to be used in DPE projects |
| 23 | PV | Part Version |
| 24 | WIP | Work in Progress |

3. Prerequisites

The list below provides the prerequisites for running the Manufacturing Hub to Engineering Hub scenario.

3.1 Server Side

- ENOVIA VPM V5 R19 or higher and Database (DB/2 or Oracle)
- ENOVIA DMU, AD2 configuration (for CGR generation for DPE), installed in the same directory as ENOVIA VPM V5)
- Webserver which supports HTTPS for secured file transfer - or
- FTP Server for non secured CGR-File-Transfer
- DELMIA Process Engineer Server (DPE 5.19 or higher) and Database (Oracle)

3.2 Client Side

- ENOVIA VPM V5 Client R19 or higher (VPC Product)
- DELMIA Process Engineer Client (DPE 5.19 or higher)
- Microsoft .NET Framework 3.0

3.3 Client and Server Sides

- Java Runtime Environment, Version 1.6 update 2

4. Launching Applications

Once the ORBIX is running on the ENOVIA VPM V5 Server Machine then you do not need to launch it repeatedly. The PPRLoader should be running on the Windows machine.

ORBIX

After a reboot, launch Orbix.

On a UNIX machine, use the command:

```
/usr/DS/B10/aix_a/code/command/.catstart -run runOrbix
```

On a Windows Machine, go to the Startup Folder and select:

Program -> Startup -> runOrbix

PPRDaemon

You find the executable in the DPE installation path

~\DELMIA\PPRClient\program\bin.

Launch the PPRDaemon.exe .

Check in the Task Manager that it has been started correctly. For testing purposes, you can start LCALoader (lcaloader.exe). When it is running, the LCA-Loader should answer with:

```
Starting CATBackBone listener service
```

```
Waiting for messages...
```

4.1 Start PPRLoader in Batch Mode

4.1.1 Command Line Application

When you select this mode, the DELMIA Process Engineer® does not have to be running. The PPR Loader starts as a standalone application.

The batch mode is always recommendable if large data quantities are to be imported within the shortest time possible. In batch mode the import speed is faster than when importing from the DELMIA Process Engineer®. Data can be transferred during the night, for example, which are then available for the user on the following day.

1) Open the prompt.

- Click on **Start/Programs/Accessories** and then click **Prompt**. Or Click on Start / Execute... and enter "cmd" in the dialog that is opening. Confirm the entry.

The prompt will open.

In the ... \program\bin directory of your DELMIA Process Engineer® client installation you will find the **PPRLoader.exe** application.

Enter this path. Example: D:\DELMIA\PPRClient\program\bin.

Now start the PPR Loader with the prompt `pprloader` and a start parameter.

64-bit PPRLoader

The currently existing PPRLoader/LCAloader is a 32-bit application. The address space of such an application is technically constrained to a maximum of 4 GB. Depending on the operating system, an even lower value, e.g. a maximum of 2GB virtual address space may be available. To overcome this limitation the importing LCAloader application (and it's underlying dlls) are migrated to 64-bit technology, thus extending the current memory limitations (2/3GB) tremendously up to 8TB on Windows 64-bit OS.

There is no other change between the 32-bit and the 64-bit PPRLoader rather than the 64-bit executable is named PPRLoader64.exe, i.e. all command line options are the very same and work as before. Same applies to the underlying LCAloader64.exe (LCAloader.exe/LCAloader64.exe is invoked from the PPRLoader.exe/PPRLoader64.exe

- 1) Open the prompt.
 - Click on *Start / Programs / Accessories* and then click **Prompt**. Or
 - Click on *Start / Execute...* and enter "**cmd**" in the dialog that is opening.
 - Confirm the entry.
- 2) The prompt will open.
- 3) In the ...\\program\\bin64 directory of your DELMIA Process Engineer® client installation you will find the **PPRLoader64.exe** application.
- 4) Enter this path. Example: D:\\DELMIA\\PPRClient\\program\\bin64.
- 5) Now start the PPR Loader with the prompt pprloader64 and a start parameter.

Limitation: LCAloader64 is supported only in batch mode

4.1.1.1 VPM V5 Import Code Separated from VPM V4 Import Code

Since the first versions, the DELMIA – ENOVIA VPM V5 Connection supported VPM export files from both, VPM V4 and VPM V5 (LCA). To reduce the impact from upcoming functionalities of the DELMIA – ENOVIA VPM V5 Connection to its VPM V4 counterpart and to simplify maintenance efforts, the code has been split in separate libraries. Still, for the end user, there's only one importing application – PPRLoader – which checks whether a VPM V4 XML file or a VPM V5 one is imported and starts VPMLoader for VPM V4 files and LCAloader for VPMV5 files.

There's no change for the end user on how to import data.

lcaloader –help or lcaloader -?

Only if the user wants to know which parameters are available for the import of VPM V5 files, he has to call LCAloader directly:

```
D:\\DELMIA\\PPRClient\\program\\bin>LCAloader -help
```

```
D:\\DELMIA\\PPRClient\\program\\bin>LCAloader -?
```

to get the following information:

```
C:\Programme\DELMIA\PPRClient>LCALoader -help
LCALoader - version 5.18
Copyright © 2001-2006, DELMIA GmbH. All Rights Reserved.
Copyright © 1999, International Business Machines Corporation and others. All Rights Reserved.
Copyright © 1999-2000, The Apache Software Foundation. All rights reserved.
version 5.18
file C:\Programme\DELMIA\PPRClient\program\bin\LCALoader.exe
time stamp Mai 3 2007 11:51:30
build time May 3 2007 11:19:48
current time Mai 14 2007 11:55:53
build info release PE 5.18, build 5.18.0.126, date 2007-05-03-09:26
[May 14 2007 11:55:53] Usage...
lcaloader
  -vpmlca <data file>          run as CAT backbone listener
                              import UPM export file
  -project [<id:short name>]    specify/print project/s
                              import in incremental/partial/complete/reference mode
                              specify root instance
  -incremental[partial|complete|reference]
  -instance [<id:short name>]
  -abf <data file>            import ABF fastener file
  -co <data file/s:directory> import CO change order file/s
  -object [<id:short name>]    specify/print root object/s
  -recalc ! -norecalc         enable/disable recalculation of absolute positions and effectivities
                              force complete recalculation of absolute positions and effectivities
  +recalc                    specify default domain for import
  -domain <domain name>       specify and switch default domain
  +domain <domain name>       import calculation models from all domains
  -calcmmodels ! +prodspecs    import calculation models from all domains, but skip product update
  +calcmmodels ! +prodspecs    specify log file/suppress log
  -log [<log file>] ! -nolog   specify filter/suppress filters
  -filter [<ids>] ! -nofilter  specify and use filter
  +filter [<ids>]             import/update only first structure levels
  -level <depth>             specify directory prefix for CGR files
  -cgrdir <directory>
  -usage                     print this usage info
  -help ! -?                 print help
  -version                   print version
  -info[rmation] ! -about    display info/about window
  -error [<number>]         print error description
```

lcaloader -info

Using this parameter general information on the LCALoader and the support address are displayed (delmia.de.support@3ds.com).



Note

*In contrast to earlier releases, it's no longer possible to use **PPRLoader** without command-line arguments as a debugging tool for PPRDaemon when importing VPM V5 files, because it starts the **VPMLoader** by default. You have to use **LCALoader** directly (without any arguments). In general, you should use **PPRDaemon** and check the various log files.*

4.1.1.2 Import a Product Structure in Batch Mode

pprloader -vpm <data file> (xml file)

Using this type of call of the PPR Loader you can retrieve export files in XML format as they are written by ENOVIA VPM V5.

pprloader -project and pprloader -object

If these two parameters are used alone, i.e. without any further options, the PPR Loader creates a list of all projects or upper top level objects available, i.e. main nodes (*root objects*). This is very useful for quickly finding the short description or ID of a project or of a desired object in order to import into a specific project or to synchronize a structure.

Example:

pprloader -project

```
C:\Programme\DELMIA\PPRClient>LCALoader -project
LCALoader - version 5.18
Copyright © 2001-2006, DELMIA GmbH. All Rights Reserved.
Copyright © 1999, International Business Machines Corporation and others. All Rights Reserved.
Copyright © 1999-2000, The Apache Software Foundation. All rights reserved.
version 5.18
file C:\Programme\DELMIA\PPRClient\program\bin\LCALoader.exe
time stamp Mai 3 2007 11:51:30
build time May 3 2007 11:19:48
current time Mai 14 2007 12:53:13
build info release PE 5.18, build 5.18.0.126, date 2007-05-03-09:26
[May 14 2007 12:53:13] Querying project list from PPR-Hub...
[May 14 2007 12:53:25] Main Thread (0039): Connect 0
[May 14 2007 12:53:25] ... Client started (1284-0)
...found 2 projects.
type      name                                short name      id              creation d
project   Trainingsprojekt                         2003            $id$(0-17406#0, 168) 08.12.2003
project   Projekts mit Zuschlägen                  New Proj.       $id$(0-127060#0, 168) 08.12.2004
```



Note

In a SSO environment use the start batch files provided in the ~\PPRClient\program\bin directory, (CATSTART.exe -run)

pprloader -vpm

imports an ENOVIA VPM V5 export file to DPE.

Example:

```
pprloader -vpm C:\Temp\IncrementalAT0-2.xml -project INC
```

pprloader -vpm ... -incremental

imports an ENOVIA VPM V5 Exportfile in incremental mode. This option can be used to force incremental mode, in case the automatic detection of an incremental file might fail. .

Example:

```
pprloader -vpm C:\Temp\IncrementalAT0-2.xml -project INC -
incremental
```

pprloader -vpm ... -complete

imports a complete file from ENOVIA VPM V5. This option can be used, in case the automatic detection of a complete XML file might fail.

Example:

```
pprloader -vpm C:\Temp\IncrementalAT0-2.xml -project INC -
complete
```

pprloader -vpm ... -partial

imports a partial file from ENOVIA VPM V5.

Example:

```
pprloader -vpm C:\Temp\IncrementalAT0-2.xml -project INC -
partial
```

pprloader -vpm ... -reference

imports a XML file from ENOVIA VPM V5 in reference mode. Multi-instancing is suppressed.

Example:

```
pprloader -vpm $VPMXMLExportFile -project $ProjectIdOrShort-
Name-reference
```



Note

Please save your modifications on the PRC before you start the Export. Only changes committed in the ENOVIA Database will be transferred to Manufacturing Hub.

4.1.1.3 Import Change Orders in Batch Mode

Given an XML change order file, the import of change orders is done through the COloader import application:

- 1) Open the prompt.
 - Click on *Start / Programs / Accessories* and then click **Prompt**. Or
 - Click on *Start / Execute...* and enter “**cmd**” in the dialog that is opening.
 - Confirm the entry.
- 2) The prompt will open.
- 3) In the ...*program\bin* directory of your DELMIA Process Engineer® client installation you will find the **COloader.exe** application.
- 4) Enter this path. Example: *D:\DELMIA\PPRClient\program\bin*.
- 5) Now start the CO Loader with the prompt **COloader** and a start parameter.

coloader -co

Imports an ENOVIA VPM V5 change order export file to DPE.

Example:

```
coloader -co C:\Temp\co-2007-06-12-12:00:00.xml -project BAW722
```

It is also possible to import a set of change order XML files or the content of an entire directory. For a set of files, you can specify the file names with wildcards ('?' for a any single character, '*' for a string of characters). The example below will import all change order XML files of that particular day (time stamp is masked by wildcard).

Example:

```
coloader -co C:\Temp\co-2007-06-12-*.xml -project BAW722
```

To import all files of a given directory, just specify the directory.

Example:

```
coloader -co C:\Temp\exports -project BAW722
```



Note

Please note there is no support for interactive transfer of change orders. Additionally, there is no incremental or partial transfer mode for change orders.

4.2 Export from ENOVIA VPM V5

4.2.1 Interactive Mode

ENOVIA VPM V5 Client

1. Launch the ENOVIA VPM V5 Client.

The login screen below appears:

Figure 1: Login Screen

2. Are you logging into this screen for the first time?

If NO, go to next step.

If YES, check the box that says *Check to choose role at logon*. Then click *Apply Role and Project* button.

Figure 2: Logon

3. Enter the user name and the password and click on *Logon*.

After some time (probably around 1-2 minutes) you will see a panel such as the one below. This is ENOVIA VPM V5 Client.

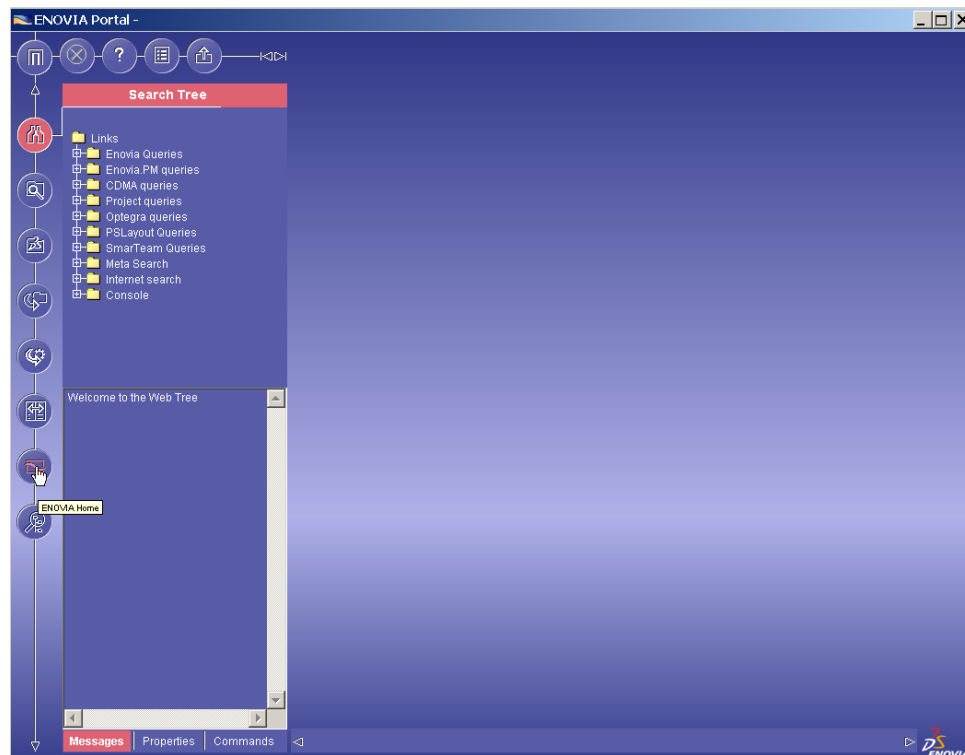


Figure 3: ENOVIA VPM V5 Client

- 4) Go to *ENOVIA Home* as shown in the image above. Then go to Engineering Life Cycle.
- 5) Open the PRC and select the particular product you want to transfer.
- 6) Make sure that the PPRDaemon is running on the Windows machine.
- 7) Select the part, open the contextual menu and select **Send To -> Manufacturing Hub** as shown below.



Note

Please save your modifications in the ENOVIA Database before you send the Product or Part to the Manufacturing Hub. Otherwise the modifications will not be transferred.

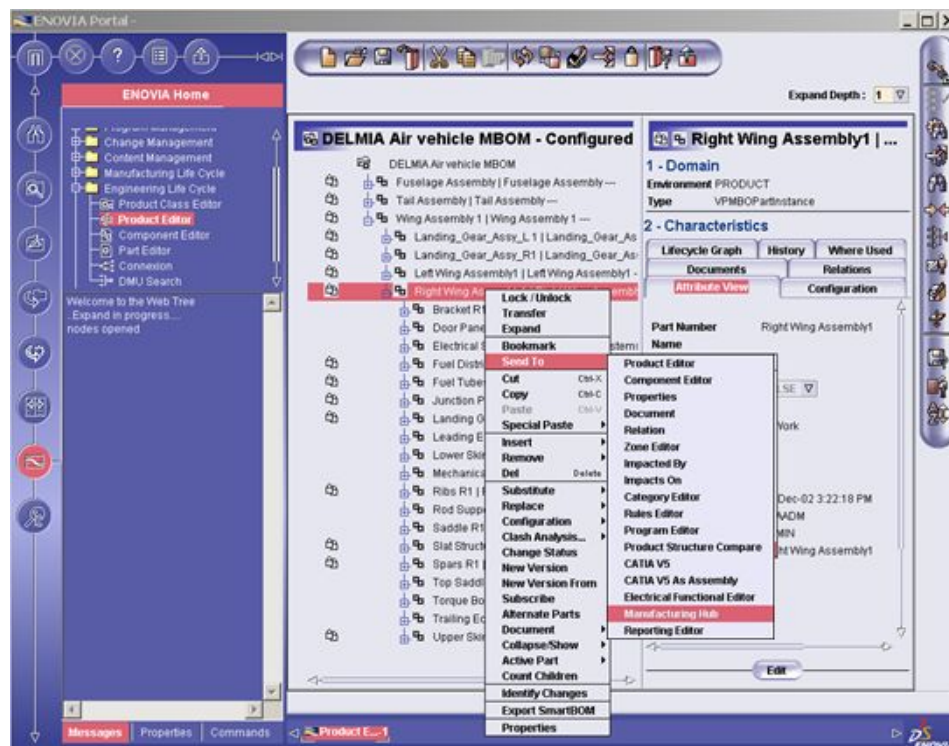


Figure 4: Manufacturing Hub

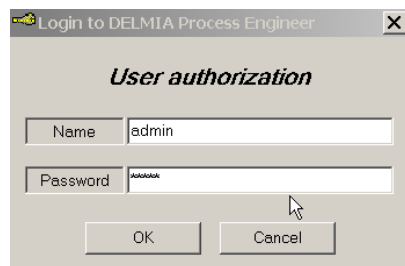


Figure 5: User Authorization

DPE Login Window pops up. You login, but do not start DPE.

Note

The system might ask for the E5 user name and Password.

8) After the login, ENOVIA VPM V5 will display the list of DPE projects.



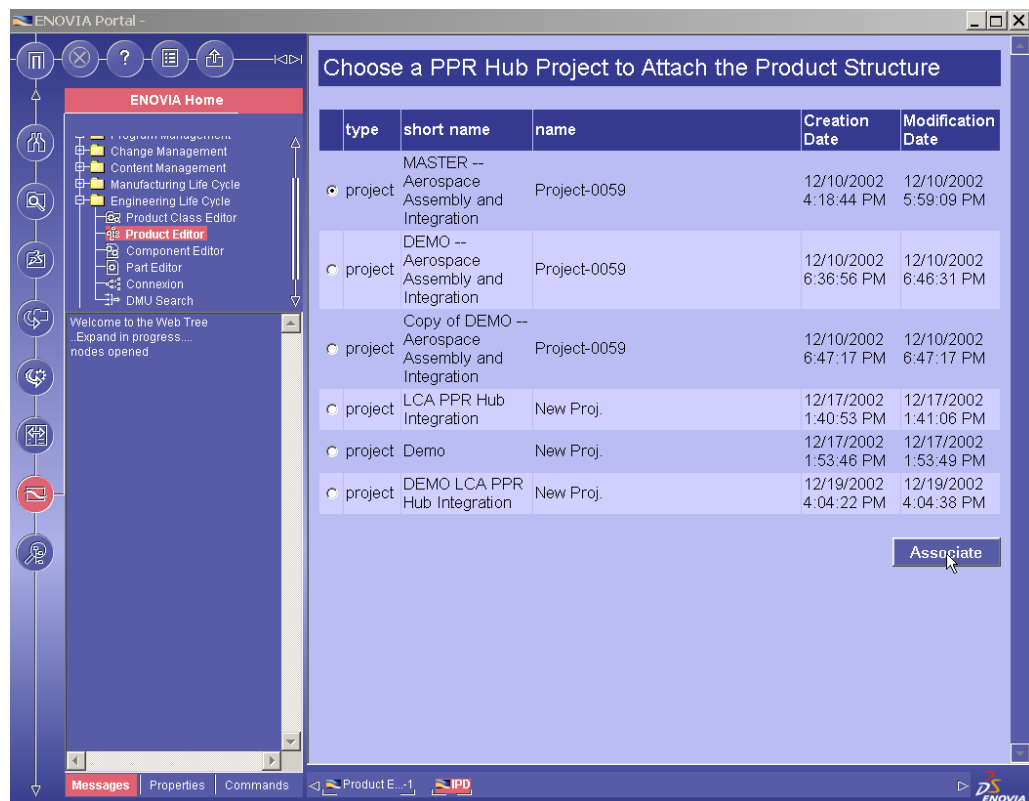


Figure 6: ENOVIA VPM V5

- 9) Check the project into which you want to import the data and click on the 'Associate' button to start the transfer.

A new console window pops up, displaying the import output from the PPRLoader.

Start DPE

Check the result in DPE: Start DPE and open the project. In the project library you can retrieve the imported product.

4.2.2 Export with Command Line Tool ProductDataGen

We can export product structures or their parts from the Engineering Hub using the command line tool “ProductDataGen”. If we call it without any arguments, we get a list of options it respects:

```
% ProductDataGen
    USAGE:
    ProductDataGen [opt] productclass <productclass_id> [incremental <modif_date>]
    ProductDataGen [opt] product <product_id> [incremental <modif_date>]
    ProductDataGen [opt] prcfile <path_to_file> [incremental <modif_date>]
    ProductDataGen [opt] part <part_id> <part_version>
    ProductDataGen [opt] instance <product_id> <part_id> <part_version> <instance_id>
    ProductDataGen [opt] instance <product_id> <part_id>
    ProductDataGen [opt] instance <instance_id>
    ProductDataGen [opt] writeprcfile <product_id>
    ProductDataGen [opt] changeorder <product_id> [<productspec_id>]
    ProductDataGen [opt] settings
```

This shows we have four different ways to export entities:

Export a complete product structure from the PRC. This is equal to an interactive “Send to Manufacturing Hub” on the PRC from the Product Editor.

Export a part. Interactively, this can be done through “Send to Manufacturing Hub” from the Part Editor.

Export a sub-structure whose parent is an item instance referencing a part within a product structure. This is a partial export. Interactively we can achieve this by “Send to Manufacturing Hub” on an item instance in the Product Editor.

Export changes to a product structure since a given date. The respective interactive command is “Send to Manufacturing Hub” on a PRC and specifying “From a given date”.

Let's say we have the following simple product structure

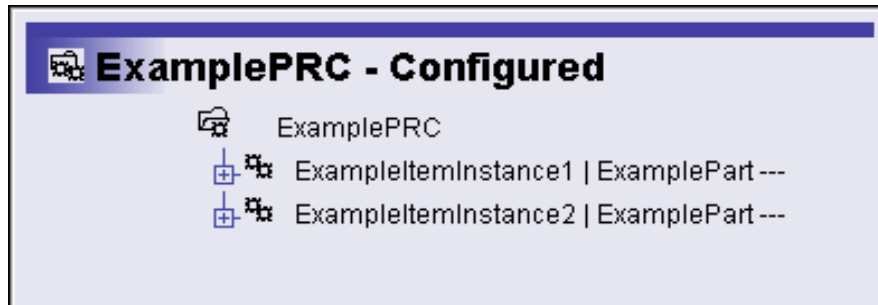


Figure 7: Product Structure

In this case the possible command line options are (each on one line)

- ` ExamplePRC 2005-03-22-10.00.00

Product structures are exported from the Engineering Hub by ProductDataGen and then imported into the Manufacturing Hub by PPRLoader. It is possible to customize the behaviour of both the exporter and the importer, e.g. it's possible to filter out parts which should not be transferred over this connection or the generation of a CGR per CATDocument can be suppressed.

The user has up to three ways to change the default behaviour of ProductDataGen

- The environment
- The configuration file
- The command line

It's only possible to change global properties of ProductDataGen through the first and the third option. More enhanced features can only be configured through the configuration file.

A command-line option takes precedence over the configuration file which in turn has a higher priority than an environment setting.

We recommend the user not to use environment settings at all with the one exception to tell ProductDataGen where the configuration file is located. Environment settings are more or less a historical burden.

Command-line options can be used to overwrite a setting in the configuration file. This may come in handy if one wants to test the effect of a changed setting. In the long run it's better to have a complete set of options in the configuration file.

It is possible to validate the configuration file – not given for the other two options – while first parsing it. This is a good way to check the validity of the configuration settings and to reduce the amount of astonishment about the output generated by ProductDataGen. To this end the configuration option - -validate has to be added to the command line prior to the export type (e.g. product).

The easiest way to see the set of possibilities one has to configure ProductDataGen is by issuing the --help option:

```
$ ProductDataGen --help
```

This is ProductDataGen
V5R19.0.0, 06-05-2008.20.00
DataNav of May 17 2008 02:35:58
SunOS 5.10
(c) 2002-2008 Dassault Systemes

USAGE

ProductDataGen [opt] productclass <productclass_id> [incremental <modif_date>]

ProductDataGen [opt] product <product_id> [incremental <modif_date>]

ProductDataGen [opt] prcfile <path_to_file> [incremental <modif_date>]

ProductDataGen [opt] part <part_id> <part_version>

ProductDataGen [opt] instance <product_id> <part_id> <part_version> <instance_id>

ProductDataGen [opt] instance <product_id> <part_id>

ProductDataGen [opt] instance <instance_id>

ProductDataGen [opt] writeprcfile <product_id>

ProductDataGen [opt] changeorder <product_id> [<productspec_id>]

ProductDataGen [opt] settings

The format of <modif_date> is yyyy-mm-dd-HH.MM.SS, e.g. 2008-06-06-16.46.42

It is possible to tweak ProductDataGen's behaviour in three ways

in descending order:

- (1) command-line option
- (2) configuration file
- (3) environment

The name of a command-line option is the same as the respective configuration setting, see below. The value has to be assigned using the equal operator. Spaces are not allowed. If a value itself contains spaces it must be quoted.

Example: --output:directory="/my/path with spaces/"

Currently understood command-line options:

| CL Option Value | Environment (ENOVIA_LCAIPD_) | Default |
|-----------------------------|------------------------------|---------|
| --apply-ch:id | | |
| --async-export | | yes |
| --block-operation:size | | 10000 |
| --block-operation:processes | | 1 |
| --block-operation:split | | no |

| | | | |
|--------------------------------------|--------------------|--------|-----|
| --cgr-transfer | CGR_TRANSFER | no | |
| --check-doc-iteration | | no | |
| --ch-export:all-domains | | no | |
| --ch-export:version | | All | |
| --config | CONFIGURATION_FILE | | |
| --stop-at-cops | COPS | no | |
| --co-export:processes | | 1 | |
| --document:vault-doc-url | | no | |
| --filter-on-effectivity | | | no |
| --history:origin | | no | |
| --idfile:size | | 1000 | |
| --idfile:processes | | | 1 |
| --separate-rl | | yes | |
| --inc-export:transfer-unchanged-docs | | yes | |
| --log:format %H:%M:%S | LOGFORMAT | %b %d | |
| --interrupt:at-level FATAL | | | |
| --log:level | LOGLEVEL | NOTICE | |
| --log:pid | LOGPID | yes | |
| --log:stderr | LOGSTDERR | ALERT | |
| --mva | MVA | yes | |
| --mva:separator | | @ @ | |
| --mva:sep-at-end | | no | |
| --mva:unique-values | | yes | |
| --output:directory | OUTPUT_DIR | /tmp | |
| --output:insert-root-id | INSERT_ROOT_ID | no | |
| --pattern:escapechar | | \ | |
| --pattern:wildchar | | * | |
| --partial-export:ignore-siblings | | no | |
| --prc-transfer | | yes | |
| --log-names | | | no |
| --refdata | | no | |
| --statistics | | no | |
| --rewrite-proc | | | yes |
| --spare-transfer | | | no |

```
--spare-transfer:attribute
--spare-transfer:report
--substitute-transfer                yes
--traverse:depth                    -1
--traverse:iid-gate                  50
--v5-login                          yes
--in-work-design                     no
--wp-transfer                        yes
--wp-transfer:tool-in-param
--wp-transfer:tool-name
--wp-transfer:tool-out-param
--wp-transfer:tool-paramlist
--wp-transfer:type                    REPORT
```

As can be seen from the output above, not all configuration settings have a counterpart in the environment.

The structure of a command line is

```
--<configuration-property>:<configuration-attribute>=<value>
```

The configuration attribute “enabled” should be omitted as shown in the output above.

5. Altering Settings

This section provides step-by-step instructions for configuring the settings necessary for Engineering Hub to Manufacturing Hub

5.1 General Mapping Information

The current DELMIA – ENOVIA VPM V5 Connection transfers the ENOVIA VPM V5 reference tree of the product structure to the DELMIA Manufacturing Hub. It maps the ENOVIA VPM V5 product structure into an instance model on Manufacturing Hub side, i.e. for each occurrence of a part reference in the ENOVIA VPM V5 product structure; the Connection creates a separate data object in Manufacturing Hub, representing the instance of the object on ENOVIA VPM V5 side (item instance).

Since all objects on Manufacturing Hub side have one origin of the very same part reference in ENOVIA VPM V5, all attributes will have the very same values. All Part Master and Part Version attributes which can be exported from ENOVIA VPM V5 are exported and can be transferred via the E5M Connection.

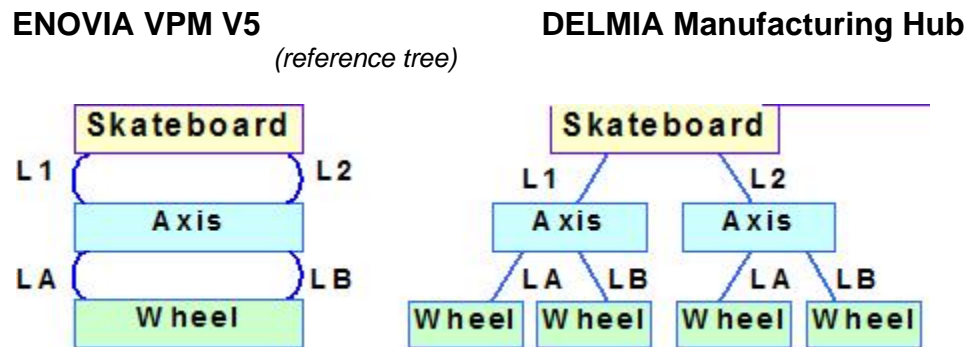


Figure 8: Reference Tree

ENOVIA VPM V5 contains a mechanism of item instances, allowing management of different attributes (beside other features) on different instances of the same part reference in the product structure.

The container for the instance attributes is a second XML export file, extracted from ENOVIA VPM V5, which is automatically imported by the PPR Loader application.

ENOVIA VPM V5 (instance attributes)

DELMIA Manufacturing Hub

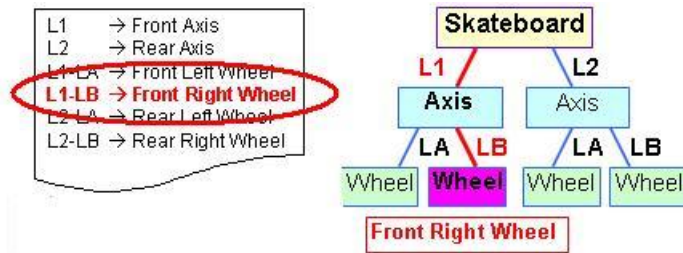


Figure 9: Instance Attributes

To support instance attributes from ENOVIA VPM V5 a second XML file form is created on UNIX side, containing the instance specific attributes of all instances from the product structure, along with their unique instance id (in the context of the selected root part). Like the current XML export file, containing the product structure, the file is fetched via FTP, then parsed and imported by PPRLoader.

No additional setting is to be done on PPRLoader/Connection side.

The name of the instance attribute XML file is the same than the product structure one, but with the prefix "Instance_". It is generated in the same export directory on ENOVIA VPM UNIX side.

Example

In case the product structure XML file name is "ODT_TstInstances.xml", the instance attribute XML file is named "Instance_ODT_TstInstances.xml".

The introduction of a second XML file will result in a second XML parser run for the instance attribute file.



Note

From R13 on the XML file structure has changed. This will cause errors when using ENOVIA VPM V5R12 XML export files. If you want to import XML export files from ENOVIA VPM V5R12, please set the registry key:

[HKCU\Software\DELMIA\ergoplan\PPRLoader\VPM]"release"="R12"

5.2 CAA Application Authentication

The EIPDClient automatically detects whether the authentication is enabled or not. To enable the EIPDClient:

Set registry value “**HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\Ergoplan System\Security\ClientAuthenticationEnabled**” to “1”.

Restart all server processes to apply the changes.



Note

If EIPDClient is enabled then new set of connection methods is used.

If disabled (setup default) the old set of connection methods, which do not authenticate the client application on the server, is used by the EIPDClient.

Client applications, which do not use the EIPDClient to connect the server, are responsible to manage the use of the correct set of connection methods by themselves. If the authentication is enabled and if they try to connect the server using insecure methods, the connection process fails returning an appropriate error code.

When using the new connection methods in the disabled authentication mode, the connection process fails returning an error code indicating the secure connection has been disabled.

If DPM client loses the connection and tries to re-connect, a new authentication is required again.

Connection Process

The new secure connection consists of two steps:

- Step 1: Retrieve the Public Key for encryption
- Step 2: Invoke the secure connecting methods

The authentication is done by the Server Pool, assigning IPDServer processes to the client, the Server Pool must already be selected by the Pooling Server in step (1) and this Server Pool selection must be transferred with the secure connection in step (2). [Figure 10](#) depicts how the secure connection is established.

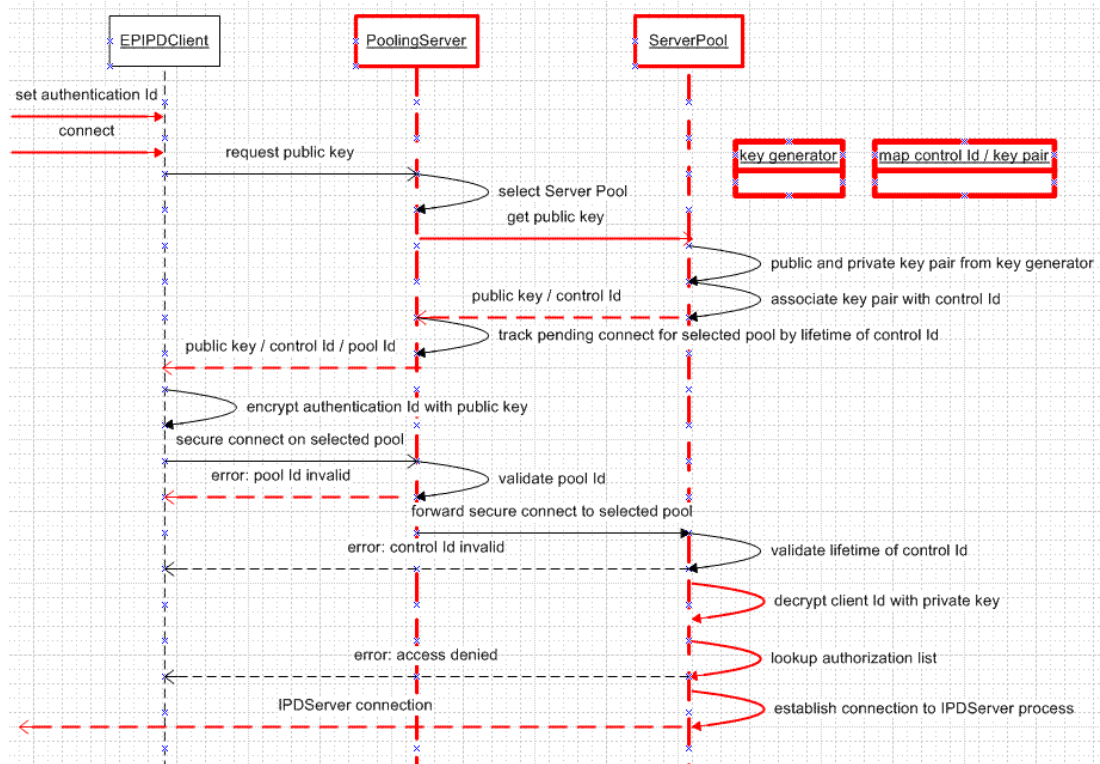


Figure 10: Secure Connection

During the first connection step, before `GetNewClientId` or `Connect` is called for the first time, the `EPIPDClient` requests the Public Key from the Pooling Server to encrypt the module authentication identifier. With the encrypted authentication identifier and an authentication control identifier that is provided by the Server Pool to identify the secure login process, the client connects to the `IPDServer` using a new set of secure connection API methods implemented by the Pooling Server and Server Pool.

After the first step has been completed, the Pooling Server needs to track, which Server Pool is part of pending connection process preventing Server Pool to reconnect to that machine. Otherwise, the connection process could unexpectedly fails. Since a pending Server Pool selection cannot be prevented to reconnect for infinite time, it has a limited life-time and becomes invalid after creation timeout configured by Pooling Server has expired. This timeout is an already existing setting:

HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\Ergoplan EPServer-Tools\serverpool\ipdserver_creation_timeout <value> in [second]

If the timeout setting is disabled (0), a timeout of 1 hour will be used.

In the second connection step the Server Pool authenticates the client module against the list of allowed client modules. This list is provided by the Pooling Server configuration and contains all client module authentications identifiers. For the look-up the Server Pool needs first to decrypt the encrypted client authentication

identifiers using the associated private key. If one of the listed identifiers matches the secure connection succeeds. If the client authentication fails, a specific HRESULT error code is returned allowing the client to evaluate the failed authentication.

For the secure connection, Pooling Server is only responsible to forward the connection calls to the Server Pool and to provide the current list of authorized client module identifiers in an encrypted manner. The implementation guarantees that no other program than the Pooling Server can provide this list. To encrypt the module identification list, the Pooling Server requests a Public Key from each Server Pool. This encryption uses the same interface as clients.

It is client's responsibility to give a proper error message that authentication has failed.

Authentication Identifier

To authenticate the client, the authentication identifier assigned to a specific client module has to be set for the EIPDClient instance calling a new interface method **SetClientModuleAuthenticationId**. This authentication identifier is

- a) Fixed for standard clients provided by the setup or
- b) Created and assigned by the administration for a new client module/application which is allowed to access the server.

For clients of type a) the interface method **SetClientModuleAuthenticationId** is called in the implementation of all client applications. The appropriate authentication identifiers are provided during development.

For clients of type b) the authentication identifier is provided by administrators.

The method **SetClientModuleAuthenticationId** needs to be invoked in both cases before calling **GetNewClientId** or **Connect** method for the first time.

The internal (EIPDClient, Pooling Server, Server Pool) authentication control identifier has a limited life-time and becomes invalid either after the second connection step or after client connection timeout configured by Pooling Server has expired. This timeout is an already existing setting:

HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\ERGOPlan PoolingServer\client_connect_timeout <value> in [second]

If the timeout setting is disabled (0), a timeout of 1 hour is used.

If the EIPDClient tries to call any secure connection method with an invalid authentication control identifier the connection process then fails and an error code is returned.

Public Key Cryptography

The Server Pool is responsible to create the Public/Private Key pair and the global unique authentication control identifier being associated with this key pair, in the first connection step.

In order to enable, the public key cryptography, three wrapper methods are needed:

- Retrieving a new public/private key pair from the operating system environment.
- Encrypt a given string with a given key (the public key)

- Decrypt an encrypted string with a given key (the private key).

All three methods internally use standard Windows © API functions which provide the requested functionality.

The key pair generation is time expensive, Server Pool creates a block of Public/Private Keys by a background task. Each of these key pairs is used within a configurable time span for all requested connections. This time span can be changed by registry setting:

HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\Ergoplan EPServer-Tools\serverpool\secure_connect_key_lifetime_sec <value> in [second]

If only one key pair still exists the Server Pool starts a new key generation. Till until generation has completed, the existing key pair is used for all connection requests.

DCOM Settings

In order to assure that PPR-Server processes are solely launched by the EPServerTools process running on the local machine – and thus “CAA Application Authentication” feature is not by-passed – administrators have to setup the following DCOM settings.

- 1) In the **DCOM Configuration Tool** the **Properties** dialog of the “DPE PPR-Server” application has to be started.

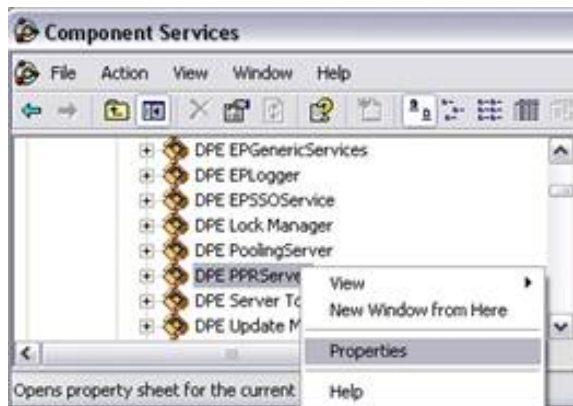


Figure 11: DCOM Setting – Step 1

- 2) On the **Security** tab of the **Properties** dialog change the **Launch and Activation Permissions**.

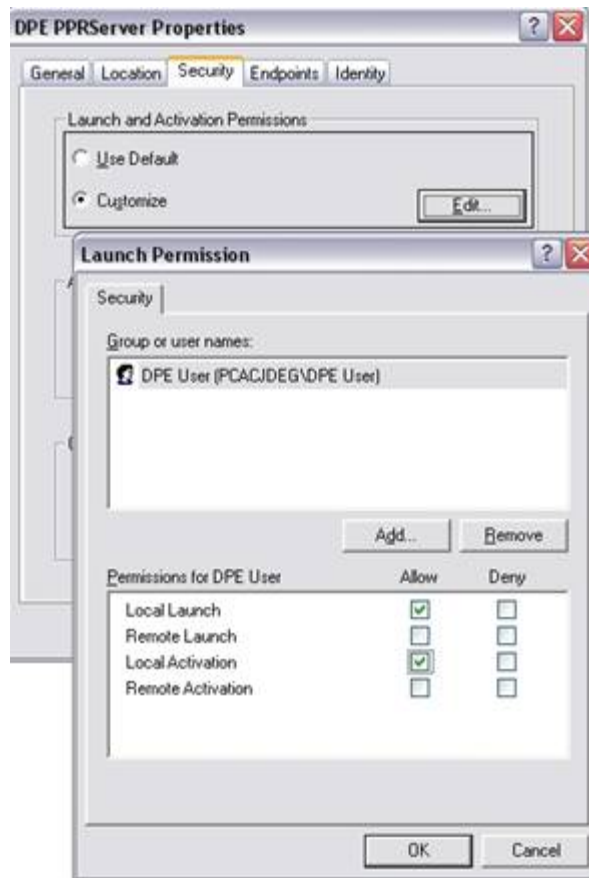


Figure 12: DCOM Setting – Step 2

- 3) The Launch Permission have to be setup in a way that “DPE PPR-Server” processes are only allowed to be started by the user account under which the “DPE Server Tools” process is running.



Figure 13: DCOM Setting – Step 3

The setting:

HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\Ergoplan
tem\Security\MaxNumberFailedClientAuth

Sys-

adjusts the maximum number of failed connects till the Manufacturing Hub is locked. Once the Hub is locked no client can connect anymore even if it could authenticate correctly. The client receives the error message indicating pooling server is locked.

This is a security restriction to prevent hostile attacks. The lock status is displayed in the ServerTools client:

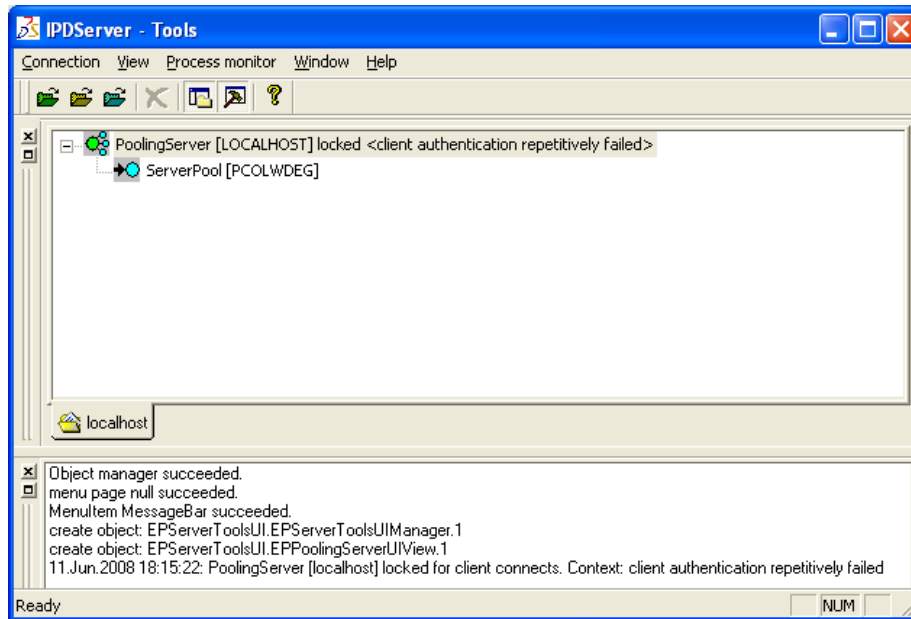


Figure 14: ServerTools client – PoolingServer Lock Status

Whenever that lock state is reached by too many failed client authentications the Manufacturing Hub must be completely restarted. The setup default setting is 10.

Pooling Server Configuration

To allow third party clients, which are not installed by the setup, to connect, when the authentication feature is enabled, the administrator can create new client module authentication identifiers and list them in the new section <<authorization>> within the Pooling Server configuration file. This new section can be placed after or before other sections like <<emergencyconfig>> or <<netconfig>>, e.g.:

```
<<emergencyconfig>>
```

```
....
```

```
<<netconfig>>
```

```
...
```

```
<<authorization>>
```

The section <<authorization>> has the following structure:

```
<<authorization>>
```

```
client_authentication:
```

```
module_ids [ '<authentication_id>', '<authentication_id>',
... ]
```

whereas <authentication_id> has to be replaced with the real GUID string between ' ' (apostrophe). The GUID string format is fixed with the following HEX char sequence:

```
XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
(8 - 4 - 4 - 4 - 12 chars from 0...9 and A...F)
```

Each identifier is separated by comma; the list is not limited, as shown in example below.

```
<<authorization>>  
client_authentication:  
module_ids [ 'C6CF5F84-700B-42fe-9015-6115F0B00962',  
             '8A0093D1-0559-448a-A083-3BEB44B08362' ]
```

The format of the authentication GUID's, being listed in module_ids, is checked by the PoolingServer configuration parser. If it is not correct an the PoolingServer configuration is invalid and is not be used; the error is logged.

Important Points

- 1) The security level, concerning hostile attacks, depends on how the client implements the assignment of the secret Authentication Id. External clients have to consider the possibilities for such attacks as well as the clients delivered with the setup. On serverside particular attention has to be paid regarding the possibilities to externally access the PoolingServer configuration if additional authentication Id's are listed after the setup installation.
- 2) The customization setting 'MaxNumberFailedClientAuth' in registry key HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\Ergoplan System\Security restricts the access on the Manufacturing Hub after that threshold is exceeded by incoming connect request. Whenever parallel connects have already passed this check successfully, they will remain valid. That restriction is effective only for the subsequent connect trials.
- 3) Once the maximum number for allowed failed authentication accesses (Max-NumberFailedClientAuth) is exceeded the Manufacturing Hub must be completely restarted to allow to access the server again, even for clients which can authenticate correctly. This restriction increases the security level concerning hostile attacks.
- 4) The first connect may be delayed in the range of few seconds due to the initial generation of encryption keys. Subsequent connects will not be delayed.

5.3 Customization Settings for Planning Context

The following are the customization settings for creating, deleting, and modifying Planning Context in E5.

- 1) Open configuration and select type **Ergo Component**.
- 2) Set property **Exclude from MCM** to **yes** for attributes.

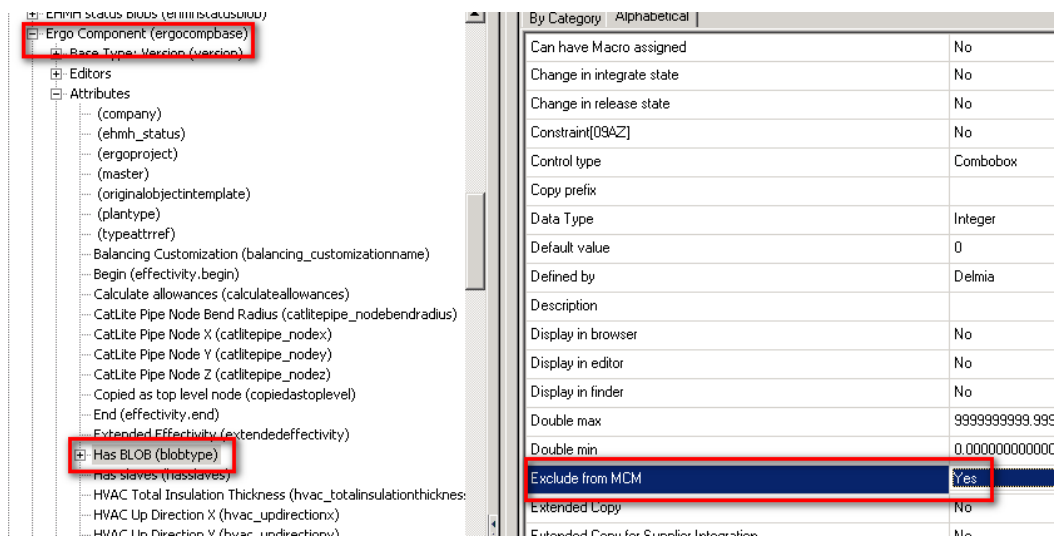


Figure 15: Set Property Exclude from MCM in Configuration

- 3) Select parent child relation “planningcontext” and set the property.

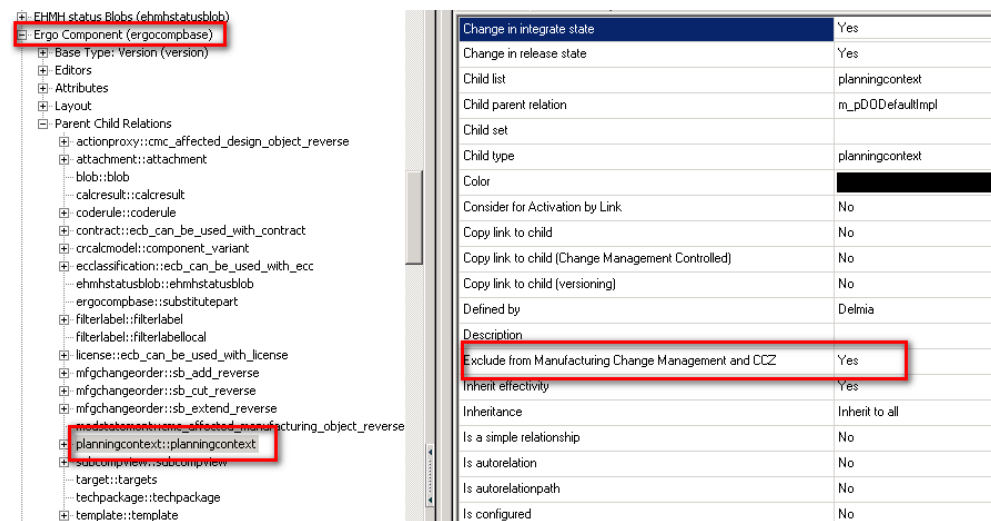


Figure 16: Set Property Exclude from MCM and CCz in Configuration

4) Select attribute “blob_as_string” and set the property.

The screenshot displays a configuration interface with a tree view on the left and a properties panel on the right. The tree view shows a hierarchy of attributes, with 'blob_as_string' selected under 'Planning Context (planningcontext)'. The properties panel shows various settings, with 'Exclude from MCM' set to 'Yes'.

Attributes Tree View:

- Planning Context (planningcontext)
 - Base Type: Base BLOB (baseblob)
 - Attributes
 - Blob Length (bloblength)
 - (blob_as_string)
 - related object or project (dodefautimpl)

Properties Panel:

| By Category | | Alphabetical |
|--|--|--------------------|
| Double max | | 999999999.9999008 |
| Double min | | 0.0000000000000000 |
| Exclude from MCM | | Yes |
| Extended Copy | | No |
| Extended Copy for Supplier Integration | | No |
| Copy prefix | | |

Flags Panel:

| Flags | |
|--|-----|
| Is read only | Yes |
| Is read only for scripts | No |
| Is printable | Yes |
| Change in integrate state | No |
| Change in release state | No |
| Is transient | No |
| In AVSet | No |
| Mandatory value | No |
| Is relevant | No |
| Exclude from MCM | Yes |
| Own Rights | No |
| Extended Copy | No |
| Extended Copy for Supplier Integration | No |

Figure 17: Set Property Exclude from MCM in Configuration



Note

These customization can be ignored for non-MCM Projects, but it is must to follow these customization for MCM Projects.

5.4 PPRLoader Configuration Settings

5.4.1 Configure the PPRDaemon

- 1) From **Tools->Settings->Maintenance Tools**, select the Current User tab.
- 2) Select VPM.
- 3) For backbonereceiver, input the name of the machine where the ENOVIA VPM V5 Client is running. You can also enter "LOCALHOST" (currently backbonesender and receiver have to be the same machine).
- 4) For backbonesender, input the name of the machine where the PPRLoader is running. You can also enter "LOCALHOST" (currently backbonesender and receiver have to be the same machine).

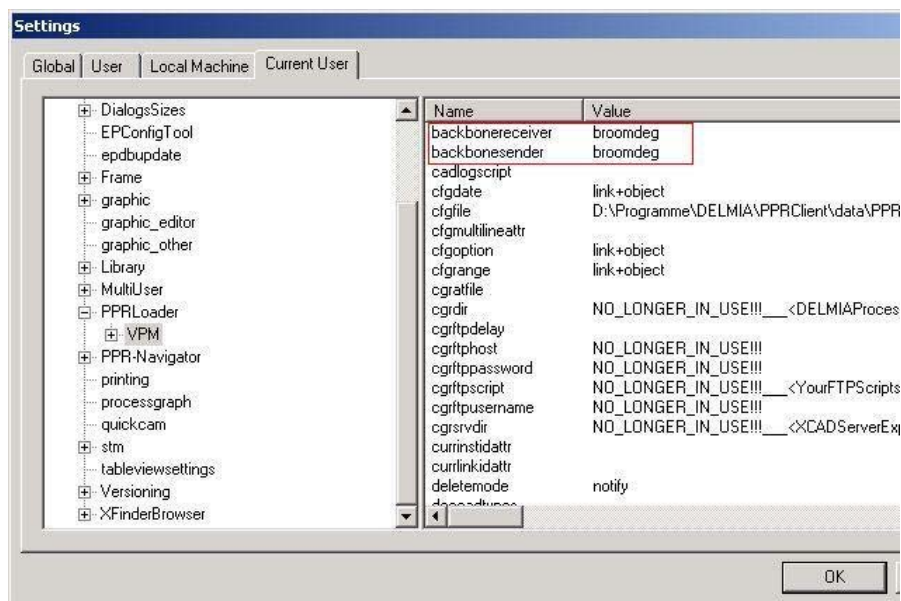


Figure 18: Backbonesender

For information about the details of PPRDaemon's role, *Please refer to the [PPRDaemon](#).*

5.4.2 Configure the CGR Settings

- 1) On the Current User tab, in the left frame, under VPM, click on the plus sign (+) to expand the tree. Under Transfer Settings, click on the plus sign (+) to expand the tree, and then select **CGR Settings**.
- 2) For **cgrdir**, enter the pathname of the directory in which you want the CGR files to be stored.



Note

CGRDIR must be equal to the global graphic_editor\cadpath or ~\product_cadpath value in order to visualize the CGRs in DPE

- 3) For cgrftpghost, enter the name of the ENOVIA VPM V5 database server.
- 4) For cgrftppassword, enter your UNIX password.
- 5) For cgrftpusername, enter your username.
- 6) For cgrlogdir, enter the path to store CGR log files
- 7) For cgrsrvdir, enter the path to find CGR files on the ENVOIA VPM V5 server.

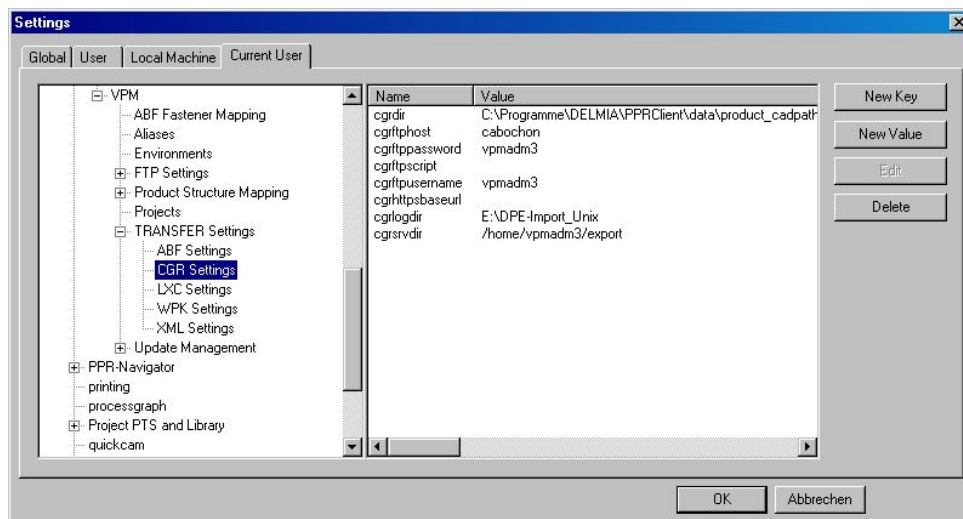


Figure 19: Enter the Path to Find CGR Files

Please refer to the [File Transfer via HTTPS/HTTP/ FTP](#)

ENOVIA VPM V5 Server running on Windows without FTP Server:

the XML-files will be stored in the default temp-directory, e.g.:

C:\Documents and Settings\\local settings\temp

Map this drive (e.g. as "XMLServerExportDirectory") and enter the full path name.

Share the drive.

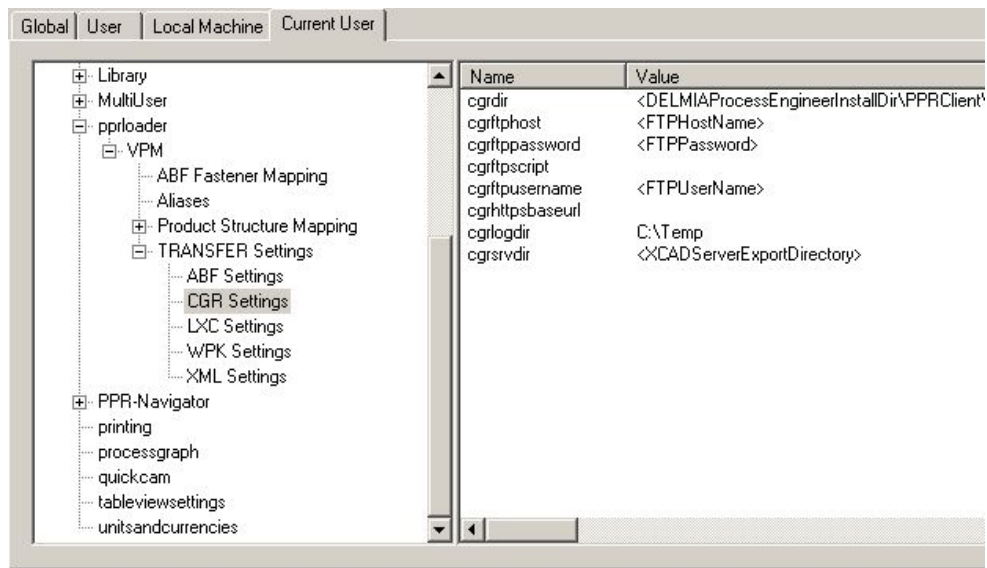


Figure 20: Share Drive

If you want to import CGR files in order to visualize them in DPE, you have to set up a FTP Server on the ENOVIA VPM V5 Server machine. In this case your settings have to be like the settings for the UNIX Server above.

The Connection fetches the CGR-files and renames them. They have to be stored in the global ~\cadpath oder ~\product_cadpath directory.

For a comprehensive discussion of FTP file transfer, *Please refer to the [File Transfer via HTTPS, HTTP, or FTP](#).*

5.4.3 Configure LXC Settings

- On the **Current User** tab, in the left frame, select **LXC Settings**.
- For **lxcdir** enter the pathname of the directory in which you want the LXC files stored.
- For **lxcdelay**, enter the delay time for a FTP.
- For **lxcdphost**, enter the name of the ENOVIA VPM V5 database server.
- For **lxcdpassword**, enter your UNIX password.
- For **lxcdusername**, enter your username.
- For **lxcdwait**, enter the time to wait for FTP.
- For **lxcdlogdir**, enter the path to store LXC log files.
- For **lxcdsrvdir**, enter the path to find LXC files on the ENOVIA VPM V5 server.

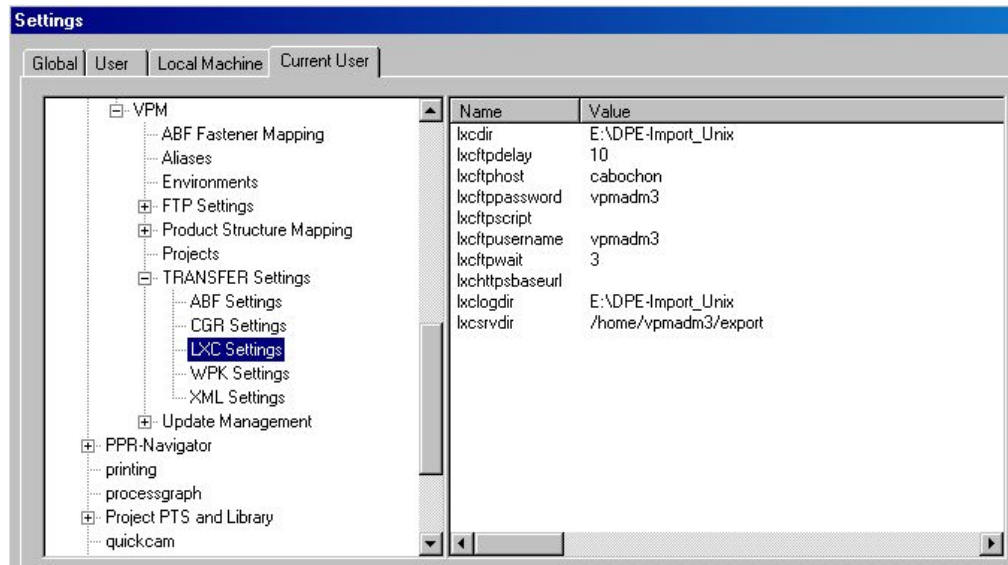


Figure 21: LXC Settings

For a comprehensive discussion of FTP file transfer, *Please refer to the [File Transfer via HTTPS, HTTP, or FTP](#).*

ENOVIA VPM V5 Server running on Windows without FTP Server: the XML-files will be stored in the default temp-directory, e.g.:

C:\Documents and Settings\<user>\local settings\temp

Map this drive (e.g. as “XMLServerExportDirectory”) and enter the full path name.

Share the drive.

5.4.4 Configure XML Settings

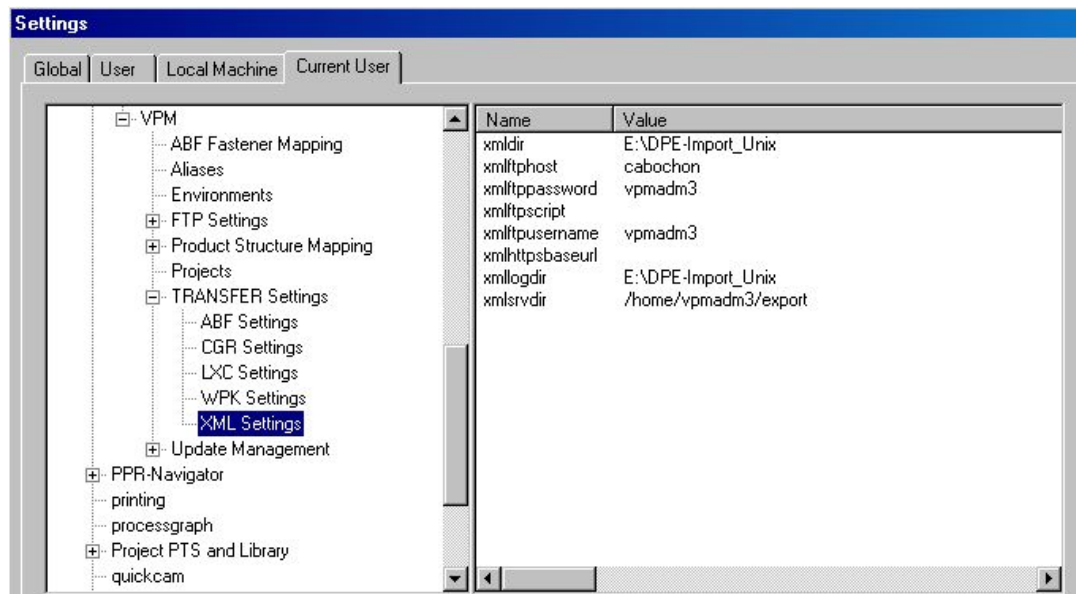


Figure 22: Configure XML Settings

- On the **Current User** tab, in the left frame, select XML Settings.

- For **xmlmdir**, enter the pathname of the directory in which you want the XML files stored.
- For **xmltphost**, enter the name of the ENOVIA VPM V5 database server.
- For **xmltppassword**, enter your UNIX password.
- For **xmlftpusername**, enter your username.
- For **xmllogdir**, enter the path to store XML log files.
- For **xmlsrvdir**, enter the path to find XML on the VPMV5 server.

ENOVIA VPM V5 Server running on Windows without FTP Server:

the XML-files will be stored in the default temp-directory, e.g.:

C:\Documents and Settings\<user>\local settings\temp

Map this drive (e.g. as “XMLServerExportDirectory”) and enter the full path name.

Share the drive.

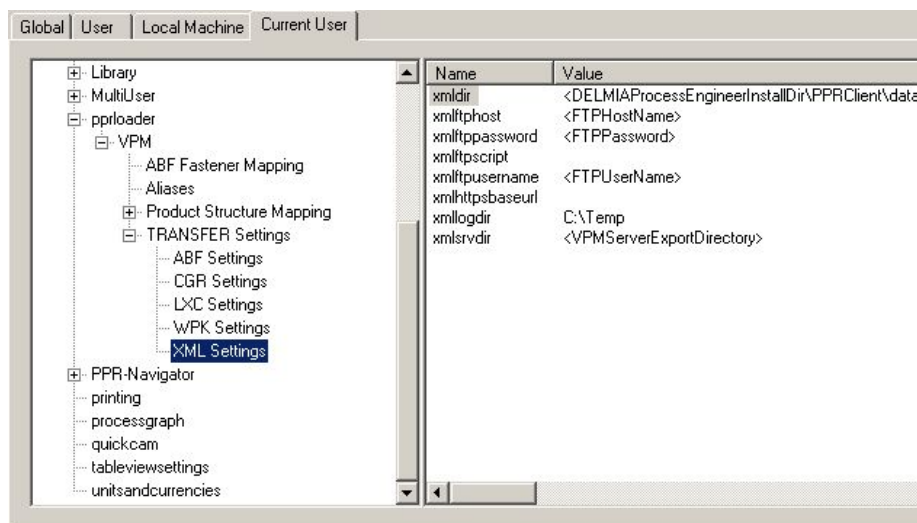


Figure 23: Share the Drive

For a comprehensive discussion of FTP file transfer, *Please refer to the [File Transfer via HTTPS, HTTP, or FTP](#)*

5.4.5 Configure WPK Settings

On the **Current User** tab, in the left frame, select **WPK Settings**.

- 1) For **wpkdir** enter the directory for work package files on the DPE client
- 2) For **wpkftphost** enter hostname of the VPMV5 database server.
- 3) For **wpkftppassword** enter your FTP password (encrypted).
- 4) For **wpkftpscript** enter the FTP script name to execute for the retrieval.
- 5) For **wpkftpusername** enter your FTP user name.
- 6) For **wpkhttpsbaseurl** enter the HTTPS host if using HTTPS instead of FTP.

- 7) For **wpklogdir** enter the directory for work package specific log outputs
- 8) For **wpksvrdir** enter the export directory for work package files on the ENOVIA VPM V5 Server.

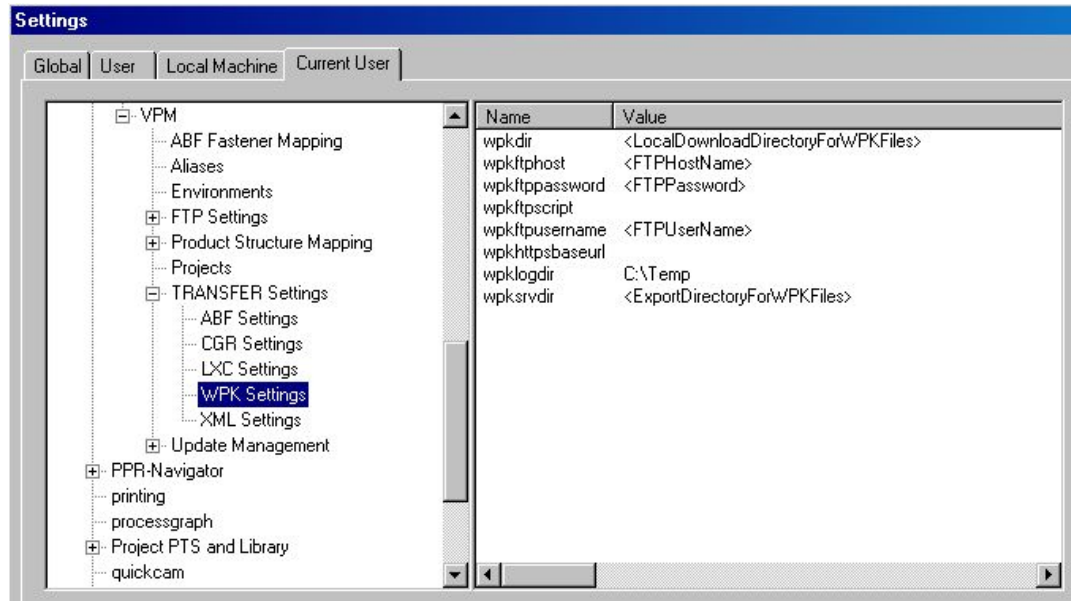


Figure 24: WRK Settings

5.5 PPRLoader Customization Settings

5.5.1 Set Attribute Mappings

- 1) Create the configuration file **lca.cfg** and save it in the
~\DELMIA\PPRClient\data\pprloader directory.
The lca.cfg follows the format below:

```
[attribute0]
general (name=attribute_2)
context (vpm=V_status;significant=true)
[attribute1]
general (name=attribute_4)
context (vpm=C_modified;significant=true)
```

In this example **attribute_2** is a DPE attribute used for the mapping of the ENOVIA VPM V5 attribute called 'Revision Status' for the document.

V_status is the real name of the ENOVIA VPM V5 attribute.

Setting **significant equal to true** activates the **update protocol** of the attribute.



Note

In the example above the configuration file is valid for all environments. If you want a configuration for a special environment, you have to add the environment type like this:

```
[attribute1]
```

```
general
```

```
(name=attribute_2)context(vpm=ENOVIALCA.PART_LIST.V_status;significant=true)
```

- 1) On DELMIA Process Engineer select **Tools->Settings ->Maintenance Tools**, and then select the Current User tab.
- 2) Select **PPRLoader/VPM** in the left frame.
- 3) In the right frame, for **cfgfile**, input the path of the directory in which you have stored the lca.cfg file.

If you do not use a config file, name and nameshort are taken to be significant by default.

If you use a config file, you have to explicitly mark name and nameshort as significant. In this case the above default is no longer valid

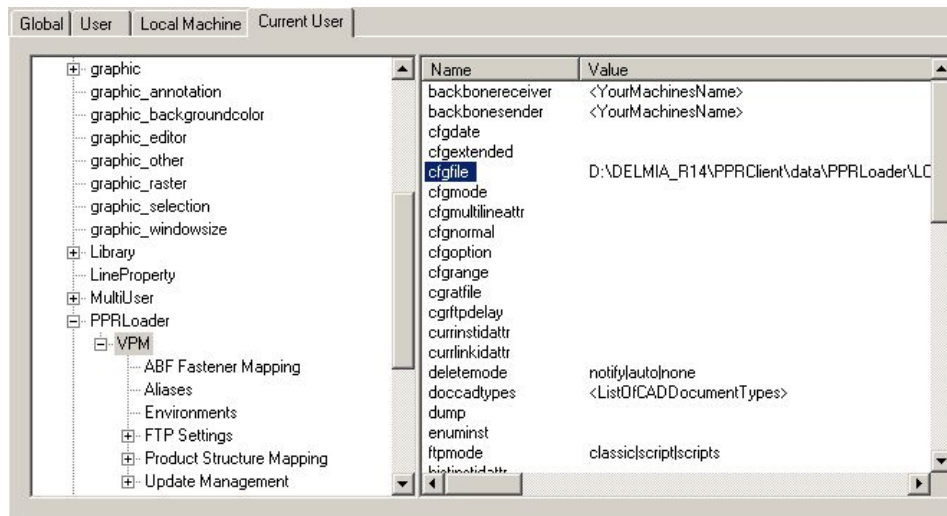


Figure 25: CLG File

For a comprehensive discussion of mapping, *Please refer to the [Product Structure Mapping](#) and [Customized/Extended Attribute Mapping](#)*

For a general discussion of how to use the configuration manager *Please refer to the Appendix B, [General Information – by Administration Manual](#)*

5.5.2 Update Protocol

To display the update state of the products imported into DPE, you have to make this attribute visible in the browser and the editor.

- 1) On DELMIA Process Engineer, select **Tools->Database Utilities->Configuration Manager**.

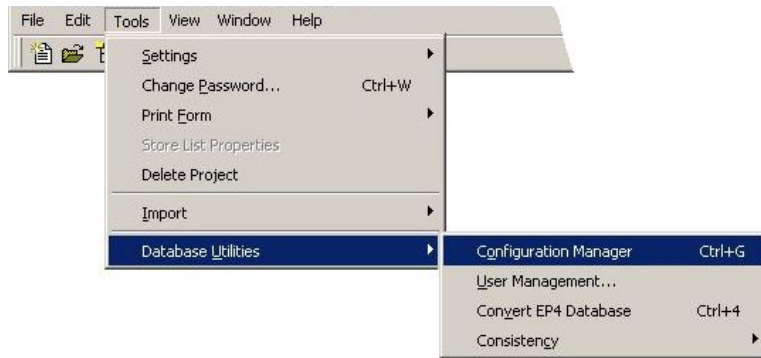


Figure 26: Config Manager

The Configuration Manager pops up.

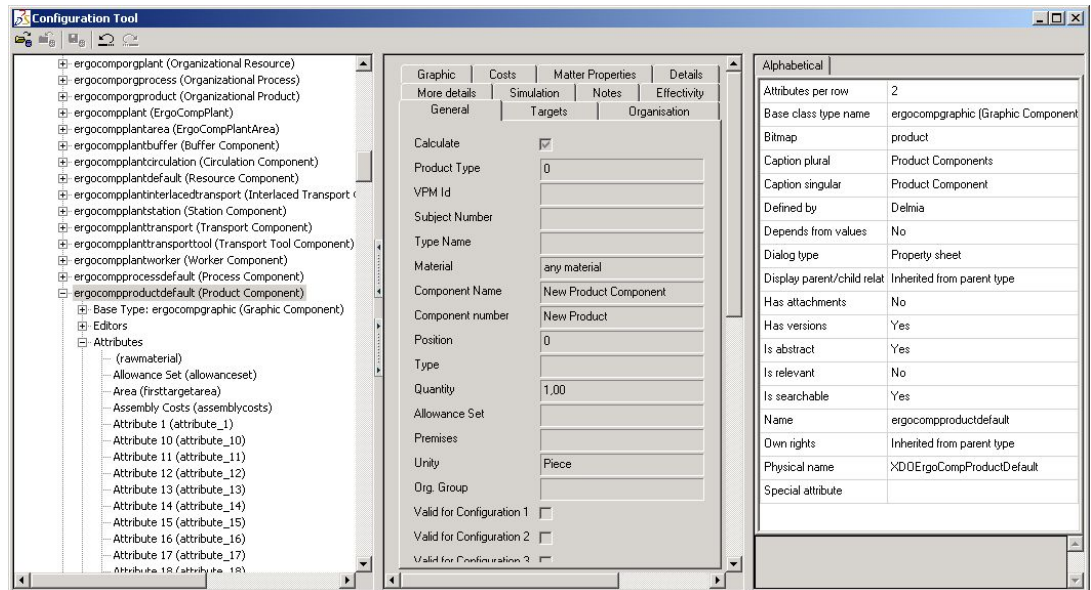


Figure 27: Config Tool

- 2) In the tree view (left frame) select type “ergocompproductdefault” and expand its attributes.
- 3) Select attribute “updatestate”.

If the type “ergocompproductdefault” has no attribute “updatestate”, go to the Base Type “ergocompbase” and overwrite the attribute “updatestate”.



How to overwrite attributes and change their properties, please refer to the [Administration Manual](#).

4) Go to the properties editor to alter the settings:

- Set “Display in browser” and “Display in editor” to “Yes”
- Select Group (1) on Page General
- Position in browser / Position in editor: 1010
- Set Prompt “Update Status”

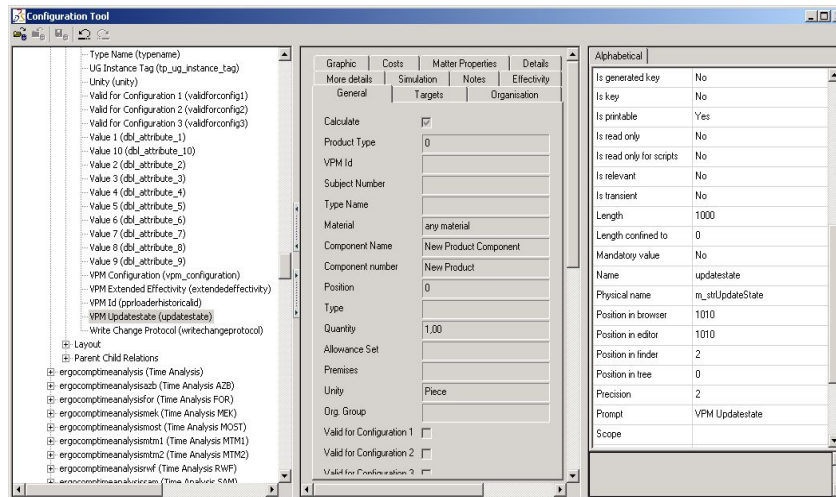


Figure 28: Config Tool

5) Select **Tools -> Settings -> Maintenance Tools**, then select the Current User tab.

6) Select **PPRLoader/VPM** in the left frame.

7) In the right frame, for **deletemode**, input: notify

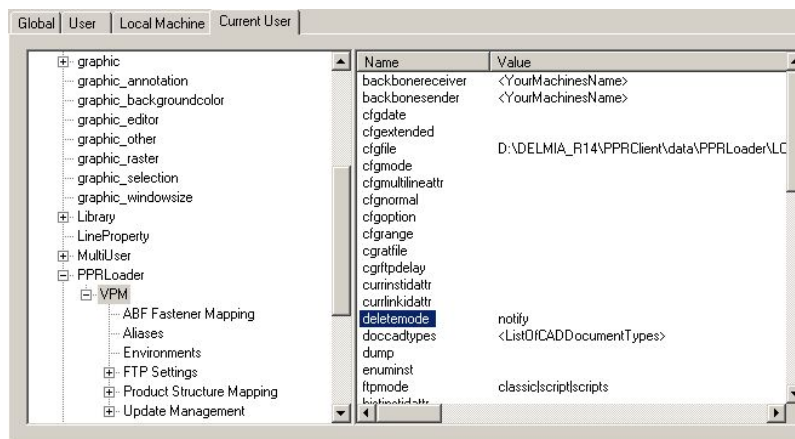


Figure 29: Deletemode

For a comprehensive discussion of the update protocol, please refer to the [Update Protocol and Management](#).

5.5.3 Product Structure Mapping

Adapt and modify the Product Structure Mappings under ENOVIA VPM V5 according to your actual ENOVIA VPM V5 scenario. This includes entering/changing the names of the desired Plantypes, corresponding to your DPE Plantype Set (see [0 Multi-Environment](#)).



Note

Please clarify that in the product structure mapping all existing Environment definitions must be complete and not empty

5.6 ProductDataGen Configuration Settings

5.6.1 Configure the Export

The customization of the export behavior is controlled through settings defined in the Export Configuration xml file.

To make use of these settings, the environment variable :

ENOVIA_LCAIPD_CONFIGURATION_FILE should be defined in the export environment.

Its value is the absolute path to the XML file containing the configuration, e.g.

```
export ENOVIA_LCAIPD_CONFIGURATION_FILE=/path/to/my/export-config.xml
```

Everything else can and should be placed in the XML file itself.

Detailed instructions on how to make use of the different export configuration settings can be found where applicable in the section: [Reference Information](#)

Please refer to the [Appendix D: XML-Based Configuration](#) for a more detailed description of the complete export configuration file.

6. Reference Information

6.1 PPRDaemon

The purpose of the PPRDaemon executable (in the ... \PPRClient\program\bin directory) is to manage all communication between the ENOVIA VPM V5 and the DELMIA Manufacturing Hub/Windows world. The one exception is the actual import of the ENOVIA VPM V5 export XML file, which is done through the PPRLoader application, i.e., PPRDaemon is a user-interface less, Windows 32 application that serves as:

- A “PPRListener,” insofar as it communicates with the ENOVIA VPM V5 Client (retrieving, analyzing and responding messages).
- A “File Fetcher,” insofar as it transfers all required files from the ENOVIA VPM V5 UNIX server side to the Manufacturing Hub/Windows world via FTP, HTTP or HTTPS

In detail, these steps are:

- Listening to messages from the ENOVIA VPM V5 Client, sent through the CATBackbone.
- Returning the list of projects to the ENOVIA VPM V5 Client.
- Fetching the ENOVIA VPM V5 export XML file from the UNIX side (ENOVIA VPM V5 export repository).
- Invoking PPRLoader with the appropriate parameters for ENOVIA VPM V5 export XML file and project id (`pprloader -vpm prc... -project id...`).

PPRDaemon itself does not import the ENOVIA VPM V5 XML file and PPRLoader is not responsible for the communication to the ENOVIA VPM V5 Client and the FTP fetches.

When the PPRDaemon invokes PPRLoader to import an ENOVIA VPM V5 XML Export File, it creates a new (console window) process. By doing so, the communication to ENOVIA VPM V5 (via the CATBackbone process and the ENOVIA VPM V5 Client) is now separated from the import, which offers the following benefits:

- No permanent console window/connection functionality is now available on demand.
- Asynchronous mode/Parallelization
- Stability/Memory Handling

PPRDaemon is based on a subset of the very same registry keys used by PPRLoader, with one exception. Reusing the registry key avoids additional customization. The only new registry key is:

```
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM] "installdir"
```

This key might contain a different path, depending on from where PPRDaemon tries to invoke PPRLoader. An example of this key with a different path is:

```
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM] "installdir"="C:\DELMIA\patch0703"
```

As in previous versions of the DELMIA – ENOVIA VPM V5 Connection, it is still possible (e.g., for debug purposes) to run PPRLoader in listener mode. Nevertheless, this is not recommended. Like PPRLoader, PPRDaemon logs all actions and messages into a log file.

The command line, executed from the PPRDaemon process, starting the PPRLoader import process is:

```
$installdir\pprloader -vpm $file -project $ProjectIdOrShort-Name
```

where: `$installdir` is either the value from the registry key or the start directory of PPRDaemon itself.

`$file` contains the XML file name, including the local data path.

`$id` specifies the project in which to import the data; that is, the object id of the project selected in the ENOVIA VPM V5 Client.

The following image shows the interaction and data flow between the ENOVIA VPM V5 Client, PPRDaemon and PPRLoader.

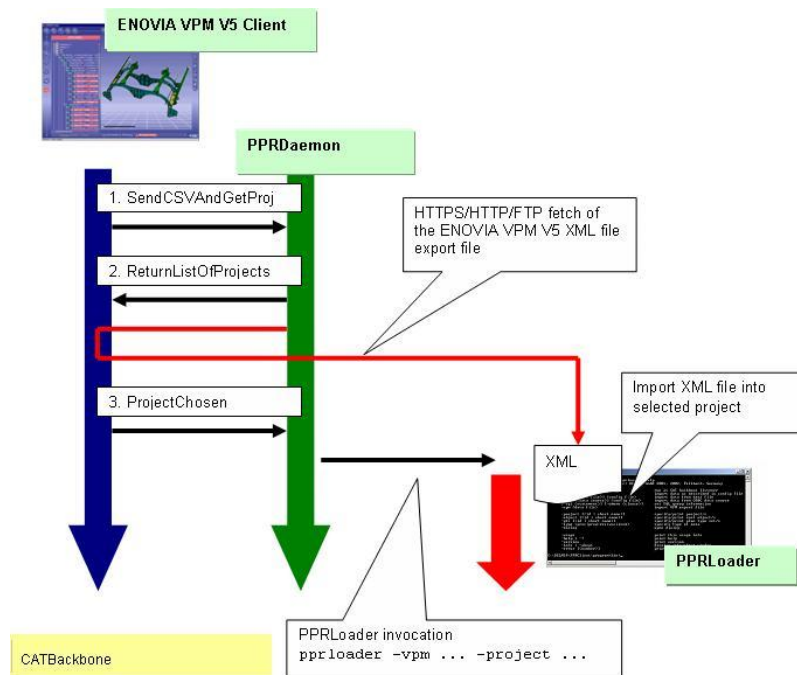


Figure 30: Interaction and Data Flow between the ENOVIA VPM V5 Client, PPRDaemon and PPRLoader

6.2 Product Structure Mapping

6.2.1 Multi Environment

The DELMIA – ENOVIA VPM V5 Connection supports multiple environments from ENOVIA VPM V5. As a result, PPRLoader now allows the individual mapping of VPM types from different Environments to different Manufacturing Hub Plantypes. Therefore, it is recommended to register all Environments from ENOVIA VPM V5 into the corresponding registry keys. Each Environment mapping must contain the table name / plantype mapping from ENOVIA VPM V5 to Manufacturing Hub.

For Documents the settings must also include the name of the attribute in which the CGR file name should be stored. Beside those two settings (for `PART_LIST` and `DOCUMENT`), an additional one for the root object must be specified (which actually has no corresponding representation in the VPM product structure).

In addition to the Environment settings, default mappings for product and resource ensure the completeness of the data import. Unlike previous versions of PPRLoader, at least the default settings must be included in the registry; there are no hard-coded default values for this. The following figure shows the corresponding registry keys.



Note

The type/key `CATIA_MODEL` is not used in the context of ENOVIA VPM V5. However, the corresponding values still must exist in the registry.

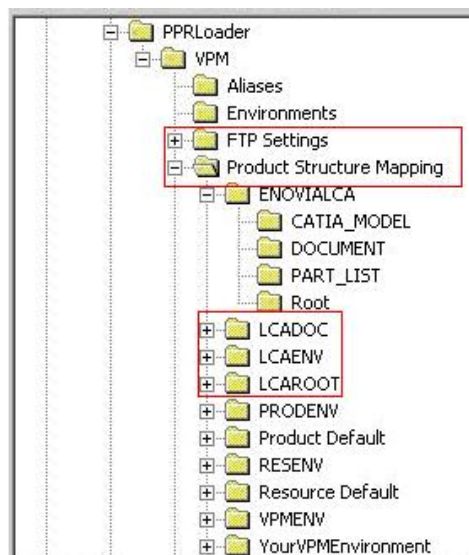


Figure 31: Registry Keys

The information, which Environment a Part belongs to, must be extracted to the XML file. Based on the availability of the environment information in the XML product structure transfer file, the Connection recognizes resources and imports them as such.

To support the Multi-Environment, the extraction from ENOVIA VPM V5 side has to provide this information through the XML file. The relation Environment - instance is not done by package (as in DELMIA – ENOVIA VPM Connection), but by instance. This means, that every part object contains the Environment information which is used to find the corresponding mapping.

Identification for ENOVIA VPM V5 export file

```
<?xml version="1.0" encoding="UTF-8" ?>
- <Package Type="OSM" Uuid="00000000-0000-0000-0000000000000000" Version="0" Name="Root">
- <Package Name="ENOVIALCA.DATA" Type="CATSpecContainer" Version="1">
+ <Instance Name="(LCAROOT.PART_LIST)MOTOR2003-4-2-18.39.52" Alias="(LCAROOT.PART_LIST)MOTOR2003-4-2-18.39.52" Type="CATPDMObject">
+ <Instance Name="_Clink_(PRODUCTS.PART_LIST)Disk Brake---2003-3-27-11.15.56" Alias="_Clink_(PRODUCTS.PART_LIST)Disk Brake---2003-3-27-11.15.56" Type="CATSpecObject" Uuid="ace4c2100000b40c3e831a4d000b0d56">
+ <Instance Name="(LCAENV.PART_LIST)Disk Brake---2003-3-27-11.15.56" Alias="(LCAENV.PART_LIST)Disk Brake---2003-3-27-11.15.56" Type="CATPDMObject" Uuid="ace4c2100000b40c3e82cf2800001917">
+ <Instance Name="_Clink_(DOCDIR.DOCUMENT)Disk Brake---2003-3-27-11.17.41" Alias="_Clink_(DOCDIR.DOCUMENT)Disk Brake---2003-3-27-11.17.41" Type="CATSpecObject" Uuid="ace4c2100000b40c3e82cf5500088c0c">
+ <Instance Name="(LCADOC.DOCUMENT)Disk Brake---2003-3-27-11.17.41" Alias="(LCADOC.DOCUMENT)Disk Brake---2003-3-27-11.17.41" Type="CATPDMObject" Uuid="ace4c2100000b40c3e82cf53000c014d">
+ <Instance Name="(LCAENV.PART_LIST)Disk Brake---2003-3-27-11.15.56" Alias="(LCAENV.PART_LIST)Disk Brake---2003-3-27-11.15.56" Type="CATPDMObject" Uuid="ace4c2100000b40c3e82cf2800001917">
+ <Instance Name="_Clink_(DOCDIR.DOCUMENT)Disk Brake---2003-3-27-11.17.41" Alias="_Clink_(DOCDIR.DOCUMENT)Disk Brake---2003-3-27-11.17.41" Type="CATSpecObject" Uuid="ace4c2100000b40c3e82cf5500088c0c">
+ <Instance Name="(LCADOC.DOCUMENT)Disk Brake---2003-3-27-11.17.41" Alias="(LCADOC.DOCUMENT)Disk Brake---2003-3-27-11.17.41" Type="CATPDMObject" Uuid="ace4c2100000b40c3e82cf53000c014d">
```

Each instance contains Environment name

Instances not belonging to an Environment and also link objects refer to the default **PRODUCT** (for Part objects) and **DOCDIR** (for Documents)

The data extraction from ENOVIA VPM V5 will store the corresponding environment for each part object. With this information the appropriate mapping in terms of Plantype (Subassembly, Part ...) and PPR area (product or resource) can be retrieved from the registry and applied to the part objects.

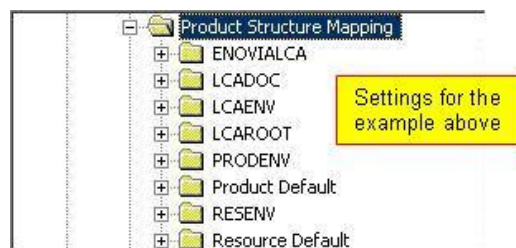


Figure 32: Product Structure Mapping

6.2.2 Fine Grained Type Mapping

In addition to the broader Multi-Environment mapping, it is also possible to fine-grain map ENOVIA components to Manufacturing Hub Plantypes

The mapping of types based on customizable attribute values is possible on the following ENOVIA types:

- Part instance
- Part master
- Part version
- Product class
- Product root
- Product spec

These attributes, which have to be specified in the export configuration file, contain the value that should be used as a mapping key by the PPRLoader process. Manufacturing Hub will map the components to their respective Plantypes defined in PPRLoader registry settings based on these attribute values.

The Connection includes a mechanism which makes sure that the import is correctly done either with environmental or fine grained mapping.

The mapping also works for PRCs which are linked to other PRCs via ProductSpecs or COPS.

Components that don't have a valid mapping are mapped to the default type.

It is possible to map PRCs to different Types of Plantypes, e.g. a PRC used with a Product Spec within the product tree of another PRC may have a Resource Plan-type.

With fine grained mapping, the user is able to map PRCs on Engineering Hub side to Manufacturing Hub plan types based on attribute values of the PRC.

Example

For example: The user wants to transfer all PRCs which have the attributes "Description" set to "Tool" to the Manufacturing Hub plan type "Resource":

Export

Before exporting, the user has to make sure that the environment variable ENOVIA_LCAIPD_CONFIGURATION_FILE points to the Export Configuration file.

For all PRCs the user wants to get mapped he has to set the attribute “**Description**” to the value “**Tool**”.

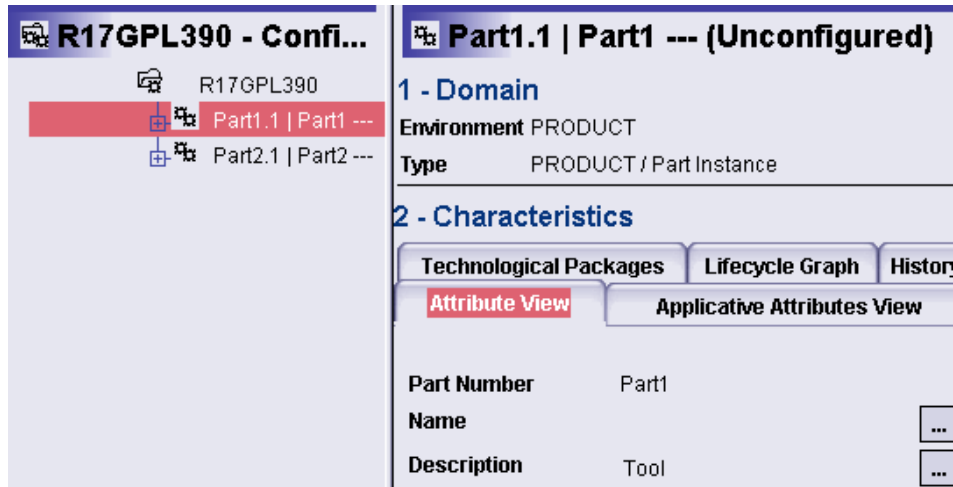


Figure 33: Set Attribute Description

The attributes for the mapping are listed in the Export Configuration file. The values used in these attributes must be defined in the registry of the importing machine in the PPRLoader section for Product Structure Mappings. If for an Engineering Hub attribute value the corresponding registry key does not exist the default type will be used. If a sub key is not defined in the registry the import stops and an error message is written in the log file.

Example: detail from Configuration file:

```
<mappings>
  <mapping context="ProductRoot">
    <attributes>
      <token>V_description</token>
    </attributes>
  </mapping>
</mappings>
```

Example: detail from the registry (incomplete):

```
[HKEY_CURRENT_USER\Software\Delmia\ergoplan\PPRLoader\VPM\Pr
oduct Structure Mapping\Resource\PART_LIST]
"Plantype"="Resource"
[HKEY_CURRENT_USER\Software\Delmia\ergoplan\PPRLoader\VPM\Pr
oduct Structure Mapping\Resource\Root]
"Plantype"="Resource"
```

Import

On the import side the mapping to the appropriate plantype as well as the Root plantype has to be defined:

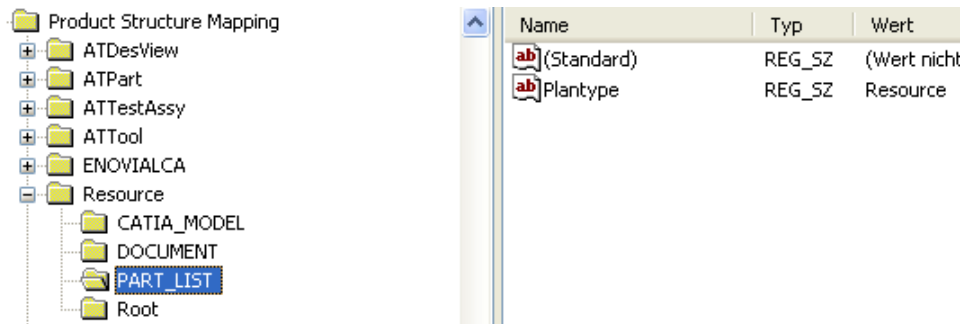


Figure 34: Import

On the import side the mapping between part classes and DPE Plantypes also has to be defined. This is done according to Environments, i.e. in the registry, below Product Structure Mapping, define a new key for each Part class, and define the respective Plan type within a `PART_LIST` key below it:

Prerequisites

It may be necessary as a prerequisite for the customer to customize the ENOVIA database in order to provide attributes which contain the mapping information.

The semi-persistent data in the XML file will be changed. In the reference XML, the unused XML attribute **Type** is used, which specifies the part class that is to be used by the importer to map the types, e.g.

```
<Instance Name="(PRODUCT.PART_LIST) PartNameXYZ"
Alias="(PRODUCT.PART_LIST) PartNameXYZ"
Type="My_Part_Class_1" Uu-
id="ace4c2cd0000cb7e41384cb00001c926">
```

Limitations

- The customization of Plantypes in the Plantypeset must be consistent with the mapping of the PRC together with its parts. E.g. if Plantype A corresponds to the PRC and Plantypes B and C to Parts, which are children of that PRC, the Plantypes B and C must be customized in the Plantypeset to be allowed children of A. This applies also for the Root Plantype: if a PRC is mapped to Plantype A then Plantype A has to be a "legal" child of the Root Plantype defined for it.
- It is not possible to link a product entity to a resource entity. It is possible to do so reversely.
- The Product Structure Mapping has to be completely defined in the PPRLoader's registry even though the keys "CATIA_MODEL" and "DOCUMENT" are disregarded during the import. That means, that also the mentioned key's String values have to be defined and must not be empty, just the attribute's values are not important (also see Related Material, demo registry file).
-

FAQ

Question: What is the behavior for updates if the value defined for Part Class changes? For example a part with class ABC is mapped to Plantype ABC. Then the Part Class changes to DEF, mapped to Plantype DEF during an update. What happens to the product structure? Do I now have two instances of this part in the tree with different Plantype mappings? Or is the old one deleted?

Answer:

If a valid DPE mapping exists, the new Plantype will be set.

If the DPE mapping is invalid, the default Plantype is set.

Question: How are resources mapped? If I specify the Plantype to map to as Resource, will the behavior be the same as using Environment mapping? Do I still need to use the Resource Environment's key?

Answer: The behavior will be the same as with the Product Environment mapping. The Resource Environment's key is used to list all Environments, which map to Resources. For Fine Grained Mapping you have to list in this key all Part Classes which map to Resources.

Question: Is this Granular Mapping supported for all transfer modes, including Incremental Update?

Answer: Mapping is independent of transfer mode.

Question: Is it supported to import into the same project xmls created by Environment mapping and xmls created by Fine-grain mapping. For example, I initially export my structure with Environment mapping, and later switch to Fine grain. Will I have duplicate data of different Plantypes in my structure?

Answer: Existing Environment mapping will be updated not duplicated.

Question: Is it possible to mix environment based mapping and fine grained mapping?

Answer: No, this is not possible. The PPRLoader clearly identifies for an XML file which version was used during the export and will switch to correct mapping mode.

6.3 Extended Part Filtering

The Extended Part Filtering mechanism filters Parts or Part instances on additional criteria. These criteria include attributes such as quality indicators, security and maturity. The objective of the filtering is to prevent the transfer of complete parts and their children (if existing) over the DELMIA – ENOVIA VPM V5 Connection.

The filtering mechanism is based on customizable attribute values on the Part Master, Part Version and Part Instance. Parts or even whole sub trees which e.g. are not yet ready for distribution or are not needed in the MH are filtered out, i.e. are not transferred through the DELMIA – ENOVIA VPM V5 Connection.

The Extended Part Filtering is supported for all transfer modes, i.e. full, partial and incremental update.

The old behavior can be restored by deleting the filter entries in the configuration file.

You can filter out sub structures which should not be transferred over the DELMIA – ENOVIA VPM V5 Connection. Several filters may be defined. The order of the filters does not matter. A different order does not lead to a different filtering. Filters that are based on different attributes are AND-connected. Filters based on the same attribute are OR-connected.

- AND-connected means, if one white list match fails, the part is filtered out (respectively: if one black list match is found).
- OR-connected filters lead to a filtering only when all white list matches fail. Comparisons are done by default case sensitive; this behavior can be changed with the optional attribute *ignorecase*. *****

Examples

If you want to transfer those Parts only (and their sub structures) which have been approved or are empty. That is, all Parts with the attribute `V_Status` that are not set to 'approved' or do not have an empty status should get filtered out:

For this make sure, that the filters are defined correctly in the configuration file of the exporter. E.g.:

White List Filtering:

```
<filters>
  <extendedfilter context="PartVersion"
attribute="V_Status" ignorecase="yes">
    <whitelist>
      <token>approved</token>
      <token></token>
    </whitelist>
  </extendedfilter>
</filters>
```

This will filter out all Part versions with a `V_status` attribute value that is not in the white list.

Alternatively black list filtering can be applied:

```
<filters>
  <extendedfilter context="PartVersion"
attribute="V_Status" ignorecase="yes">
    <blacklist>
      <token>In Work</token>
    </blacklist>
  </extendedfilter>
</filters>
```

This will filter out all Part Versions with a V_status attribute value = "In Work".

Interactive and batch export show the same behavior.

There is no dependency between the "Fine grained mapping" functionality and the "Extended filtering" functionality. Attributes which are used for type mapping can be used for filtering as well and vice versa.

If parts have already been transferred over the DELMIA – ENOVIA VPM V5 Connection and are then filtered out later on, they will be marked "deleted" in the Manufacturing Hub.

Example:

On the ENOVIA VPM-Server the user has to edit the exporter's configuration file to define the extended filters. Empty token values are allowed and allow the transfer of parts which have no or an empty value for the filter attribute. The extended filter has to be defined in the filters section of the configuration file:

```
<filters>
  <extendedfilter context="PartMaster"
attribute="V_attributeName" ignorecase="yes">
    <whitelist>
      <token>Value 1</token>
      <token>Value 2</token>
      <token></token>
    </whitelist>
  </extendedfilter>
  <extendedfilter context="PartVersion"
attribute="V_another_attribute" ignorecase="no">
    <blacklist>
      <token>abc</token>
      <token>xyz</token>
    </blacklist>
  </extendedfilter>
</filters>
```

The "**context**" describes the object type, where the "**attribute**" is found. This can be ProductRoot, PartMaster, PartVersion or PartInstance. The attribute "**ignore-case**" is optional. When missing, comparisons will be done case sensitive.

This filter definition would filter out all parts, whose PartMaster attribute "V_attributeName" and whose PartVersion attribute "V_another_attribute" have a value which is listed in the black list given.

Errors in the definition of the extended filters like "unknown attribute", "empty white list" will be ignored, that is the filter will not be applied. If it is impossible to parse a filter definition (i.e. unknown context, bad XML token, etc.) the exporter will abort execution.

This means for Interactive Mode, that the configuration file should be validated to meet all syntactical constraints before the Engineering Hub server is started to make sure it can be parsed by the exporter program.

Limitations

- Multi Value Attributes are not allowed. Attribute types other than String are converted to Strings first (as the export is doing by default) and then the filtering algorithm is applied.
- Wildcards are not supported.
- Only one black list or white list per filter is allowed. Additional lists will be ignored.

6.3.1 Boolean Attribute Filtering

You can filter out sub structures which should not be transferred over the EH-MH connection.

If you want to transfer those parts only (and their sub structures) which are Standard Part, i.e. all parts with the attribute V508_isStandardPart that is set to false should get filtered out,

For this make sure, that the filters are defined correctly in the configuration file of the exporter.

```
<filters>
  <extendedfilter                                context="PartMaster"
attribute="V508_isStandardPart">
    <whitelist>
      <token>true</token>
    </whitelist>
  </extendedfilter>
</filters>
```

This will filter out all Partmaster with a V508_isStandardPart attribute value that is false, that is not in the white list. Alternatively black list filtering can be applied:

```
<filters>
  <extendedfilter                                context="PartMaster"
attribute="V508_isStandardPart">
    <blacklist>
      <token>true</token>
    </blacklist>
```

```
</extendedfilter>
</filters>
```

This will filter out all parts whose version have V508_isStandardPart attribute value = false.

Interactive and batch export show the same behaviour.

If parts have already been transferred over the EH-MH Connection and are then filtered out later on they will be marked "deleted" in the MH.

Customization

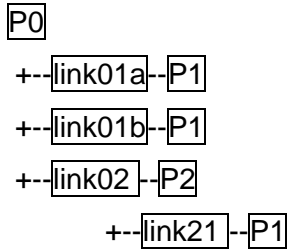
On the EH-Server edit the exporter's configuration file to define the extended filters. The extended filter has to be defined in the filters section of the configuration file:

```
<filters>
  <extendedfilter      context="PartMaster"      attribute="
V508_isStandardPart" ignorecase="yes"
    filtermissing="no">
    <whitelist>
      <token>true</token>
    </whitelist>
  </extendedfilter>
</filters>
```

6.4 Multi Instancing

The DELMIA – ENOVIA VPM V5 Connection maps the ENOVIA VPM V5 product structure reference tree into a Manufacturing Hub instance model.

The figure below represents the ENOVIA VPM V5 and Manufacturing Hub product structure.



Each Part or Document in the product structure can be identified through a unique identifier from ENOVIA VPM V5, containing the environment, its table name, and its OID (object Id). The format in which the ID is stored in Manufacturing Hub is: `ENOVIAEnvironment.TableName.OID`.

The attribute used to store the part (or document) identifier is “externalid” on the “ErgoCompBase” object.

Example

```
P1=ENOVIALCA.PART_LIST.ACE4C2100000B40C3E83391100046A9D_...
```

Links have a similar identifier. It contains the table name of the child object, encapsulated within “CLink(“ ... ”)” as the ENOVIA VPM V5 identifier. The attribute used to store the link identifier is “externalid” on the “SubCompltem” object.

Example

```
link02=ENOVIALCA.CLink(PART_LIST).ACE4C2100000B40C3E8339540001D80F_...
```

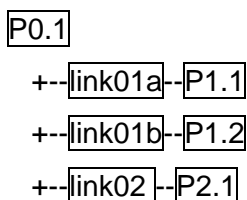
ENOVIA VPM V5 can store attributes specific for an instantiation of a part.

To allow the same functionality (plus providing functions such as code rule inheritance between product and process, or different process relations on different product instances), it is necessary to map part references from ENOVIA VPM V5 into part instances in Manufacturing Hub, each of which is identified by a unique instance ID.

To identify part instances from ENOVIA VPM V5 uniquely, it is necessary to assemble the path of link identifiers from the root object to the instance. The instance identifier of an instance is the path of link identifiers from the root to the instance, separated by a ‘+’ character.

The instance identifier is stored on the attribute “vpm_instanceid1”.

The figure below represents the Manufacturing Hub product structure.



+--link21--P1.3

Example

```
link02+link21=
ENOVIALCA.CLink(PART_LIST).ACE4C2100000B40C3E8339540001D80F_
...+
ENOVIALCA.CLink(PART_LIST).ACE4C2100000B40C3E8339E60001CE8E_
...
```

The usage of the attribute “vpm_instanceid1”, used to store the instance identifier, can be overwritten through the following registry key:

```
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM] "currinstid"
```

Example

```
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM] "currinstid"="attribute_1"
```

The correct multi instance behavior can be checked in several ways:

Each object/component occurs once, and only once in the product structure. This can be checked by using the “Find Usage” function from the DPE context menu of the selected instance or by script.

When the name (or any other attribute) is changed on one instance, this has no impact on any other instance in the DELMIA Product Engineer product structure.

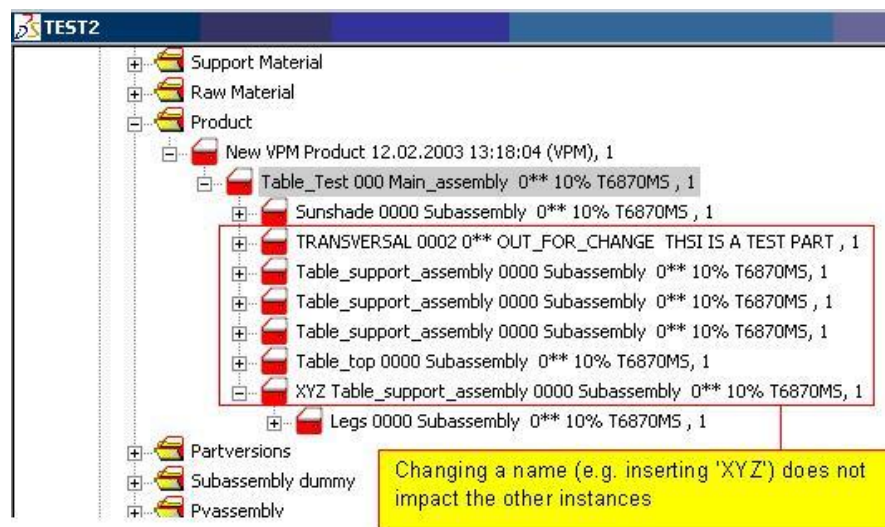


Figure 35: Product Structure

If a Part is moved from one branch of the product structure to another (especially when the hierarchical level inside the structure is changed), there is no way to re-identify the instances beneath.

Example

P0.0

+--link01a--P1.1

+--link01b--P1.2

+--link02--P2.1

+--link21--P1.3

+--link21'--P1.?

All Manufacturing Hub Part instances of a given ENOVIA VPM V5 Part reference do not know about the other part instances, except that they have an identical ENOVIA VPM V5 identifier. They also do not have a common data source (e.g., they do not share a set of common attributes).

6.5 Resource Support

Based on the multi environment support, the DELMIA – ENOVIA VPM V5 Connection recognizes resources and imports them as such.

The handling of resources includes the independent mapping between table names and the corresponding Manufacturing Hub Plantypes, done through a registry key, specifying which environments contain resources and should be treated as such. The relation currently used to link resources to product data in Manufacturing Hub is "plant_provides_prod".

Example

```
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM\Product Structure Mapping]"Resource Environments"="RESENV+TOOLING+PLANT"
```

The registry key

```
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM]"resenvs"
```

is out of use.

6.6 Computing Manufacturing Context

The Manufacturing and planning contexts of a process can be calculated by server interface that offers the possibility to get set of products assembled before the selected process.

The Manufacturing context supports the following process to product relations:

Process creates Product (PCP), Process first processes Product (PFPP), Process processes Product (PPP), and Process removes Product (PRP).

The relations "PCP", "PFPP" and "PPP" are considered to integrate products to the manufacturing context, and hence are referred to as "integrating relations". The corresponding processes are referred to as "integrating processes". The relation "PRP" removes products from the assembly, and hence is referred to as a "removing relation".

The product has four different states: **Integrated**, **Removed**, **Undefined**, and **Untouched**.

The state of a product is **integrated** (and hence belongs to the manufacturing context), if:

- At least one predecessor process is linked with the product via an "integrating relation" (PCP, PFPP, or PPP), and no other predecessor process is linked with the product via a "removing relation". For example, "Prod11" in [Figure 36](#).

- The product is linked with some predecessor processes via both “integrating” and “removing” relations. Further, there exists at least one “integrating” process for each “removing” process, such that the “integrating” process is a successor of the “removing” process. For example, “Prod121” in Figure 36.

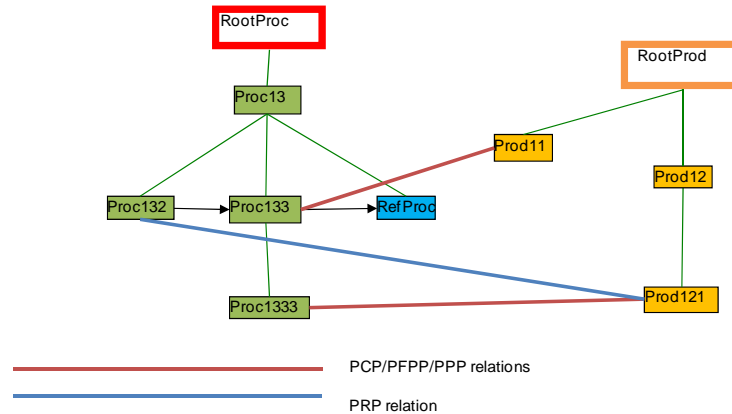


Figure 36: Integrate State

The state of a product is **removed** (and hence does not belong to the manufacturing context), if:

- At least one predecessor process is linked with the product via a “removing relation”, and no other predecessor process is linked with the product via an “integrating relation”. For example, “Prod13” in Figure 37.
- The Product is linked with some predecessor processes via both “integrating” and “removing” relations. Further, there exists at least one “removing” process for each “integrating” process, such that the “removing” process is a successor of the “integrating” process. For example, “Prod141” in Figure 37.

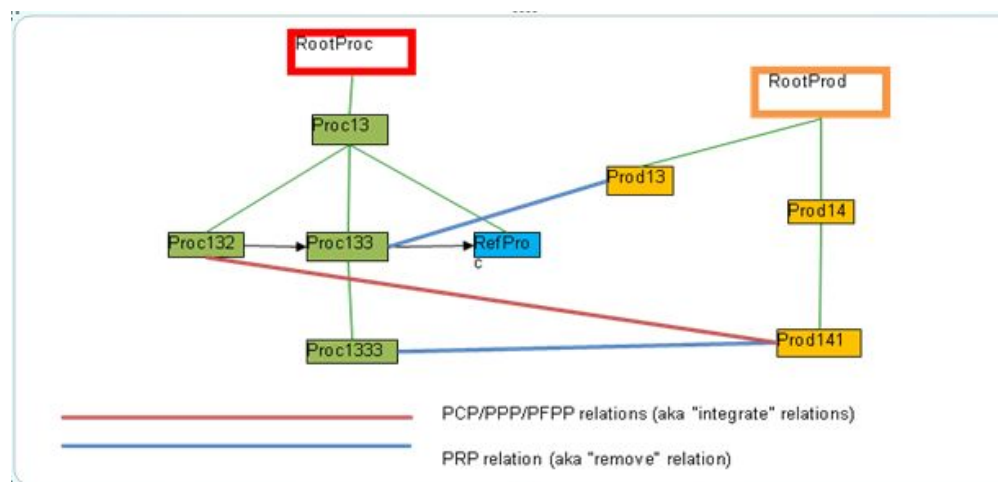


Figure 37: Removed State

The state of a product is **undefined** (and hence does not belong to the manufacturing context), if the product is linked with predecessor processes via both “integrating” and “removing” relations. Further, the last “integrating” process and the last “removing” process are concurrent processes. For example, see “Prod15” in [Figure 38](#).

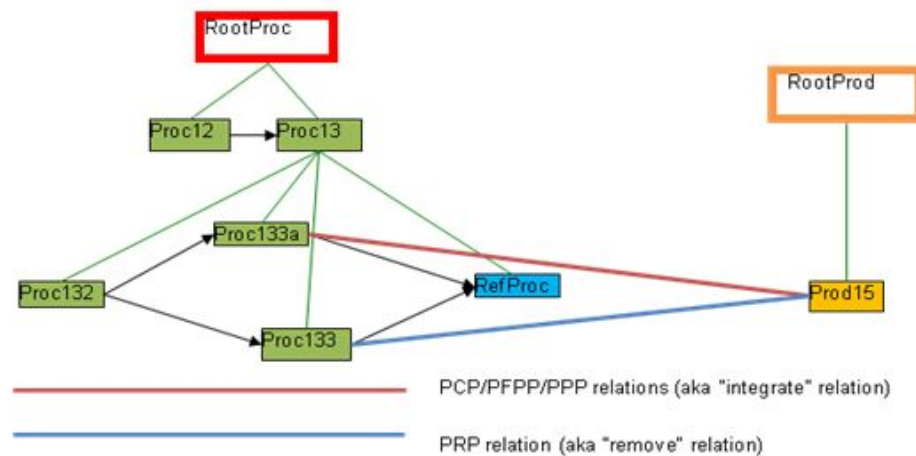


Figure 38: Undefined State

The state of a product is **untouched** (and hence does not belong to the manufacturing context), if the product is not linked with any predecessor processes (via “integrate” or “remove” relations). For example, “Prod16” in [Figure 39](#).

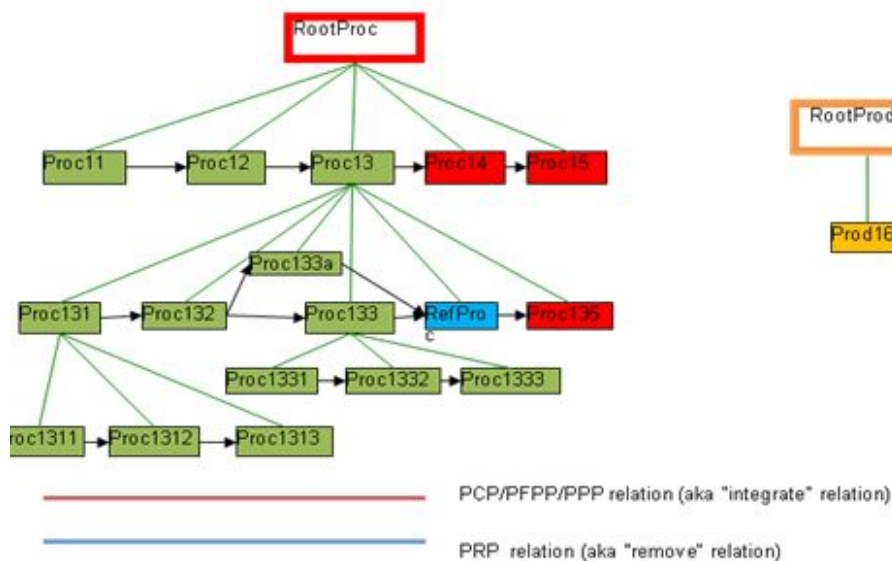


Figure 39: Untouched State

Limitations

- 1) The context consists only of Product objects (not Resource objects).
- 2) The Process structure should obey Hierarchical Order Constraints.
When one process precedes another, then all children of the first process implicitly precede all children of the other.
- 3) The Manufacturing Context computation will consider only those processes that are either direct or indirect predecessors to the reference process, and that are within the defined root process.

6.7 Customized/Extended Attribute Mapping

To use Customized/Extended Attribute Mapping you must create a configuration file, similar to the regular ones (used for standard PPRLoader imports) that describes the specific mapping of the ENOVIA VPM V5 attributes and the Manufacturing Hub attributes (administered with the Configuration Manager in DELMIA Process Engineer). This file has to be defined using the following registry key:

```
HKCU\Software\Delmia\ergoplan\PPRLoader\VPM] "cfgfile"="..."
```

The example below illustrates how to use this file. Below is a portion of an ENOVIA VPM V5 export file. The default environments are PRODUCT for parts and DOCDIR for documents.

```
- <Attribute Name="V_user" Type="String">
  <Value Value="APV" />
</Attribute>
- <Attribute Name="V_organization" Type="String">
  <Value Value="ADMIN" />
</Attribute>
- <Attribute Name="V508_AppDomaine" Type="String">
  <Value Value="ADMIN" />
</Attribute>
```

This example shows three attributes: V_user, V_organization, and V508_AppDomaine. Each is defined at the PART_LIST table inside ENOVIA VPM V5.

Each attribute has an [attribute...] entry in the config file. In our example, the entries could look like this:

```
[attribute0]
general(name=nameshort)
context(vpm=PRODUCT.PART_LIST.V_user)
[attribute1]
general(name=attribute_12)
context(vpm=PRODUCT.PART_LIST.V_organization)
[attribute2]
general(name=attribute_18)
context(vpm=PRODUCT.PART_LIST.V508_AppDomaine)
```

Name of the ENOVIA VPM V5 attribute (including environment and table name)



Note

There is only one configuration file for all types and table names. Document attributes will have a different table name prefix (DOCDIR.DOCUMENT for documents from the default environment).

As described above, for reference attributes which are mapped, the mapping description in the config file must contain the ENOVIA VPM V5 environment and type information (see [attribute2] in the following figure). If this is not specified (see [attribute3]), the attribute describes a mapping for instance attributes and in addi-

tion serves as a mapping template, valid for all ENOVIA VPM V5 environments and types.

The order of the attributes is of no importance, but keep in mind that the sequence of attributes must be continuous starting with 0 ([attribute0]).

PPRLoader tries to convert the ENOVIA VPM V5 attributes (always exported as strings) according to the attribute type in Manufacturing Hub. Currently, PPRLoader supports conversion into five different types:

Double/float, integer, Boolean, date/time, and string (default)

The type of the Manufacturing Hub attributes is read from the Manufacturing Hub configuration database, so no additional information is required in the configuration file for the type conversion.

Example

```
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM] "cfgfile"="D:\DELMI\data\lca.cfg"
```

The following text shows a section from a config file used for attribute mapping between ENOVIA VPM V5 and the Manufacturing Hub (non-default environment ENOVIALCA):

```
;ENOVIALCA
[attribute0]
general (name=attribute_17)
context (vpm=ENOVIALCA.PART_LIST.V_status)

[attribute1]
general (name=attribute_17)
context (vpm=ENOVIALCA.DOCUMENT.V_status)

[attribute2]
general (name=attribute_12)
context (vpm=ENOVIALCA.PART_LIST.V_type)

[attribute3]
general (name=attribute_12)
context (vpm=ENOVIALCA.DOCUMENT.V_type)
```

It is also possible to map attributes from the ENOVIA link to the instance in Manufacturing Hub (since there is only a limited number of free attributes on the sub-compitem/link object in DPE). To accomplish this, the environment and type `PRODUCT._CLink_(PART_LIST)` is required, independently from the real environment the link/part belongs to. I.e. there will be no `ENOVIALCA._CLink_(PART_LIST)` entry in the mapping file. In addition, there will be no `PRODUCT._CLink_(DOCUMENT)` contexts.

```
[attribute0]
general(name=attribute_24)
context(vpm=PRODUCT._CLink_(PART_LIST).V_volume_x1)
```

Mapping for link attributes start always with PRODUCT._CLink_(PART_LIST)

```
[attribute1]
general(name=attribute_25)
context(vpm=PRODUCT._CLink_(PART_LIST).V_volume_y1)
```

```
[attribute2]
general(name=attribute_26)
context(vpm=PRODUCT._CLink_(PART_LIST).V_volume_z1)
```

...

```
[attribute7]
general(name=attribute_12)
context(vpm=V_nbsubstitute)
```

```
[attribute8]
general(name=attribute_13)
context(vpm=V_issubstitute)
```



Note

All registry settings under

[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM\Environments]

are obsolete. The attribute mapping is contained in one and only one configuration file, which has to be registered in:

[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM]"cfgfile"="..."

6.7.1 Attribute Length

In case you need to visualize the ID attributes coming from ENOVIA VPM V5 (instance ID, external ID ...), you have to ensure that the identifier attribute is able to store the entire (string) value of the ID. While the external ID from ENOVIA VPM V5 does not exceed 64 characters, the instance IDs can reach several hundreds of characters quite easily.

As a reminder, the following formula can be used:

$$L_{id} = D_{VPM} * 100$$

where L_{id} is the length of the instance ID attribute in characters and D_{VPM} is the depth of the product structure from ENOVIA VPM V5.

The value of 1024 characters would e.g. support product structures in ENOVIA VPM V5 up to 10 levels. This length has to be set in the properties of the corresponding attribute, when switching it to 'visible' in the Configuration Manager Tool of DPE.

For further information, *Please refer to the [Administration Manual](#).*



Note

All these settings have no impact on the Manufacturing Hub database itself (especially not for ORACLE).

6.7.2 Configuration File

For reference attributes which are mapped, the mapping description in the configuration file must contain the ENOVIA VPM V5 environment and type information (see [attribute2]). If this is not specified (see [attribute3]), the attribute describes a mapping for instance attributes and in addition serves as a mapping template, valid for *all* ENOVIA VPM V5 environments and types.

```
context(vpm=PRODUCT.PART_LIST.V_order)
```

ENOVIA VPM V5 environment and type for reference attribute mapping.

```
[attribute2]
```

```
general(name=attribute_2)
```

```
context(vpm=PRODUCT.PART_LIST.V_status)
```

Just the attribute name for instance attributes and reference attributes templates, valid for ALL ENOVIA VPM V5 environments and types.

```
[attribute3]
```

```
general(name=dbl_attribute_2)
```

```
context(vpm=V_level)
```

```
[attribute4]
```

```
general(name=dbl_attribute_2;type=Part)
```

```
context(vpm=V_level)
```

The type, indicating that this mapping applies only to this particular plantype.

```
[attribute5]
```

Please note, that the plantype definition for the attribute

```
general(name=attribute_2;type=Subassembly)
```

has a different meaning now as in previous releases.

Some possible obstacles and the corresponding warnings are described in the following.

In case the plantype is defined in the attribute definition (either reference or instance attribute), this definition replaces the Plantype, specified through the Product Structure Mapping).

```
[attribute4]
```

```
general(name=attribute_2;type=Part)
```

```
context(vpm=PRODUCT.PART_LIST.V_order)
```



Caution

Plantype definition 'Part' for attribute 'attribute_2'
 ('PRODUCT.PART_LIST.V_order') replaces the registered plantype
 'PRODUCT.PART_LIST' (product structure mapping 'Subassembly'), see
 [attribute4]

When an attribute is defined multiple times, the second, ambiguous definition is ignored.

```
[attribute2]
general (name=attribute_2)
context (vpm=PRODUCT.PART_LIST.V_status)
...
[attribute4]
general (name=attribute_4)
context (vpm=PRODUCT.PART_LIST.V_status)
```



Caution

Ambiguous mapping for VPM attribute 'PRODUCT.PART_LIST.V_status' ignored ('attribute_4' <> 'attribute_2'), see [attribute2] and [attribute4]

In case an invalid reference attribute is specified (the attribute is not found on the registered plantype), the following warning is displayed

```
[attribute6]
general (name=attribute7)
context (vpm=PRODUCT.PART_LIST.C_created)
```



Caution

Attribute 'attribute7' not found on plantype 'Subassembly', see [attribute6]

For invalid instance attributes no warning is dumped during parsing the configuration file, since the corresponding plantype is unknown (it's an instance or reference template attribute mapping!).

```
[attribute7]
general (name=attribute8)
context (vpm=C_created)
```

Instance attribute cannot be checked upfront! ; (plantype is unknown at parsing time)

In this situation no warning is dumped during parsing the configuration file, since the corresponding plantype is unknown (it's an instance or reference template attribute mapping!).

At the very end of the data transfer, there are also warnings for unused *reference* attribute definitions.

```
[attribute8]
general (name=attribute_9)
context (vpm=PRODUCT.PART_LIST.V_level)
```



Caution

-> attribute 'Subassembly::attribute_9' ('PRODUCT.PART_LIST.V_level') not used during import, see [attribute8]

Unused *instance* attributes, do not raise warnings, since it is unknown whether the attribute has been used as a reference template (attribute mapping for reference attribute, independent of ENOVIA VPM V5 environment and type!).

[attribute9]

general (name=attribute_10)

context (vpm=V_order)

6.8 Substitute Parts

New attributes for substitute Parts to ensure the correct transfer of these parts:

- “PrimeUUID”: For substitute Parts the Prime UUID (“PrimeUUID”) of the prime part. This attribute is mapped into the MH by default. The corresponding attribute on MH side is “primeinstanceid” (class ergocompbase). The prime UUID contains the ENOVIA VPM V5 instance UUID of the prime part it points to (stored in MH attribute “vpm_instanceid2”).
- “V_issubstitute”: (boolean); indicates whether a part instance is a substitute part or not. This attribute is not mapped to MH by default.
- “V_nbsubstitute”: (integer); number of substitute parts a base part has. By definition this number is “0” for substitute parts. This attribute is not mapped to MH by default.

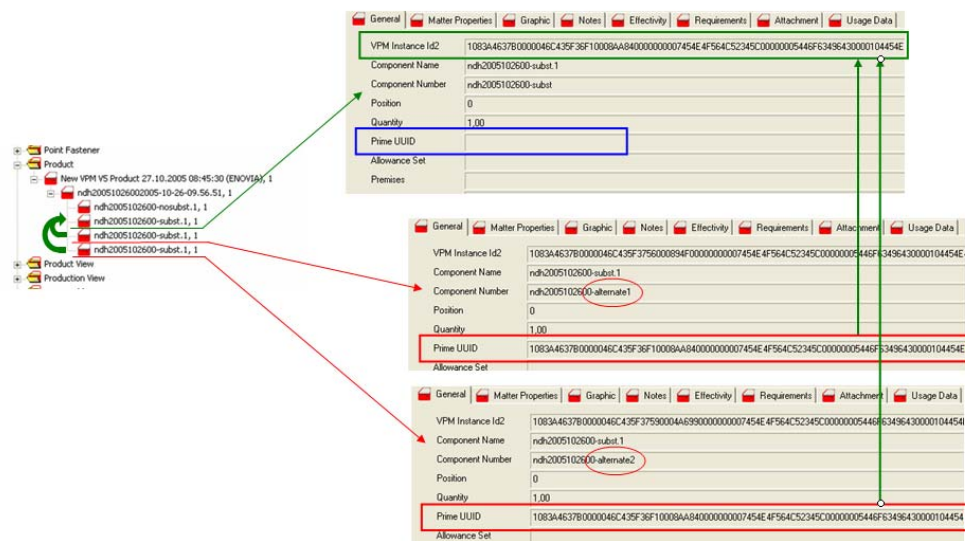


Figure 40: Substitute Parts

Substitute parts, are referring to the prime.

6.9 Support Management of In Work Design

The customer has the need to distinguish within a product structure between parts which have been *approved* and parts that are marked as so-called *work in progress* (WIP). In order to allow this, it's necessary that on both the Engineering Hub and the Manufacturing Hub respective effectivities and filter capabilities are given, the DELMIA – ENOVIA VPM V5 Connection has to transfer the necessary information. The connection transfers both *approved* and *in-work* effectivities from the Engineering Hub to the Manufacturing Hub so the customer is able to work with this data on the Manufacturing Hub.

- **WIP:** *work in progress*, in contrast to *approved*. Both are states in the life-cycle of an object.

The WIP token used to mark *in-work* effectivities is not added automatically to the token list. The customer has to add it manually in order to filter on these effectivities in the Manufacturing Hub.

The life-cycle status is taken from the reference.

Parts can be promoted into an *approved* state during their life-cycle in the Engineering Hub. This doesn't mean that they can't be changed later in time. Still, the new work is not yet approved and thus still in *in work* state.

In the ENOVIA client the customer will see two effectivity terms for such a part, one in the *approved* domain, the other in the WIP domain. For [example](#):

- R(1 - 100) in the *approved* domain
- R(1 – ∞) in the *work in progress* domain

When such a part is transferred over the DELMIA – ENOVIA VPM V5 Connection, a single effectivity will be created which contains a concatenation of the separate terms. The concatenation operator is "OR", the only logical operation possible. To the WIP term an additional term WIP is added through an "AND" operation.

The resulting effectivity term can be seen below. It is put into the XML file as attribute of the assembly relation instance (node CLINK in the XML file) and will be transferred from the Engineering Hub to the Manufacturing Hub.

Example:

```
<Attribute Name="Node Effectivity" Type="String">
<Value Value="(R(1 - •) AND WIP) OR (R(1 - 100)) AND NOT
WIP">
</Attribute>
```

In this example you see an *in-work* effectivity from one to infinity and an *approved* effectivity of R(1-100).

All effectivity terms have to match, i.e. the single effectivity terms, *approved* respectively *WIP*, must be the same. In the MH you should see a concatenation of the single terms. The concatenation operator is "OR".

The described feature is disabled by default. It has to be activated by setting the environment variable

```
ENOVIA_LCAIPD_WIP=1
```

on the export side (on the ENOVIA server for interactive transfers and in the environment in which ProductDataGen is run for batch transfers).

If the values are `0`, `no`, or `false`, the known DPE R15 behavior with only one effectivity, which resembles the in-work Part, will be transferred over the Connection.

6.10 Document-Object- and Document-Attachment Mapping

With R13 the default behavior to map documents into IPD objects has changed from product nodes to attachments.

Two attributes have been introduced which allow to identify the appropriate document in IPD as the “preferred” one.

The first one is on the link between part object and document.

```

- <Instance Name="(ENOVIALCA.DOCUMENT)Disk Brake---2003-3-27-11.17.41"
  Alias="(ENOVIALCA.DOCUMENT)Disk Brake---2003-3-27-11.17.41" Type="CATPDMObject"
  Uuid="ace4c2100000b40c3e82cf53000c014d">
+ <Attribute Name="$COID" Type="String">
+ <Attribute Name="$COMPID" Type="String">
+ <Attribute Name="S_NAME" Type="String">
+ <Attribute Name="MODEL_DESCRIPTION" Type="String">
+ <Attribute Name="C_TYPE_REP" Type="String">
- <Attribute Name="id" Type="String">
  <Value
    Value="10ACE4C2100000B40C3E82CF53000C014D0000000007454E4F564C52345C00000"
  </Attribute>
- <Attribute Name="type" Type="String">
  <Value Value="VPMTRepresentation" />
</Attribute>
+ <Attribute Name="V_externalID" Type="String">

```

Attribute "type", containing the type information, used for the ENOVIA LCA – IPD – DELMIA V5 interoperability

The type information will be available in Manufacturing Hub on the attribute “vpmttype”.

The second new attribute “Exposed” belongs to the actual document itself and indicates, that this document is the one to be taken into account by the DELMIA V5 – ENOVIA VPM V5 interoperability. The exposed attribute will be available in DPE on the attribute “vpmexposed” on the document instance.

```

- <Attribute Name="C_TYPE_REP" Type="String">
  <Value Value="CATPart" />
</Attribute>
- <Attribute Name="id" Type="String">
  <Value
    Value="102984960C0000675A3EF806B4000556110000000007454E4F564"
  </Attribute>
- <Attribute Name="Exposed" Type="Boolean">
  <Value Value="1" />
</Attribute>
- <Attribute Name="V_externalID" Type="String">
  <Value Value="" />
</Attribute>

```

Attribute "Exposed", containing whether this is the "preferred" document or not, used for the ENOVIA LCA – IPD – DELMIA V5

6.11 Transfer of Multi Value Attributes

(e.g. “V_discipline” in ENOVIA VPM V5)

The uses of Multi Value Attributes (MVA) are manifold, e.g. they are needed for the support of part types or the process engineer has to see which suppliers are defined for a certain part. Another important use case is the data sharing with suppliers where only a subset of all suppliers is allowed to receive the data. And this subset is defined in ENOVIA by selecting one or more dedicated entries from the complete list of suppliers in a customized part attribute.

As the export does not support multi value attributes natively another solution for providing visibility in the DELMIA tools has been found. The MVAs are concatenated to a (e.g. comma) separated list which can be stored in a String attribute. This way visibility and compatibility within the DELMIA tools is ensured.

Necessary Settings:

- 1) First of all, the environment variable ENOVIA_LCAIPD_MVA has to be set to “1” on the ENOVIA LCA server machine in order to activate the functionality at all for the export. The valuation must be done in the V5 environment file located under the CATEnv directory.
- 2) The attribute mapping of the DELMIA – ENOVIA VPM V5 Connection has to be modified in such a way that a multi-value attribute defined on ENOVIA side is mapped to a String attribute on DPE side. This can be done by a configuration file. The attribute, to be used as MVA, has to be listed in it.

Example: Add this line to your LCA.cfg file:

```
[attribute8]
general (name=attribute_30)
context (vpm=V_discipline;significant=true)
```

- In DPE open the configuration manager and change the settings of attribute_30 of type ergocomproductdefault:
 - Prompt: MVA
 - Control type: multiline edit
 - Display in browser and editor: yes
 - Page: general
- 3) On the machine where the PPRLoader process runs, some Registry settings have to be made for MVA (e.g. to display it properly with multiple lines in the DPE client) as follows:

```
[HKEY_CURRENT_USER\Software\DELMIA\ergoplan\PPRLoader\VPM]
"mvamultiline"="1"
"mvaseparator"=" "
"mvalinebreak"=" "
```

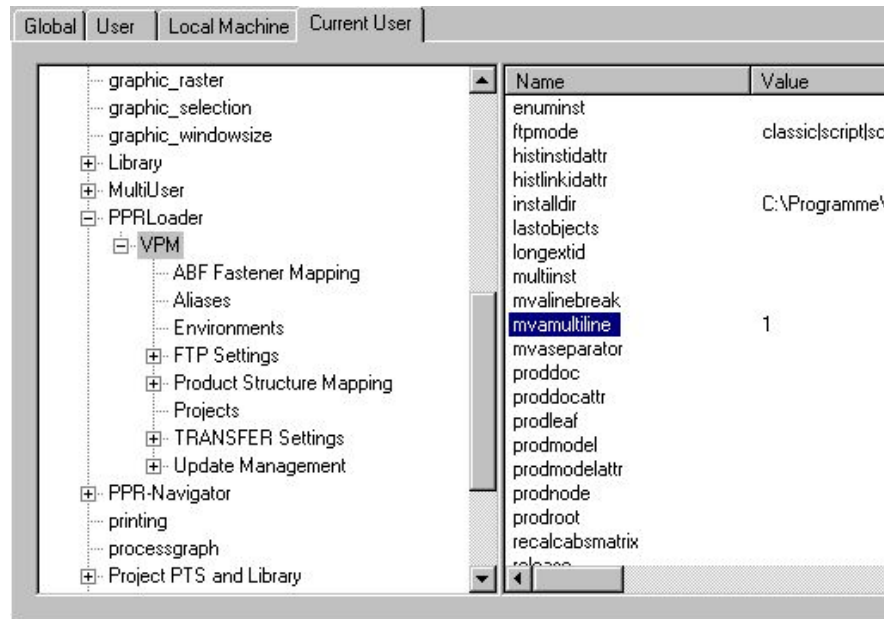


Figure 41: Settings

The default MVA line break character is “\r\n”. Without any changes every value of the attribute is shown in a separate line.

The default character used as a separator for MVA is “@@”. It can be customized by setting the environment variable ENOVIA_LCAIPD_MVA_SEP on the Export side. Please note that the registry key `mvaseparator` has to be changed accordingly in order to get the desired result.

The maximal size of string fields in a vanilla DPE database is 1000 characters. This can be pushed up to 4000 characters through customization of the database.

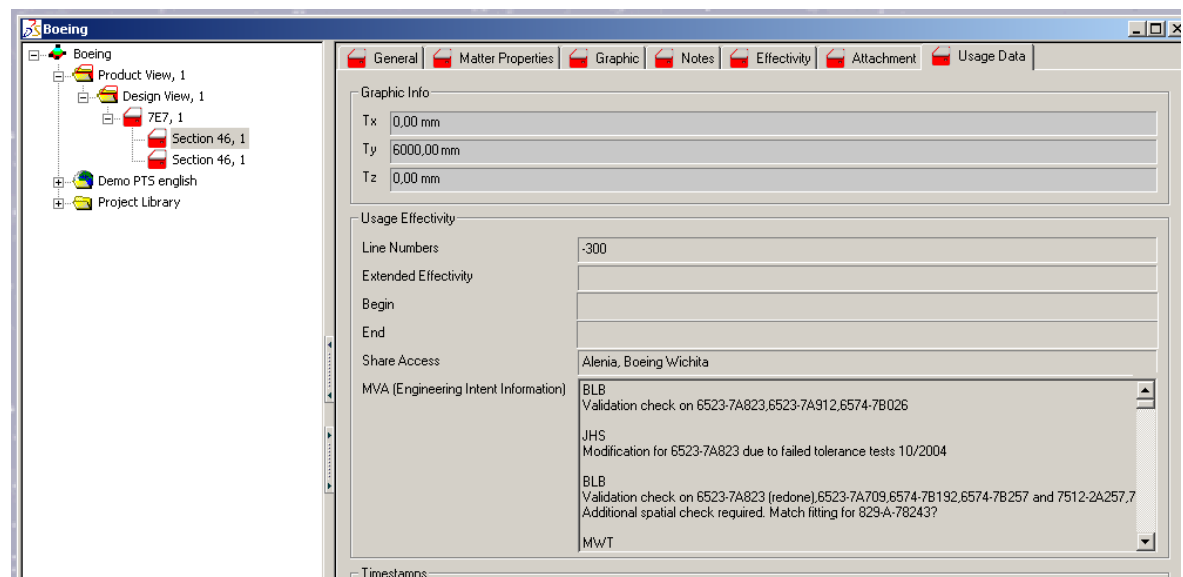


Figure 42: Customization of the Database

6.12 Update Protocol and Management

The Update Protocol includes functionalities that determine the status of an object or link after an import/synchronization. The identification of those changes is part of the DELMIA – ENOVIA VPM V5 Connection. Beyond that, all additional features such as gathering the modified objects and links (e.g., by script or finder) and notifying the end user (e.g., by changing the bitmap or creating reports) is not under the control of the PPRLoader application.

The Update Protocol uses a dedicated attribute “updatestate” to store the status of the last update/synchronization to mark each instance object.

6.12.1 Supported String Values/Attributes

Currently, only string values/attributes are supported. Please see also the registry file template for details.

The Update Protocol supports the following states:

- **New** (for objects and links, indicated by the value “new”)
All part instances and the corresponding links, that were not in the database before and therefore have been created by this import/update are marked “new”. Exception: after the initial import, the update status is empty.
- **Updated** (for objects, indicated by the value “update”)
In case one or more attribute values of a Part instances and Part References have been modified (different values in the database and the current XML export file), those part instances will be marked with status “update”
- **Repositioned** (only on links, indicated by the value “position”)
For a positional change (transformation matrix is different in database and XML export file), the link is marked with “position”
- **Deleted** (currently only for links, indicated by the value “delete” in case the delete mode is set to “notify”). If a link is not identified during the update/synchronization, the Connection assumes that this link has been deleted on ENOVIA VPM V5 side and marks it as “delete”.
- **Config** (for objects and links, indicated by the value “config”)
If the effectivity for a link or instance has changed, the Connection marks this at the corresponding object as “config”.
- **Geometry** (for parent objects of documents, indicated by the value “geometry”)
To identify those modifications, the importing PPRLoader application checks the “last modification” time stamp on the documents meta data section. In addition, the Connection checks in general, whether the document representing the geometry of a part has been changed/replaced. This is done by comparing the Document id of the “exposed” Document.

**Note**

This mechanism works only for links imported previously from ENOVIA VPM V5, i.e. the external ID is non-empty. Manually or otherwise created links in Process Engineer stay untouched.

The corresponding Part instance inherits this update state, in case its own status is empty.

**Note**

Per default only two attributes are transferred from ENOVIA VPM V5 to DPE side: the PART_NUMBER (on DPE side “shortname”) and the INSTANCE_ID (on DPE side “name”). Additional attributes (e.g. the reference ID) can be configured in the configuration file.

Example:

```
general(name=attribute_40)
context(vpm=PRODUCT.PART_LIST.S_NAME; significant=true)
```

6.12.2 Priorities of Update Status

Priority of Update Status for **parts**:

new + delete > geometry > update > position > config

Priority of Update Status for **links**:

new + delete > position > config

6.12.3 Significant Attributes

Please note that only significant attributes are checked for modifications to determine the update status “update”. By default, only the (Manufacturing Hub) attributes “name” and “nameshort” are configured that way (hard-coded, built-in significant attributes). To set an attribute to “significant”, use the following syntax in the ENOVIA VPM V5 config file (section):

```
Enovia;
[attributeN]
general (name=...)
context (vpm=...;significant=true)
```

Only attributes that have this *context* are checked, i.e. they are compared with the old values in order to mark objects as changed. To set all attributes to “significant”, you can use the following statement in the config section:

```
[config]
defaultcontext=significant=true
```

This applies the context significant to all attributes.

6.12.4 Customized Attributes for Update Status

It is possible to specify a different attribute than the default “updatestate” for the Update Status attribute for links (SubCompltem) and components (ErgoComp-Base). When doing so, please ensure that those attributes exist. Even though the status of links is transferred to the child object, it is especially important to verify that the attributes exist.

There are no free attributes (“attribute_1” to “attribute_55”) on links. For instance, a setting like:

```
HKCU\Software\Delmia\ergoplan\PPRLoader\VPM] "updatestateattr link"="attribute_19"
```

results in a Connection internal use of the links update state, but the state cannot be stored in the Manufacturing Hub database (on the SubCompltem object).

The set of available attributes can be checked in the DPE Configuration Manager Tool.

6.12.5 Modes for Deleted Objects

Since the distinction between update status for links and instances is confusing for the end-user and it is actually not required in a multi-instance scenario, the status for the link is also set on the instance – in case the instance update status is empty.

For objects deleted in ENOVIA VPM V5, PPRLoader supports four different modes:

- **“auto”** deletes automatically all links that are no longer in use. This is the default setting. The corresponding instances stay in the Manufacturing Hub database, to avoid data loss (component stays alive in project library).
- **“notify”** mark deleted components with the attribute “updatestate” set to “delete”, if an additional “propagatedelete” flag is set, all children of the sub-structure are marked as “deleted”, too
- **“attachment”** combination of modes “notify” for part components and “auto” for attachments (documents). Deletes physically attachments, product components are marked as “delete”.
- **“none”** neither deletes nor notifies the deleted/removed objects in DPE.
- **“deep”** The “deep” delete mode performs a real “delete” on the delete component (and all children within its sub-structure, if any). It combines the automatic removal of the product component from the structure and the effects on the sub-structure when using “propagatedelete”. *Please refer to the [Deep Delete](#)*

Select the mode using the following registry key:

```
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM] "deletemode"=" .
.."
```

To use the notify mode, set the value of this registry key to "notify" (or "0" or "false"). All other values will use the default mode, which is "auto".

For example:

"deletemode"="notify"

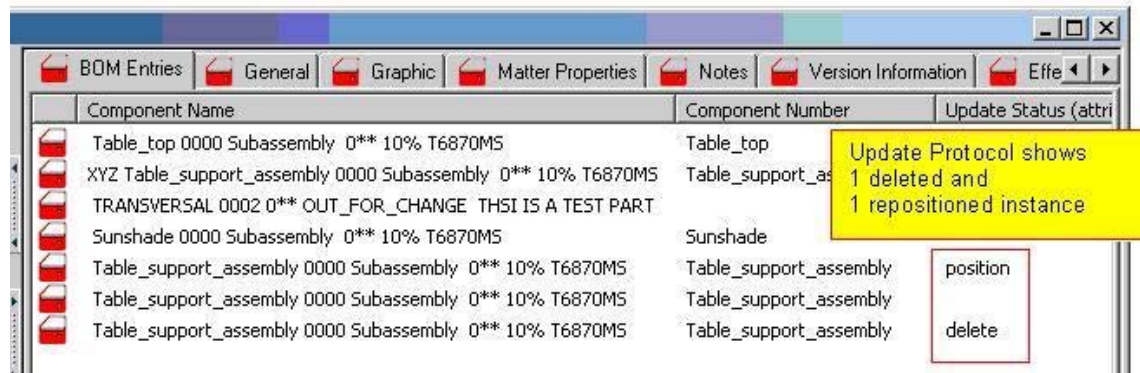


Figure 43: Notify Mode

Since the DELMIA – ENOVIA VPM V5 Connection cannot compare content, size, or modification date of a CGR geometry attached to a part, there is no dedicated update state for "geometry updated". Nevertheless, the gateway can be configured so that a comparison of the ENOVIA VPM V5 attributes "C_lastmod" or "C_modified" DOCUMENT objects is performed.

The following section demonstrates as an example the usage of the context token "significant" on those ENOVIA VPM V5 attributes, to setup the required behavior:

```
; ENOVIALCA
[attribute16]
general(name=attribute_27)
context(vpm=ENOVIALCA.DOCUMENT.C_lastmod;significant=true)
[attribute17]
general(name=attribute_28)
context(vpm= ENOVIALCA.DOCUMENT.C_modified;significant=true)
```



Note

It is not possible to SET the Manufacturing Hub attributes modification date "modificationdate" or creation date "creationdate" (since they are handled internally by the Manufacturing Hub database/the server). A comparison to track modifications therefore makes less sense, since the value from ENOVIA VPM V5 is not reflected in those attributes at all. Nevertheless, if you want to track changes on this attribute, store the ENOVIA VPM V5 value in one of the free default attributes ("attribute_1" to "attribute_55") by adapting the config file and compare the values from ENOVIA VPM V5 and Manufacturing Hub based on this. In addition, setting and comparing configuration attributes (like "effectivity.start", "tailnumber", ...) does not yield meaningful results. These attributes are handled a totally different way and must be handled separately.

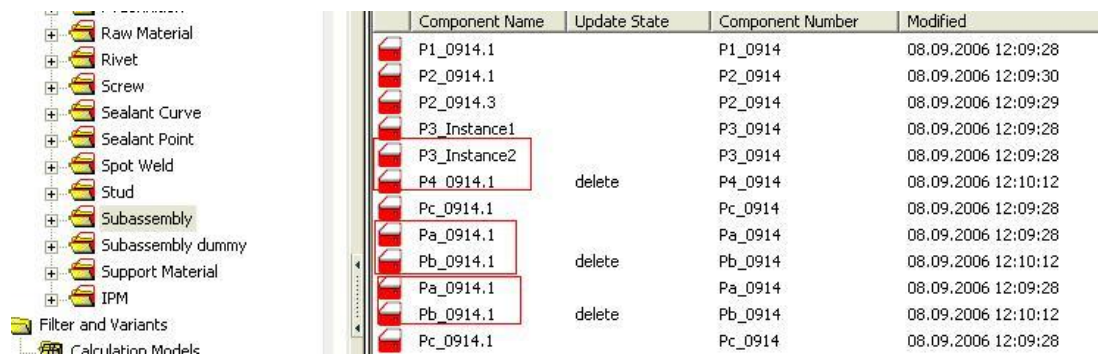
6.12.5.1 Deep Delete

The “Deep Delete” mode reduces the number of unused/out-dated objects in the MH dramatically. This avoids e.g. confusion for the user, when querying objects through DPE finder, since deleted objects are no longer found any more.

During the import/update of the product structure the DELMIA – ENOVIA VPM V5 Connection stores the list of deleted components (objects which are present in the MH database, but not in the XML export file) as entry points for the substructures to be deleted.

In difference to the current delete modes, where objects are just marked as “deleted” respectively only the first (root) component of a substructure to delete is removed from the product structure (by deleting the link/relationship to it), the new *deep delete* really deletes the entire substructure physically from the database.

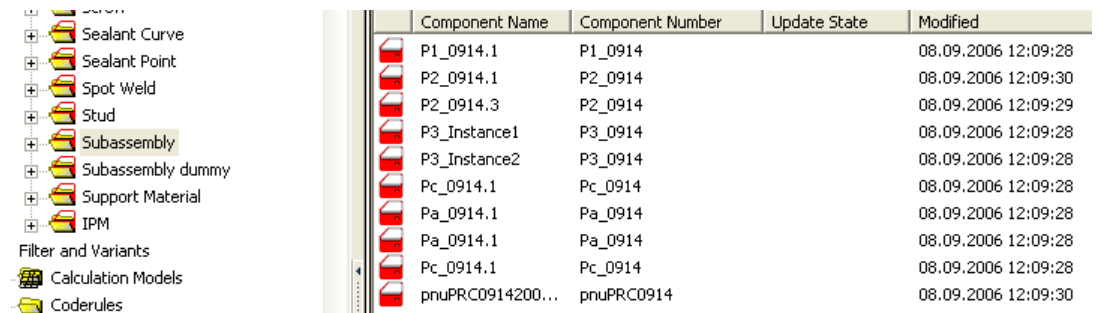
Image showing the previous behavior: the library contains the three “Subassembly” components marked as “deleted”.



| Component Name | Update State | Component Number | Modified |
|----------------|--------------|------------------|---------------------|
| P1_0914.1 | | P1_0914 | 08.09.2006 12:09:28 |
| P2_0914.1 | | P2_0914 | 08.09.2006 12:09:30 |
| P2_0914.3 | | P2_0914 | 08.09.2006 12:09:29 |
| P3_Instance1 | | P3_0914 | 08.09.2006 12:09:28 |
| P3_Instance2 | | P3_0914 | 08.09.2006 12:09:28 |
| P4_0914.1 | delete | P4_0914 | 08.09.2006 12:10:12 |
| Pc_0914.1 | | Pc_0914 | 08.09.2006 12:09:28 |
| Pa_0914.1 | | Pa_0914 | 08.09.2006 12:09:28 |
| Pb_0914.1 | delete | Pb_0914 | 08.09.2006 12:10:12 |
| Pa_0914.1 | | Pa_0914 | 08.09.2006 12:09:28 |
| Pb_0914.1 | delete | Pb_0914 | 08.09.2006 12:10:12 |
| Pc_0914.1 | | Pc_0914 | 08.09.2006 12:09:28 |

Figure 44: Library Subcomponents

Deep delete mode: no “delete” update states, subassemblies “P4_0914” and both “Pb_0914” have been deleted completely.



| Component Name | Component Number | Update State | Modified |
|------------------|------------------|--------------|---------------------|
| P1_0914.1 | P1_0914 | | 08.09.2006 12:09:28 |
| P2_0914.1 | P2_0914 | | 08.09.2006 12:09:30 |
| P2_0914.3 | P2_0914 | | 08.09.2006 12:09:29 |
| P3_Instance1 | P3_0914 | | 08.09.2006 12:09:28 |
| P3_Instance2 | P3_0914 | | 08.09.2006 12:09:28 |
| Pc_0914.1 | Pc_0914 | | 08.09.2006 12:09:28 |
| Pa_0914.1 | Pa_0914 | | 08.09.2006 12:09:28 |
| Pa_0914.1 | Pa_0914 | | 08.09.2006 12:09:28 |
| Pc_0914.1 | Pc_0914 | | 08.09.2006 12:09:28 |
| pnuPRC0914200... | pnuPRC0914 | | 08.09.2006 12:09:30 |

Figure 45: Deep Delete Mode

The deletion of those components and their substructures is performed AFTER the regular product structure update/import traversal (similar to the recalculation of absolute effectivities e.g.).



Note

Please note, that with delete mode set to “deep”, the “propagatedelete” registry setting is ignored.

Customization

The Deep Delete mode is switched off by default. It must be enabled by setting the following registry key:

```
[HKEY_CURRENT_USER\Software\DELMIA\ergoplan\PPRLoader]"deletemode"="deep"
```

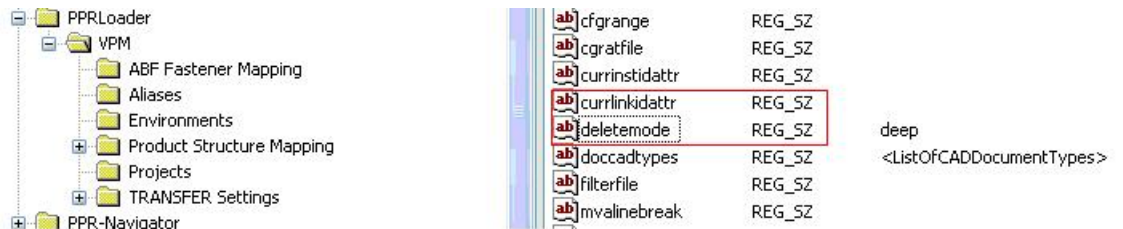


Figure 46: Customization

Limitation

- For projects which are under MCM control, there might be product components which can not be deleted due to MCM relations referring this object.
- If the deep delete mode is enabled, the “propagatedelete” registry setting is ignored.
- Only the entry point of a deleted substructure is logged in the PPRLoader.log file. The children of this root component are not traced anywhere.

6.12.6 Recalculation of Absolute Instance Positions

To calculate the correct absolute instance positions, which are necessary to display the products geometry correctly in the context of an MBOM, the PPRLoader calls the server method “recalcabsmatrix” after the import/update. This will make the server update the absolute instance positions, based on the relative transformations from each link.

Since this recalculation can take quite a while (the entire product structure needs to be traversed), it is possible to skip this, by using the following registry setting:

```
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM]"recalcabsmatrix"="0"
```

To skip the recalculation use the a value of “0”, all other values will use the default mode, which is to recalculate the instance positions.

The same behavior can be accomplished by using the command line options – recalc respectively –norecalc.

```
pprloader -vpm ... -project ... -norecalc
```

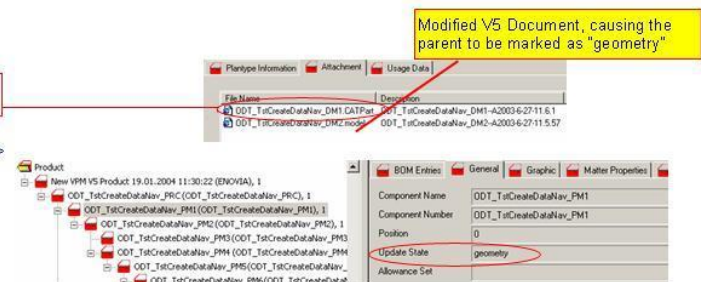
Please note, that the command line option will overwrite the behavior from the registry setting without changing the registry, for this particular import/update run.

6.12.7 Update Protocol for Geometry

From DPE R5.14 on the Connection identifies geometrical changes of V5 documents. To identify those modifications, the importing PPRLoader application checks the “last modification” time stamp on the Documents meta data section and marks the corresponding document attachment in the Manufacturing Hub as “update”. In addition, the corresponding parent node will be marked with the update status “geometry”. The Connection checks in general, whether the document representing the geometry of a part has been changed/replaced. This is done by comparing the document id of the “exposed” document.

In both cases, the update status for the parent part of the document is set to “geometry”, if no other modification/update status is identified before (e.g. positional change “position”). \

```
<Instance Name="(ENOVIALCA.DOCUMENT)ODT_TstCreateDataNav_DM1-
Alias="(ENOVIALCA.DOCUMENT)ODT_TstCreateDataNav_DM1-
Uuid="5bdc66ef0000db43efbd7ae000013aa">
+ <Attribute Name="$COID" Type="String">
+ <Attribute Name="$COMPID" Type="String">
- <Attribute Name="S_NAME" Type="String">
  <Value Value="ODT_TstCreateDataNav_DM1" />
</Attribute>
+ <Attribute Name="MODEL_DESCRIPTION" Type="String">
+ <Attribute Name="C_TYPE_REP" Type="String">
+ <Attribute Name="id" Type="String">
+ <Attribute Name="Exposed" Type="Integer">
- <Attribute Name="C_created" Type="String">
  <Value Value="2003-6-27-11.5.42" />
</Attribute>
+ <Attribute Name="C_modified" Type="String">
  <Value Value="2003-6-27-11.6.1" />
</Attribute>
+ <Attribute Name="V_externalID" Type="String">
```



Functionality Scope and Limitations

- The update status of the document is not set to “geometry”, but to “update”. Only the update status of the parent object contains “geometry”.
- In case of modifications, the update status of the parent objects is “update”, which has a higher priority than “geometry”. E.g. when you use the default attribute mapping, where the name of the part contains a modification time stamp. In this case, the part’s name is also modified, since a modification of the geometrical representation touches also the parent part.

Example

DELMIA – ENOVIA VPM V5 Connection

- 1) Start DPE.
- 2) Create a new project (“New Project”), using the plantypeset “Default-V5R18”.
- 3) Enter “ENOVIA VPM” as project name and “EV5” as project number, specify “Standard” as coderule mode, click “OK”.
- 4) Commit changes (diskette icon!).
- 5) If PPRDaemon is not running, start it.

- Check that project was created correctly (e.g. by typing “pprloader –project” in a command shell).
- 6) Start ENOVIA VPM V5 and log in.
- 7) Select appropriate PRC and open a product in the Product Editor.
- 8) Select a part inside the product structure.
- 9) Open it in CATIA V5 and create the part’s geometry.
- 10) Save the Part back in ENOVIA VPM V5.
- 11) In ENOVIA VPM V5 select the PRC and “Send to Manufacturing Hub” command.

From the displayed list of projects, select the one, just created

- Check that two files, \$PRCNameAndTimeStamp...”.xml” and “Instance_“ \$PRCNameAndTimeStamp...”.xml” are created on ENOVIA VPM V5 side AND transferred via FTP/HTTP/HTTPS to Windows / Manufacturing Hub.
 - Check that the selected PRC from ENOVIA is transferred to the Manufacturing Hub side. Check that the V5 documents are transferred as document attachments to Manufacturing Hub.
- 12) In ENOVIA VPM V5 select the same part again.
 - 13) Send it to CATIA V5 and change the part’s geometry.
 - 14) Save back to ENOVIA VPM V5.
 - 15) Select “Send to Manufacturing Hub” command.

From the displayed list of projects, select the one, just created

- Check that the selected PRC from ENOVIA is transferred to the Manufacturing Hub side. Check that the Update status of the corresponding parent object is set to “geometry”.

| BOM Entries | | General | | Matter Properties | | Notes | | Effectivity | | Attachmen | |
|------------------|--------------------------|---------|--|-------------------|--|-------|--|-------------|--|-----------|--|
| Update State | geometry | | | | | | | | | | |
| Component Name | ODT_TstCreateDataNav_PM1 | | | | | | | | | | |
| Component Number | ODT_TstCreateDataNav_PM1 | | | | | | | | | | |
| Position | 0 | | | | | | | | | | |
| Quantity | 1,00 | | | | | | | | | | |

Figure 47: Set Corresponding Parent Object to Geometry

6.13 Initial Import

The first time a product is imported into DPE, a Top Level Product Folder will be created in the Project Library. Under this product node, the top level subassembly representing the Product Root Class (PRC) will be created

Importance of the Subassembly node, representing the Product Root Class

The existence of the Subassembly node representing the PRC indicates whether to perform an update or an initial import (create everything new). I.e. in case of the absence of the PRC node in the Project Library, the import mechanism tries to create a new one, e.g. it starts an “initial import” (that creates a new Top Level Product Folder, the PRC node and the complete product structure).

The product folder itself can be considered just as a structuring element. Date and time of the import are stored in the name of the product folder. This gives a better orientation to the user when navigating across complex project structures after an import.



Note

You can delete the product folder (with option “component, flat”), this does not influence the update mechanism. On the other hand, the update mechanism will not create a new Product Folder since it takes its information only from the node representing the PRC.

If you delete the node, representing the PRC in DPE, the update process cannot work correctly. In this case it is recommended to delete the whole product with all components and to start a new initial import.

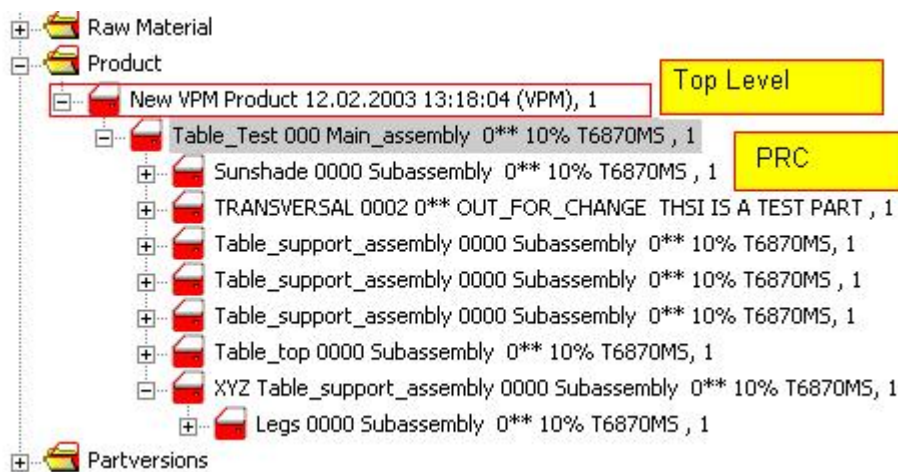


Figure 48: PRC in DPE



Note

From R15 on, the Update Status of the product and the subassembly after the initial import is not “new” but empty.



Note

When transferring a PRC to a CMC project into DELMA Process Engineer, ensure that the “Has versions” attribute of the Product plantype is set to “No”.



To accomplish this, open the Library, navigate to the Product plantype and select Edit from the context menu.

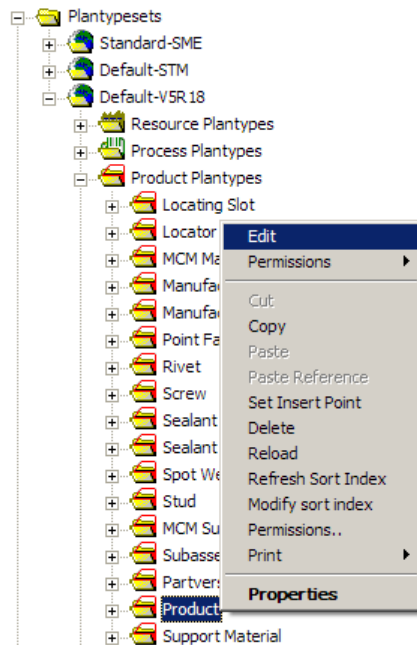


Figure 49: Select Edit from the Context Menu

The Configuration Tool opens and the attributes for the selected plantype are displayed. Select “No” for the Has versions attribute.

| Flags | |
|---------------|----|
| Has versions | No |
| Is searchable | No |

Figure 50: Config Tool

6.14 Partial Product Data Transfer

With this functionality it is possible to select single sharing objects in a product structure and update. The main objective is to allow the possibility to re-create the context of the extracted instance/partial product structure. This will allow a split of an entire product into areas of interest which are updated/synchronized more often than the rest of the product.



Note

The application of this functionality is limited to product structures, which are not completely flat.

It is recommended to use this functionality only in batch mode.

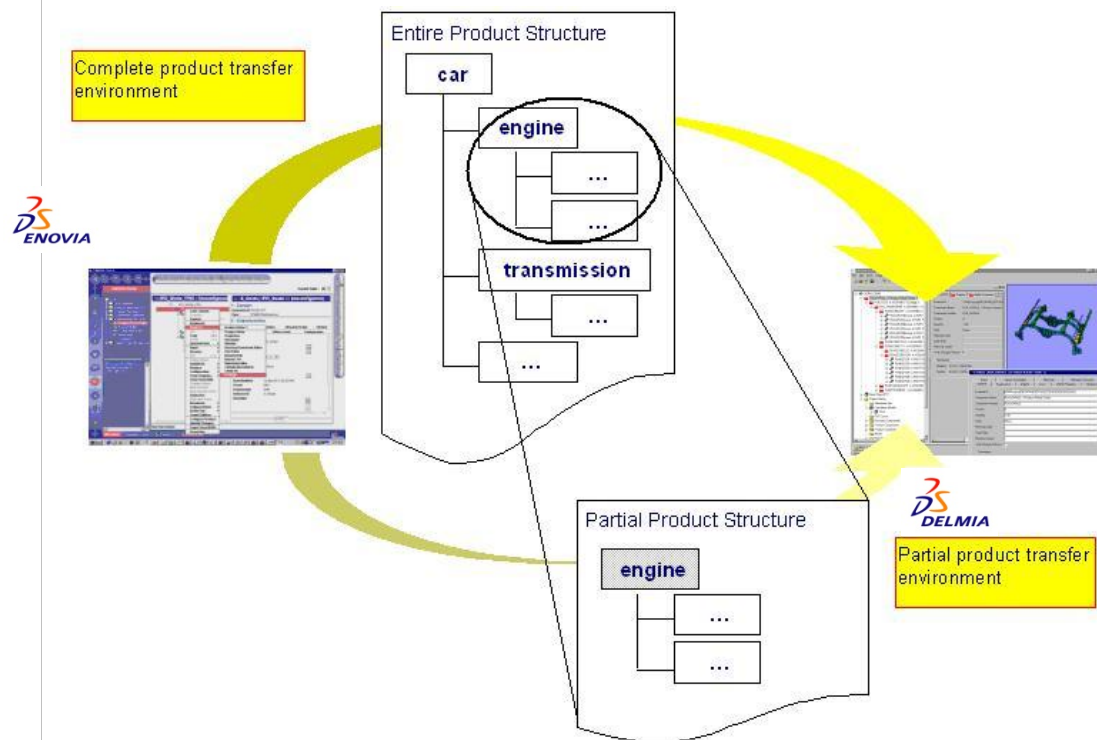


Figure 51: Data Transfer

In difference to the full product XML file, the generated XML file will not contain any sibling of the requested instance. Still, the partial data transfer XML file contains all required instances and links up to the root (PRC) to allow the correct identification of the selected instance through the path of link ids.

ENOVIA VPM V5 – PRC/Product Structure

Manufacturing Hub

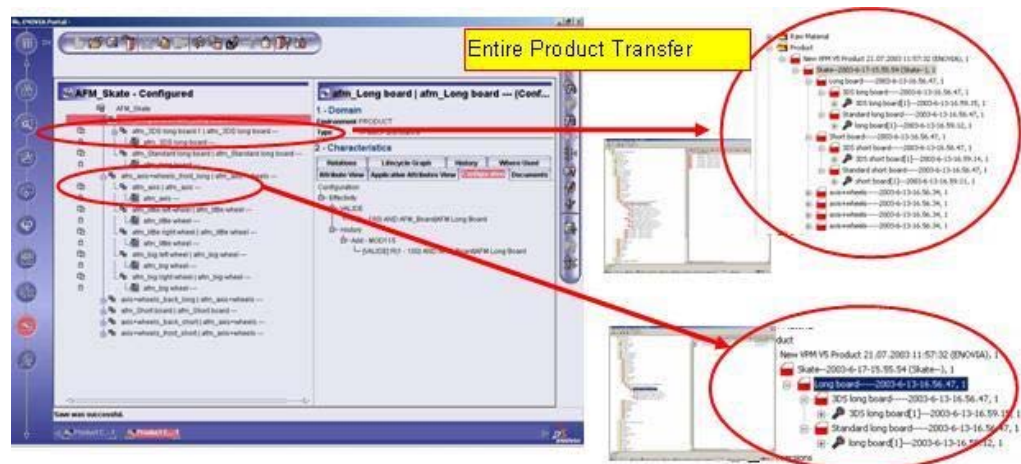


Figure 52: Partial Product Transfer

Note

All parent nodes up to the root level (PRC) are imported/updated on Manufacturing Hub side.

The Update Protocol will only work correctly on the transferred section of the product structure. All missing data, i.e. data that is out of scope for this partial data transfer is **NOT** under control of the Update Protocol, to avoid e.g. the deletion of unsent data (or marking those instances as deleted).

This functionality is for define and manage the set of required sub-structures of interest, which will allow the complete transfer of data in portions.

```
pvrloader -vpm C:\Temp\IncrementalAT0-2.xml -project INC -
partial
```

Example

- 1) Create a new project ("New Project") using the Plantypeset "Default-V5R18".

2) Enter “ENOVIA VPM” as project name and “EV5” as project number, specify “Extended” as coderule mode, click “OK”.

3) Commit changes (diskette icon!).

4) Open a command shell (DOS box), go to the installation directory (...\\DELMIA\\PPRClient\\program\\bin) and enter the following command:

```
pprloader -vpm ...\\ODT_TstReferenceTree_PRC-2003-6-20-12.26.51.xml -project EV5
```

- Check that the project was created correctly (e.g. by typing “pprloader – project” in a command shell).

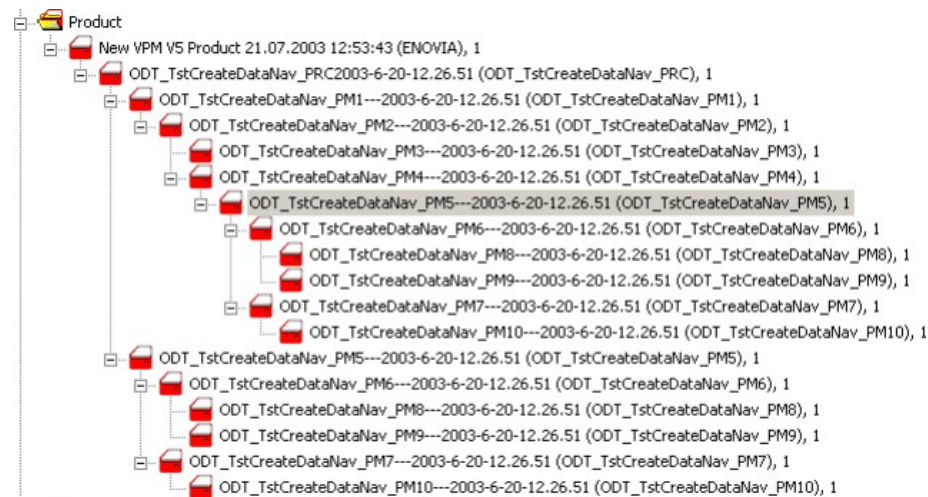
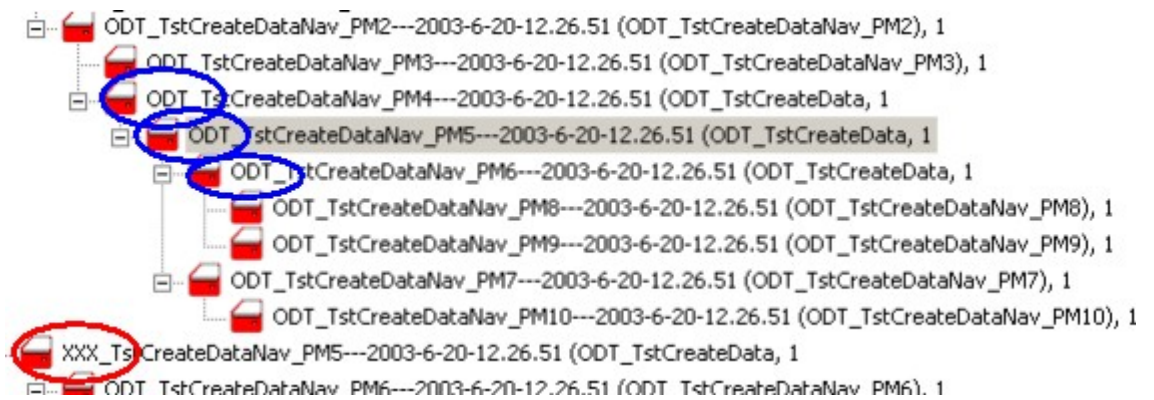


Figure 53: Project Structure

1) Enter the following command.

```
pprloader -vpm ...\\ ODT_TstPartialDataTransfer_PM5-2003-7-1-11.31.54.xml  
-project EV5 -partial
```

- No new instances were created, but only the appropriate instances in the Manufacturing Hub’s product structure were updated:
The selected instance ODT_TstCreateDataNav_PM5 and its child
- ODT_TstCreateDataNav_PM6 are updated as part of the selected data section transferred.
- ODT_TstCreateDataNav_PM4 is transferred, since it’s part of the path from the PRC to the selected instance.
- The second instance of ODT_TstCreateDataNav_PM5 is not updated, since it’s not referred in the requested area-of-interest for this partial product transfer.
- After partial update.

**Figure 54: Product Transfer**

6.15 Block Data Transfer

In R18 the maximum capacity of the exporter ProductDataGen is about 50000 instances per 1 GB virtual address space. The address space of such an application is technically constrained to a maximum of 4 GB. Depending on the operating system, an even lower value, e.g. a maximum of 2GB virtual address space may be available.

- 1) Through a first change in the way ProductDataGen works the number of instances per GB can be doubled for product structures which have a broad structure (assuming 50% of all parts are first level parts).
- 2) In a second step a block approach is implemented which allows a better parallelisation of the import of a product structure into the Manufacturing Hub and hence a reduction of run time. This can be achieved because with today's multiprocessor/multicore machines it is more efficient to import a set of medium sized XML files (e.g. 5x 400MB file size) instead of one very large XML (e.g. 1x2GB file size).

In regards to memory consumption the bottleneck for ProductDataGen export is the traverse process. This is a single process which reads the information of all Part Instances (II) and, depending on defined set of filters, also their respective Part Masters (PM) and Part Versions (PV). This leads to the afore-mentioned upper boundary for the maximal number of instances that can be handled.

Therefore, we now change the process in such a way that we start a new sub-process whose only task is to export information about the Part Instances on the top-level (Root Item Instances) so they can be handled by later processes. We call this new process a Block Id Generator, because it collects the Ids of the Part Instances and writes it into files which contain a block each. The size of each block can be customized. This process now is the memory bottleneck but it pushes the number of item instances higher.

Transferring big product structures can take quite a while. Parallelization can help to reduce the time necessary. To this end a block approach can be adopted. This is different from and not to be mixed up with the partial mode already existing in ProductDataGen. The information created by the Block Id Generator can be used as input to a block export which each only handles the slice of the product structure. As an effect instead of two files, reference and instance, which describe the product structure a set of files is generated which have to be imported in order to get all information. The various import processes can be run in parallel.

The change to first write out the identifier of the root Item Instances has no effect at all to the external view. It is transparent to the user. It is used in interactive mode also.

If the block mode is activated, see Customization below, a set of files is generated which look like

- PRC-DATE.xml, Instance_PRC-DATE.xml: The PRC with link information
- PRC-DATE_0001.xml, Instance_PRC-DATE_0001.xml: Data of the first block
- PRC-DATE_0002.xml, Instance_PRC-DATE_0002.xml: Data of the second block

In order to get an update of the complete product structure the whole set of files has to be imported. At the moment a separate PPRLoader process has to be used for each generated pair of instance and reference file. The resulting structure in the Manufacturing Hub is independent of the order this set of files is imported. It is not required to import the PRC and link information at the beginning or the end of the set.



Note

Block split cannot be used in interactive mode.


In order to influence the block operation mode a new global property has been added to the configuration file. It is called "block-operation". It understands two attributes.

The first, required attribute is named "size". It describes the number of Root Item Instance Identifier in each block. The default value is 10000.

The second, optional attribute is called "split". It allows the user to change the default behavior of generating one big XML file into a set of files.

Here are the contents of a sample configuration file:

For example if a PRC in the Engineering Hub has 30000 root instances below it



```
<?xml version="1.0" encoding="UTF-8"?>
<configuration xmlns="urn:ds:ENOVIpIntegration">
  <global>
    <properties>
      <block-operation size="10000" split="yes"/>
    </properties>
  </global>
</configuration>
```

(which also may have sub structures beneath them) the default block size of 10000 will lead to the creation of three pairs of reference/instance XML files if the "split" attribute is set to "yes" (30000/10000=3).

Note

The update state of the block mode will be "new" (in case the block mode import is done in an empty project) for the initial load case.

6.16 Incremental Update

To reduce the amount of data transferred between ENOVIA VPM V5 and the DELMIA Manufacturing Hub through the DELMIA – ENOVIA VPM V5 Connection, it is now possible to transfer only the delta between a past data transfer and the current image of the ENOVIA VPM V5 database.



Note

It is strictly recommended to use this functionality only in batch mode.

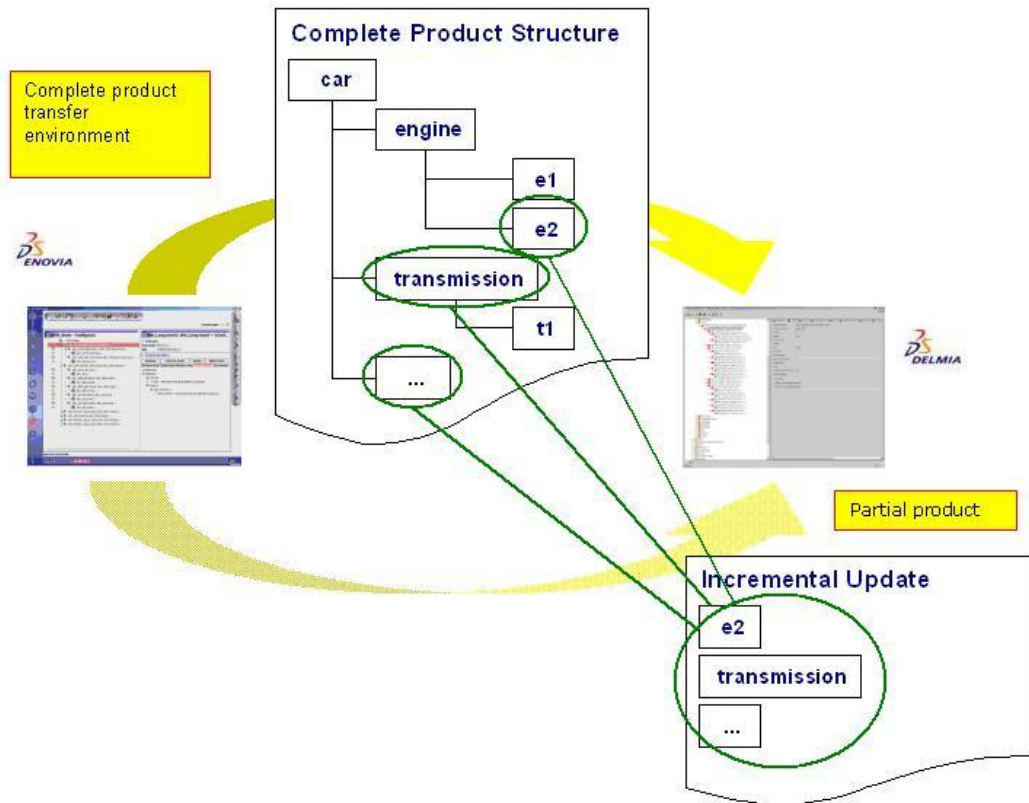


Figure 55: Data Transfer

In difference to the full product XML file, the generated XML file will not contain *all* data, but just the modified objects from the ENOVIA VPM V5 database, within the context of the select PRC.

From R13 on the Connection can identify modifications in the ENOVIA VPM V5 database and instances and objects, independently from the transfer of a structural / hierarchical context. If an incremental update of the PRC's product data is performed, the Connection can identify the objects to be updated.

It is not possible to synchronize data between EH and MH correctly, when a part instance has been deleted in an unconfigured EH product structure (unconfigured cut operation). It is recommended to use incremental update in configured PRCs only.

ENOVIA VPM V5 – PRC/Product Structure

Manufacturing Hub

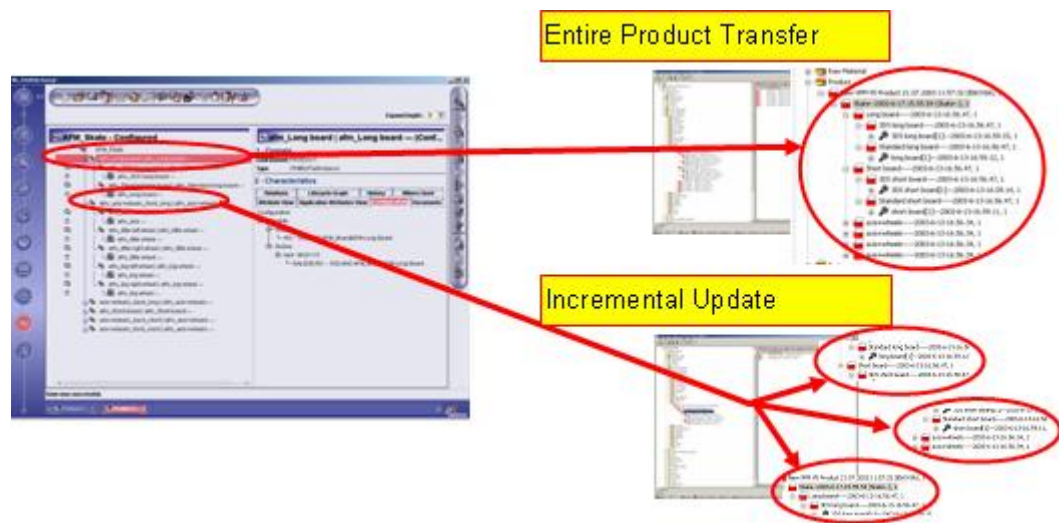


Figure 56: ENOVIA VPM V5 – PRC/Product Structure

The following object types from ENOVIA VPM V5 are tracked by their modification date:

- Item Instances
- Parts References (Part Versions)
- Assembly Relations (Links)
- Documents (Document Revisions)
- Options (Specifications)
- Configuration Handlers and Filters

An incremental XML file can be identified through the fact, that the contained structure is not complete. In some cases, this identification mechanism fails, since the structure seems complete, even if it is only an incremental file. For this reason please use the incremental update only in batch mode.

**Note**

The incremental update is not a replacement for partial data transfer. Incremental update serves different intentions and purposes and might be used in parallel to improve the overall transfer performance.

6.16.1 User Interface

Export

On the exporting ENOVIA VPM V5 Engineering Hub side, the user has to specify the date, since all modifications should be gathered. Due to the interactive and batch transfer mode, the user needs to do that either in the ENOVIA VPM V5 client through a dialog.

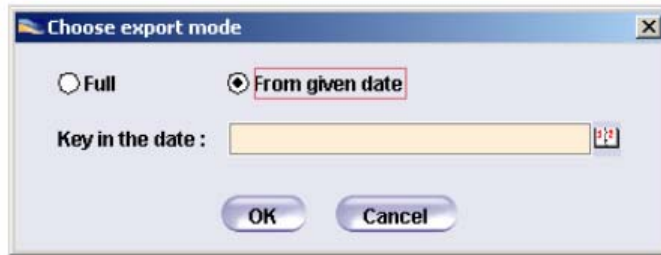


Figure 57: Expoer Mode

or when running the ProductDataGen application with an additional command line parameter

```
ProductDataGen incremental <productId> <modificationDate>
```

where the format of the modification date needs to match YYYY-MM-DD-hh.mm.ss, e.g.

```
ProductDataGen incremental prc2 2005-01-17-11.00.00
```

This will include all modifications done since January, 17th 11AM in the ENOVIA VPM V5 XML export files.

Import

Even though there is an automatic detection, whether the transferred XML export files represent a complete data transfer or an incremental update, it is strongly recommended to explicitly force the usage of the incremental algorithms. This can be done through the command line parameter `-incremental`.

```
pprloader -vpm <xmlFile> -project <projectId> -incremental
```

e.g.

```
pprloader -vpm C:\Temp\prc2-2004-08-11-12.42.32.xml -project P2 -incremental
```

6.16.2 Techniques of Interactions

Export

To extract an incremental XML file from the ENOVIA VPM V5 side, please use the *ProductDataGen* extraction tool, provided along with the regular ENOVIA VPM V5 setup. The syntax to extract a PRC in incremental mode is:

```
ProductDataGen incremental $ProductId $PreviousExportDate
```

e.g.

```
ProductDataGen incremental IC_1 2003-10-17-11.20.00 ...
```

Please see the usage and help of the *ProductDataGen* for more details.

It is the responsibility of the end user/administrator to maintain a complete gapless chain of incremental data transfers, i.e. it's up to the end user/administrator to check, that the specified time stamp matches the time of the last data transfer.

Please note, that any changes done before are not part of the transfer.

A chain of exports using batch export could look like this.

```
t0 ProductDataGen ... (initial load, i.e. complete transfer)
```

```
t1 ProductDataGen incremental $t0 ... (to detect all changes since t0)
```

```
t2 ProductDataGen incremental $t1 ... (to detect all changes since t1)
```



Note

Incremental XML export files are not useful for initial loads.

Import

On the MH import side, it is not required to specify the modification time again. Still, as already mentioned, it is recommended to use the `-incremental` command line option to force incremental algorithms to be used. Please note that this only applies to batch transfers. It is not possible to control the import behavior in interactive scenario.

To force the PPRLoader application to treat this file as incremental, use the `-incremental` command line option, which will make PPRLoader apply the appropriate synchronization algorithms:

```
pprloader -vpm $IncrementalXMLFile -project $ProjectIdOr-ShortName
```

```
-incremental
```

e. g.

```
pprloader -vpm C:\Temp\IncrementalAT0-2.xml -project INC -incremental
```



Note

The incremental update is not supported for unconfigured PRCs on ENOVIA VPM V5 side.

6.17 Support of Product Specifications and COPS

(COPS = Carry Over Product Specifications)

ENOVIA VPM V5 allows the usage of products inside other PRCs. This helps the user to better structure huge products and in addition to reuse products in several different contexts, without multiplying the data.

A Product Specification is a resolved (filtered) product within a parent PRC. Since the Product Specification (PS) is already resolved, it is not possible to apply a different configuration to it in the context of the parent PRC. Product Specifications are used when a particular instance of a sub-structure is fixed in this specific context, e.g. since the sub-structure is under the responsibility of a supplier.

Product Specifications are managed by the regular Connection algorithms. There is no specific treatment and handling of Product Specifications required.

In difference to that, Carry Over Product Specifications (COPS) are Product Specifications with an empty filter applied, meaning that the concrete configuration is *carried over* from the parent PRC. The parent PRCs filter is used to also resolve the COPS sub-structure. In this context a COPS structure is *not* fixed and depends on the specified filter of the root PRC.

The feature allows the user to split a structure during the export from the Engineering Hub, import the Parts in the Manufacturing Hub in arbitrary order, and receiving the same (instantiated) structure.



Note

COPS require a specific ENOVIA VPM V5 customization. The ENOVIA VPM V5 default installation does NOT support COPS. For more details, please refer to the ENOVIA VPM V5 documentation directly.

Example:

- In the following example COPSs are used to reference PRCs like sections by **AAA**. COPSs can also be used within other PRCs like “Wing” to “Sub-Wing-1”.
- Different positions depending on effectivities: Section 46 has two different positions relative to AAA. For the short plane (effectivity: -300) the X-position of Section 46 is 10. For the long plane (effectivity: -400) the X-position of Section 46 is 50.
- Product Specification on Product: the PRC “Lavatory” is referenced through different Product Specifications depending on effectivities.
- The AAA PRC contains so called “Fake parts”. The purpose of these Fake parts is to hold engineering intents between Parts of different sections.
- The AAA PRC contains system components for electrical and other purposes.

Example

Product structure with COPS (AR = Assembly Relation; II = Item)

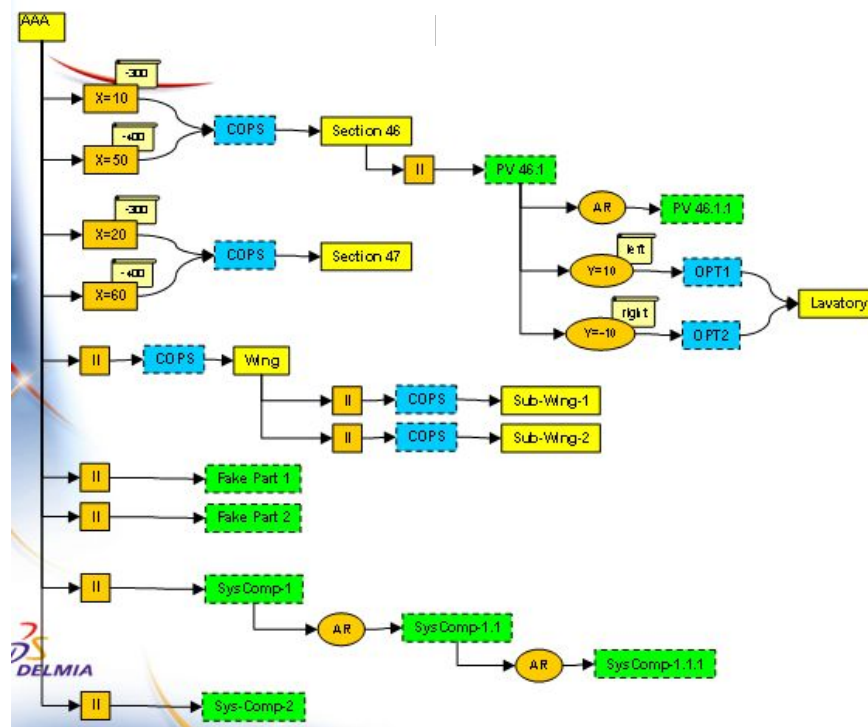


Figure 58: Product Structure with COPS

The previous behavior of the DELMIA – ENOVIA VPM V5 Connection was that the Manufacturing Hub Import created a full expanded product instance structure in Manufacturing Hub. If for example a product component was referenced two times in Engineering Hub then the product component was created two times in Manufacturing Hub. Together with the product component the whole sub tree that belonged to the product component was duplicated. The reason for this behavior was that Processes planned in Manufacturing Hub are linked directly to Product components. If two different Processes exist that do different things with the two times referenced Product component then the Product component must exist two times in Manufacturing Hub to be able to store links between the Processes and the Product component in Manufacturing Hub.

From R15 on the Engineering Hub Export recognizes a new environment variable ENOVIA_LCAIPD_COPS (Exclude COPSs during the transfer).

If the Engineering Hub Export is started with this option activated, the transfer stops at COPS and does not transfer the omitted PRCs.

Up to R17 for every not written object listed above an entry was written into a protocol file CATPRF – one for each parent PRC containing item instances referencing COPSs. Each file contained all the information of the not transferred links and PRCs. After all PRCs have been imported into the Manufacturing Hub it was the task of an administrator to manually re-create a link and add the information stored in the CATPRF-file.

This information was the guide for the user to manually transfer the omitted PRCs and linking the complete structure in the Manufacturing Hub.

6.17.1 Improved Handling of COPS

From R18 on this is no longer necessary. All information is now contained in the product structure files themselves.

The improved handling of COPS does not remove any feature of the DELMIA – ENOVIA VPM V5 Connection.

If the configuration option to split the structure at COPS is activated, c.f. the configuration file in the Related Material section, the child product structure is not exported with the parent and needs to be exported separately. The child PRC (only the node, none of its sub structure) and the link towards it, i.e. the COPS, are exported together with the parent structure. This is similar to having only the PRC node itself defined in the Engineering Hub, but no structure yet. Through this change the information at the link is stored in the normal structure files and no additional file has to be written and examined.

Of course, it is still necessary to export child product structures and import them separately. The order of export and import does not matter, i.e. it makes no change whether the children are transferred prior or after the parent. The child PRC will be queried for and used if it already exists in the Manufacturing Hub.

The update state of the child PRC is only changed when the parent structure is imported. The update state is not touched if the child structure itself is imported (or updated). For example, if we have two PRC, PRC1 and PRC2 of which PRC2 is linked into the structure of PRC1.

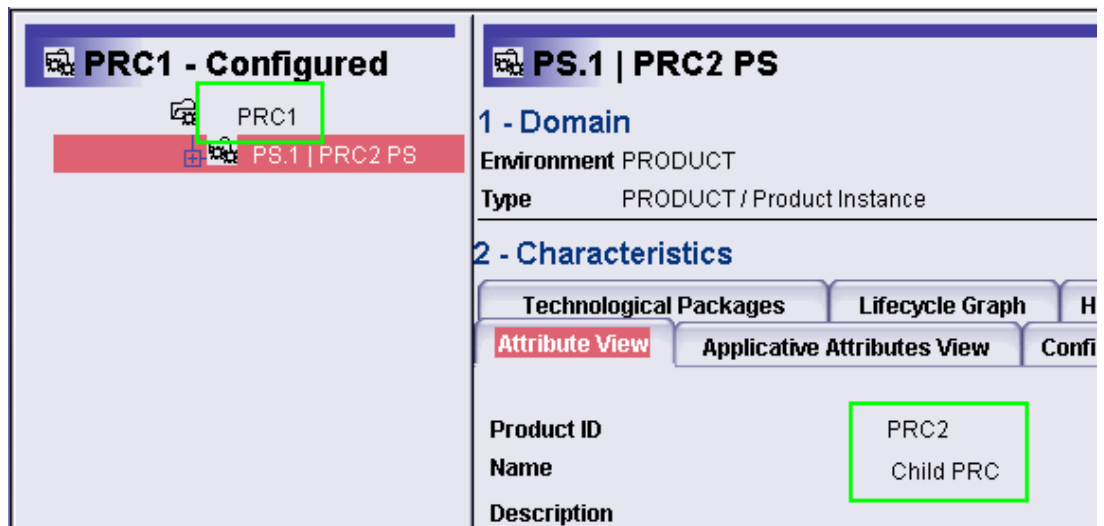


Figure 59: PRC Structure

If PRC1 is imported into the Manufacturing Hub first and PRC2 is imported into the Manufacturing Hub afterwards, the update state of PRC2 is not changed by the second import.

Configuration

The ability of the export to split a product structure at COPS must be configured. This is normally done through a configuration file similar to

```
<configuration>
  <global>
    <properties>
      <stop-at-cops enabled="yes"/>
    </properties>
  </global>
</configuration>
```

The configuration has to be addressed through the environment variable
 ENOVIA_LCAIPD_CONFIGURATION_FILE=/absolute/path/to/config.xml
 1

Client Process Step by Step:

Target is to transfer a product structure that corresponds to the principle structure shown in figure 1 from Engineering Hub to Manufacturing Hub and to have a product structure in Manufacturing Hub that fulfils the customer's need: a section appears only once in Manufacturing Hub to be able to do the process planning for this section.

0: Activate the exclusion of COPS transfer by setting the environment variable ENOVIA_LCAIPD_COPS to "1".

1: In the ENOVIA Product Class View create two configured PRCs named ParentPRC and ChildPRC1. Add a COPS to ChildPRC1.

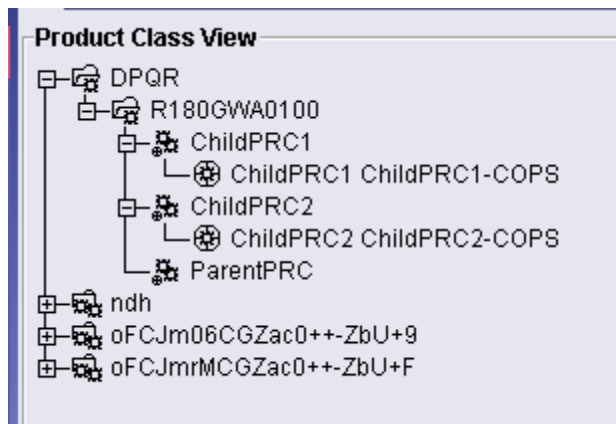


Figure 60: Product Class View

0: Send ChildPRC1 to the Product Editor and add two first level instances to it.

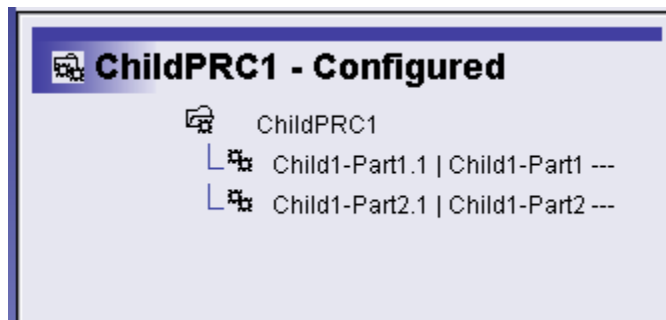


Figure 61: Child PRC1 Configured

0: Send ParentPRC to the Product Editor and add a first level instance to it.

Configure the instance.

Insert ChildPRC1 through the COPS both under the PRC itself and under the part.

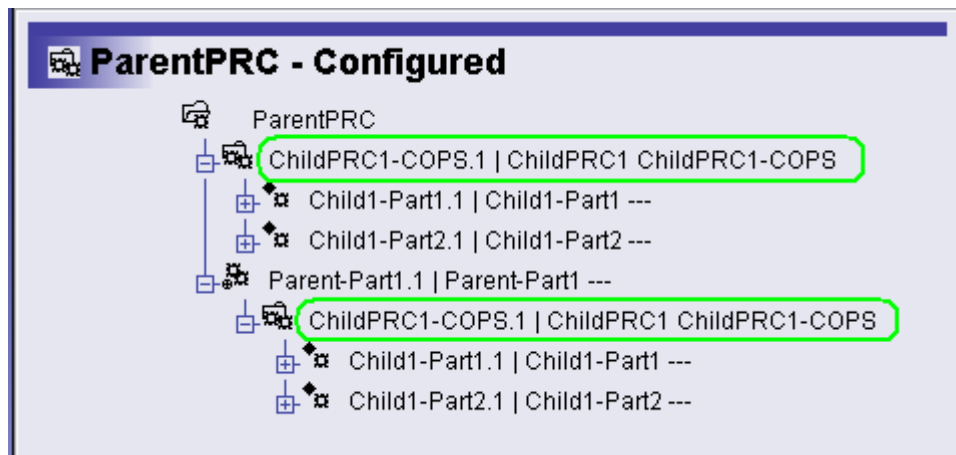


Figure 62: ParentPRC Configured

0: Add configuration data, e.g. a range interval to the top level COPS ChildPRC1-COPS.1 and a date interval to the instance Parent-Part1.1.

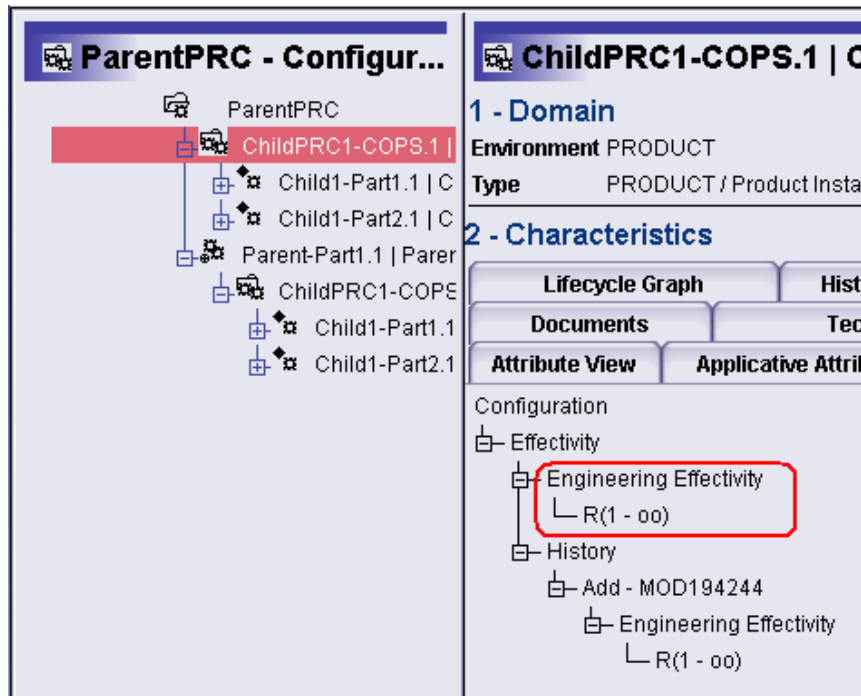


Figure 63: ChildPRC1-COPS

0: Export both ParentPRC and ChildPRC1 using the configuration option to stop at COPS.

- Store the sample configuration to /path/to/config.xml
- export ENOVIA_LCAIPD_CONFIGURATION_FILE=/path/to/config.xml (use "set" on Windows instead of "export")
- Open a Command Shell and perform the export:
ProductDataGen product ParentPRC
- ProductDataGen product ClientPRC1

0: In DPE create a new project R180GWA0100, import ParentPRC and import ChildPRC1 afterwards.

- After import of ParentPRC

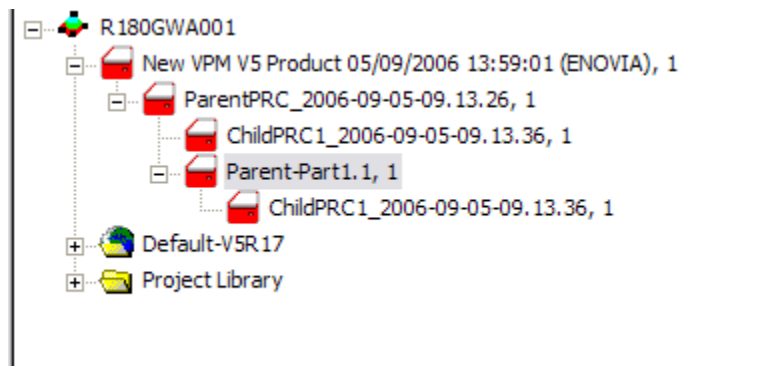
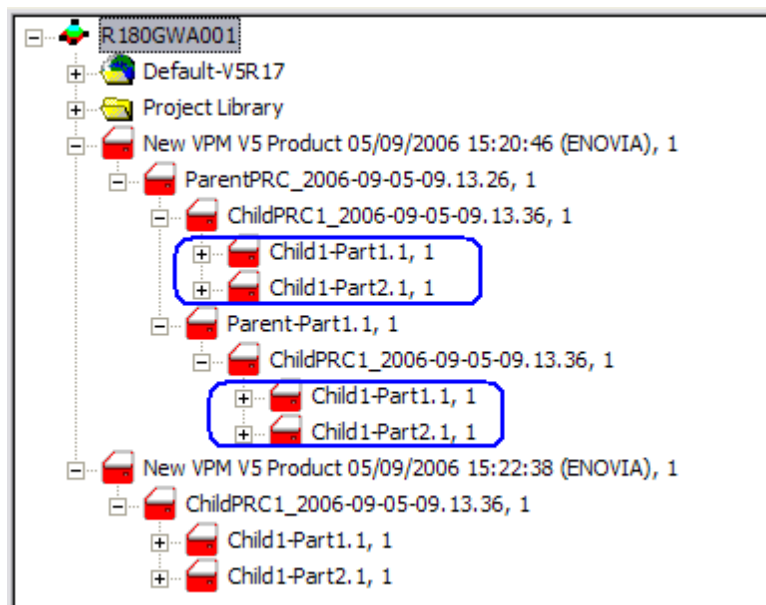


Figure 64: Import of ParentPRC

b) After import of ChildPRC1

**Figure 65: Import of ChildPRC1****Result:**

The structure including all configuration and geometric information exists in the Manufacturing Hub the same way it's given in the Engineering Hub.

6.18 Support Best-So-Far (BSF) Strategy in ENOVIA VPM V5

The way synchronization happens in the Engineering Hub may be customized. Up to and including R15 only the automatic synchronization of the product structure was supported by the DELMIA – ENOVIA VPM V5 Connection. Starting with R16, another synchronization methodology, i.e. BSF, is supported.

Using this methodology it's no longer true that the version of a part referenced by an item instance below a Product Root Class (PRC) is the latest one.

Thus, the part version information has to be gained in a different way, i.e. through the item instance.

R15

If we created a product structure in the EH and versioned one or more parts within it, the item instances referencing the part would be synchronized automatically. Thus, there's only one part version which is the correct one, namely the latest one.

R16

Exactly this last sentence may give a headache. Let's say, a part has been designed and the result has been approved by some qualified person. Now, the designer tries to improve the current design and takes several stabs in this direction. Clearly, all those might have their drawbacks and should be verified and approved by an authorized instance. As long as this has not happened, other people should not see this new version, but the version best so far. So, immediate synchronization has to be suppressed and additionally not the latest, but the BSF part version has to be transferred over the Connection.

Export

This change has consequences for the export part of the DELMIA – ENOVIA VPM V5 Connection. We are no longer able to traverse the reference tree and ask for the correct part version, since this would return us the latest one, but we have to extract the information from the item instance referencing the correct version.

There is a way to compute the correct item instance referencing an assembly relation. Therefore, the change is straight forward. Instead of extracting the part version via the part master, we have to ask the item instance.

Compatibility

The new methodology also works correct if ENOVIA VPM V5 is customized to be used in supersede mode with automatic synchronization. For further information please see the ENOVIA VPM V5 Documentation.



Note

*As a side effect of this change the **transfer of a part**, triggered out of the **part editor** or through the `part` command line option to `ProductDataGen`, now stops after the first level, i.e. only the part itself and all documents attached to it are transferred, but no sub structure which might be connected to it. This step is necessary, because we do not have the instance information which is needed to correctly traverse the structure.*

The partial transfer of a sub structure out of the product editor is not impacted by this change.

6.19 Support Export of Multiple PRCs in One Call

Until R15 the export part of the DELMIA – ENOVIA VPM V5 Connection only allows the export of a single Product Root Class (PRC) in full and incremental mode. This leads to a much too high time consumption especially of the incremental export due to the large overhead existing in this mode for a call to the export tool `ProductDataGen`. For R16 the `ProductDataGen` has been extended to perform an export of multiple PRCs in a single call.

In batch mode the export part of the DELMIA – ENOVIA VPM V5 Connection (`ProductDataGen`) now understands a new command switch “`prcfile`” as in

```
ProductDataGen prcfile </path/to/file_with_ids> [incremental  
<modification_date>]
```

This allows the user to export all product roots specified in a text file. The PRCs need to be specified one per line, see below.

In incremental mode the exporter queries the ENOVIA VPM V5 database for all changes which happened since the specified modification date. This is done for every type of change the tool understands, e.g. Part Version or Assembly Relation. After this the result sets are reduced by removing all objects which are not directly or indirectly referenced by the PRC given. If the export feature “Stop at COPS” is activated, the same steps have to be performed for every single PRC. Here, the new mechanism kicks in. All PRCs listed in the specified file are taken into account, thus reducing the amount of time spent for the queries to the database and the reduction.

After the mapping of changed objects to PRCs is finished, each PRC is exported. This is done one after the other, so this happens the same way a set of subsequent calls to `ProductDataGen` would work.

The export of multiple PRCs is also possible if the `prcfile` option is used without specifying an incremental date. In this case the given PRCs are exported one after the other in full export mode.

All PRCs listed in the specified file will be exported “in parallel”. Only one log file will be created for the complete export which is named after the first PRC in the file. Herein you will find log sections for each exported PRC. There will be no difference in contents of the generated XML/catprf files compared to a set of subsequent incremental/full exports of the specified PRCs.

The product structures which have one of the specified PRCs as parent are exported in parallel.

Performance

The function heavily improves the scalability for exporting multiple PRCs in incremental mode in the case the export feature “Stop at COPS” is activated.

This means that the smaller the PRCs are the greater is the benefit when using the new multi PRC export methodology.

If the export feature “Stop at COPS” (though `ENOVIA_LCAIPD_COPS`) is not activated and the specified PRCs are linked to each other (Product on Product), the sub structures in common will be exported more than once. This results in a time regression.

Limitations

This functionality is only available in batch export mode (via `ProductDataGen`) but can't be used within the ENOVIA VPM V5 client.

The generated files will be created in one directory which is customizable via `ENOVIA_LCAIPD_OUTPUT_DIR`.

User Interface

In the batch mode, run `ProductDataGen` with the command line switch “`prcfile`” and the path to the file containing the identifiers of the PRCs you want to export. It's always a safe bet to use an absolute path.

```
ProductDataGen prcfile </absolute/path/to/file_with_ids>
[incremental <modification_date>]
```

If the optional incremental part is not provided, the tool will work subsequently through the list of PRCs given in the file and performs a full export for each of them. In this case there's no internal parallelization, it is more like a convenient methodology to export multiple PRCs.

The format of the file containing the identifiers is as follows:

- Each identifier has to be given on one line
- The hash sign (#) starts a comment. A comment lasts till the end of the line
- Leading and trailing white space is ignored
- Empty lines are ignored

Example of a valid PRCFILE

----- cut here -----

```
# Root node
```

```
787
```

```
# List of first level COPS
```

```
Left Wing
```

```
Right Wing
```

```
Section 41          # causes ProductDataGen a headache
```

```
Section 40          # another big one
```

----- cut here -----

6.20 Improved Support of Product Classes Export

It is now possible to transfer an ENOVIA Product Class including its sub structure of product classes to the MH. All contained PRCs will be created in the MH as "place holders" (meaning that only the root PRC node will be created along with its parent) which can be populated with product data later on. For the PCs the known Fine Grained Mapping functionality is supported so that Product Class nodes in the MH can be created with arbitrary (but PTS compatible) Plan Types, dependent on a customized product class attribute. This way it is possible to put the top PC node into the resource view and populate its sub tree automatically.

The contained PRC nodes have to be transferred independently from the PC transfer.

The DELMIA – ENOVIA VPM V5 Connection will now transfer not only the PRCs data, but also the structure and meta information from the PC super-structure, containing those PRCs.

The order whether PRCs are transferred first or the PCs in which the PRCs are contained does not matter. The detection algorithm of the DELMIA – ENOVIA VPM V5 Connection ensures a correct linkage between the two object types.

The described export of ENOVIA product classes is supported for full update batch mode only via the command line tool ProductDataGen. Interactive transfer via the ENOVIA VPM V5 client is not supported. Partial and incr. update modes are not supported for the transfer of product classes. This should not be a big limitation as the size of product class structures is much smaller than the size of PRCs.

This means that PCs have to be exported via full export mode but PRCs may be exported in a different mode (e.g. incremental).

Export

With the support of PCs, the DELMIA – ENOVIA VPM V5 Connection is capable to transfer PCs and – independent from that – the contained PRCs. The PC structure is a super-structure of sub-PCs and contained PRCs, linked into the root PC hierarchy.

The schema of the Product Class XML export file, generated on the EH side is similar to the one for regular PRC transfers. One main difference is, that the PC export stops when visiting PRC node. Therefore the overlap of a PC export file and the export file of one of the contained PRCs is only the PRC node itself.

The export of a PC structure is also similar to the export of a PRC structure. Especially the Fine grained mapping is also possible for PC transfers. The DELMIA – ENOVIA VPM V5 Connection must know where to find the Customization file. Therefore the new environment variable ENOVIA_LCAIPD_CONFIGURATION_FILE must contain the path to the Customization file.

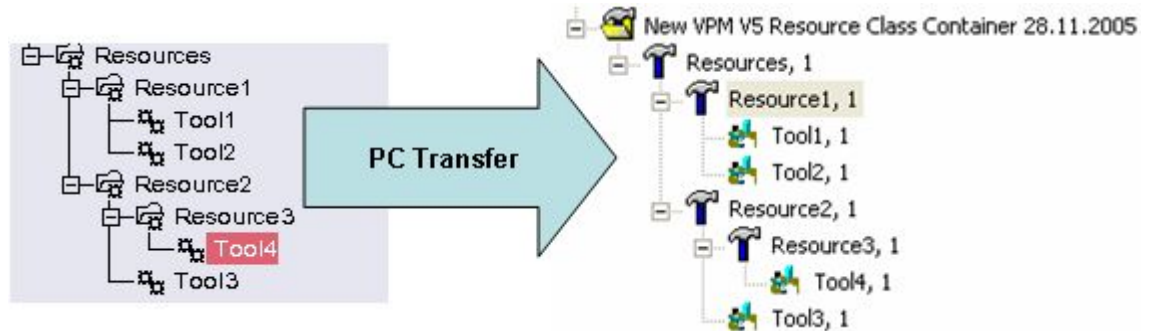


Figure 66: Export of a PC Structure

The Customization file contains a section of mapping information. For the customized attributes the values are used as type information.

In the picture above the complete tree is transferred. “Resources” is the top PC node and e.g. is mapped to an organizational Plantype OrgPC with the name “Resources” in DPE. This has to be linked to the Resource View manually after the first transfer. The rest of the tree is automatically populated according to the tree structure in ENOVIA VPM V5, provided that the mapping is correct.

For this example the mapping part of the Customization file looks as follows:

```
<mappings>
  <mapping context="ProductClass">
    <attributes>
      <token>name</token>
    </attributes>
  </mapping>
  <mapping context="ProductRootClass">
    <attributes>
      <token>name</token>
    </attributes>
  </mapping>
</mappings>
```

The export of a PC export file is possible in batch mode only

The batch export is available through the ProductDataGen tool and its new command line option “productclass”:

```
ProductDataGen productclass ...
```

Import

It is not required to specify an XML file import as a PC transfer. The importing PPRLoader application identifies a PC transfer file through the specific hard-coded identifier `PRODUCT_CLASS` in the environments name. Therefore, `PRODUCT_CLASS` is a reserved term and not allowed as an environments name (see Limitations). PC transfers do not come along with an instance XML file, so the regular warning on this situation is suppressed.

The import of the PC XML export file is to some extent similar with the import of a PRC XML export file. PC nodes are being created in MH according to their specific Fine Grained Mapping (if customized on export side) or the settings done for the (pseudo) environment `PRODUCT_CLASS` (if fine grained mapping for product classes is not customized on export side). The PT definition used for PC object creation is taken from the setting `[...\PART_LIST]Plantype`, analogue to the import of regular nodes in case of a PRC transfer. If no "`PRODUCT_CLASS`" mapping is available the "Product default" mapping will be used instead.

Also, a root node named "New ENOVIA VPM V5 Product/Resource Class Container..." is created above the top level PC. Its PT definition is taken from the `[...\Root]Plantype` setting of the corresponding Fine Grained Mapping of the root PC or, again, from the (pseudo) environment `PRODUCT_CLASS`.

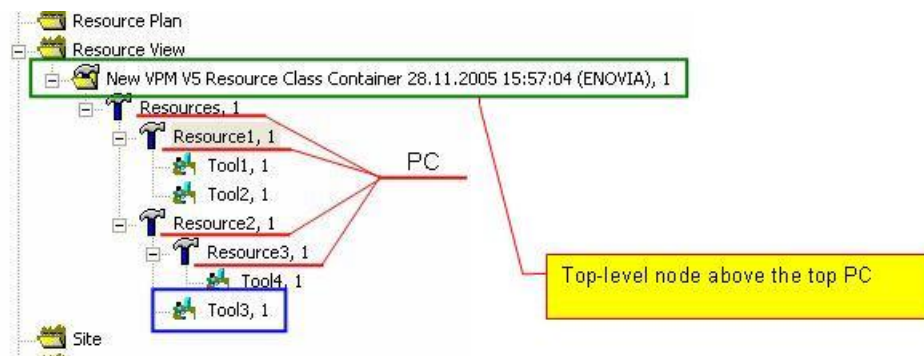


Figure 67: New ENOVIA VPM V5 Root Node

In difference to a regular PRC import, the PRC nodes within the PC structure (named Tools in image above) will have all the very same instance ids ("`vpm_instanceid1`" and "`vpm_instanceid2`"), set to "`VPM.PRODUCT.00000000000000000000000000000000`". This allows the later on re-identification of those PRCs.

| General Investment Simulation Notes Effectivity Attachment | |
|--|---|
| Cycle Time | 0,00 sec |
| External ID | PRODUCT.PART_LIST.83A4637B0000439843269467000C729A_80A2B3BD0000520C383BE498000986CE |
| VPM Instance Id 1 | VPM.PRODUCT.00000000000000000000000000000000 |
| VPM Instance Id 2 | VPM.PRODUCT.00000000000000000000000000000000 |
| Resource Name | Tool1 |
| Resource Number | Tool1 |
| Length | 0,00 m |

Figure 68: PRC Nodes within the PC Structure

All sub-PC nodes within the top-level PC are marked with the regular ids. Please note, that – since there is no instance data – the attribute "`vpm_instanceid2`" con-

tains the concatenation of the links External Ids in long format (including environment and table information) and not an ENOVIA instance UUID.

| | |
|-------------------|--|
| Cycle Time | 0,00 sec |
| External ID | PRODUCT_CLASS.PART_LIST.83A4637B00004398432696F2000310BD_80A2B3BD0000520C38 |
| VPM Instance Id 1 | 83a4637b000043984326971300079038 |
| VPM Instance Id 2 | PRODUCT_CLASS.CLink(PART_LIST).83A4637B000043984326971300079038_80A2B3BD0000520C38 |
| Resource Name | Resource1 |
| Resource Number | Resource1 |
| Length | 0,00 m |

Figure 69: Sub-PC Nodes within the Top-Level PC

If one of the PRCs contained within the PC structure is transferred later on, the PRC node is automatically identified inside the already transferred PC structure (by its External Id which is of course the same for PC and PRC transfer, since it is the very same PRC object!). Through this, it is not required to manually link PRC nodes to its PC super structure (which needs to be maintained manually as of R16, by the way).

Even though the PRC node already exists, the import is treated as an initial import in case the PRC node does not have any children (at least no children for the re-

levant relationships “nodes” and the product-resource one, which is set to “plant_provides_prod” by default). This will create the PRCs sub-structure completely new and also create the top-level object for the PRC node, with a PT definition taken from the [...\Root]”Plantype” setting of the corresponding Fine Grained Mapping of the transferred PRC or from its corresponding environment. In this case the node representing the PRC will be a child of two parent objects (i.e. it is linked two times): first from the PC structure and its parent PC, second from the top-level of the PRCs own (initial) import. The intention of this is to generate the very same results in Manufacturing Hub; independently from the order the actual imports are preformed.

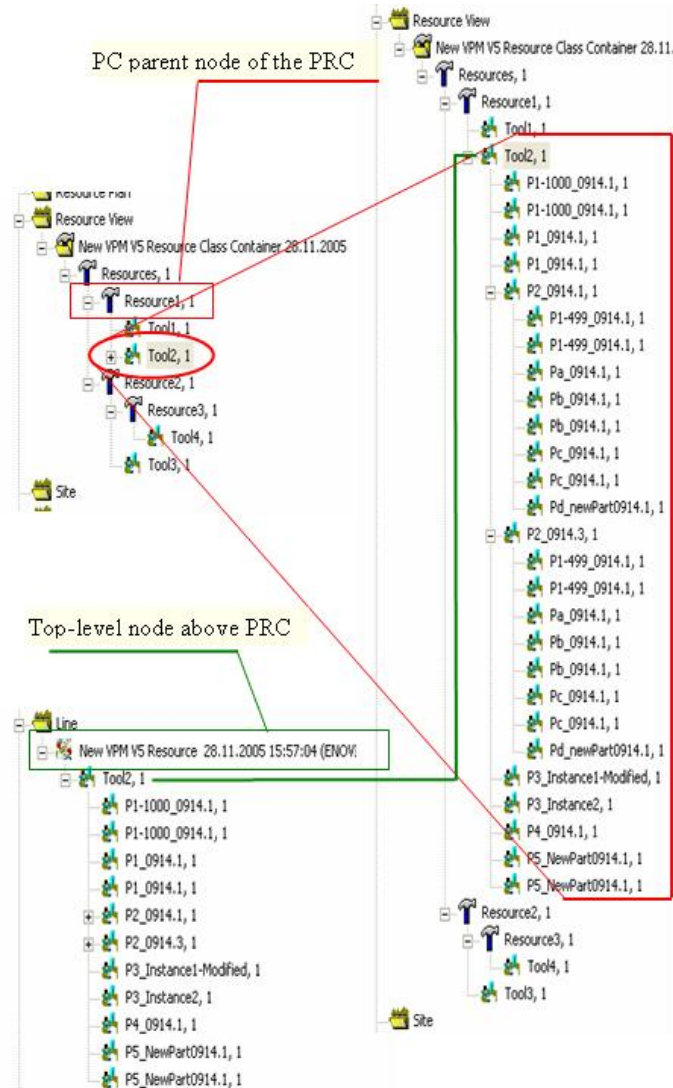


Figure 70: PRC Import

This mechanism is independent from the order in which the imports are preformed. If the PRC structure is transferred first, the PC import will also identify the

contained PRC nodes and bind them into the overall PC structure. In this case, the deletion of the PRCs sub-structure (or the marking of it as “delete”d) is of course suppressed, since the sub-structure is not transferred within the PC XML export file.

As already mentioned, there are several differences between the import of a PC XML export file and a regular PRC export file from EH:

- The recalculation of absolute instance effectivities and transformations matrices is disabled
(in the EH, PRCs do not have configuration and position data within the context of a PC).
- If a PC export XML file contains PRC nodes, which are not yet in the MH's PC structure, those PRC nodes will be queried from the database, based upon their “externalid” attribute, before creating them new. This ensures the same end result for PC and PRC transfers, independent from the order of imports.
- The instance ids of transferred PRC, contained in the PC are all set to “VPM.PRODUCT.0000000000000000”. This will allow an automatic identification and linkage of the PRC, when its sub-structure is transferred later on.
- PRC nodes are not instantiated, since they might be used from other PCs as well.
- Since there are no PRC instances, there is also no instance XML file for PC transfers. Functions and messages/warnings in the area of instance data are therefore disabled (e.g. the warnings as “Instance file missing” or “No instance data for component ...” are suppressed).
- The root object “New ENOVIA VPM V5 Product/Resource...” is created, even when the PRC node is already existing in MH, if the PRC node has no children (the relevant child lists for this check are “nodes” and the product-resource relationship, which is by default “plant_provides_prod”). There is no check on any id or attribute, whether the children represent ENOVIA VPM V5 parts or manually created nodes.
- If a PC structure is imported into MH, containing already transferred PRC nodes, the PRCs sub-structures are not deleted resp. marked “delete”d.

Customization

Export

On the EH-Server the user has to define an environment variable ENOVIA_LCAIPD_CONFIGURATION_FILE pointing to the Customization file. The Customization file contains a list of attributes used for fine grained mapping of PRCs and PCs.

The mapping for PCs behaves exactly as the fine grained mapping for PRCs.

For example, the list may consist of mapping mapping_attr_1 and mapping_attr_2, e.g.

```
<mappings>
  <mapping context="ProductClass">
    <attributes>
      <token>mapping_attr_1</token>
      <token>mapping_attr_2</token>
    </attributes>
  </mapping>
</mappings>
```

Now, if the exporter comes across a PC which contains an attribute mapping_attr_1 with a non-empty value the following happens:

The PC will be sent over the DELMIA – ENOVIA VPM V5 Connection and the value of mapping_attr_1 is used as mapping information, no matter whether mapping_attr_2 exists and has a value assigned to it.

If mapping_attr_2 shall be used as mapping information, the attribute must contain a value. Additionally, mapping_attr_1 may exist but must not have a value assigned to it or the order of the tokens in the attribute list has to be reversed.

The mapping attribute(s) customized in the Customization file have to be defined on the PC.

The PCs are transferred by the DELMIA – ENOVIA VPM V5 Connection together with the tree structure.

Import

On the import the mapping between PRC classes and plan types is defined. The mapping between PC classes and plan types is defined the same way.

The creation of a top-level root node (e.g. "New VPM product ...") can be suppressed by using the -noroot command line option.

```
pprloader -vpm ... -project ... -noroot
```

This option is valid for both, PC and PRC data transfers. The root nodes may not be needed any longer due to the availability of fine grained mapping of PCs and PRCs and therefore can be suppressed.

Limitations

- The mapping of a PC has to use an existing PT. It is possible to leave the customized attribute value empty, so a default PT is taken. If there is a value, this value has to have a corresponding entry in the registry, which has to lead to an existing PT. If this chain breaks, the import stops and creates an error log file entry.

The customization of PTs in the PTS must be consistent with the mapping of the PC together with the PRCs and their parts. E.g. if PT A corresponds to a PC and PTs B and C to PRCs, which are children of that PC, the PTs B and C must be customized in the PTS to be allowed children of A.

This applies also for the Root PT: if PT A corresponds to the top node PC then PT A must be an allowed child of the Root PT.

There is no consistency check on the PTS during the import. It's up to the customer/administrator to ensure a working mapping between the Fine Grained mapping types/environments and the projects PTs in MH.

Most probable errors in this area are unknown/inexistent plantypes, which would result in failures while creating the corresponding objects, or insufficient PT properties (flag "recursive" on PT, invalid parent-child-relationships between PTs), which would block the creation of the corresponding links, this leading to multiple creation of the related child objects, since they're not found during the traversal of the structure and therefore identified as "new".

- The Product Structure Mapping has to be completely defined in the PPRLoader's registry even though the keys "CATIA_MODEL" and "DOCUMENT" are disregarded during the import.
 - The transfer of PCs is only available in batch mode using the Product-DataGen export tool. Interactive transfer mode is not supported.
 - The transfer of PCs is only available in Full Update mode. Neither Partial nor Incremental modes are currently supported.
- If a child PC of an already transferred root PC is imported, the corresponding PC sub-structure of the child PC will not be identified and therefore, it will be created new in parallel to the existing structure of the root PC. Even though, since PRC nodes are not instantiated, the PRC nodes are correctly identified and still exist only once in the MH database.

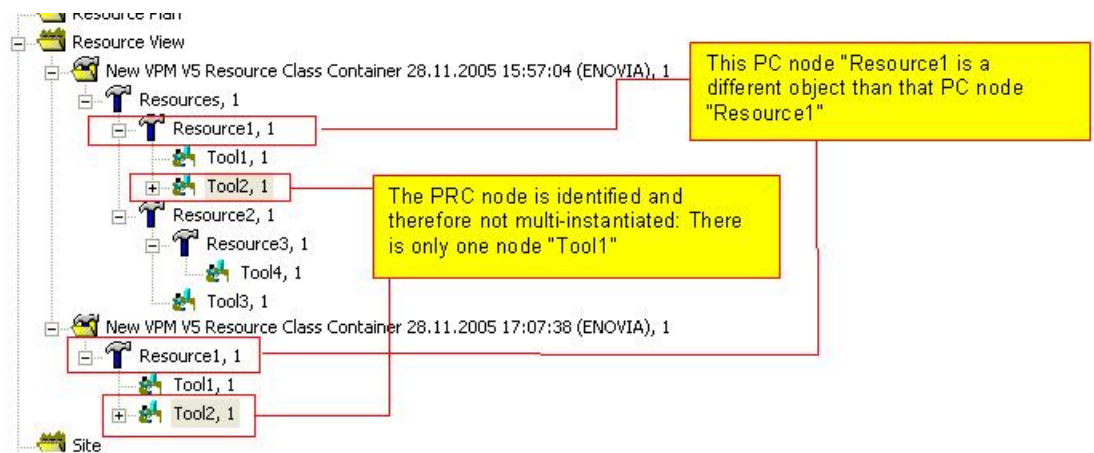


Figure 71: PCs Transfer

- The term PRODUCT_CLASS in the environment name identifies a PC transfer file. Thus, it is not allowed to have a real environment named PRODUCT_CLASS, since this would make the importing PPRLoader application recognize all regular XML files containing such parts as PC transfer files.

Example 1: Transfer of a PC Structure

- 1) Create a new project in DPE using the Default V5 plan type set.
- 2) In ENOVIA V5 create a new PC structure, containing a PC "Resources" and, below that, two PCs "Resource1" and "Resource2". Inside "Resource1" insert two PRCs "Tool1" and "Tool2". Inside "Resource2" insert another PC "Resource3" and, parallel to that, insert another PRC "Tool3". Beneath "Resource3" insert a second level PRC "Tool4".
- 3) Put "Resource" in the description of the PCs. Put "Tool" in the description of the PRCs.

The structure should look similar to the one in the following image:

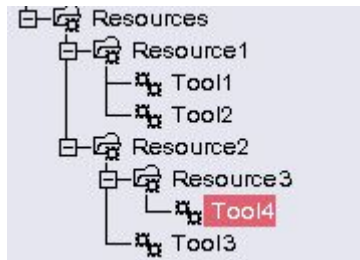


Figure 72: Transfer of a PC Structure

- 4) Define the environment for the exporter respectively on the server through the customization file, specified by the environment setting ENOVIA_LCAIPD_CONFIGURATION_FILE. In the customization file define in the section mappings as follows:

```

<mappings>
  <mapping context="ProductClass">
    <attributes>
      <token>V_description</token>
    </attributes>
  </mapping>
  <mapping context="ProductRootClass">
    <attributes>
      <token>V_description</token>
    </attributes>
  </mapping>
  <mapping context="PartVersion">
    <attributes>
      <token>V_description</token>
    </attributes>
  </mapping>
</mappings>

```

- 5) On the MH importing side, adapt the registry settings in [HKEY_CURRENT_USER\Software\DELMIA\ergoplan\PPRLoader\Product

Structure Mapping] as follows:

Insert the environment “Resource” and set the key “Plantype” in “PART_LIST” to “Resource”. In the type “Root”, set the key “Plantype” to “Resource View”. Create a second environment “Tool”. Set the key “Plantype” for the type “PART_LIST” to “Station” and for “Root” to “Resource”.

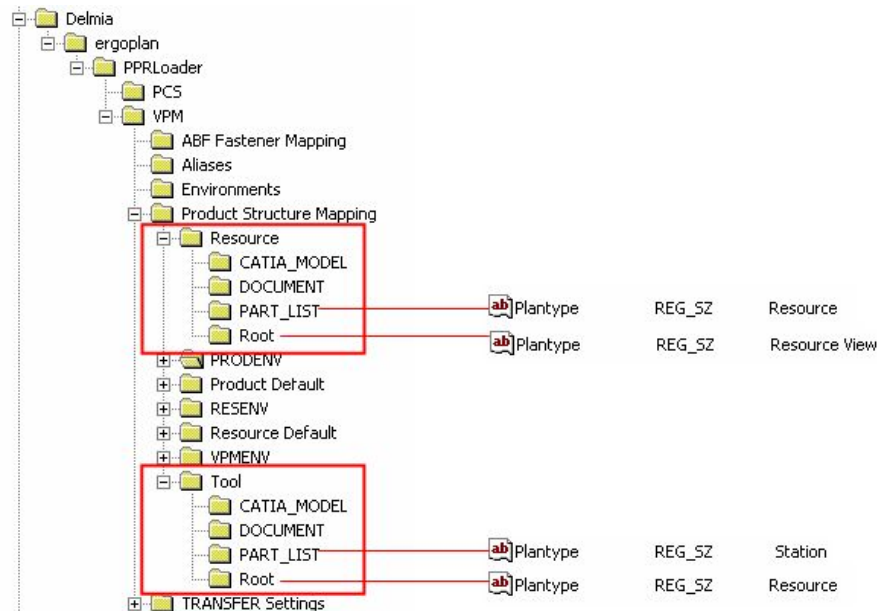


Figure 73: MH Importing Side

Transfer the PC structure from the EH to the MH using the batch mode: On the ENOVIA V5 side, enter the following command line:

```
ProductDataGen productclass ...
```

On the MH import side, the PPRLoader application is used, along with the EXAMPLE1_TRANSFER_OF_A_PC_STRUCTURE project as destination:

```
pprloader -vpm ... -project  
EXAMPLE1_TRANSFER_OF_A_PC_STRUCTURE
```

Check in the PPRLoader log file the correct identification of a PC transfer. The file must contain the following lines

```
VPM V5 XML export file contains Product Class structure.
```

resp.

```
components total: ... (... product class objects, ... PRCs)
```

If these lines don't exist, the test has failed.

In DPE, in project EXAMPLE1_TRANSFER_OF_A_PC_STRUCTURE, check the creation of the PRC structure.

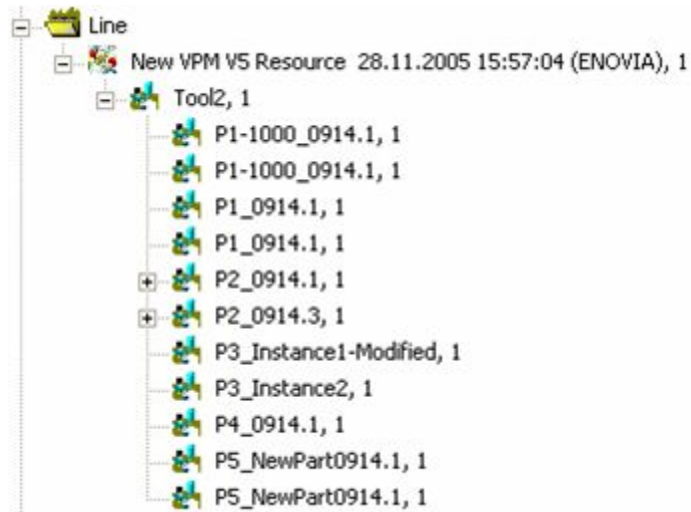


Figure 76: Project EXAMPLE1_TRANSFER_OF_A_PC_STRUCTURE

Check the creation of a top-level root node “New ENOVIA VPM V5 Resource” with plantype “Line” above the top PRC node.

Check also, that the PRC “Tool2” has correctly been linked in the PC structure, transferred example 2.

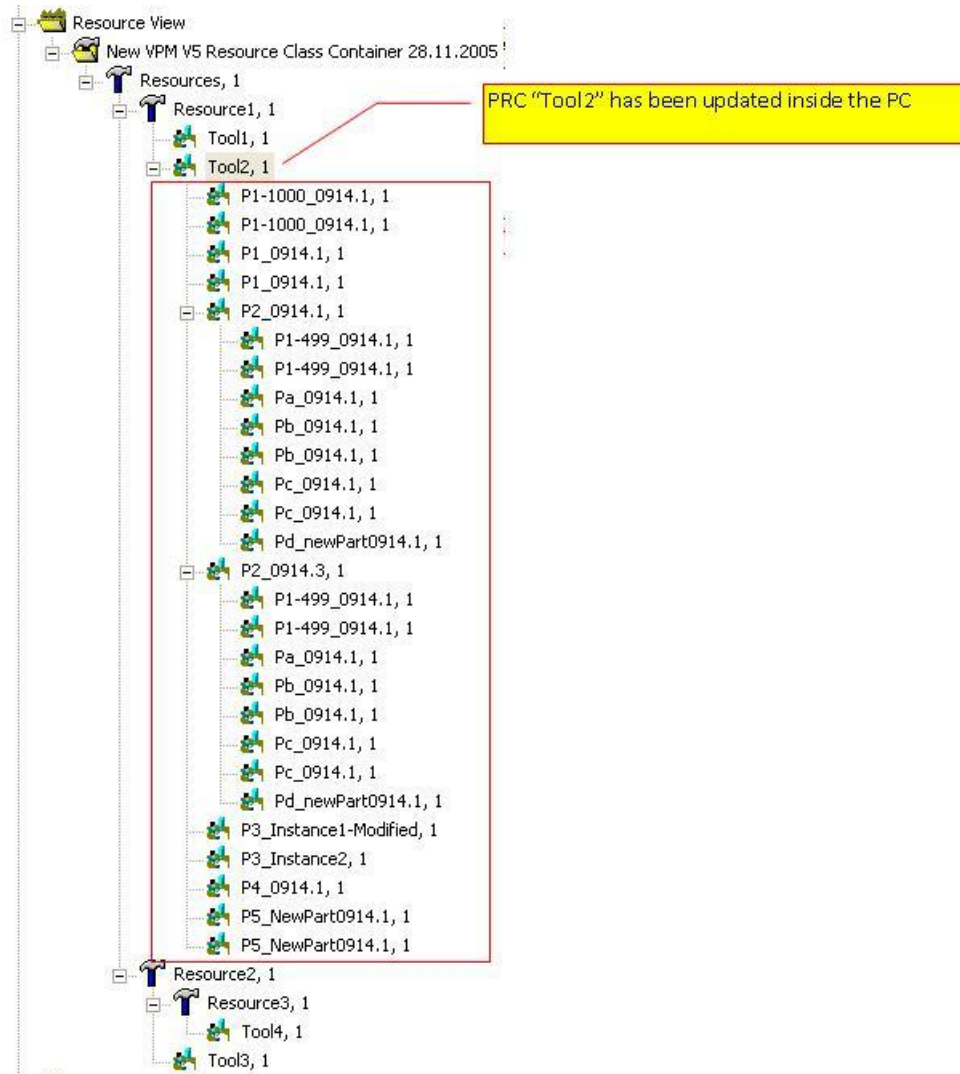


Figure 77: Check the Creation of a top-Level Root Node

Example 3: Correct creation of the PC structure and its PC sub-structures, if the order of imports is reversed (import PRC first, and then import PC super-structure).

- 1) Create a new project in DPE: EXAMPLE3_PC_STRUCTURE using the Standard V5 plan type set.
- 2) In ENOVIA V5 use the data created for Example 1.
- 3) Use the very same customization and registry settings as for Example 1.

Transfer the PRC structure of PRC “Tool2” from the EH to the MH using the batch mode:

```
ProductDataGen product ...
```

Use the EXAMPLE3_PC_STRUCTURE project as destination.

In a second step, transfer the PC structure from the EH to the MH using the batch mode:

```
ProductDataGen productclass ...
```

Again, on the MH import side, the PPRLoader application is used, along with the EXAMPLE3_PC_STRUCTURE project as destination:

```
pprloader -vpm ... -project EXAMPLE3_PC_STRUCTURE
```

In DPE, in project EXAMPLE3_PC_STRUCTURE, check the correct creation of the PC structure and the correct linkage to the PRC structure (imported as a first step).

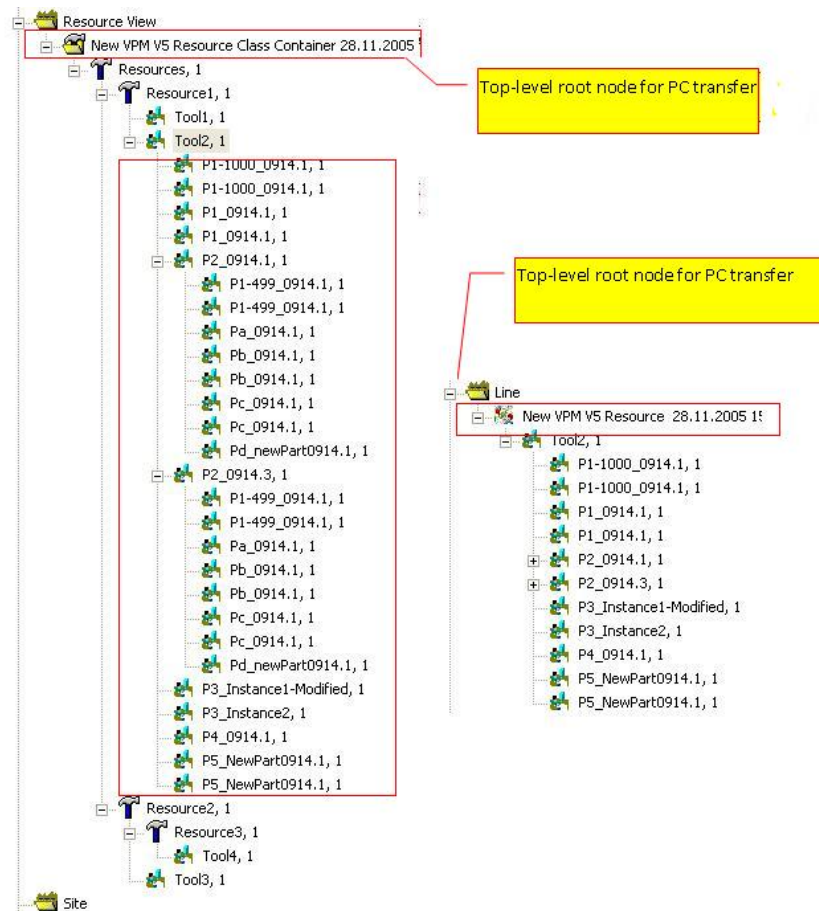


Figure 78: Project EXAMPLE3_PC_STRUCTURE

Check in the PPRLoader log file the correct identification of the existing PRC through the following line

```
VPM V5 XML export file contains Product Class structure.
```

Check, that there is no different, neither in structure not in attributes between the PC and PRC structures in the projects GPL380A and GPL380B. Check e.g. that both top-level nodes, one for the top PC and one for the transferred PRC exist.

Example 4: Correct update of a transferred PC structure

- In DPE use the project created for Example 1.
- Use the ENOVIA V5 data created for Example 1
- Use the very same customization and registry settings as for Example 1.
- In ENOVIA V5 add another PRC "Tool5" under PC "Resource1" and delete PRC "Tool1". In addition, add another PC "Resource4" with a PRC "Tool6" as a child.
- In DPE rename PRC "Tool2" to "Tool2_modified"

Transfer the PC structure from the EH to the MH using the batch mode:

```
ProductDataGen productclass ...
```

On the MH import side, the PPRLoader application is used, along with the EXAMPLE1_TRANSFE_OF_A_PC_STRUCTURE project as destination:

```
pprloader -vpm ... -project EXAMPLE1_TRANSFE_OF_A_PC_STRUCTURE
```

Check in the PPRLoader log file the correct identification of a PC transfer. The file must contain the following lines

```
VPM V5 XML export file contains Product Class structure.
```

```
resp.
```

```
components total: ... (... PRCs, ... parts, ... models, ... documents )
```

In DPE, in project EXAMPLE1_TRANSFE_OF_ A_PC_STRUCTURE, check the correct update of the PC structure.

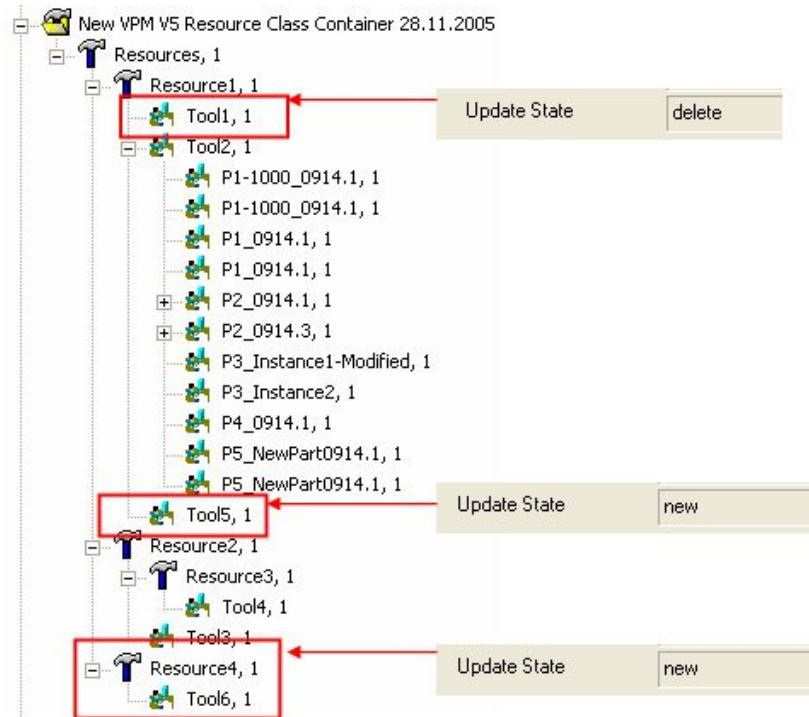


Figure 79: Project EXAMPLE1_TRANSFE_OF_ A_PC_STRUCTURE

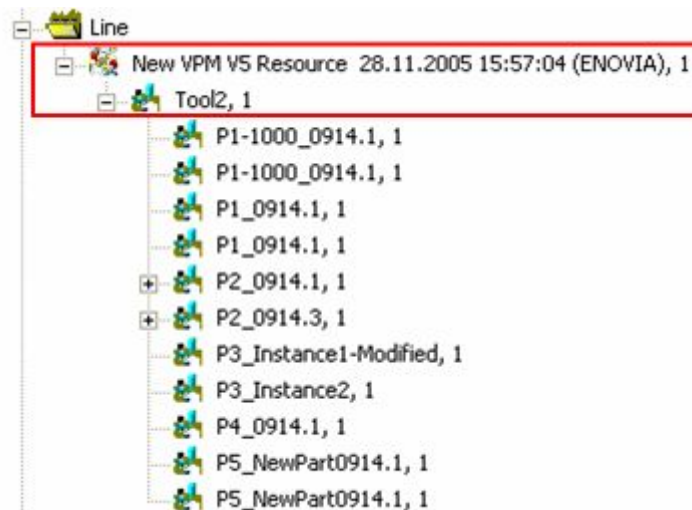


Figure 80: New Line VPM Resource

Check the creation of the new PC and PRC objects (PC “Resource4” and PRCs “Tool5” and “Tool6”, update state is set to “new”).

Check the deletion of the PRC “Tool1”, resp. check that it is marked as “deleted” (see attribute “updatestate”).

Check also, that the PRC “Tool2_modified” has been correctly changed back to “Tool2”.

Check further, that no object beneath the PRC “Tool2” has been “delete”d or marked that way.

Example 5: Batch transfer of a PC structure without the Fine Grained Mapping.

- 1) Create a new project in DPE: EXAMPLE5_BATCH_TRANSFER using the Standard V5 plan type set.
- 2) In ENOVIA V5 create a new PC structure, containing a PC “Resources” and, below that, two PCs “Resource1” and “Resource2”. Inside “Resource1” insert two PRCs “Tool1” and “Tool2”. Inside “Resource2” insert another PC “Resource3” and, parallel to that, insert another PRC “Tool3”. Beneath “Resource3” insert a second level PRC “Tool4”.
- 3) Put “Resource” in the description of the PCs. Put “Tool” in the description of the PRCs.

The structure should look similar to the one in the following image:

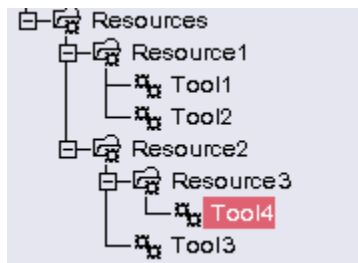


Figure 81: Batch Transfer of a PC Structure

Unset the environment setting ENOVIA_LCAIPD_CONFIGURATION_FILE.

- 4) On the MH importing side, adapt the registry settings in [HKEY_CURRENT_USER\Software\DELMIA\ergoplan\PPRLoader\Product Structure Mapping] as follows:

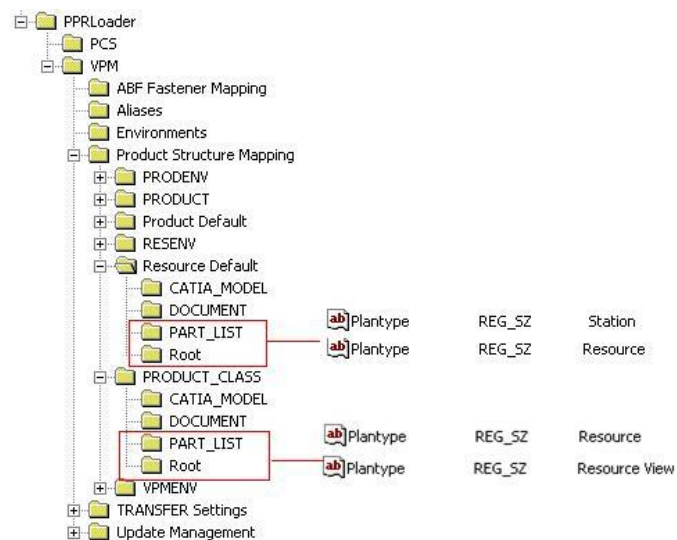


Figure 82: MH Importing Side

Insert the environment “PRODUCT_CLASS” and set the key “Plantype” in “PART_LIST” to “Resource”.

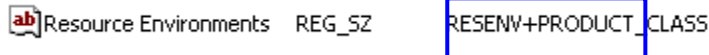
In the type “Root”, set the key “Plantype” to “Resource View”.

In the environment “Resource Default” set the key “Plantype” for the type “PART_LIST” to “Station” and for “Root” to “Resource”.

Register the “Environment “PRODUCT_CLASS” as resource

(enter “PRODUCT_CLASS” into the “Resource Environments” key:

[HKEY_CURRENT_USER\Software\DELMIA\ergopl an\PPRLoader\Product Structure Mapping]
 “Resource Environments”=“VPMENV+PRODUCT_CLASS”



Transfer the PC structure from the EH to the MH using the batch mode using the very same command as in Example #4:

```
ProductDataGen productclass ...
```

On the MH import side, the PPRLoader application is used, along with the EXAMPLE3_PC_STRUCTURE project as destination:

```
pprloader -vpm ... -project EXAMPLE5_BATCH_TRANSFER
```

Check in the PPRLoader log file the correct identification of a PC transfer. The file must contain the following lines

```
VPM V5 XML export file contains Product Class structure.
```

```
resp.
```

```
components total: ... (... product class objects, ... PRCs)
```

If this line does not exist, the test has failed.

In DPE, in project EXAMPLE5_BATCH_TRANSFER, check the creation of the structure equal to Example #5 (“Resource View” object and a sub-structure, made of “Resource”s and “Station” s).

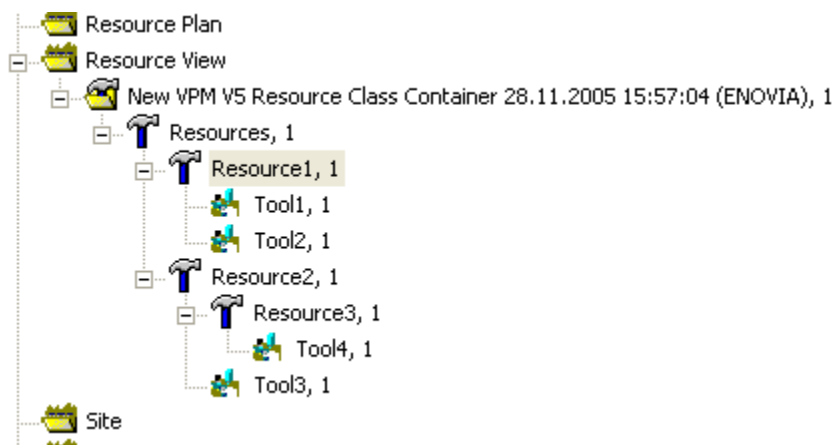
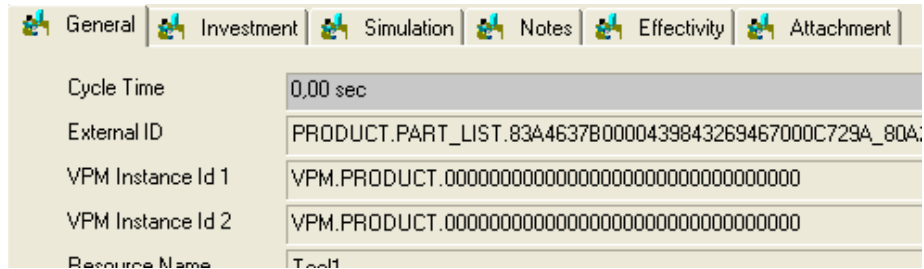


Figure 83: Project EXAMPLE5_BATCH_TRANSFER

Check the creation of a top-level root node named “New ENOVIA VPM V5 Resource Class Container...” above the top PC.

Check the attributes “vpm_instanceid1” and “vpm_instanceid2” for the PRC nodes. They must all contain the value “VPM.PRODUCT.0000000000000000”



| | |
|-------------------|--|
| Cycle Time | 0.00 sec |
| External ID | PRODUCT.PART_LIST.8344637B0000439843269467000C729A_80A |
| VPM Instance Id 1 | VPM.PRODUCT.00000000000000000000000000000000 |
| VPM Instance Id 2 | VPM.PRODUCT.00000000000000000000000000000000 |
| Resource Name | Tadit |

Figure 84: vpm_instanceid1

All PRCs referenced by the PC, directly or indirectly, are collected. The update then is done for each of the affected PRCs. This implies that we might get (nearly) empty XML files in incremental mode, if no changes within a PRC have been made since the last export.

In batch mode the export part of the Connection (ProductDataGen) now understands a new command switch “productclass” as in ProductDataGen productclass <product_class_id> which allows the user to export all Product Root Classes referenced by a Product Class. Also, in the incremental mode a Product Class ID may be given as argument.

All Product Root Classes referenced by a Product Class, i.e. that have this PC as one of their parents, direct or indirect, are transferred from the EH to the MH.

User Interface

In the batch mode, run ProductDataGen with the command line switch “productclass” and the id of the product class.

```
ProductDataGen productclass <product_class_id>
```

In order to support incremental exports of both Product Classes and Products command line arguments have been changed.

The two calls

```
ProductDataGen productclass <product_class_id> incremental  
<timestamp>
```

```
ProductDataGen product <product_id> incremental <timestamp>
```

are supported.

The known command line option, now deprecated and to be removed in coming releases,

```
ProductDataGen incremental <product_id> <timestamp>
```

can only be used to export Products. It has no effect on Product Classes, even if they have the same name as a Product.

Limitations

- This functionality is only available in batch mode (via ProductDataGen) but cannot be used within the ENOVIA VPM V5 client.
- Every PRC is transferred into the same project since the target is selected only once for the product class itself. As a workaround, in interactive mode, create substructures and export them separately, or use the batch mode to create multiple XML files which can be imported into different projects in the MH.
- The products are not structured in the MH, i.e. all PRCs are to be found in parallel. The Product Class structure is lost.
- Partial export is not supported in conjunction with the export of a Product Class. But this is not a real limitation as the partial update is only useful to export a sub portion of a PRC. In case of a Product Class export only complete PRCs are being exported.
- When performing an incremental update, there's no way to prevent the occurrence of empty XML files, i.e. files only containing the header information, but no actual changes, from being transferred.
- Due to the fact that the known command line option to ProductDataGen, incremental <id> <timestamp>, only supports products but not product classes, it's been deprecated in this release. It will be removed in coming releases.

Example

Open a shell window and set the appropriate ENOVIA VPM V5 environment. Then call

```
ProductDataGen productclass GPL130
```

All seven product root classes, GPL130-prc00 - GPL130-prc20, have to be transferred over the DELMIA – ENOVIA VPM V5 Connection into the selected project. They have to be found as “siblings” in the project's library.

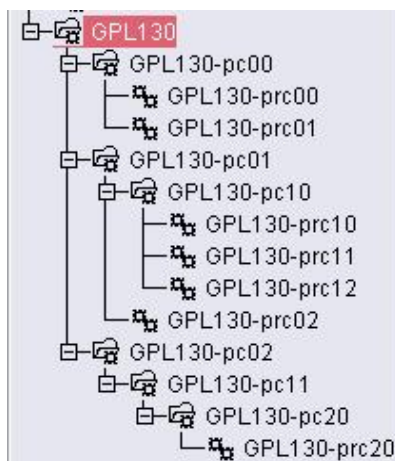


Figure 85: Siblings” in the Project’s Library

6.21 Support Consumption of ENOVIA Work Packages

When storing a product structure from CATIA to ENOVIA, it is possible to group substructures into so-called “Work Packages” or “black boxes”. These closed objects are represented as single objects in ENOVIA without any structural information. Nevertheless the content of a Work Package is stored inside ENOVIA, even if it is not displayed.

Until DPE R15 Work Packages (WP) were treated as single objects, i.e. Documents. After the import to DPE, the Work Package was represented only by its name. As a result in the Manufacturing Hub the structural information of a Work Package was not existent.

From DPE R16 on Work Package structures are exported as well. The Work Package content is transferred to DPE first and then attached to the proper Product component.

The Work Package content will be exported in a separate XML file and attached to its parent object in Manufacturing Hub. If a special Work Package content of a product already exists in Manufacturing Hub, it will be updated.

A partial transfer of a product referencing a Work Package is possible. Since the Work Package itself is handled as a single document (an attachment in the MH), a partial transfer of the package only is not possible.

An incremental transfer of a product containing references to a Work Package is also possible. As stated above, if the product described in the Work Package has been changed after the last update, a new iteration of the Work Package will be transferred to the MH.

We understand two types of work package documents. Both have a MIME type of either xml or text/xml and are related to a V5 document.

- Secondary formats of a primary V5 document.

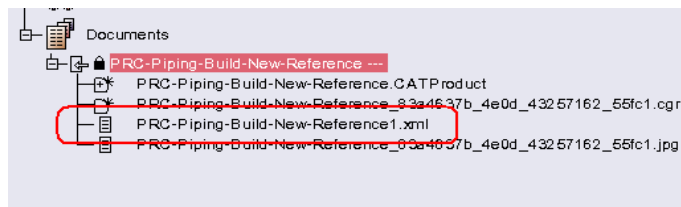


Figure 86: Secondary Formats

- CATIA reports related to this document.

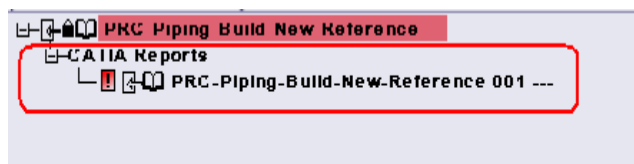


Figure 87: CATIA Reports

Which type of Work Package (or both) should be exported can be specified by the user through the environment variable ENOVIA_LCAIPD_WPTYPE, cf. Customization below. By default CATIA reports are taken into account.

The contents of the Work Packages will be stored as an attachment on the product component in Manufacturing Hub. A data model change in ENOVIA and DELMIA is not necessary for that.

The XML file which represents a report of the work package will be attached directly to the product component that represents the product node in ENOVIA referring the work package. The name is built in the following scheme:

ENOVIA5_<V_ID of object master>_<OID of object revision>_<OID of iteration>_<C_modified date of revision>.xml

The parts <...> will be replaced by the described values.

For **example**:

ENOVIA5_PRC-Demo-workpackage-product-
Refer-
ence_601BCA4E00000DDC423073F7000007C7_601BCA4E00000DDC423074
0900000D87_2005-3-10-17.21.30.xml

If the environment variable ENOVIA_LCAIPD_INSERT_ROOT_ID is set to 1 the identifier of the top entity, e.g. the PRC, is inserted into the generated name right after "ENOVIA5_". The name then is "ENOVIA5_<ROOT_ID>_....". By default, this setting is off.

Customization

On Manufacturing Hub:

For the Transfer Settings on DPE side, *Please refer to the [Configure WPK Settings](#).*

On Engineering Hub:

The export of work package XML files is activated by default. It can be activated on the export side, either in the ENOVIA VPM V5 Server environment for interactive transfers or in the environment used for running the ProductDataGen batch tool or both. The deactivation is achieved by setting the environment variable

ENOVIA_LCAIPD_WP_TRANSFER=0

If activated the Work Package XML file(s) may be generated in several ways:

- a) Manually attached as a secondary format to the primary V5 Work Package document in the ENOVIA VPM V5 client.

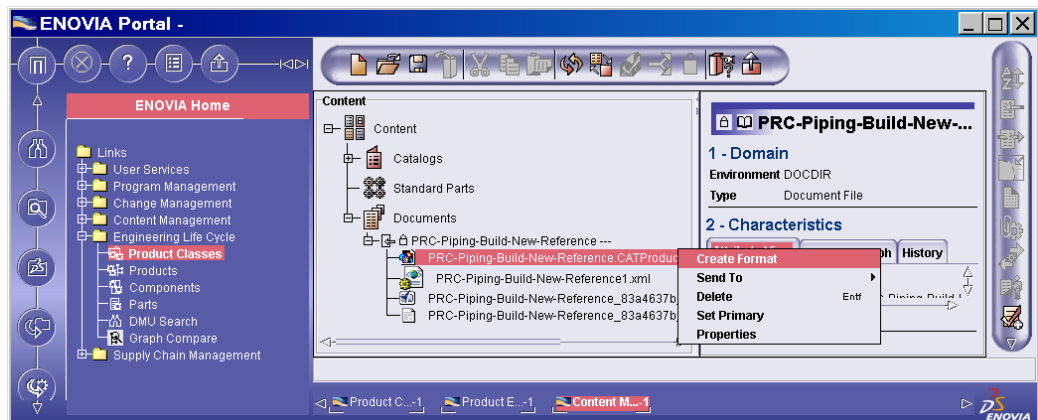


Figure 88: V5 Work Package document in the ENOVIA VPM V5 Client

- b) Manually created via "Send To Work Package report generator" in the ENOVIA VPM V5 client.

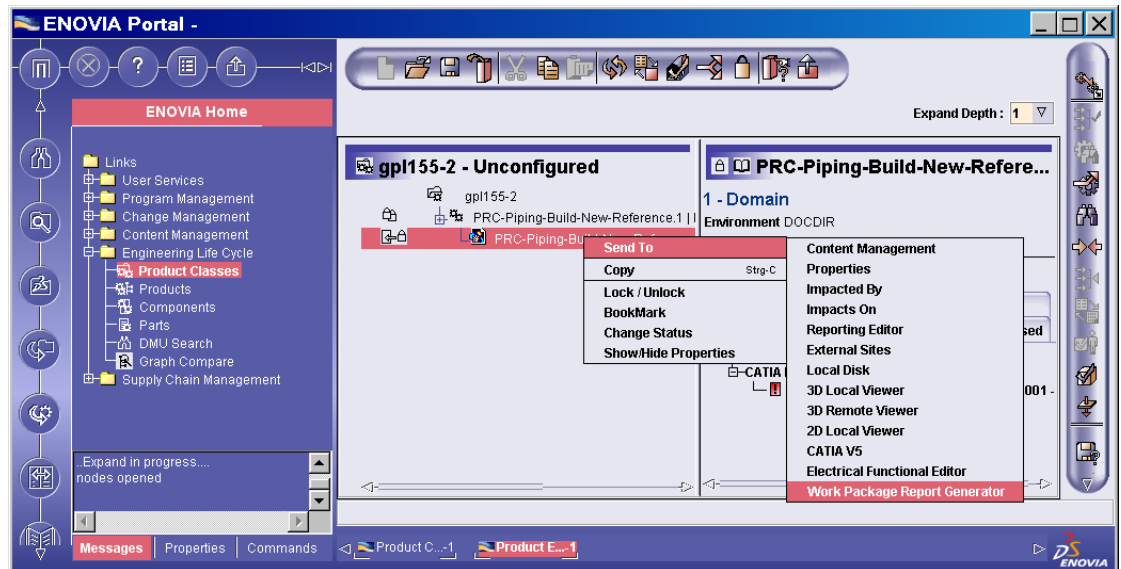


Figure 89: Send To Work Package Report Generator

- c) Automatically generated via a file introspection tool when the Work Package is saved from CATIA to ENOVIA. Such a tool can be activated on the ENOVIA server to create the XML files on the fly. It is important that the file introspection tool stores the created XML reports in the same location as one of the manual methods described above. The MIME type of the generated documents has to be either "xml" or "text/xml".

For more information on file introspection see the corresponding ENOVIA VPM V5 documentation.

- d) The introspection tool can also be specified to be used by the ENOVIA export module directly. Therefore the following environment values have to be defined on the ENOVIA server:

- ENOVIA_LCAIPD_WPTOOL_NAME=<The name of the introspection tool without extension>
- ENOVIA_LCAIPD_WPTOOL_IN_PARAM=<The name of the parameter which defines the input file path>
- ENOVIA_LCAIPD_WPTOOL_OUT_PARAM=<The name of the parameter which defines the output file path>
- ENOVIA_LCAIPD_WPTOOL_PARAMLIST=<All other parameters>



Note

These environment values are mandatory when file introspection has not been activated for work packages. If they are undefined or have the wrong values, the work package support will not work.

Example for using the default introspection tool 'CATRpmPSU'

- ENOVIA_LCAIPD_WPTOOL_NAME=CATRpmPSU

- ENOVIA_LCAIPD_WPTOOL_IN_PARAM=-f
- ENOVIA_LCAIPD_WPTOOL_OUT_PARAM=-of
- ENOVIA_LCAIPD_WPTOOL_PARAMLIST=-iUser demo iPwd demo -iServer lcaserver:9018 -iRole VPMADMIN.ADMIN.DEFAULT



Note

*The value expression has to be quoted on **UNIX** if it contains spaces as above:*
 ENOVIA_LCAIPD_WPTOOL_PARAMLIST="-iUser demo -iPwd demo -iServer lcaserver:9018 -iRole VPMADMIN.ADMIN.DEFAULT")

*Quoting **must not** be done on **Windows**.*

If something is going wrong with the generation of the XML file it can be seen in the traces named 'BatchIPD'.

- The work package export can be deactivated by setting the environment variable ENOVIA_LCAIPD_WP_ENABLED=0

on the export side (on the ENOVIA server for interactive transfers and in the environment in which ProductDataGen is run for batch transfers).

Customizations of the XML File Location used for Export

The type of Work Package file which should be exported can be specified through the environment variable ENOVIA_LCAIPD_WPTYPE. Possible values are

- SECONDARY: secondary formats which have a MIME type of either xml or text/xml are exported; from the list of secondary formats which are attached to the currently exported Work Package document all XML files with the correct MIME type interpreted as Work Package XML file and transferred.
- REPORT: CATIA reports related to a V5 document get exported if they have a primary MIME type of either xml or text/xml. All XML reports with the correct MIME type found in the CATIA Report TOC are transferred.
- ALL: both of the above types.

The default setting is REPORT.

Example 1:

In DPE create a project for receiving the product data from ENOVIA V5 Engineering Hub.



- Start the ENOVIA VPM V5 Client and navigate to the product structure that contains the Work Package. The Work Package is marked with a special icon.
- Select the root of this structure, open contextual menu and select "Send to Manufacturing Hub".
- From the upcoming project list select your project and confirm the transfer.
- There should be no critical error messages in all software modules of the Connection.

- Open your project in DELMIA Process Engineer.
- In the Project Library you will find the imported product structure.
- Verify that the data file is attached to the Work Package object: Open the properties window of the Product component in DELMIA Process Engineer. Navigate to the 'Attachment' page and check if the work package XML file is there.
- Compare the content of this attachment with the file that is located in ENOVIA VPM V5 Database Server export directory.

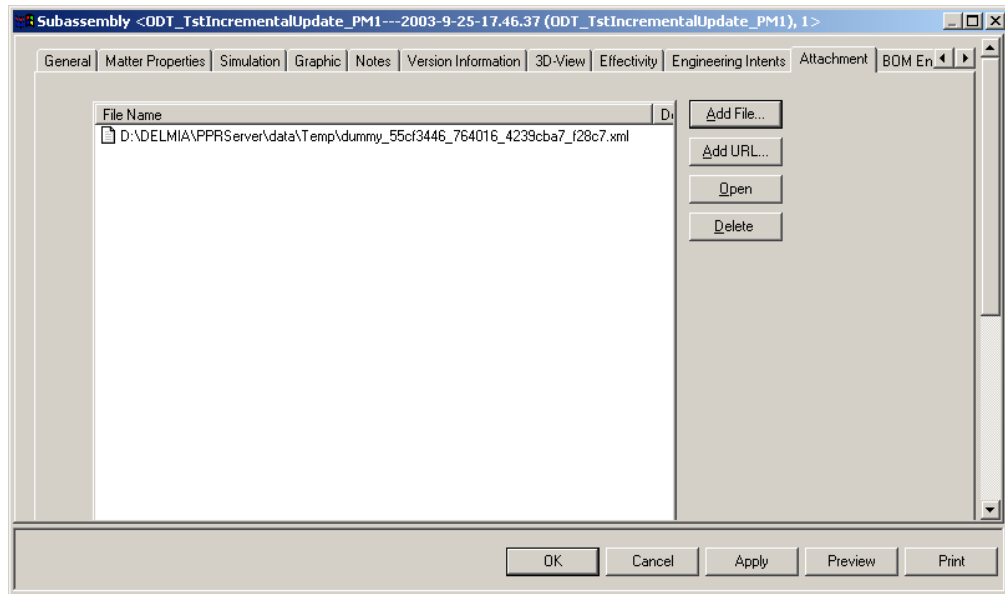


Figure 90: Comapre Contents

Example 2:

In DPE create a project for receiving the product data from ENOVIA V5 Engineering Hub.



- Start the ENOVIA VPM V5 Client and navigate to the product structure that contains the Work Package. The Work Package is marked with a special icon.
- Create CATIA report on this document (WP template and WP query are taken to exist in the database):
- Select BB-WP document and send to "Work Package Report Generator"
- On the Template tab, search for your report template (document with type "WP-Template"). Select it but DO NOT click OK.
- On the Query tab, search for your report query (document with type "WP-Query"). Select it and click OK.
- Click Save to database (with no unlocks).
- With the BB-WP document still highlighted, select the Document tab on the right-hand side of the product editor.

- Expand the CATIA Reports node to expose your report.

There will be a yellow icon to the left of it which indicates the report is still "processing".

- Continue refreshing from database and checking the icon to the left of the report until it turns green which indicates the report has finished processing and the XML report is done. If a red icon is displayed instead, this indicates a failure in the batch process.
- Save the product, set the environment variable ENOVIA_LCAIPD_WPTYPE to either REPORT or ALL, and send the product to the MH.

6.22 Support Product Maintenance Lifecycle

Data necessary to support Product Maintenance in the Manufacturing Hub is transferred from the Engineering Hub. In the Engineering Hub the list of Product Specifications of a given Product Root Class is extracted by ProductDataGen into XML files (for each Product Specification one XML file is generated), containing the following data:

- Product Specifications associated to the Product Root Class,
- The corresponding lists of Change Orders which themselves affect the Product Specifications,
- Information about the type of modifications to be made ("add", "cut", and "extend"),
- Identifiers of affected part instances

The XML files are imported into the Manufacturing Hub using the PPRLoader import application. The PPRLoader reads and parses the XML files and creates

- Change Orders along with their type information.
- Relations between Change Orders and Calculation Models (the Manufacturing Hub analogue to Product Specifications).
- Relations between Change Orders and the affected Part Instances.



Note

The MH data model differs from the corresponding EH one. Thus, not all objects and data are transferred from EH to MH, such as Actions and Modifications.

Export

The new functionality is available through a new command-line option "changeorder" to ProductDataGen. It accepts the id of a PRC.

```
ProductDataGen changeorder <product_id>
```

If only the change orders associated with a single product specification of a PRC are wanted, the id of the PS has to be added as a third argument

```
ProductDataGen changeorder <product_id> <productspec_id>
```


By default the output is placed into the standard temporary directory defined by the operating system. As usual it is possible to change this behaviour via configuration settings. The files follow the naming scheme:

CO_<product_id>_<productspec_id>-<modification_date_of_ps>.xml

Beginning with a Product Root Class all associated Product Specification are queried (a Carry-Over Product Specification is ignored).

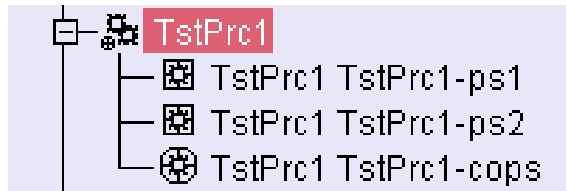


Figure 91: PRC with a COPS and two PS

For each Product Specification those Change Orders are queried which affect it. The relation between a Product Specification and a Change Order can have one of six states (In Work, Available, Incorporated, Non-Incorporated, Removed and Obsolete). They are reflected in the Manufacturing Hub data model as different types of relations. We iterate over the list of Change Orders and get Change Orders and, recursively, the associated Actions and Modification Statements. The Modification Statements can describe either an “add” or a “cut” operation of a part. Identifiers of all Item Instances referencing the affected parts in the current Product Root Class are exported. The information goes into a file. We generate one file for each Product Specification to allow a simple means for parallelization during the import.

Figure 92: Change Order with two affected Product Specifications

A reduced schema will be written into the file. Since actions and modification statements are not supported in the Manufacturing Hub, they won't be written into

the output. As a consequence, in the Manufacturing Hub the affected parts are directly connected to the Change Order.

Only the minimal necessary information will be placed in the generated files. For example, since each Product Specification has already to exist in the Manufacturing Hub, only an identifier will be used to allow the importer to search for them. The same is true for all affected Part Instances.

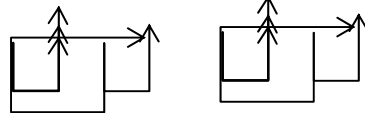
The generated files will follow a structure very similar to the one of the known transfer files. They will be transferred in binary format.

The whole process also works for Service Bulletins that require more than one Change Order to define the necessary configuration steps.

Import

Given an XML change order file, the import of change orders is done through the COLoader import application:

```
coloader -project $ProjectId -co
$Path\ChangeOrderFileName.xml
```



Where \$ProjectId is the id or number of the project, into which the Change Order information is to be imported. \$Path and \$ChangeOrderFileName.xml specify the location and file name of the XML file name to import, e.g.



Note

The import of COs requires an MCM project. Versionings for product plantypes in this project must be disabled, to allow the import of the corresponding product structure before. Versioning can be disabled for a specific plantype through the configuration manager (open "Library" > select required plantype > select "Edit" in context menu).

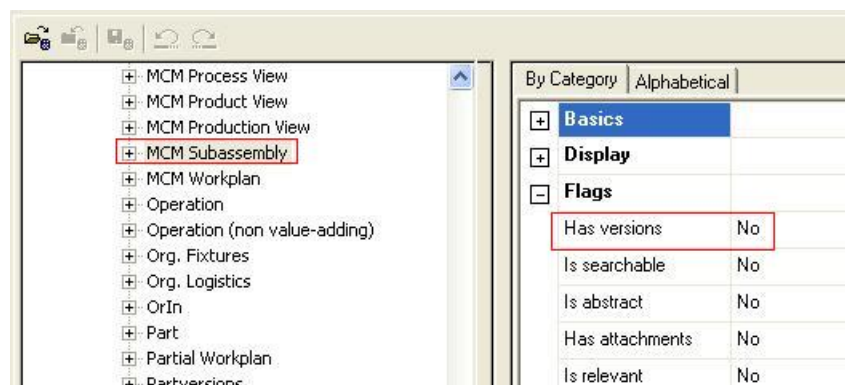
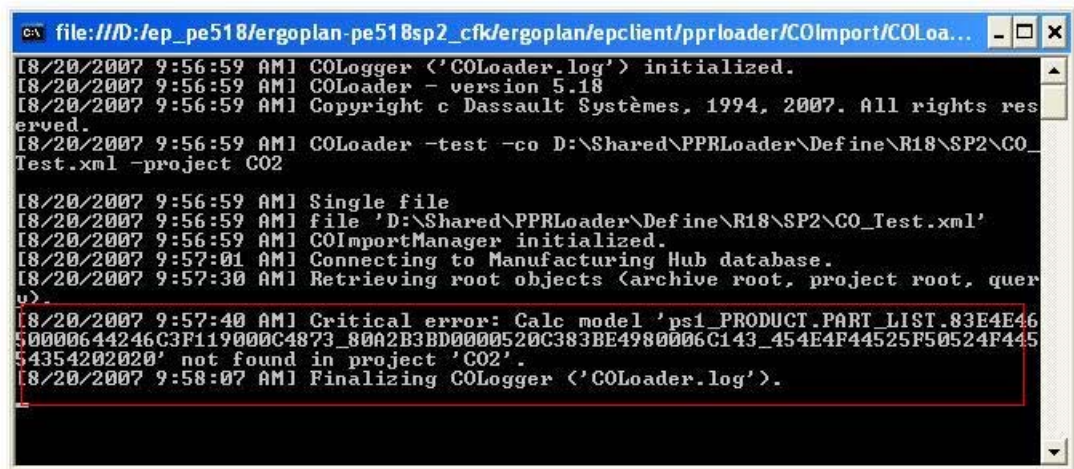


Figure 93: Config Manager

From the generated XML files, the importing COLoader application queries the corresponding Calculation Model in the given project (which is specified through the command line of the COLoader invocation). If a Calculation Model from the

Change Order export file does not exist in MH, COLoader will not create it, but dump a warning in the log file.



```

file:///D:/ep_pe518/ergoplan-pe518sp2_cfk/ergoplan/epclient/pprloader/COImport/COLoa...
[8/20/2007 9:56:59 AM] COLogger ('COLoader.log') initialized.
[8/20/2007 9:56:59 AM] COLoader - version 5.18
[8/20/2007 9:56:59 AM] Copyright c Dassault Systèmes, 1994, 2007. All rights reserved.
[8/20/2007 9:56:59 AM] COLoader -test -co D:\Shared\PPRLoader\Define\R18\SP2\CO_Test.xml -project CO2
[8/20/2007 9:56:59 AM] Single file
[8/20/2007 9:56:59 AM] file 'D:\Shared\PPRLoader\Define\R18\SP2\CO_Test.xml'
[8/20/2007 9:56:59 AM] COImportManager initialized.
[8/20/2007 9:57:01 AM] Connecting to Manufacturing Hub database.
[8/20/2007 9:57:30 AM] Retrieving root objects (archive root, project root, query)
[8/20/2007 9:57:40 AM] Critical error: Calc model 'ps1_PRODUCT.PART_LIST.83E4E4650000644246C3F119000C4873_80A2B3BD0000520C383BE4980006C143_454E4F44525F50524F4454354202020' not found in project 'CO2'.
[8/20/2007 9:58:07 AM] Finalizing COLogger ('COLoader.log').
  
```

Figure 94: Log File

Now, for the existing Calc Models, the related COs will be created and linked to the Calc Models they are assigned to. If they already do exist in MH, they will be left untouched (no update for CO objects!). The relation between CM and CO also carries the state attribute, e.g. “available” or “incorporated” in MH. COs which are no longer existent or do no longer affect the specified PS are deleted.

Since COs might have other CO children in ENOVIA, this structure is rebuilt in MH. Child COs are created and linked to their parent CO.

Once the Change Orders are imported, the affected part instances are “updated”. There are three cases to distinguish, “add”, “cut” and “extend” operations. In all cases, a relation between the part instance and the corresponding CO is created. The existing part instance is not touched at all. The operation does not create any parts in MH, even if they are linked by an “add” or “extend” operation to a CO. All part instances required or used by one of the imported COs must have been created by a previously run product structure update/import. COLoader will only treat COs and the corresponding relations, but will not create or delete part instances. E.g. if a part instances which is mentioned within a CO XML file is not present in MH (due to an error during the previous PPRLoader product update/import), an error is logged in the later COLoader run, but the part instance is NOT created by the COLoader, since COLoader cannot access all sufficient data to create the part instance properly

If a part is being deleted by the regular product structure import/update of the EH-MH Connection, the link to its related COs is lost, too. No further notification is given to the user (rather than the normal “Delete” message in the PPRLoader.log. Please note, that the link between CO and the part instance stays alive, if the EH-MH Connection uses the delete mode “notify”, i.e. if the part instance is not physically deleted but just marked so.

If the part instance to “add”, “cut” or “extend” is not present in MH, a warning is written to the COLoader.log file.



Note

A change order import will not affect the “updatestate” attribute of the related part instances at all, i.e. there’s neither an update state “new” for “add” or “extend” operations nor an update state “delete” for “cut” operations.

The PPRLoader creates CO objects of type “mfgchangeorder” in MH. Relations between COs and the corresponding CMs are of types “XDOSimpleRelation”. This relation carries the state attribute, e.g. “available” or “incorporated”.

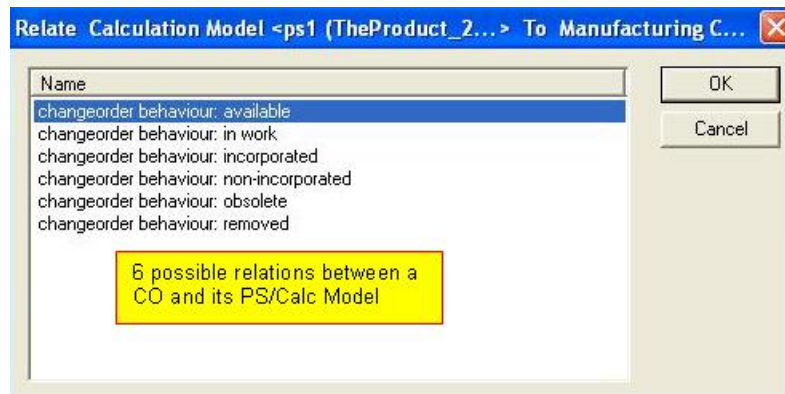


Figure 95: Relation between COs and CMs

COs and their children COs are linked through the relation “mfgchangeorder”.

For the relations between the COs and the added respectively cut part instances, relations of type “sb_add”, “sb_cut” and “sb_extend” are used (“sb” for service bulletin).

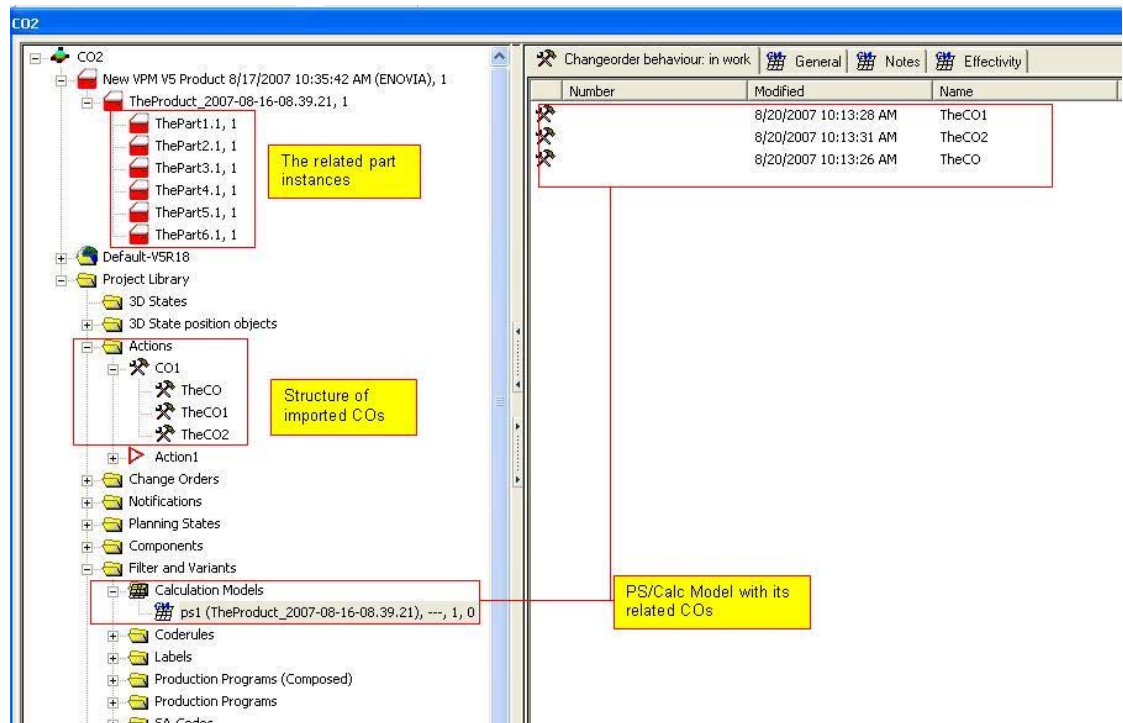


Figure 96: COs and their Children COs

6.23 ENOVIA Filter for EAR Data

It is possible to generate partner XML data with filtered EAR sensitive attributes. This is possible in one shot based on the partners' ECCN values, together with the unfiltered Connection XML data. For ENOVIA attribute values in the partner XML data which the partners must not receive, a customizable replacement text is used.

The DELMIA – ENOVIA VPM V5 Connection must know the ECCN values (permissions) for the Partners who should get XML Data. Therefore a Customization file is used where this information can be stored in a Permission Table.

This filter mechanism is independent of other customization options, e.g. Type Mapping. The defined attribute filters are only applied to parts which are sent over the DELMIA – ENOVIA VPM V5 Connection, e.g. they are ignored for parts which are filtered out due to additional extended filters.

Positioning with respect to Client Process

The generation of XML data for Partners will respect their ECCN permissions as defined in the Customization file's Permission Table.

Table 1: Customization File's Permission Table

| Terms | Description |
|--------------------|--|
| EAR | Export Administration Regulations |
| ECCN | The ECCN number or Export Control Classification Number is a unique number listed in the US Department of Commerce Export Administration Regulations (EAR). When information contained in design, manufacturing planning and support data contains export sensitive information under the jurisdiction of the United States Department of Commerce EAR, it must be marked with the correct ECCN. |
| ECCN value | Classifies a part regarding export control. Any Partner having the permission for this ECCN value may read all the attributes of this part. This is stored in a regular ENOVIA attribute. |
| Filtering | In this document filtering means replacement of EAR sensitive attribute values with a replacement value, e.g. "This value has been replaced". |
| XML data | The DELMIA – ENOVIA VPM V5 Connection exports data into XML files. These are referred to as XML data. |
| Partner | A partner in the context of this feature is an entity which has permissions describing which ENOVIA data may be seen. |
| Partner XML data | The partner XML files may contain filtered attributes depending on the partner's permissions. |
| Permission Table | Table which defines all Partners' permissions. Consists of a set of e.g. partners and their permission values. A permission value may be an ECCN number. |
| Customization file | This is an xml file containing information about the Permission Table, the replacement text and other customization information. |
| Attribute Filter | The ECCN values for the EAR sensitive attributes have to be stored in an attribute. This attribute can be customized in the Customization file. |

An environment variable sets the path for the Customization File:

```
ENOVIA_LCAIPD_CONFIGURATION_FILE=<path of customization file>
```

The Customization File contains a section of PRCs with corresponding Partners. It also contains a section defining the permissions which each Partner has. For the customized combinations of PRCs and Partners Partner specific XML files are generated which reflect the permission of the Partner.

For impacted attributes which the Partner has no permission for a text defined in the Customization File is used instead of the original content.

To achieve the described result the exporter has to apply a logic similar to the following:

For each part

```

    If <customized ECCN attribute> is empty
        overwrite EAR sensitive attribute values with replacement text
    Else
        For each ECCN value in <customized attribute>
            If value is not in the Partner's ECCN value list:
                overwrite EAR sensitive attribute values with replacement text
            End if
        End for each ECCN
    End if

```

End for each part

If there is no environment variable ENOVIA_LCAIPD_CONFIGURATION_FILE set, the DELMIA – ENOVIA VPM V5 Connection works as before (it generates unfiltered XML data) but does not create any Partner XML files.

Partner XML files are saved in the same directory as the reference file and have the name of the reference file together with the customized name in the export token, connected by “-”.

For example, if the reference xml file has the name xyz.xml, the partner file with the customized export token “Partner1” gets the name xyz-Partner1.xml.

Techniques of Interactions

If the environment variable ENOVIA_LCAIPD_CONFIGURATION_FILE is set correctly, and if there are PRCs with corresponding partners in the Customization file, the DELMIA – ENOVIA VPM V5 Connection generates Partner XML files.

Defining the path to the Customization file

On the EH-Server the user has to define an environment variable ENOVIA_LCAIPD_CONFIGURATION_FILE which contains a path to the Customization file. If this variable does not exist or if it has an empty value the DELMIA – ENOVIA VPM V5 Connection will work as before and create unfiltered XML data. If the file referred to is not found or is corrupt an error message will be put in the log file and the DELMIA – ENOVIA VPM V5 Connection will terminate.

Creating/Editing the Customization file

The Customization file contains the information from the Permission Table, the information about the connection between PRCs and Partners, the EAR sensitive attributes and for each attributes the replacement text, and other customization information. It may be created/edited with any text editor or with specialized XML editors depending on the user's preference.

For each "export" element defined for a PRC one additional reference XML file is generated.

Structure of the Customization file

The Customization file is an XML file. An example, *Please refer to the [XML-Based Configuration](#).*

Limitations for EAR Filter

- EAR filter is not available in interactive mode.
The purpose of the interactive mode is to update the MH on the same machine immediately and not to create Partner XML data to distribute to partners.
- Exactly one attribute filter is allowed, either an attribute of part version or an attribute of part master. It is not allowed to customize two attribute filters in the same customization file.
- Only String- and Multi Value Attributes can be used as discriminator attributes.
- As mentioned in the section Customization/Export, this feature produces partner reference files. That means: No partner instance files are created.

Example: Generation of a partner XML file.

DPE

Create two new empty projects "EXAMPLE_1_EAR_FILTER" and "EXAMPLE_2_EAR_FILTER" using the Standard-V5 plan-type set.

ENOVIA:

Create a new product structure, containing a PRC R17GPL375 and three parts, Part_R17GPL375_1, Part_R17GPL375_2 and Part_R17GPL375_3.



Figure 97: New Product Structure

In Attribute description of Part_ R17GPL375_1 insert
"AAA02 AAA03"

| 2 - Characteristics | |
|---------------------|------------------------|
| Part Number | Part_ R17GPL375_1 |
| Name | EAR sensitive data ... |
| Description | AAA02 AAA03 ... |

Figure 98: Attribute Description of Part_ R17GPL375_1

Leave Attribute description of Part_ R17GPL375_2 empty

In Attribute description of Part_ R17GPL375_3 insert
"AAA01"

| 2 - Characteristics | |
|---------------------|------------------------|
| Part Number | Part_ R17GPL375_3 |
| Name | EAR sensitive data ... |
| Description | AAA01 ... |

Figure 99: Attribute Description of Part_ R17GPL375_3

Put "EAR sensitive data" to the Attribute Name. Together with the Attribute V_user, which originally contains the user name, these attributes are referred to as sensitive EAR attributes below.

Administrative task:

For convenience make a small change in Plantype Subassembly of the Plantype-set DefaultV5-R16:

Make the attributes attribute_6, attribute_7 and attribute_8 visible in Browser (Go to the Library, select the Plantypeset, choose "Edit" from the context menu of the Subassembly in the Product Plantypes. Navigate to the attributes; Set "Display in Browser" to yes and Position in Browser to a small value.)

Protocol

Define the environment for the exporter respectively on the server as follows:

- ENOVIA_LCAIPD_CONFIGURATION_FILE= "<Path to Customization file>".
- Use the attached customization file "export-configPES375.xml"

```
<configuration>
  <prcs>
    <prc id="R17GPL375">
      <exports>
        <token>Export01</token>
        <token>Export02</token>
      </exports>
    </prc>
  </prcs>

  <permissions>
    <permission id="Export01" description="Test Export Per-
missions 01">
      <token>AAA01</token>
      <token>AAA02</token>
    </permission>
    <permission id="Export02" description="Test Export Per-
missions 02">
      <token>AAA02</token>
      <token>AAA03</token>
    </permission>
  </permissions>

  <filters>
    <attributefilter context="PartMaster"
attribute="V_description" delimiter="" ignorecase="no">
      <impact attribute="V_name">
        <replacement>POTENTIAL EAR-SENSITIVE DATA|SEE THE
MBD FOR COMPLETE DEFINITION</replacement>
      </impact>
      <impact attribute="V_user">
        <replacement>POTENTIAL EAR-SENSITIVE DATA|SEE THE
MBD FOR COMPLETE DEFINITION|FILLER TEXT</replacement>
      </impact>
    </attributefilter>
  </filters>
```



```
</configuration>
```

- ENOVIA_LCAIPD_OUTPUT_DIR = "<Path to export directory>".

Start the DELMIA – ENOVIA VPM V5 Connection (command: ProductDataGen product R17GPL375).

Import the xml-files created for the partner into the resp. DPE projects (command: pprloader –project <the project numbers see above> -vpm <the partner xml file-name, see below>).

- Use the attached import customization file “import-configPES375.cfg” for import. You have to insert the absolute path to the file in the registry in the key:

```
[HKEY_CURRENT_USER\Software\Delmia\ergopl an\pprloader\VPM
] cfgfile=<absolute path>
```

Result on DPE Side

Two Partner XML files have been generated:

<originalfilename>-Export01.xml and

<originalfilename>-Export02.xml.

(e.g. EXAMPLE_2_EAR_FILTER005-12-08-14.33.58-Export01.xml)

Open project “EXAMPLE_1_EAR_FILTER” in DPE. Navigate to the Products in the project library.

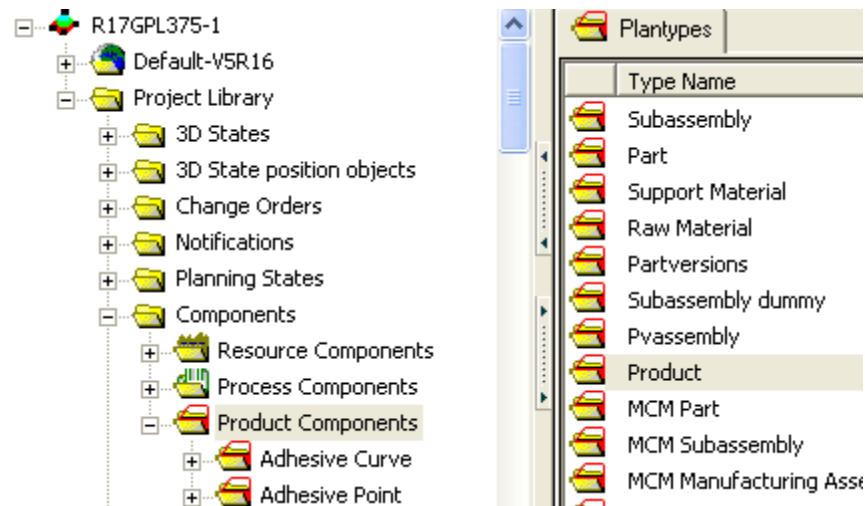


Figure 100: Navigate in Project Library

Open the Product folder, where the PRC has been transferred to below the node New VPM V5 Product with the date and time of the import. There the structure should look like this:



The screenshot shows a software interface with a tree view on the left and a table on the right. The tree view shows a hierarchy starting with 'Product', followed by 'New VPM V5 Product 15.12.2005', then 'R17GPL3752005-12-08-14.3', and finally three sub-components: 'Part_R17GPL375_1.1', 'Part_R17GPL375_2.1', and 'Part_R17GPL375_3.1'. The table on the right has four columns: 'Component Number', 'Attribute 6', 'Attribute 7', and 'Attribute 8'. It contains three rows of data for the components.

| Component Number | Attribute 6 | Attribute 7 | Attribute 8 |
|------------------|-------------|----------------------------------|----------------------------------|
| Part_R17GPL375_1 | AAA02 AAA03 | POTENTIAL EAR-SENSITIVE DATA ... | POTENTIAL EAR-SENSITIVE DATA ... |
| Part_R17GPL375_2 | | POTENTIAL EAR-SENSITIVE DATA ... | POTENTIAL EAR-SENSITIVE DATA ... |
| Part_R17GPL375_3 | AAA01 | EAR sensitive data | GWN |

Figure 101: Resulting Structure 1

The sensitive EAR attributes in Part_ R17GPL375_3 have not changed because "Export01" has the permission for "AAA01".

The sensitive EAR attributes in Part_ R17GPL375_1 have been overwritten with the customized replacement because "Export01" does not have the permission for "AAA03" as well as the sensitive EAR attributes in Part_ R17GPL375_2 because an empty permission attribute will always lead to the replacement of the impacted data.

Open project "EXAMPLE_2_EAR_FILTER" in DPE. Navigate to the Products in the project library:

Open the Product folder, where the PRC has been transferred to below the node New VPM V5 Product with the date and time of the import. There you should get the following result:

| Component Number | Attribute 6 | Attribute 7 | Attribute 8 |
|------------------|-------------|--------------------------------|------------------------------|
| Part_R17GPL375_1 | AAA02 AAA03 | EAR sensitive data | GWN |
| Part_R17GPL375_2 | | POTENTIAL EAR-SENSITIVE DAT... | POTENTIAL EAR-SENSITIVE D... |
| Part_R17GPL375_3 | AAA01 | POTENTIAL EAR-SENSITIVE DAT... | POTENTIAL EAR-SENSITIVE D... |

Figure 102: Resulting Structure 2

The sensitive EAR attributes in Part_ R17GPL375_1 have not changed because "Export02" has the permission for both "AAA02" and "AAA03".

The sensitive EAR attributes in Part_ R17GPL375_2 have been overwritten with the customized replacement text, again because the permission attribute value is empty, as well as the sensitive EAR attributes in Part_ R17GPL375_3 because "Export02" does not have the permission for "AAA01".

With the provided Configuration File the ENOVIA attribute Description is mapped to the attribute "attribute_6" in DPE, Name is mapped to the attribute "attribute_7", the ENOVIA attribute V_user is mapped to the attribute "attribute_8" in DPE.

6.24 Customer Specific File Markings

When DELMIA – ENOVIA VPM V5 Connection XML files are distributed to partner companies, it may be required to mark these files according to their sensitivity before exporting them to the partners. This is necessary because the XML files are human readable files and therefore anybody opening such a file (e.g. with an editor/viewer) shall be clearly indicated that the proprietary/sensitive data is present.

The user will be able to specify a fixed note, which will be included in all XML files that are generated by/under control of the DELMIA – ENOVIA VPM V5 Connection. The described capabilities are supported for batch mode only. If markings are defined in the customization file on the ENOVIA server side used for interactive transfers, they are ignored.

The user has to customize a configuration file to define the file markings. If this file does not exist or no markings are defined within it, no file markings will be created and the behavior is the same as in R16.

| | |
|---------------------------------------|---|
| every xml file the exporter generates | Refers to reference and instance XML file as of today; this does not include files which are retrieved from the ENOVIA vault, e.g. work package XML files |
|---------------------------------------|---|

Example:

Fixed text file marking: The text will be included in every xml file the exporter generates.

I.E.:

THE INFORMATION CONTAINED HEREIN IS PROPRIETARY TO THE XYZ COMPANY AND SHALL NOT BE REPRODUCED OR DISCLOSED IN WHOLE OR IN PART OR USED FOR ANY PURPOSE EXCEPT WHEN SUCH USER POSSESSES DIRECT, WRITTEN AUTHORIZATION FROM THE XYZ COMPANY

On the EH side no effect of the file markings will be visible.

Customization

The file markings are defined in a customization file. The path and name of this file has to be set in the environment variable ENOV_LCAIPD_CONFIGURATION_FILE.

Each file marking has to be defined in the customization file:

Fixed Text Files Marking:

```
<markings>
  <marking>THE INFORMATION CONTAINED HEREIN CONTAINS
  TECHNICAL DATA.</marking>
</markings>
```

Limitations

- Only human readable files will be marked.
- Only String- and Multivalue-Attributes can be used to control the variable file marking.
- The described capabilities are supported for batch mode only. If a marking is defined in the customization file on the ENOVIA server used for interactive transfers it is ignored.

6.25 Specification Alias Handling

Specifications are used as Boolean operators which can be used to filter parts of a product structure. Specifications are linked to a category. In the EH it is possible to define an alias on a specification. It is possible to use the same alias in the MH after the product has been transferred. Defined specification aliases are honored. It is possible to switch between a specification ID and a given alias in subsequent runs of the EH to MH transfer.

Specification aliases are taken into account both in effectivity terms and filters. If a specification alias is defined in the EH, references to it are changed: Given a category with two specifications of which one has an alias defined:

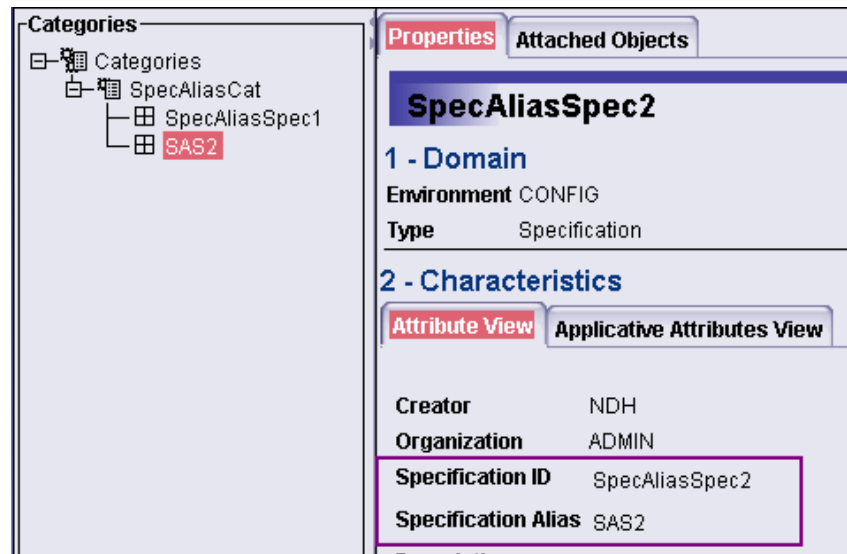


Figure 103: Category with two Specifications and an Defined Alias

If an attached product contains a part with an effectivity that references the specifications, the alias is seen instead of the identifier:

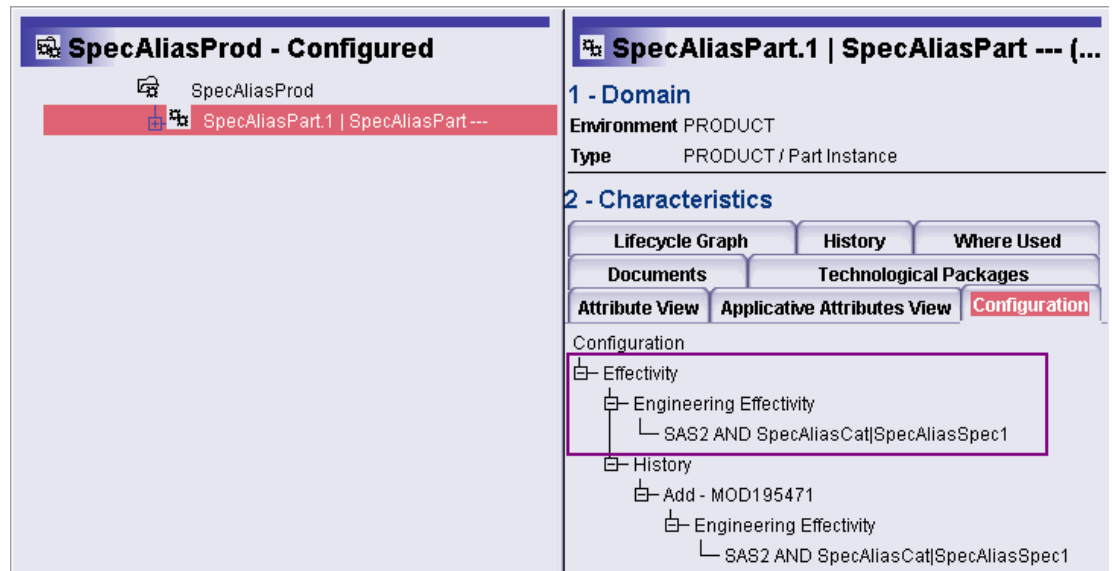


Figure 104: Visible Alias

In the same way the specification alias is used in product specifications and filters:

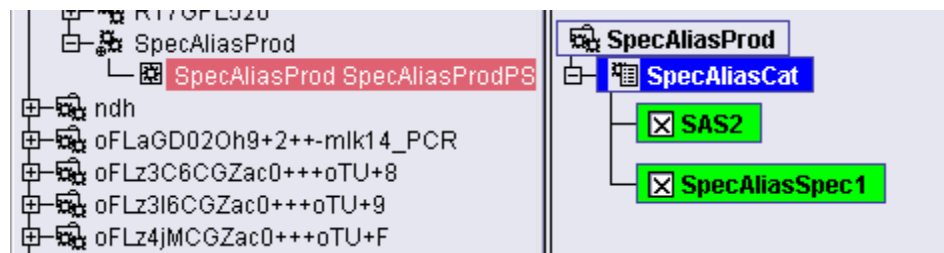


Figure 105: Specification Alias used in Product Specifications and Filters

Both changes are visible in the MH as well. After a transfer the extended effectivity of the part looks like:

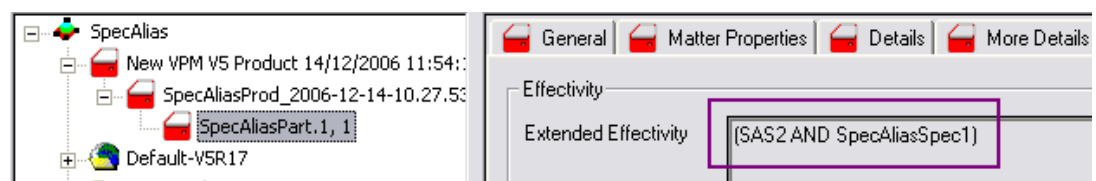


Figure 106: Changes Visible in the MH

The model configuration of a filter generated from the above defined product specification is:

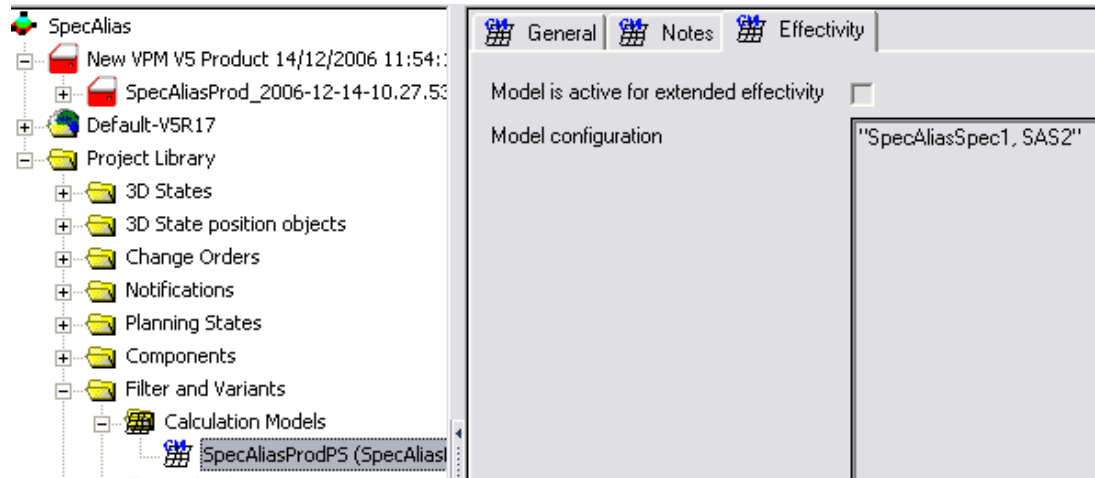


Figure 107: Model Configuration of a Filter



Note

Specification identifiers are replaced by specification aliases if they are defined. The specification alias is visible (and used) in the MH in the same way it is in the EH. If a specification alias is removed the original specification identifier will be reused.

Limitation

Special characters like a pipe (|) or an ampersand (&) are allowed in the EH but should not be used, since they can be mixed with filter operators in the MH.

6.26 New Binary Format for XML Product Structure Files

The R17 release includes a new DELMIA – ENOVIA VPM V5 Connection binary format implemented to increase the reliability and security of the DELMIA – ENOVIA VPM V5 Connection.

The DELMIA – ENOVIA VPM V5 Connection data format will change from a human readable XML format to a binary format. On the export side of the DELMIA – ENOVIA VPM V5 Connection the generated data will be written in a binary format. No readable ASCII structure data is written to storage devices. The purpose is to prevent unauthorized use of the XML data that is being used to transfer product data between Engineering and Manufacturing Hub.

Both Batch Mode and Interactive Mode are supported by this highlight. Only XML files used for DELMIA – ENOVIA VPM V5 Connection internal product data transfer are affected. Data files extracted from the ENOVIA Vault (e.g. Work Package XML files) will not be touched.

On EH-side the XML data is written in a special file format. The name of the binary file is not changed compared to previous releases, it consists of the exported PRC name followed by a timestamp.

The compatibility with existing data is given. The code will detect ASCII XML files and process them appropriately. Incorrect binary files will result in a program abort.

Limitation

Only XML files generated by the DELMIA – ENOVIA VPM V5 Connection will get transformed.

6.27 File Transfer via HTTPS, HTTP, or FTP

The usage of HTTPS will ensure the secured data transfer between ENOVIA VPM V5 Engineering Hub side and DELMIA Manufacturing Hub side.

On the ENOVIA VPM V5 side, it must be ensured, that the data export repository (i.e. the directory where all XML, cgr and log files - such as lxc and lxc_log - are stored) is accessible by the http daemon.

On the DELMIA side it must be possible to switch between the current FTP file transfer mechanism and an HTTPS based one (disable FTP, enable https). The required information (HTTPS host, access information, etc.) must be registered, like today the analogous information for FTP (e.g. registry settings).

The switch between FTP and HTTPS is part of the customization of the DELMIA – ENOVIA VPM V5 Connection.

Only one connection can be used at one time. The bunch of XML-files has to be stored before sot (= start of transfer) in the same main directory. Subdirectories with different contents are recommended, but a clear and unequivocal nomenclature has to be done.

Because of transferring the data in a factorized format, or if you prefer, in small well-formed XML-portions like pearls, no serious problem arises by any interruption of data lines. Perfectly transferred file-pearls are fulfilling the XML-Specifications and need not to be transferred again. The factorized or pearl-based-transfer helps to overcome the horizon of impossible transfers for large systems represented in the XML-file-structure. The XML-Validation can be done easily by already existing software, resp. APIs.

The usage of HTTPS instead of FTP requires different customization, as well on ENOVIA side (ensure the data export repository is accessible by an http client) as

well as the DELMIA side (register https host and access information instead of the FTP access data). Once this customization is done, there's no difference between the FTP and https data transfer for the end users perspective.

Prerequisites

On the ENOVIA VPM V5 side, it must be ensured, that the data export repository (i.e. the directory where all XML, cgr, ABF fastener and log files - such as lxc and lxc_log - are stored) is accessible by the http daemon.

On the DELMIA side it must be possible to switch between the current FTP file transfer mechanism and an HTTPS based one (disable FTP, enable https). The required information (HTTPS host, access information, etc.) must be registered, like the analogous information for FTP (e.g. registry settings).

The switch between FTP and HTTPS is part of the customization of the DELMIA – ENOVIA VPM V5 Connection.

The transferred data have to be inserted into the Manufacturing Hub. None of the transfer-files is intended to residue in a repository.

PPRDaemon fetches files from the ENOVIA VPM V5 side.

The following files have to be transferred

- The ENOVIA VPM V5 XML export file
- The .lxc_end file (which indicates that the XCAD server has completed the generation of the CGR files)
- The CGR files (which allow the geometrical representation of the product in DPE)

In previous versions, the FTP mechanism relied on the direct customization of scripts. The current version reads the required settings from the registry and executes the corresponding HTTPS/HTTP/FTP command directly. The modified approach is reflected by new registry settings. Settings for the secured transfer have to be done on DPE side only.

1. securetransfermode: value 0 = FTP; value 1 = HTTPS

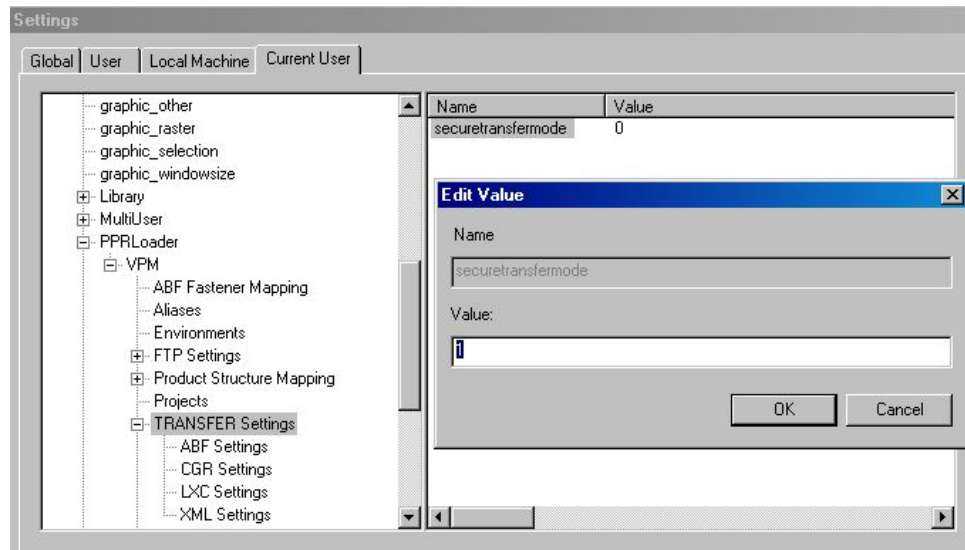


Figure 108: Transfer Settings

2. xmlhttpsbaseurl: value: HTTPS://sunlcadeg/<shared xml repository>

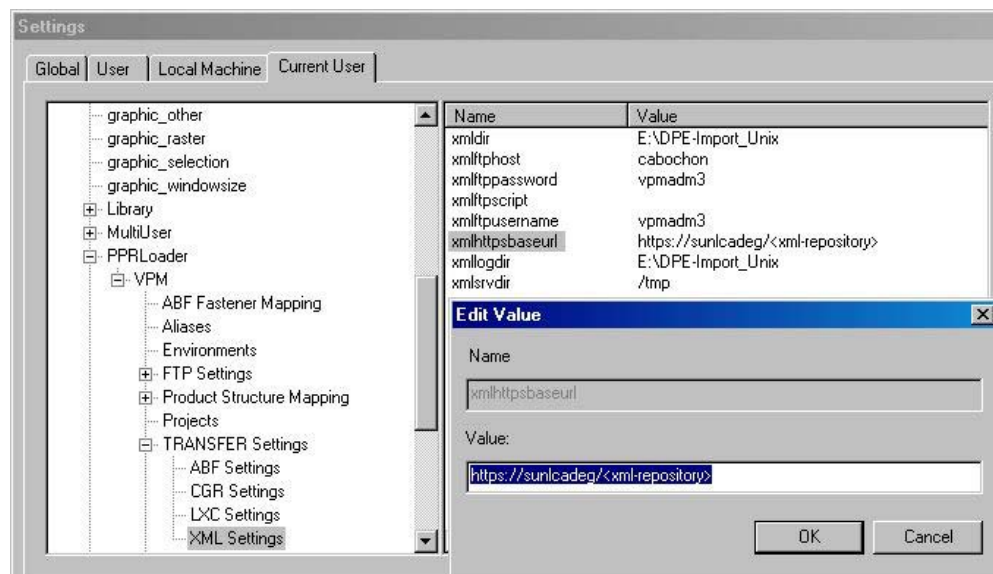


Figure 109: XML Settings

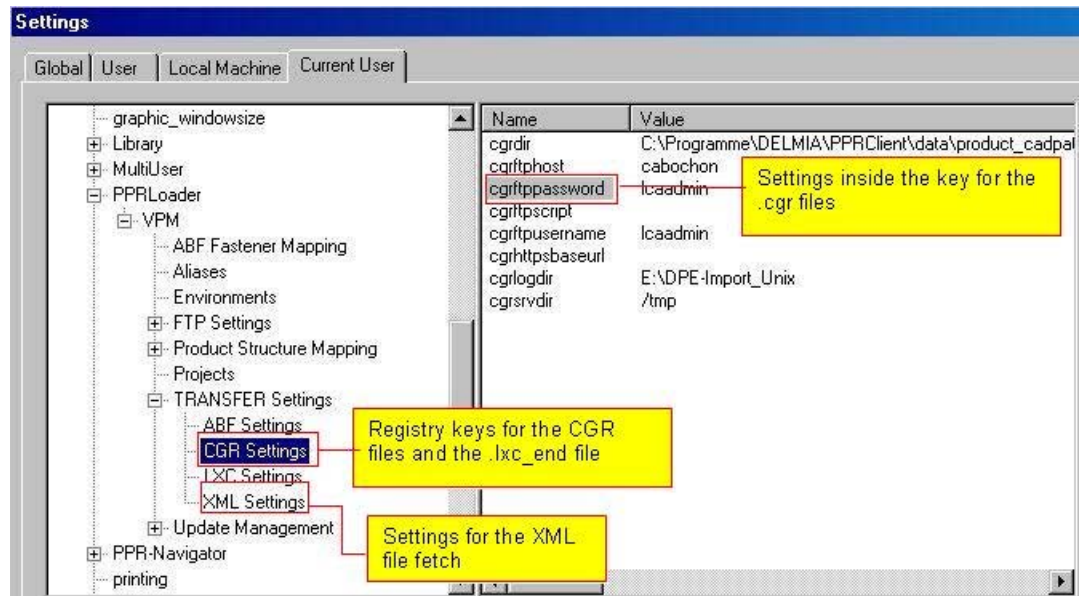


Figure 110: Registry Keys

The registry keys in detail:

- xml/cgr/lxcdir: the directory where the fetched file(s) is/are finally stored
- xml/cgr/lxcftp host: the ENOVIA V5 server host to connect through HTTPS or FTP
- xml/cgr/lxcftp password and xml/cgr/lxcftp username: the FTP account information to be used to login to the ENOVIA V5 server
- xml/cgr/lxclogdir: the log directory where temporary and log files are created
- xml/cgr/lxcsvrdir: xml/cgr/lxc files are stored in this directory on Unix Server side
- lxcftp wait: the number of tries for the FTP fetch of the .lxc_end file before passing by. A value "infinite" indicates that PPRLoader tries forever.
- lxcftp delay: contains the time span in seconds between two tries for the .lxc_end file

The registry keys lxcftp delay and lxcftp wait do not have any counterpart in the registry key for the XML file and the CGR files.

As soon as the lxc_end file is created on the ENOVIA VPM V5 server side, the cgr files can be transferred to the DELMIA Process Engineer side. I.e. these settings permit to control indirectly the cgr transfer.

As in the previous version, the registry keys and settings regarding the .lxc_end file are a little bit misleading, since they are still named lxc. The .lxc file itself, as well as the .lxc_log file from the DMU Utilities are not used by PPRLoader, a fetch of those files via FTP is not required.

**Note**

XMLDIR, XMLLOGDIR, LXCLOGDIR, LXCDIR must be equal.

CGRDIR must be equal to the Global/graphic_editor/cadpath value in order to visualize the CGRs in DPE, e.g. PPRClient\data\product_cadpath

6.28 Secure Registry Settings

The new security features of R15 also include passwords set by the DELMIA – ENOVIA VPM V5 Connection. To match customers' security requirements, all the password registry settings must be stored encrypted in the registry and the transfer of the passwords has to be done encrypted.

- The user has to enter the registry settings for the passwords which have to be secure. To be able to enter them the administrator has to use the administration tool "SimpleCryptAdmin" available for this action.

Handling the password to get access to the Manufacturing Hub Server:

Doesn't matter if there is an update from an earlier release to release 15 or if the release is newly created. At first the administrator has to update/enter the corresponding registry key value. If it is a release update, the password is not encrypted with the encryption algorithm valid from release 15 on. If it is new, the password has to be entered anyway.

```
[HKEY_CURRENT_USER\Software\Delmia\ergoplan\PPRLoader]
```

Change the value of the entry "password" so that the password can be read in readable format.

Now the administration tool

```
"SimpleCryptAdmin" -ek [0|1] RegistryKey  
[AES|3DES|DES|RC4|RC2]
```

can be used to encrypt the value with the new encrypting algorithm.

**Note**

"pprloader -login user/password" is NOT supported any more!

- In case of a release update to R15 or R16, the passwords used before have to be encrypted in the following manner. Otherwise this part is handled by the given registry file newpprloader.reg:

For the **migration** to release 15 or 16 the registry key "FTP settings" has to be replaced by "TRANSFER Settings" in registry directly (see at picture below where this is already replaced).

The same action, i.e. replace "ftp" by "transfer" within the strings of the registry entries, has to be done for all strings under registry key

```
[HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings\CGR Settings]
```

```
[HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings\LXC Settings]
```

```
[HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings\XML Settings]
```

```
[HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings\ABF Settings]
```

- In case of a **new installation** of R15 or R16 use the administration tool "SimpleCryptAdmin" -ek [0|1] RegistryKey [AES|3DES|DES|RC4|RC2] to encrypt the value with the new encrypting algorithm four times:

```
[HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings\CGR Settings]
```

```
"cgrftppassword"="<Password>"
```

```
[HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings\LXC Settings]
```

```
"lxcftppassword"="<Password>"
```

```
[HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings\xML Settings]
```

```
"xmlftppassword"="<Password>"
```

```
[HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings\ABF Settings]
```

```
"abfftppassword"="<Password>"
```

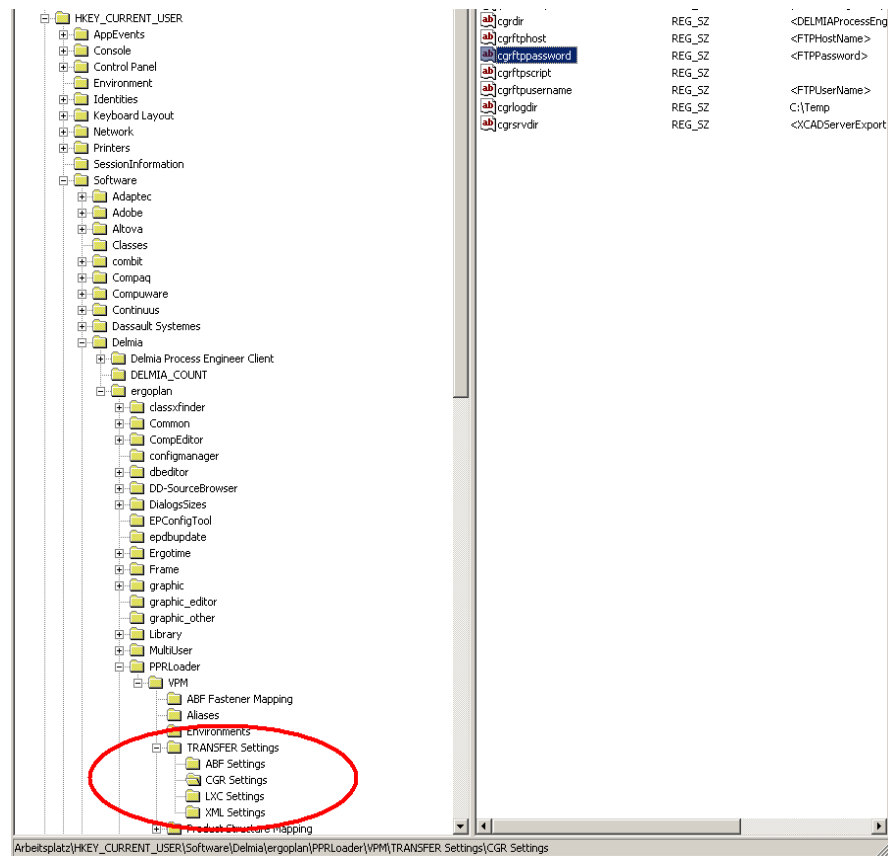


Figure 111: Transfer Settings

Generally to decrypt the secure registry settings for the designated registry keys the entered key values have to be handled by a component called *Session Data Manager*. This is a uniform module handling decrypting of the key values in a uni-

form way. Once the key values are encrypted, they are written to the registry where they can be seen only in the encrypted format. Afterwards if the key values are needed by the program, the encrypted key values are taken from the registry and decrypted by the *Session Data Manager* module. So they can be treated as usual. Beside the decryption the *Session Data Manager* module also treats the write/read actions to the registry. Thus all requests done to the registry are handled by the *Session Data Manager* module. The transfer of the password to get access to the Manufacturing Hub Server will also be done in an encrypted form, what means that a hash function is used to encrypt this password before the transportation to the server. On server side the originally entered password located in the data base of the server was also encrypted by this hash function. Thus the encrypted passwords can be compared.

The transfer of the passwords used to get information via FTP/HTTPS is encrypted within the HTTPS protocol. The web server containing the data which is the target of the HTTPS protocol uses its own algorithm to encrypt the password after unpacking it from HTTPS protocol.



Note

Encrypted data (e.g. passwords) are valid only on the machine where they are created; they cannot be decrypted on a different machine

Example:

```
"SimpleCryptAdmin" -ek [0|1] RegistryKey
[AES|3DES|DES|RC4|RC2]
```

-ek: Encrypt the value of the given registry key using the given encryption algorithm

0: machine key container would be used for encryption

1: user key container would be used for encryption

RegistryKey: Name of the registry key

Remark: Choose "0" and "3DES", the RegistryKey has to be typed in without brackets and string name included.

Example

```
SimpleCryptAdmin -ek 0
HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings\xmltransferpassword 3DES
```

or alternatively:

```
SimpleCryptAdmin -ekv 0
HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings\xmltransferpassword <klartext password> 3DES
```

If the password found in registry differs from the clear text password, the encryption was done.

Check if the data transfer via the HTTPS connection was successful. Thus decryption was done and workflow as usual.

6.29 Reference Import

ENOVIA VPM V5
(reference tree)

DELMIA Manufacturing Hub
(instance tree)

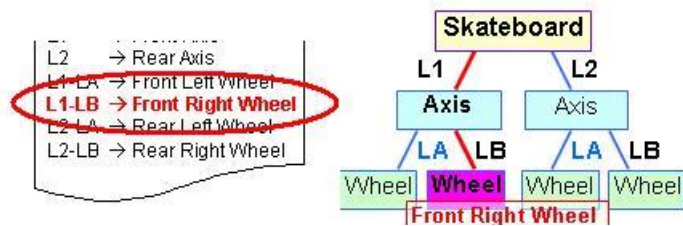
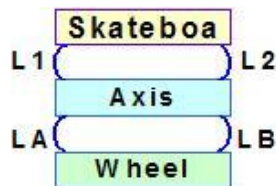


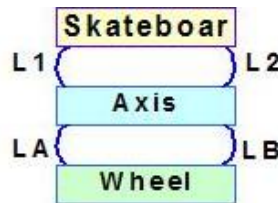
Figure 112: Reference Import

In some cases (e.g. for resources) it might be required to transfer the data as references into the Manufacturing Hub. This results in a reference tree in Manufacturing Hub.

ENOVIA VPM V5 (reference tree)



DELMIA Manufacturing Hub (reference tree)



Using a reference tree import dramatically reduces the amount of data (number of objects) to be imported into Manufacturing Hub. As a drawback, there are several, critical restrictions, which will make this feature only usable in some cases/customer scenarios.

To import a VPM XML export file as a reference tree into DELMIA Manufacturing Hub, a new command line option “**-reference**” is introduced to PPRLoader:

```
pprloader -vpm $VPMXMLExportFile -project $ProjectIdOrShort-Name -reference
```

No further settings / customization are required.

- It is up to the user to ensure a reference update is not performed against a regular instance based product structure in Manufacturing Hub. This would result in undetermined behavior.
- Also, reference imports do not support incremental and partial updates.
- It is also obvious that all instance related functionality cannot be applied to this feature (e.g. instance attributes, transformation management and several configuration/effectivity related features).
- This functionality is only available in batch mode.



Note

From V5R13 on the XML file structure has changed. This will cause errors when using ENOVIA R12 XML export files. If you want to import XML export files from ENOVIA V5R12, please set the following registry key

```
[HKCU\Software\DELMIA\ergoplan\PPRLoader\VPM]“release”=“R12”
```


6.30 Transfer of Configuration Data

To support properly configuration data from ENOVIA VPM V5, use Extended Code Rules for the desired project. For a detailed overview on how configuration data is mapped to Manufacturing Hub, *Please refer to the [Appendix A](#)*

[Configuration Scenarios](#) –
for details.

In ENOVIA VPM V5 all configuration data is related to the link, but the DELMIA – ENOVIA VPM V5 Connection can be customized to support two different ways of storing configuration data:

- On the links (relative effectivities)
- The instances themselves (absolute effectivities, this ensures the support for drag'n'drop to an MBoM and product–process configuration inheritance mechanism)

Basically, links (=Manufacturing Hub SubCompltems) and components (=Manufacturing Hub ErgoCompBase, e.g. ErgoCompProductDefault) contain one attribute, which is used to store configuration data in the case of the DELMIA – ENOVIA VPM V5 Connection:

- extendedeffectivity // e.g. '(AFR&{1-20})|(ACD&<2005/07/01-*>)'

As already mentioned, this attribute contains the relative (extended) effectivity for links, while it contains the absolute effectivity for instances.

Beside the extended effectivity attribute, there are four additional attributes, which can be useful sometime to store configuration data. Their usage matches the behavior in the DELMIA – ENOVIA VPM Connection:

- effectivity.begin // start date effectivity; e.g. '31-01-2005'
- effectivity.end // end date effectivity; e.g. '31-03-2006'
- tailnumber // range effectivity; e.g. "1-5", "10-20"
- coderulestring // options, e.g. DIESEL+SUNROOF+4DOOR

Please note, that the usage of those attributes has some strong limitations for complex effectivities (*Please refer to the [Appendix A](#)*

[Configuration Scenarios](#) –

for further details). In addition, using those attributes will not create correct absolute instance effectivities, thus causing problems in an MBoM context.

To define which of the two ways to store configuration data should be used, please use the following registry keys:

```
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM] "cfgextended" = "
```

```
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM] "cfgnormal" = " "
```

If none of the above is specified, the only the extended effectivity attribute is used, but none of the regular ones. If you want to get the regular configuration attributes (similar the DELMIA – ENOVIA VPM V4 Connection), use the following setting.

```
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM] "cfgnormal"="1"
```

With this, the extended effectivity is still set. To suppress it, use in addition

```
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM] "cfgextended"="0"
```

Up to DPE 5 R15 the effectivities in the Engineering Hub and the Manufacturing Hub differ, at least in their respective representation. This caused some confusion. From R16 on the effectivity terms defined in the EH are transferred to the MH. The user will see the same terms on both sides of the DELMIA – ENOVIA VPM V5 Connection.

The user now is able to compare effectivity terms in EH and MH without trouble, because they are the same.

Consider the following effectivity term in the ENOVIA Engineering Hub:

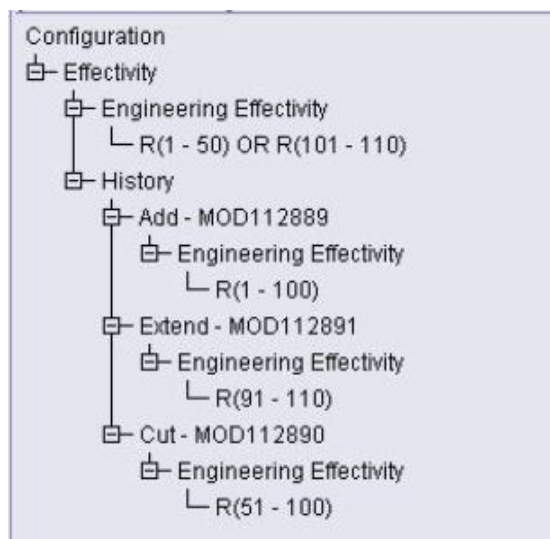


Figure 113: Effectivity Term in EH

In R15, this leads to the following effectivity expression in the DELMIA Manufacturing Hub:

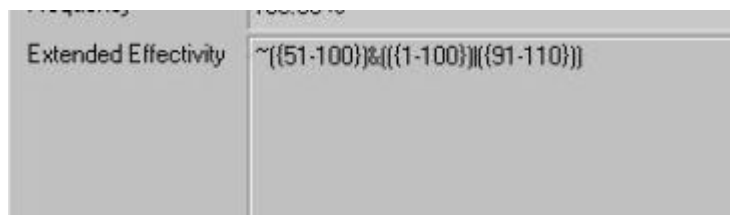


Figure 114: Effectivity Representation in MH as of R15

This is logically the same, but the user has to understand both representations and map one onto the other.

In DPE R16, the same expression will be seen in the MH:



Figure 115: Effectivity Representation in MH as of R16

The effectivity terms will be stored in the MH. This is only a change of the existing representation; no new data has to be created.

No special treatment is necessary to activate this feature or make use of it.

In MH clients the customer has the ability to choose which terminology should be used (is understood) when filtering on a product structure. This is a project attribute.

The default setting: Use E5 logical operators "&" (ampersand) and "|" (pipe).

ENOVIA_SYNTAX_ALLOWED=1.

Now in addition to above, "AND" and "OR" are also allowed logical operators.

ENOVIA_SYNTAX_ONLY=1.

In this mode, only "AND" and "OR" are valid logical operators.

Existing data is not touched by the new highlight. In the export code an additional attribute will be inserted into the XML file which is transferred from the EH to the MH. The new attribute contains the effectivity term referenced in the assembly relation currently looked at.



Note

*This is **not** the computed effectivity, but only the result of all local modifications.*

Taking the example above, the additional data looks like

```
<Attribute Name="Node Effectivity" Type="String">
  <Value Value="(R(1 - 50)) OR (R(101 - 110))" />
</Attribute>
```



Note

Transfer of Configuration Data impacts ENOVIA VPM V5 users only. The behavior for ENOVIA VPM V4 users is unchanged. Older XML-files result in DPE R15 behavior (ENOVIA VPM V5 users only).

6.31 Domain Effectivity

ENOVIA VPM V5 is capable of managing different effectivities for different domains (e.g. like Engineering, Manufacturing, Maintenance, etc.). Still, there's just one domain effectivity transferred across the Connection and stored in Manufacturing Hub. The definition of which domain effectivities to export is customizable on the ENOVIA VPM V5 side.

ENOVIA VPM V5 Connection must know where to find the Export Configuration File. The new environment variable ENOVIA_LCAIPD_CONFIGURATION_FILE must be set and contain the path to the Export Configuration File.

The Configuration file contains a section of effectivity domain information. This contains a list of ENOVIA effectivity domains which will be written into the export files.

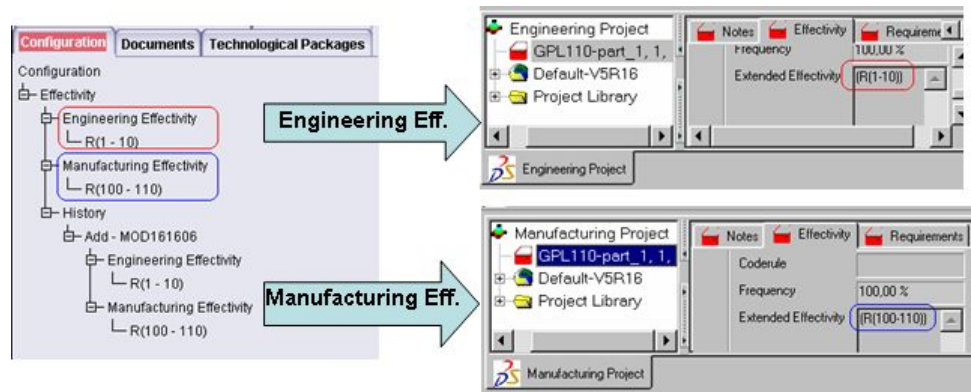


Figure 117: Configuration File

In the example above the export files contain the data of two effectivity domains, the Engineering Effectivity and the Manufacturing Effectivity. Each of those is used for an import into its own Manufacturing Hub project and a representative result may be seen in the pictures above.

For this example the effectivity domain part of the Export Configuration File looks as follows:

```
<effectivitydomains default="Engineering Effectivity">
  <token>Engineering Effectivity</token>
  <token>Manufacturing Effectivity</token>
</effectivitydomains>
```

The VALIDE domain is not treated specifically but can be specified just like any other effectivity domain, too.

Import

It is not required to customize the effectivity domain which shall be used for the import part of the DELMIA – ENOVIA VPM V5 Connection. If not customized the default domain marked in the export data will be used.

If a different effectivity domain shall be used for the import it may be customized in the Windows registry of the machine running the import tool PPRLoader.

A MH project is not aware from which domain the effectivity of its product components is originating: if you change the registry entry which defines the effectivity domain to be imported, all components will get an updated effectivity depending on the changed configuration data.

If the effectivity domain registry setting is defined such that the data which shall be imported does not contain the defined domain the import will abort with an appropriate error message.

Customization

Export

On the EH-Server the user has to define an environment variable ENOVIA_LCAIPD_CONFIGURATION_FILE pointing to the Export Configuration File. The Configuration file contains a list of effectivity domains which will be written into the EH-MH Connection export data.

For example, the list may consist of effectivity domain Engineering Effectivity and Manufacturing Effectivity, e.g.

```
<effectivitydomains default="Engineering Effectivity">
  <token>Engineering Effectivity</token>
  <token>Manufacturing Effectivity</token>
</effectivitydomains>
```

Now, if the exporter comes across an assembly relation the following happens:

The assembly relation object will be queried for all customized domains and for each of those (if also customized) the WIP and the Approved effectivity will be extracted and written into the export data. The effectivities for multiple domains are extracted independently from each other. So a single mod statement may contain effectivity data for multiple domains. Still only the node effectivity is transferred, no single mod statements.

If no Export Configuration File is used or no effectivity domains are defined in the file the "Engineering Effectivity" is transferred.

If non existing effectivity domains are defined in the "effectivitydomains" section the export will abort with an error message.

The "default" attribute of the "effectivitydomains" element is mandatory. It has to contain an existing eff. domain, otherwise the export will abort with an error message.

Import

On the import side a new customization setting is introduced in the Windows registry database:

```
[HKLM/Delmia/PPRLoader/VPM]
"effectivitydomain"="Engineering Effectivity"
```

This new setting is used to define the effectivity domain which will be used by the import tool PPRLoader. If this setting is not defined the default domain defined on the export side will be used for the import.

If effectivity is defined in the registry which is not existent in the transfer data used for the import the PPRLoader tool will abort its operation with an error message.

The eff. domain which is imported will be logged in the PPRLoader.log file.

Example

Test the transfer of two effectivity domains via the DELMIA – ENOVIA VPM V5 Connection and the import of the default effectivity domain into an empty Manufacturing Hub project.

Create a new project in DPE with name and number set to "R17GPL110" using the Standard V5 plan type set and using extended coderule filtering mode.

Set username and password in the Windows registry for automatic PPRLoader login, e.g.

```
[HKEY_CURRENT_USER\Software\Delmia\ergoplan\pprloader]
"username"="admin"
"password"="admin"
```

Define the environment for the exporter respectively on the server as follows (you can also use the attached configuration file):

ENOVIA_LCAIPD_CONFIGURATION_FILE pointing to the Export Configuration File

In the Configuration file define the section effectivitydomains, the complete file looks like this (a suitable configuration file is included in the "Related Materials" chapter):

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<configuration release="17" servicepack="0" hotfix="0">
  <effectivitydomains default="Engineering Effectivity">
    <token>Engineering Effectivity</token>
    <token>Manufacturing Effectivity</token>
  </effectivitydomains>
</configuration>
```

In ENOVIA create a Product Class "DPQR" and below this one another ProductClass "R17GPL110".

Below this PC create a PRC "R17GPL110".

Below the PRC create two Part Instances "GPL110-part_1.1" and "GPL110-part_2.1". Configure "GPL110-part_2.1".

Below "GPL110-part_2.1" create two more instances "GPL110-part_2_1.1" and "GPL110-part_2_2.1" to test feature for second level parts.

Assign effectivities as given in the following table:

Table 2: Assign Effectivities

| Part Instance | Engineering Effectivity | Manufacturing Effectivity |
|-------------------|-------------------------|---------------------------|
| GPL110-part_1.1 | R(1-10) | R(100-110) |
| GPL110-part_2.1 | R(2-20) | R(200-220) |
| GPL110-part_2_1.1 | R(2100-2199) | R(2200-2299) |
| GPL110-part_2_2.1 | null | null |

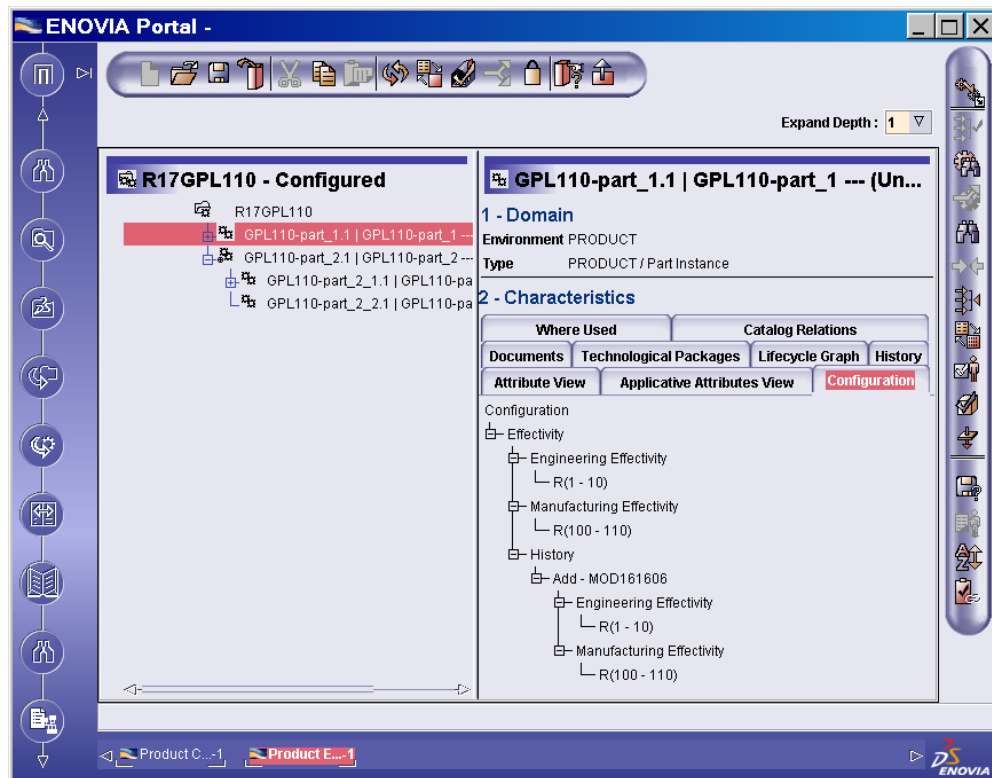


Figure 118: ENOVIA Portal

Transfer the PRC from the EH to the MH using the batch mode: On the ENOVIA V5 side, enter the following command line in a command window:

```
ProductDataGen product R17GPL110
```

- On the MH import side make sure that the registry key

[HKLM\Delmia\PPRLoader\VPM] "effectivitydomain"

is not set. This is to test the import of the default domain defined on the export side.

Run the PPRLoader application in a suitable command shell:

```
pprloader -vpm R17GPL110xxx.xml -project R17GPL110
```

Verify that the imported product structure has the correct structure.

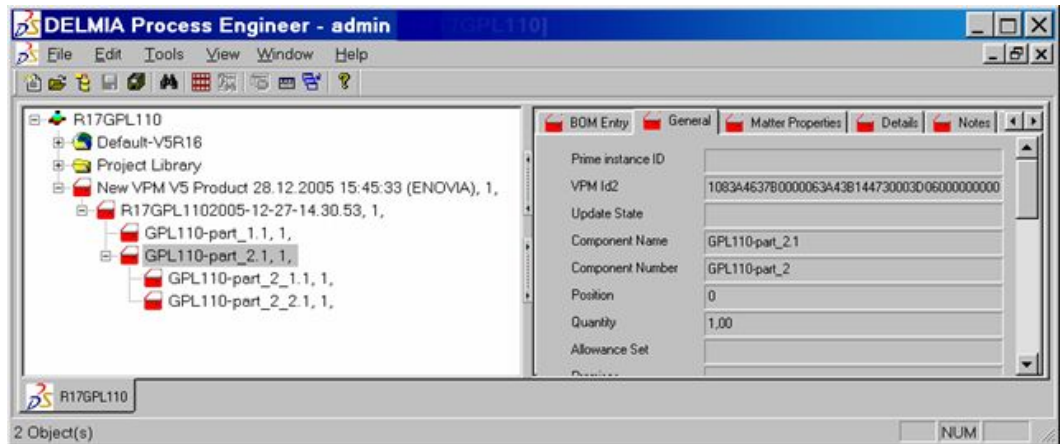


Figure 119: Imported Product Structure

Verify the Engineering effectivities created in ENOVIA VPM V5 in the corresponding Manufacturing Hub parts

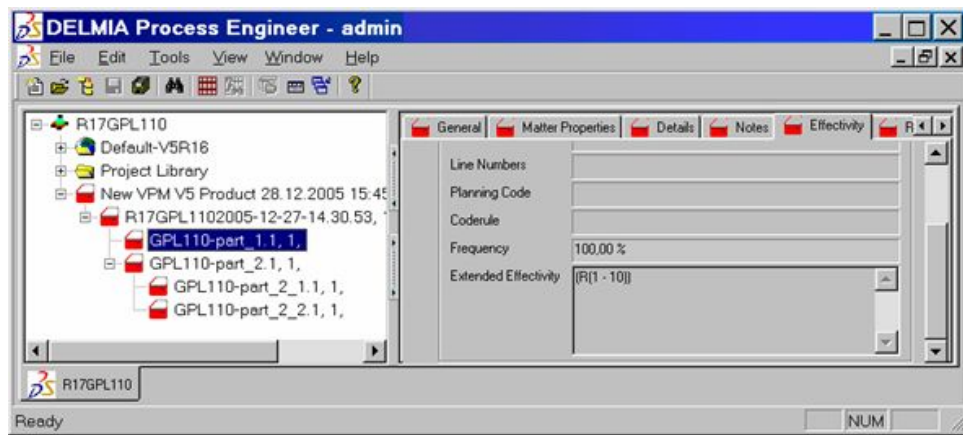


Figure 120: Engineering Effectivities

and for the other parts according to the table listed above.

Test the transfer of two effectivity domains via the DELMIA – ENOVIA VPM V5 Connection and the import of a customized effectivity domain into a Manufacturing Hub project with preexisting data.

Example 2

Example #1 must have run successfully before this one.

Reuse the export data created by Example #1

On the MH import side set the registry key

[HKLM/Delmia/PPRLoader/VPM]

"effectivitydomain"="Manufacturing Effectivity"

This is to test the import of the "Manufacturing Effectivity" effectivity data defined in ENOVIA VPM V5.

- Run the PPRLoader application in a suitable command shell:

```
pprloader -vpm R17GPL110xxx.xml -project R17GPL110
```

Verification

Verify that the imported product structure still has the correct (unchanged) structure.

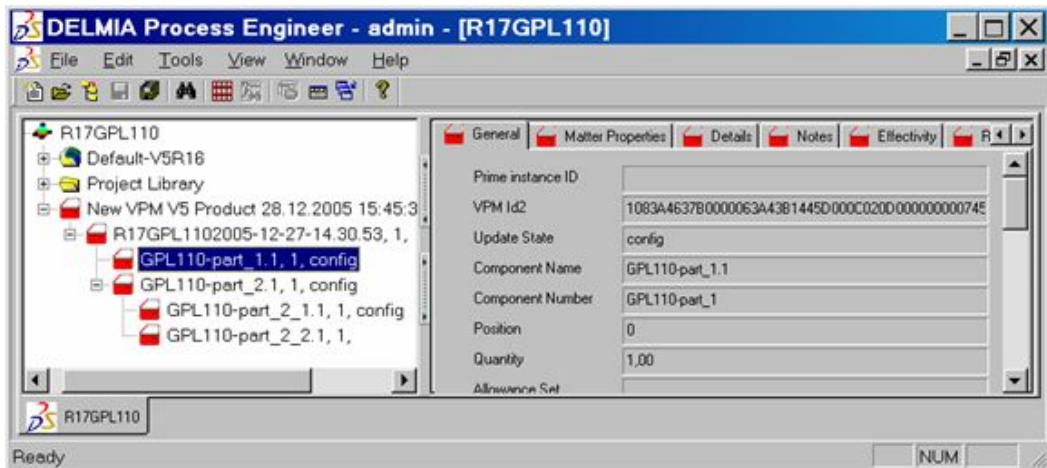


Figure 121: Imported Product Structure Unchanged Structure

Verify that all parts except "GPL110-part_2.2.1" have an update state "config".

Verify the Manufacturing effectivities created in ENOVIA VPM V5 in the corresponding Manufacturing Hub parts:

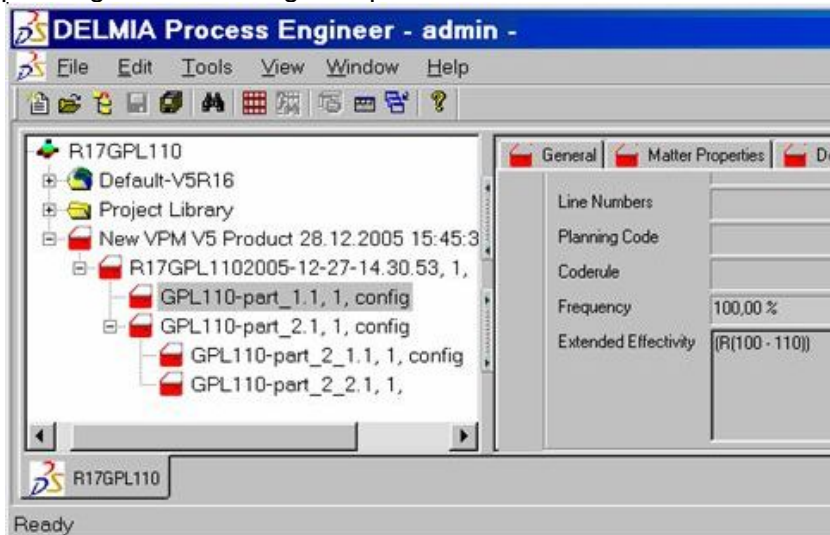


Figure 122: Manufacturing Effectivities

and for the other parts according to the table listed in Example 1.

6.33 Enhanced Extended Filtering on Export

Currently it is possible to filter Parts while exporting them from the Engineering Hub (EH). This is done via either a blacklist or a white list of values a certain attribute must not have or is allowed to have. Both the name of the attribute and the values are configurable. If an attribute can have a lot of values the configuration is tedious and error-prone. Also, so-called Multi Value Attributes (MVA) were not supported so far.

Therefore, *Enhanced Extended Filtering on Export* has improved in two ways:

- Multi Value Attributes are supported if one value matches the filter fires.
- A wildcard character (asterisk, *) can be defined to replace an arbitrary substring in the configured values. An escape character (backslash, \) is defined to mask this behavior of the special wildcard character.

Customization

From now on it is possible to use a wildcard character (default asterisk, “*”) in the tokens denoting the attribute values which discriminate on the filter behavior of a Part. The wildcard replaces an arbitrary number of characters including zero. In the rare cases in which an asterisk has to be read verbatim, it has to be prefixed by an escape character (default backslash, “\”). Both wildcard and escape character can be configured via the global property “pattern”. C.f. the following example.

It is now also possible to use a Multi Value Attribute as filter attribute. This has not been supported in prior releases.

An example configuration:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<configuration>
  <global>
    <properties>
      <enhanced-filter enabled="yes"/>
      <pattern wildchar="*" escapechar="\\"/>
    </properties>
  </global>
  <filters>
    <extendedfilter context="PartInstance" attribute=
"V_discipline">
      <whitelist>
        <token>MO*</token>
      </whitelist>
    </extendedfilter>
  </filters>
</configuration>
```

The Part instance attribute “V_discipline” can have a set of values. If at least one of those matches the pattern “MO*” the Part referenced by this part instance is exported.

Example

In the Manufacturing Hub create a new empty project R17GPL530

In the Engineering Hub

- Create a PRC R17GPL530
- Insert three Parts “Part01”, “Part02”, and “Part03” on the first level
- Give each Part the name according to its ID.
- Add the Discipline “BASELINE” to each part
- Add the Discipline “MOCKUP” to “Part01”
- Lock “Part03” and repeat step 2. Use “Part1x” as names.
- Change the status of the Parts “Part02”, “Part03”, “Part12”, and “Part13”; promote their respective references.
- The resulting structure should look like.

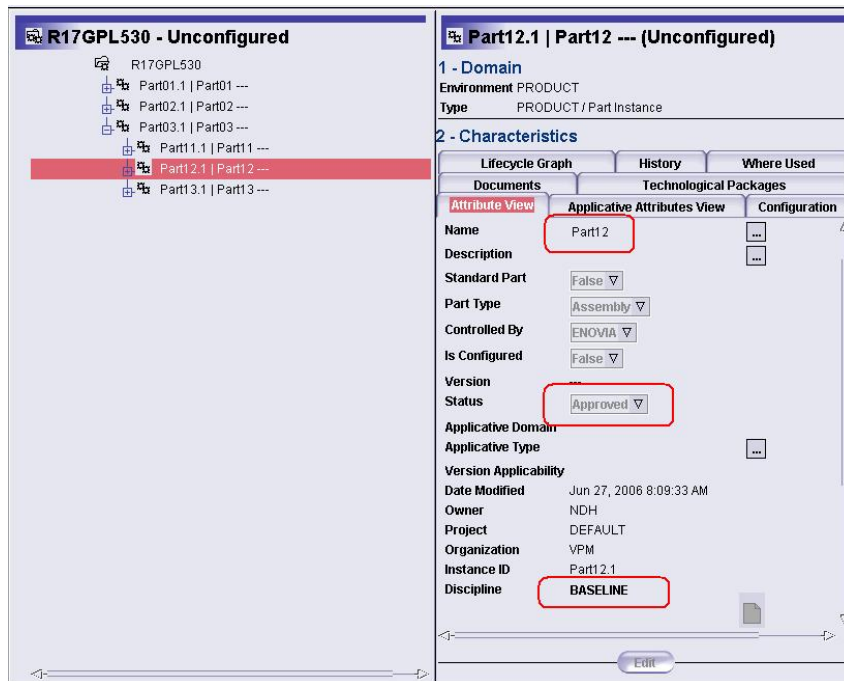


Figure 123: Resulting Structure

- Copy the configuration which is attached in Part 5 to the hard disk
- Reference the configuration through
ENOVIA_LCAIPD_CONFIGURATION_FILE
- Run ProductDataGen product R17GPL530
- Import the generated XML files into the MH

- The resulting Product structure must only contain the two Parts “Part03” and “Part13”. All other Parts must not be transferred as they don’t meet all filter criteria.

6.33.1 The Configuration file

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
- <configuration>
  - <global>
    - <properties>
      <log level="notice" />
    </properties>
  </global>
- <filters>
  - <extendedfilter context="PartInstance"
    attribute="V_discipline">
    - <blacklist>
      <token>MO*</token>
    </blacklist>
    </extendedfilter>
  - <extendedfilter context="PartVersion"
    attribute="V_status">
    - <whitelist>
      <token>Approved</token>
      <token>Completed</token>
    </whitelist>
    </extendedfilter>
  - <extendedfilter context="PartMaster"
    attribute="V_name">
    - <whitelist>
      <token>P*3</token>
      <token>ABC</token>
    </whitelist>
    </extendedfilter>
  </filters>
</configuration>
```

6.34 Multiple Parts Filtering on Import

The Multiple Parts Filtering on Import uses a new filter configuration file, which contains the filter criteria (based upon attribute values) and probably the project

references, in case different projects use different filter settings. The name and location of this file is registered in the registry, similar to the existing attribute mapping configuration file.

During the update/import run of the DELMIA – ENOVIA VPM V5 Connection the XML export files to be imported is read as regular. Still, during the parsing of the XML export files, each component read is checked against the corresponding list of filters; components filtered out are marked.

In the second step, the traversal (resp. the initial load) of the Product structure such objects filtered out are treated as non-existent. The blockage of one Part by a filter causes its entire sub-structure to be suppressed, even though portions of it might have passed the filter.

Filters are applied to Part references, Part instances and Documents. If a Part reference is blocked, all its instances are suppressed, too.

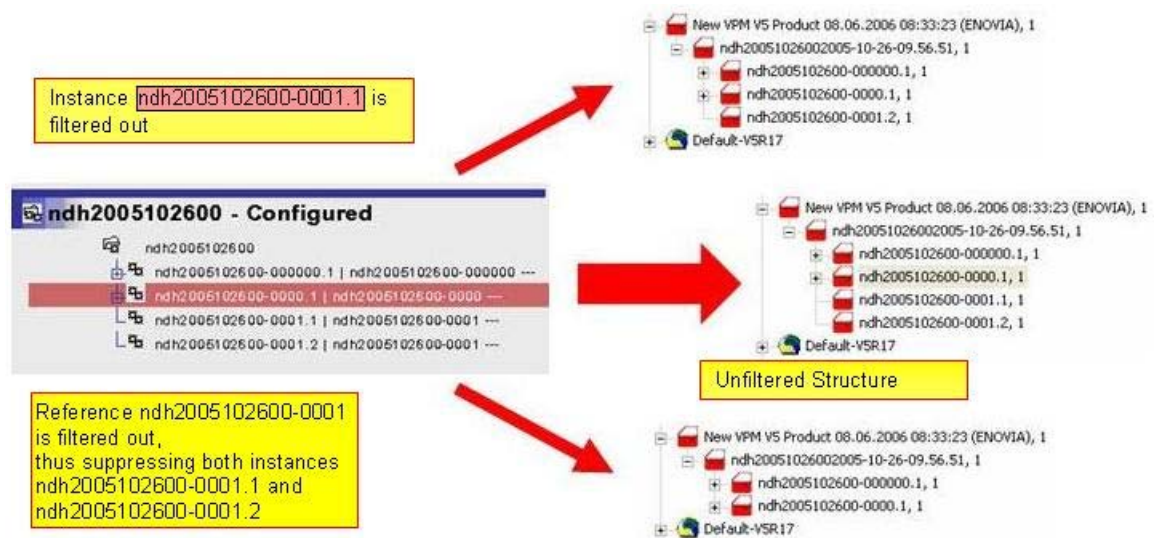


Figure 124: Parts Filtering

The capability of import side filtering allows having different product structures (in different projects) for different purposes (such as design, process planning or maintenance and services) with one XML export on the ENOVIA V5 Engineering Hub side. This reduces the number of exports to be performed on customer side, since one XML export file can be re-used to support several projects (or partners) at once.

6.34.1 The Filter Config File

Filters are defined in a filter configuration file through filter criteria based upon attribute values, either on Part references (reference XML file), Part instances (instance XML file) or Documents (reference XML file).

Filter criteria can either be defined as pass filters (white list) or blocking filters (black list) on attribute values. It is possible to use wildcards (in a limited way). It is also possible to combine several filters.

Additional conditions for filters can be applied (e.g. instance and reference filters, filters only for specific environments and types). These conditions can be defined

on a filter to check only parts from specific environments or specific types (ENOVIA VPM V5 fine grained mapping).

6.34.1.1 Registry Settings

The filter config file is registered in the registry, similar to the current attribute mapping config file. Still, in difference to the attribute mapping config file, the new filter config file is XML formatted and is capable of containing different filters for different projects.

```
[HKEY_CURRENT_USER\Software\Delmia\ergoplan\PPRLoader\VPM] "filterfile"="..."
```



Figure 125: Filter File

During the parsing of the ENOVIA VPM V5 XML export files (reference and instance) each component (incl. documents) is checked against the list of filters.

In case of components, the entire substructure beneath a component blocked-out is suppressed, even though, child components might have passed all filters.

A component which matches a block filter (blacklist) criteria is considered as “out”. In case of a pass filter, the component is filtered out, if it does NOT matching the filter criteria.

If a Part reference or Document is filtered out (from the XML reference file), all corresponding instances/MH components are affected. If a Part instance (from the XML instance file) is filtered out, only this particular component gets blocked. A Part instance is blocked, if the corresponding Part reference has been filtered out, even though the Part instance data itself indicated the opposite.

It's not possible to reset the status of an already filtered “out” component back to “in”, i.e. all filters are additive (AND concatenated).

Components filtered out are logged in the PPRLoader.log file, if tracing is enabled.

The filter config file XML schema describes filters and projects, which use those filters during update/import. For a complete description on the XML format, *please refer to the [Example of a Filter Configuration File](#).*

6.34.1.2 Filter Config Template

A filter config template file “import-config.xml” is provided with the regular DPE installation (in ... \PPRClient\data\PPRLoader).

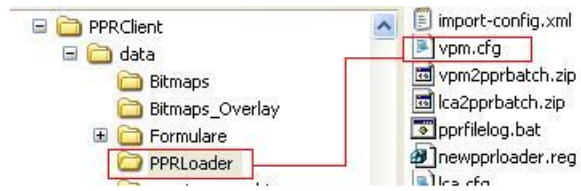


Figure 126: Config Template

6.34.1.3 Example of a Filter Configuration File

```

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
- <configuration>
- <global>
- <properties>
  <pattern multiwildchar="*" singlewildchar="?" escapechar="\" />
</properties>
</global>
- <filters>
  <!-- pass filter for part instances, only status "Approved" and "Released" pass through -->
  - <extendedfilter id="filterA" context="instance" environment="" table="" type="" attribute="V_status" ignorecase="no"
    filtermismatch="out">
    - <whitelist>
      <token>Approved</token>
      <token>Released</token>
    </whitelist>
  </extendedfilter>
  <!-- block filter for part instances, admin part instances are filtered out -->
  - <extendedfilter id="filterB" context="instance" environment="" table="" type="" attribute="V_organization" ignorecase="yes">
    - <blacklist>
      <token>admin</token>
    </blacklist>
  </extendedfilter>
  <!-- pass filter for part references, customizable type "engineering" is allowed -->
  - <extendedfilter id="filterC" context="reference" environment="" table="PART_LIST" type="" attribute="BOEACType"
    ignorecase="no" filtermismatch="out">
    - <whitelist>
      <token>Engineering</token>
      <token>Design</token>
    </whitelist>
  </extendedfilter>
  <!-- block filter for part references, "design" projects are filtered out -->
  - <extendedfilter id="filterD" context="reference" environment="" table="PART_LIST" type="Engineering" attribute="V_project"
    ignorecase="yes">
    - <blacklist>
      <token>design</token>
    </blacklist>
  </extendedfilter>
  <!-- combined block and pass filter, all states ending with "ed" are allowed (e.g. "Approved",
    "Designed", "Planned"), except "Released" -->
  - <extendedfilter id="filterE" context="reference" environment="" table="" type="" attribute="V_status" ignorecase="no">
    - <blacklist>
      <token>Released</token>
    </blacklist>
  </extendedfilter>
</filters>
- <projects>
  - <project id="GPL520A">
    - <filterlist>
      <filter>filterA</filter>
      <filter>filterC</filter>
    </filterlist>
  </project>
  - <project id="GPL520B">
    - <filterlist>
      <filter>filterB</filter>
      <filter>filterD</filter>
    </filterlist>
  </project>
  - <project id="GPL520C">
    - <filterlist>
      <filter>filterE</filter>
    </filterlist>
  </project>
</projects>
</configuration>

```

At the beginning the global properties define wildcard characters and the escape sequence, which can be used for the filter criteria.


```

- <global>
- <properties>
  <pattern wildchar="*" escapechar="\\" />
</properties>
</global>

```

In the next section filters are defined as a set of extended filters. Each extended filter definition contains several attributes. The “id” attribute is the filter’s name, which is used later on to apply several filters to different projects.

```

- <extendedfilter id="filterA" context="instance" environment="" table="" type="" attribute="V_status" ignorecase="no"
  filtermismatch="out">
- <whitelist>

```

The “context” attribute describes the objects source, either “reference” or “instance”.

```

- <extendedfilter id="filterA" context="instance" environment="" table="" type="" attribute="V_status" ignorecase="no"
  filtermismatch="out">
- <whitelist>

```

If context is set to “reference” this filter is only applied to the reference file, if the context is “instance” it’s only valid for the instance file. The context “internal” specifies that the filter is to be applied on MH attributes. If context is left empty or not specified (or contains any other invalid character string), the filter is applied to both XML export files (reference and instance).

The next three attributes “environment”, “table” and “type” can be used to limit each filter to a specific set of objects. They directly refer to the corresponding entries in the ENOVIA VPM V5 XML reference export file.

```

- <extendedfilter id="filterC" context="reference" environment="" table="PART_LIST" type="" attribute="BOECACType"
  ignorecase="no" filtermismatch="out">
- <whitelist>
  <token>Engineering</token>
  <token>Design</token>
</whitelist>
</extendedfilter>

```

The following image shows a part from environment “BOECACPrdt”, with table “PART_LIST” (in difference to “DOCUMENT” e.g.) and type “Engineering”.

```

      Uuid= 554fc36c0020a0c041b80ec5000aeba7 >
- <Instance Name="(BOECACPrdt.PART_LIST)Blue Spoiler---2004-12-9-9.37.30"
  Alias="(BOECACPrdt.PART_LIST)Blue Spoiler---2004-12-9-9.37.30" Type="Engineering"
  Uuid="554fc36c0020a0c041b80ec5000aeba7">
+ <Attribute Name="$COID" Type="String">
+ <Attribute Name="$COMPID" Type="String">
+ <Attribute Name="S_PART_NUMBER" Type="String">
+ <Attribute Name="S_NAME" Type="String">

```

These three attributes are optional. Since there’s no environment, table and type information within the instance XML file, these attributes are only useful on “reference” context.

The attribute “attribute” itself defines which (ENOVIA) attribute is to be checked against the filter criteria.

```

- <extendedfilter id="filterE" context="reference" environment="" table="" type="" attribute="V_status"
  ignorecase="no" filtermissing="no">
- <blacklist>

```

If context is set to “internal”, this value reflects an MH attribute name (e.g. “attribute_3”), not the name of the attribute from the ENOVIA VPM V5 XML export file.

The attributes “ignorecase” and “filtermismatch” contain information about case sensitivity (“yes”: no case sensitivity, “no”: lower/upper case is checked) and the behaviour, if the filter attribute is not existent at all on a component (“block” or “out”: component is filtered out, if filter criteria attribute is missing, “pass” or “in”: component may pass, if specified attribute is not found).

```
- <extendedfilter id="filterA" context="instance" environment="" table="" type="" attribute="V_status" ignorecase="no"
  filtermismatch="out">
- <whitelist>
  <token>Approved</token>
  <token>Released</token>
```

The next list in the filter config file is the list of tokens, separated into “blacklist” (tokens defining block criteria) and “whitelist” (tokens defining pass criteria).

```
ignorecase= no filtermismatch= no
- <blacklist>
  <token>Released</token>
</blacklist>
```

In the above example, the checked component can only pass, if the specified attribute has a value ending with “ed”, except “Released”, which defined the component as “out”. So, values as “WIP” and “Integrate” would cause the component to be blocked (string do not end on “ed”), allowed values would be “Approved”, “Planned”, .etc.).

“blacklist” and “whitelist” should only occur once for an extended filter, but can contain several token entries:

```
- <whitelist>
  <token>Approved</token>
  <token>Released</token>
</whitelist>
```

The conversion to MH attribute types from the XML string values is not affected by the filt4ering, i.e. the filtering will always check and compare string values. With that, it’s e.g. not possible to define range filters on double values.

Errors in the definition of the extended filters like an empty white list will be ignored that is the filter will not be applied. If it is impossible to parse a filter definition (i.e. unknown context, bad XML token, etc.) the importing PPRLoader application will abort the execution.

The “projects” section of the filter config XML file defines the filters, the specific projects use for update/import. Each “project” contains again an “id” attribute, which corresponds to the projects “nameshort” (“Number”) attribute in MH.

```
- <projects>
- <project id="GPL520A">
  - <filterlist>
    <filter>filterA</filter>
    <filter>filterC</filter>
  </filterlist>
</project>
```

The filter list contains the ids of the extended filters, which should be applied to the given project. If the list contains ids, which do not correspond to one of the extended filters, the entry is ignored.

```

- <projects>
- <project id="GPL520A">
- <filterlist>
  <filter>filterA</filter>
  <filter>filterC</filter>
</filterlist>
</project>

```

Filters can be set inactive, either in the filter definition directly by setting the attribute “active” to “false”

```

- <extendedfilter id="filterX" active="false" context="instance" attribute="V_status" ignorecase="yes">
- <blacklist>
  <token>out-block</token>
</blacklist>
</extendedfilter>

```

or in the projects filter list, also using an attribute “active”, set to false for the filter to deactivate.

```

- <project id="GPL520">
- <filterlist>
  <filter active="false">refA</filter>
  <filter>instD</filter>
  <filter>filterX</filter>
</filterlist>
</project>

```



Note

If no filter config file is specified or if it is empty or contains no filter criteria, all components are updated/imported. No filter check is performed, this guarantying full compatibility to the previous behavior.

Limitations

- The maintenance of the XML based new filter config file is up to the user/customer. Only a template is provided with the regular DPE installation.
- A Part or Document that has already been transferred to the MH via the EH-MH Connection, but is blocked by one of the filter criteria's in a later on update/import run (since either the filter itself has changed or a modified attribute value makes the filter suppress this Part/Document now) is treated as a regular deleted object. The concrete action upon this delete depends on the given delete mode.
- Filters can only be defined on Parts (Part references and instances) and Documents. It is not possible to define filters for links/relations or any other objects, content of the data transfer such as Calculation Models.
- Filters are additive, i.e. all filters are concatenated by a logical AND. If one filter blocks a particular Part, the match result for all other filter criteria is irrelevant. This is important in case one filter blocks out a Part reference, while a pass filter explicitly allows one of its instances to be transferred. Still, as a result, this instance is skipped during update/import, since it needs to pass ALL filters.

This also means that the order in which the filter criteria are defined in the filter config file is of no interest for the filter result at all.

- The conversion to MH attribute types from the XML string values is not affected by the filtering. Filtering is always based on string value comparison, e.g. it's not possible to define range filters on double values (e.g. component is filtered when double attribute is within range -2.8 to +4.2).
- Please note, that MVAs (multi-value attributes) are transferred as one single string. To filter for a particular entry (sub-string) within an MVA, the usage of wildcards is required.
- Filters work only on mapped attributes (through the config file, plus some hard-coded mapped ones). Therefore, filter attributes should appear in the attribute mapping config file as well as in the filter config file.

6.35 Filter on Effectivities

This new feature of R18 will allow the user to filter Parts during the import of a product structure into a project, if their computed effectivity is either NULL or completely unresolved (empty) in a certain **effectivity domain**. The effectivity domain to be used for the filtering as well as the filter criterion to be used (NULL, empty or both) can be specified during the import run in the import configuration file (see the Customization section). This is handled by the standard import filter functionality.

The computed effectivity of each Part is fetched during the export and analyzed whether it's empty or NULL. The gained information is placed into the **reference data file**. This is done for each configured effectivity domain. Since this operation takes some time it is off by default and has to be activated in the configuration (see the Customization section).

Assuming the export has written the effectivity information, as a result the importer knows whether a Part's effectivity in a specified domain is empty or NULL.

It's possible to specify an effectivity domain for which during the import of a product structure Parts that have either a NULL effectivity, no effectivity set at all, or both will get filtered out. As a result these Parts are not present in the Manufacturing Hub. As usual, filtering a Part out will also remove the complete sub-structure associated to this Part as well as all Documents attached to it.

The user can export a product structure from the Engineering Hub specifying multiple effectivity domains, c.f. the Customization section. Then it's possible to use the same product structure files to import a product structure into several projects in the Manufacturing Hub. The projects can have a different effectivity domain (one of the set specified during the export). If the effectivity filter is activated both during the export and the import, the resulting product structures in the Manufacturing Hub can look different.

The filter capabilities can be activated during the import of a product structure into the Manufacturing Hub. It is possible to get different results in the Manufacturing Hub from one set of export data based on the chosen effectivity domain. It's necessary to run a specifically configured export beforehand, though.

Example

In the Engineering Hub create a PRC with two Parts:

- The first Part only has an Engineering Effectivity 'R(1-oo)'
- The second Part only has a Manufacturing Effectivity 'R(1-oo)'
- Export this PRC, specifying both effectivity domains 'Engineering Effectivity' and 'Manufacturing Effectivity' and activate the **filter-on-effectivity** feature.
- Create two projects in the Manufacturing Hub
- Import the product structure with the filter-on-effectivity feature activated
 - Into the first project using the switch '+domain "Engineering Effectivity"'
 - Into the second project using the switch '+domain "Manufacturing Effectivity"'
- The structure in the first project only shows the first Part, the structure in the second project only shows the second Part.

Export

The export part of the DELMIA – ENOVIA VPM V5 Connection has to be configured in order to receive necessary additional information in the generated reference file. It is mandatory to activate the filter-on-effectivity property. Additionally, especially if more than one effectivity domain is to be taken into account, each domain has to be specified in the "effectivitydomains"-section. If no domains are given, the domain taken from the environment setting, a defined user exit, or the default domain "Engineering Effectivity" is used.

```
<configuration>
  <global>
    <properties>
      <filter-on-effectivity enabled="yes"/>
    </properties>

    <effectivitydomains>
      <domain name="Engineering Effectivity"/>
      <domain name="Manufacturing Effectivity" default="yes"/>
    </effectivitydomains>
  </global>
</configuration>
```

Import

The configuration of the import is done by specifying certain filter criteria for the import side. Computed effectivities are read from the XML export file and stored into the transient attribute "_computedeffectivity_domain_\${DomainName}", depending on the domain this effectivity comes from.

E.g. the computed effectivity from domain "Manufacturing Effectivity" will be stored in "_computedeffectivity_domain_manufacturing effectivity" on links.

Transient attributes are not stored into Manufacturing Hub, but can be used for filtering purposes. The following example shows a section from an import config.xml file, used to block null respectively unresolved effectivities for the domains "Engineering Effectivity" and "Manufacturing Effectivity":

```

</global>
- <filters>
  <!-- block filter for node relations, computed NULL effectivities (for Engineering
  domain) from ENOVIA are blocked -->
  - <extendedfilter id="filterA" context="internal" environment="PRODUCT" table="_CLink_(PART_LIST)"
  type="attribute" attribute="_computedeffectivity_domain_engineering effectivity" ignorecase="yes"
  filtermissing="no">
    - <blacklist>
      <token>NULL</token>
      <token>UNRESOLVED</token>
    </blacklist>
  </extendedfilter>
  <!-- block filter for node relations, only valid computed effectivities (for
  Manufacturing domain) from ENOVIA can pass -->
  - <extendedfilter id="filterB" context="internal" environment="PRODUCT" table="_CLink_(PART_LIST)"
  type="attribute" attribute="_computedeffectivity_domain_manufacturing effectivity" ignorecase="yes"
  filtermissing="no">
    - <blacklist>
      <token>NULL</token>
    </blacklist>
  </extendedfilter>
</filters>
- <projects>
  - <nniact id="GWANN3A">

```

Both filters are defined as “internal” (since they do not apply to attributes from the XML export file directly). Environment “PRODUCT” and table “_CLink_(PART_LIST)” relate these filters to links (node relations). As described above, the filter attribute names refer to the transient computed effectivity attributes for each domain, while the tokens are defined in the regular manner.

The configuration file has to be specified in the registry, similar to the attribute-mapping configuration-file:

```
[HKEY_CURRENT_USER\Software\Delmia\ergoplan\PPRLoader\VPM]"filterfile"=""
```

Of course, the calculation of the computed effectivity during the export takes time. Therefore, activating this feature will slow down the export part of the DELMIA – ENOVIA VPM V5 Connection. Currently, we are calculating the computed effectivity on each linkable object and check, whether its value is empty, null, or different from these. The calculation of the computed effectivity is expensive in terms of time. We expect an increase in time of the sub process around 15 to 20 percent.

This increase in time can be reduced to around 5 percent if the existing ENOVIA API method `CATICfgManager::IsComputedEffectivityNull` can be enhanced in two ways:

0: Allow the caller to specify the effectivity domain which should be checked

1: Instead of returning a `CATBoolean` in case of an unresolved or null effectivity, return an enumeration type which lets the caller differentiate on the effectivity’s actual value.

Unfortunately, no customization options exist to activate this change. It requires coding efforts.

Limitation

If the export is not configured to insert the effectivity information into the product structure files, the import filter does not work. This can lead to completely empty product structures in the Manufacturing Hub.

6.36 Spares Configuration Data

A Spare Part is an alternate Part which fulfils a certain (user-defined) Boolean requirement. Alternate Parts are linked to a primary Part. The relation exists between Part masters (PM).

During the export, the product structure is traversed. It is checked whether a referenced PM has alternatives. If this is given, in the alternate PM is checked whether a configured Boolean attribute is set to a true value, see Customization and Limitation below. If a valid spare is encountered, a spares report attached to the prime part as supplementary document is searched and exported.

In order to identify the report a name schema must be defined. The ID of the report has to be specified in the configuration file, c.f. the Customization section below. The report must have a mime type of either xml or text/xml. If the report does not exist at time of export nothing will happen, (*Please refer to the [Limitation](#)*). During import the report is treated the same way as a Work Package report (black-box report), since it will be used by the same application (BOM export).

Information about Spare Parts is transferred over the DELMIA – ENOVIA VPM V5 Connection as additional XML files. They are stored as attachment at the respective Prime Part. This allows subsequent applications to generated reports about Spare Parts.

Customization

The feature must be activated through customization. To this end the new property “spare-transfer” has been added to the configuration file.

A sample configuration looks like

```
<configuration>
  <global>
    <properties>
      <pattern wildchar="*" escapechar="\\"/>
      <spare-transfer enabled="yes" attribute="V508_isStandardPart"
        report="SpareReport*" />
    </properties>
  </global>
</configuration>
```

The value of the key “attribute” specifies the attribute which is checked on the alternate Parts in order to decide whether they are a spare Part or not.

The “report” attribute specifies the name pattern of the report ID to look for. Wildcards are allowed. They can be configured using the “wildcard” configuration property. The default is an asterisk (*). The identifier is the name of the Document master.

The DTD has been updated accordingly.

Limitations

- The XML report describing the Spare Parts must exist prior to the export. No report is generated on the fly during the export.
- The report must be a supplementary document, it must have the mime type xml or text/xml, and the identifier of the Document master must match the configured pattern. If any of these criteria is not met the report will not be found.
- The attribute denoting whether an alternate Part is also a spare Part must be Boolean. Other attribute types do not work.
- The spare report will be treated the same way as a Work Package file. Therefore, during import the same transfer settings apply.

Example

In ENOVIA VPM V5 create a new product R17GPL510 and insert a new instance R17GPL510-PrimePart below it.

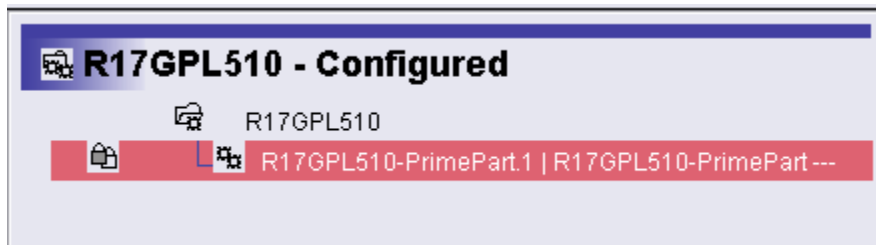


Figure 127: New Instance

- 1) In the Part Editor create three parts R17GPL510-Alternative1, 2, and 3. Mark two of them as “Standard Part”, the other one not. “Standard Part” will act as spare discriminator.
- 2) Open the part R17GPL510-PrimePart in the part editor and add the three alternatives as Alternate Parts.

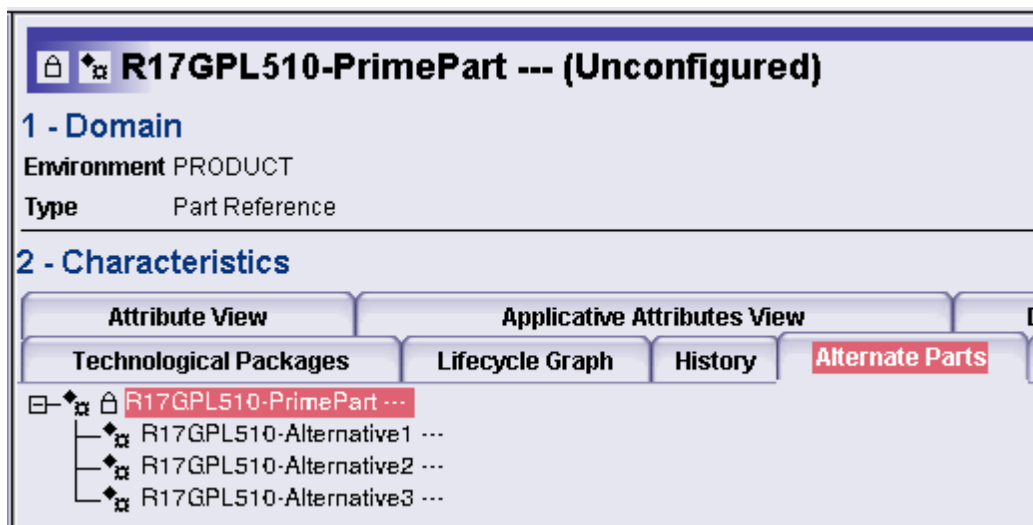


Figure 128: Alternate Parts

- 3) Add a supplementary Document to R17GPL510-PrimePart. Give it the ID SpareReport-1. Make sure the mime type is xml or text/xml. Use any existing XML document to be stored in the Vault.

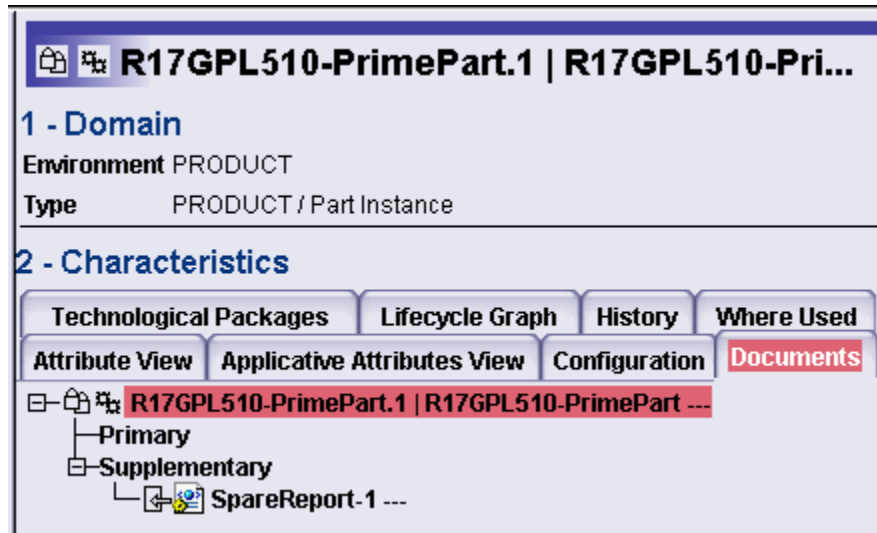


Figure 129: Add a Supplementary Document

Export the product structure.

- a) Copy the configuration to /path/to/config.xml
 - b) `export ENOVIA_LCAIPD_CONFIGURATION_FILE=/path/to/config.xml`
(use `set` on Windows)
 - c) `ProductDataGen product R17GPL510`
 - d) If necessary: copy files into a directory wherefrom the importer can access them.
- 4) Generate a new empty Project R17GPL510 in DPE. Import the exported structure into this project.

The structure must be the same in DPE as in ENOVIA VPM V5

The Spares Report file must get exported and imported.

6.37 Version Management Support

The Engineering Hub to Manufacturing Hub Connection identifies and handles the versioning of the parent assembly in Engineering Hub, thus avoids the replanning on unchanged child parts.

In a PRC which has one child Item Instance (II1) which in turn has three children Item Instances (II11, II12, and II13). II1 points to a Part Version PV1 which is connected to a Part Master PM1. II11, II12, and II13 has similar associations. In the reference model exists a connection, the Assembly Relation AR1 between the two parts from Part Version PV1 to the Part Master PM11. Similarly, assembly relations exist among all the children. **Figure 130** is the product structure that is transferred over the Bridge into the Manufacturing Hub.

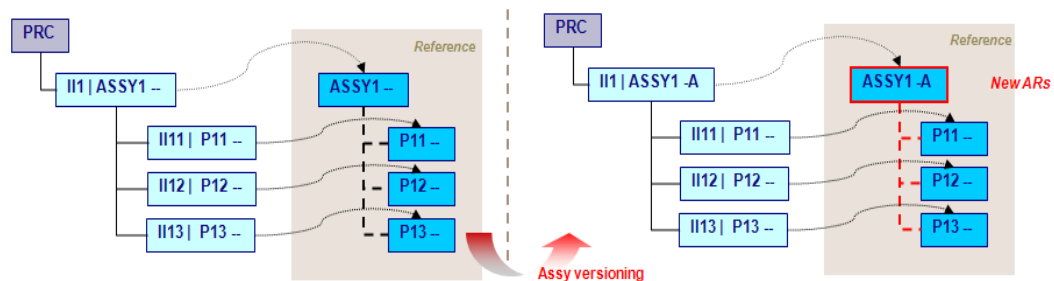


Figure 130: Product Structure transferred over the Bridge into the Manufacturing Hub

If in the Engineering Hub Part Version PV1 is versioned to PV1' a new Assembly Relation AR1' to the Part Master PM11 of the respective child II11 is created. Similar behaviour happens for all the children.

When this new product structure is transferred over the Bridge (Run ProductDataGen and generate the export XML files before and after versioning the product in Engineering Hub. Configure the export so that the origin of part versions and assembly relations are transferred to the generated XML files), then the Item Instance II11 and the Assembly relation AR1 are identified with the help of the 'historical id' stored on the object and the link in the Manufacturing hub respectively and treated appropriately.

The correct handling of the versions is also done.

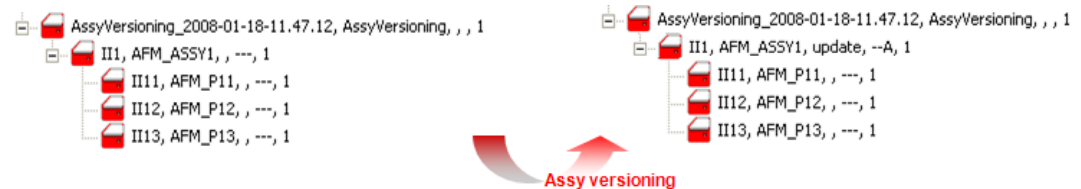


Figure 131: Correct Handling of the Versions

In the above case, the versioned assembly get updated and all its unmodified children are untouched and maintain the links to the processes. In case, the children are modified then the modified objects will be handled appropriately.

Customization

The version support can be turned off by using the following registry settings.

[HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM]

"supportversions"="0 or false or no "

The export (ProductDataGen) has to be configured in order to get the historical ID written into the transferred data. A sample configuration is:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<configuration xmlns="urn:ds:ENOVipdIntegration">
  <global>
    <properties>
      <history origin="yes"/>
    </properties>
  </global>
</configuration>
```



Note

The versioning of the documents are not supported. The export can be configured to export the origin id only from R19. The behaviour explained above can be achieved only with R19 data (provided that the export was configured to transfer the origin id).

Appendix A

1 Configuration Scenarios – DELMIA – ENOVIA VPM V5 Connection

Configuration Mapping

The DELMIA – ENOVIA VPM V5 Connection ensures the correct provision of all configuration information in the correct objects and attributes. This will allow the filter module in the PPRServer to reproduce the very same filtering results than the ENOVIA VPM V5 Configuration Modeler.

Scenarios with Extended Effectivities

Extended effectivities were introduced to solve the known restrictions of the regular configuration attributes (see following sections for details on this). A functional expression of options, dates and ranges F is transferred into another functional expression F' , formatted in a different way, but reproducing the very same filter behavior on the Manufacturing Hub side, than F on the ENOVIA VPM V5 side.

Effectivity → Extended Effectivity

$F(\text{option}, \text{date}, \text{range}) \rightarrow F'(\text{option}, \text{date}, \text{range})$

The filtering approach in the PPRServer is based on a attribute “extendedeffectivity”, contains a configuration string expression, which is evaluated by the filter engine/parser. In difference to the existing effectivity attribute “coderulestring”, the engine/parser is also enabled to recognize expressions containing date, range and label effectivities (please note that labels are out of scope for the DELMIA – ENOVIA VPM V5 Connection).

- The R 5.12 Multiline Effectivity workaround attribute and functionality stays untouched by this.

The following figure shows a sample for a Multiline Effectivity attribute value in comparison to the Extended Effectivity, using standard operators.

| | |
|--------------------------|---|
| VPM Configuration | <code>[[A+E2C] AND {1-200}] OR [[E2C+FR] AND '01.07.2003 15:27:32' to '+infinity']</code> |
| VPM Extended Effectivity | <code>('A,E2C'& '{1-200}')]('E2C,FR'& '<2003/07/01->')</code> |

Figure 132: Multiline Effectivity Attribute Value

Please note, that in difference to the “extendedeffectivity” attribute, the attribute “vpm_configuration” is not taken into account for filtering.

To access a correct view of the complex multiline effectivity in DPE, please display the attributes “vpm_configuration” and “extended effectivity”.

How to display these attributes *please refer to the* [Administration Manual](#).

Simple Scenarios with Regular Effectivities

The following two sections describe the mapping, when using the regular effectivity attributes (simple scenarios) and also the limitations with this approach (complex scenarios).

- There is no -infinity to date1/range1 in ENOVIA VPM V5.

- A range of “N to infinity” in ENOVIA VPM V5 is transcribed to “N-2147483647” ($N-2^{31}-1$). This is done internally by the DPE Server and cannot be blocked by the PPRLoader.
- Milestones must also include to infinity. A milestone itself is just a date or range.
- Dates and ranges cannot be mixed in one effectivity.
- Expressions such as “(option1 OR option2) AND (option3 OR option4)” are not possible; they have to be rearranged.
- All scenarios describe the mapping of an effectivity or configuration object (like the dictionary) to its counterpart in Manufacturing Hub (if possible). Additional Manufacturing Hub-specific filtering capabilities (such as labels or –infinity to date1) are not mentioned.
- Modification effectivities are not handled by PPRLoader.
- The three filter scopes option, range and date are concatenated with AND. Therefore, all filter conditions option/code rule AND range/tail number AND date must be valid, to see the link/object in Manufacturing Hub.
- When using regular configuration attributes AND extended effectivity, ALL 4 filter conditions (extended effectivity AND option AND range AND date) must be valid

Option Effectivity → Code Rule (and Token)

option1 → option1

Along with the option effectivity/code rule itself on the link in Manufacturing Hub, a token (option1) has to be created in Manufacturing Hub (similar to create the option along with a category in a dictionary in ENOVIA VPM V5). Currently, tokens are only created when the dictionary is exported, so options defined in a dictionary outside the exported scope are mapped into code rules, but are not handled as tokens.

A possible solution would be to create *all* options as tokens, not just the ones included in dictionaries.

Date Effectivity → Date Effectivity

date1 to date2 → date1 to date2

date2 to infinity → date2 to infinity

Range Effectivity → Tailnumber

range1 to range2 → range1-range2

range2 to infinity → range2-infinity

Milestone → Date Effectivity/Tailnumber

milestone1 to infinity → date1 to infinity/range1-infinity

Milestones are not exported as milestones, but only as their corresponding date or range values. Depending on that, PPRLoader should handle them in the appropriate way.

Mix from Option and Date or Range Effectivity

Mix from Option, Date Effectivity and Tailnumber

option1 and date1 to date2 → **option1 and date1 to date2**

option1 and range1 to range2 → **option1 and range1 to range2**

Again, a mix from date and range is not possible in the regular code rule mode. The cases date/range to infinity were covered before.

Complex Scenarios with Regular Effectivities

Complex scenarios cannot be mapped through the Manufacturing Hub-internal regular effectivity mechanism in general. The following examples indicate situations, where you have to use the Extended Effectivity mode. For details on the Extended Effectivity mode and how to define the operator tokens, *please refer to the Project Library Manual*.

Complex Option Effectivity → Code Rule

NOT option1 → **!option1**

option1 AND option2 → **option1+option2**

option1 OR option2 → **option1/option2**

Complex Date Effectivity → Code Rule Expression

(option1 AND option2) OR (option3 AND option4)

→

(option1+option2)/(option3+option4)

NOT(option1 AND option2) OR (option3 AND option4)

→

!(option1+option2)/(option3+option4)

Complex Range Effectivity → Tailnumber Expression

range1, range2 → **range1, range2**

range1 to range2 OR range3 to range4

→

range1-range2, range3-range4

Package from Option and Date or Range Effectivity

→

Mix from Option, Date Effectivity and Tailnumber

**(option1 AND option2) and date1 to date2 OR
(option3 AND option4) and range1 to infinity**

→

**(option1 AND option2) and date1 to date2 OR
(option3 AND option4) and range1 to infinity**

Configuration Objects

Dictionary

→

No Mapping

The ENOVIA VPM V5 dictionary object itself has no counterpart in Manufacturing Hub.

Config Rule → **No Mapping**

Config Rules are not mapped by PPRLoader, since they cannot be handled by Manufacturing Hub.

Config Matrix → **No Mapping**

Config Matrices are not mapped by PPRLoader, since they cannot be handled by Manufacturing Hub.

Categories → **Token Group**

Categories are stored as group descriptions on tokens.

Config Handler → **Calculation Model**

option1 → **option1**

option2,option2,option3 → **option1,option2,option3**

Calculation Models in Manufacturing Hub can only contain a list of options/tokens. Neither date nor range effectivity can be handled.

Coderule Modes

The following overview describes the various combinations of the projects code-rule mode and the used configuration attributes and their impacts.

Project Coderule-Mode „Extended“, using extended effectivity attribute

(cfgnormal="0", cfgextended="1", default for ENOVIA VPM V5 data)

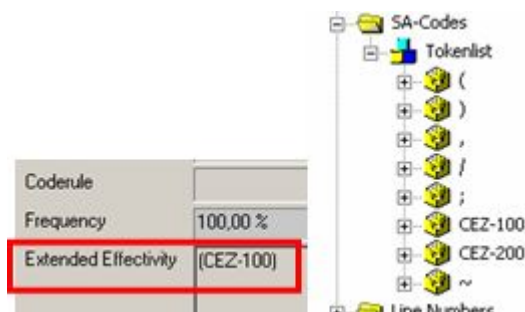


Figure 133: Tokens are generated, special characters are kept

Project Coderule-Mode „Extended“, using regular configuration attributes

(cfgnormal="1", cfgextended="0")

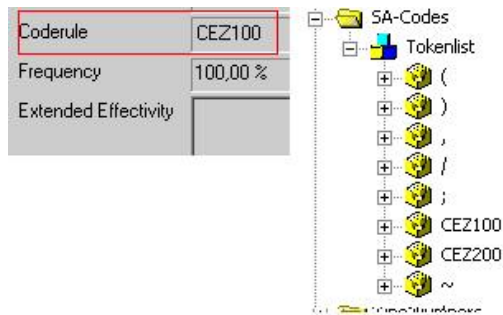


Figure 134: Tokens are generated, but special characters are removed

Project Coderule-Mode "Standard", using both, extended effectivity and regular configuration attributes

(cfgnormal="1", cfgextended="1")

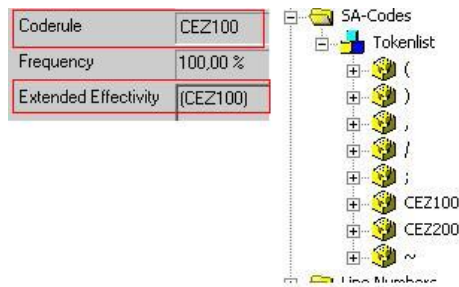


Figure 135: Tokens are generated, but special characters are removed. I.e. setting "cfgnormal" rules over "cfgextended"

Project Coderule-Mode "Standard", using extended effectivity attribute

(cfgnormal="0", cfgextended="1")

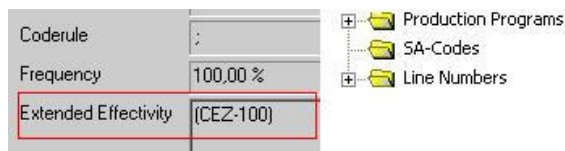


Figure 136: Tokens are not generated

Project Coderule-Mode "Standard", using regular configuration attributes

(cfgnormal="1", cfgextended="0", similar default for ENOVIA VPM V4 data)

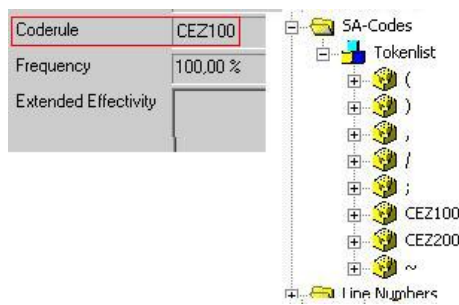


Figure 137: Tokens are generated, but special characters are removed

Project Coderule-Mode "Standard", using both, extended effectivity and regular configuration attributes

(cfgnormal="1", cfgextended="1")

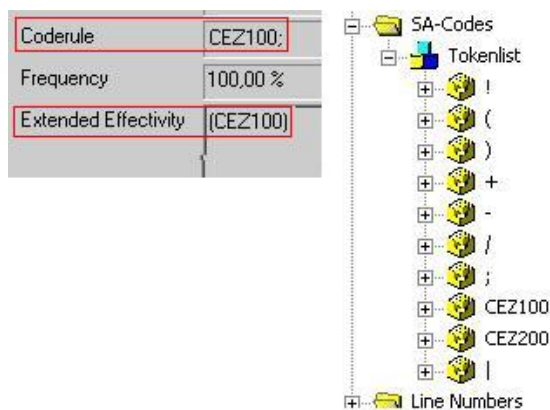


Figure 138: Tokens are generated, but special characters are removed

In other words, the setting "cfgnormal" rules over "cfgextended".

Appendix B

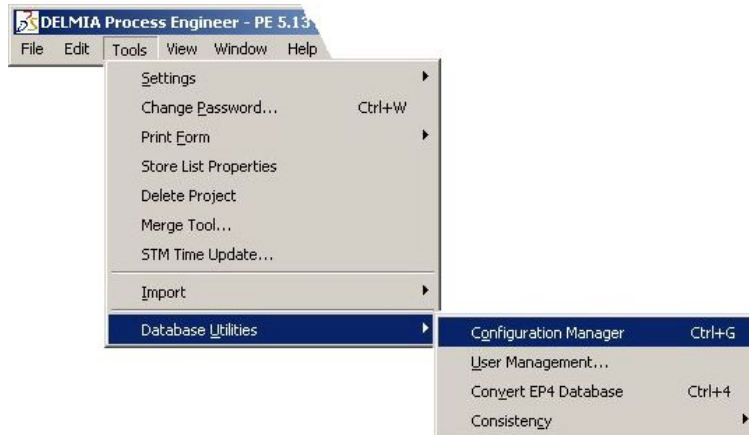
6.38 PPR Manufacturing Hub Configuration and DPE Configuration Manager

Please see also the corresponding sections/chapters in the DPE/Manufacturing Hub manual (especially in the [Administration Manual](#)).

General Information – by Administration Manual

The Configuration Manager, which allows the customization of the PPR Manufacturing Hub database, can be accessed via

DPE > Tools > Database Utilities > Configuration Manager.



For further information about how to display these attributes and change their properties, please refer to the [Administration Manual](#).

The configuration of the attribute “updatestate” is described in the following paragraph.

Special Attributes and Bitmap Adaptation for the Update Protocol

The DELMIA – ENOVIA VPM V5 Connection Update Protocol stores the status of the last synchronization on each link (SubCompltem) and object (ErgoCompBase) in the dedicated attribute “updatestate”.

To allow a simple notification for the user on changes within this attribute, it is possible to modify the bitmap of an object, based on the update status. This section describes how to set this up.

You can do the configuration in the configuration manager or in the PlanTypeSet. If you configure the attribute “updatestate” in the configuration manager, all plantypes derived from *ergocompproductdefault* are affected. In the consequence all derived plantypes obtain equal icons. The configuration of the attribute in the PlanTypeSet gives you the possibility to change the bitmaps of selected plantypes only. This allows a more exact control of the display.



Note

You cannot use both ways of configuration at the same time.

In the PlanTypeSet

The modification of the bitmap for an object is related to a “Special Attribute” that has to be defined on the “Plantype Information” page of the “Attributes & Groups & Pages” dialog.

- 1) Right-click on a plantype.
- 2) Select the “Edit” menu item to open the Attributes & Groups & Pages window.

Once you have this dialog box open, you should:

- 3) Click “Select” button in the “Attribute & Bitmap Selection” group and select the required attribute from the list, shown in the “Select Special Attribute” dialog.
- 4) In the “List of Attributes” page of the “Table Properties” dialog, select the required attribute. Either double-click on the attribute list entry or click the “Properties” button to open the attributes “Properties” dialog.
- 5) Switch to the “Advanced Properties” page and click the “Value List” button to define the list of values for the special attribute.
- 6) In the following “List of Values” dialog, click “New” to define a “Value Item.” Then enter the name, value and sort number (must be unique) for the new value item, which represents one possible status value, set by the Update Protocol mechanism.

The values used by the DELMIA – ENOVIA VPM V5 Connection Update Protocol are for Assembly:

Table 3: DELMIA – ENOVIA VPM V5 Connection Update Protocol Values

| Item Name | Item Value | Display Order | Bitmap |
|------------|------------|---------------|--------|
| “new” | new | 10 | |
| “update” | update | 20 | |
| “position” | position | 30 | |
| “delete” | delete | 40 | |
| “geometry” | geometry | 50 | |
| “config” | config | 60 | |
| EMPTY | | 70 | |

- 7) After clicking the “Select” button, the following dialog allows the import and selection of the requested bitmap, representing the corresponding update status on each object, synchronized by the DELMIA – ENOVIA VPM V5 Connection.

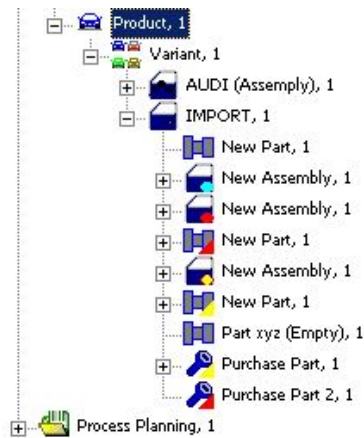


Figure 139: Import and Selection of the Requested Bitmap

Result of the configuration within the PlanTypeSet

In the Configuration Manager

- 1) Select the Attribute *updatestate*.
- 2) Change the Control type from *Edit* to **Combobox**, picking the corresponding entry from the list.



Figure 140: Change the Control Type

- 3) Now create a value list with the entries from [Table 3](#). Please refer to the [Administration Manual](#).



Note

Customized bitmaps can be created using any graphics program (e. g. Paint). The bitmaps must have a size of 18 x 17 pixels (width x height) and must be saved in 256 color mode.

- 3) Now select the type **ergocomplantdefault** and change the properties of the type.
- 4) Search the entry **Special attribute**. From the list now opening select the attribute *updatestate*.

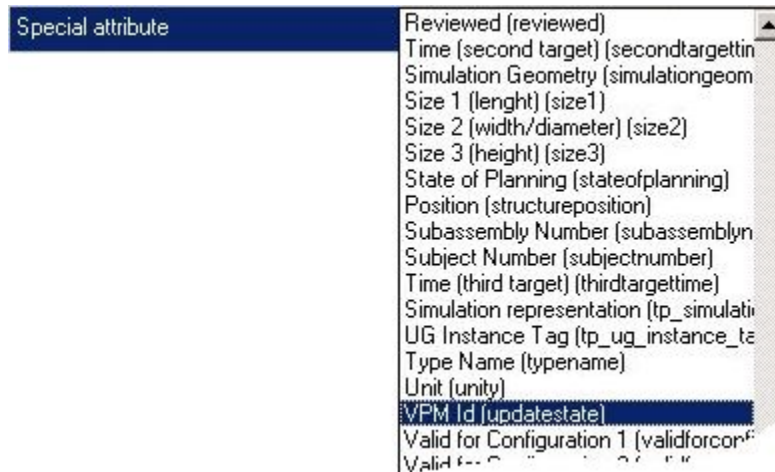


Figure 141: Special Attributes

5) Search the entry ***Depends on values***. Set it to Yes.



Figure 142: Set entry Depends on values to Yes

6) Save the changes.

- In the PPR Navigator the products are now displayed with different icons. The assigned icons identify the *updatestate*.

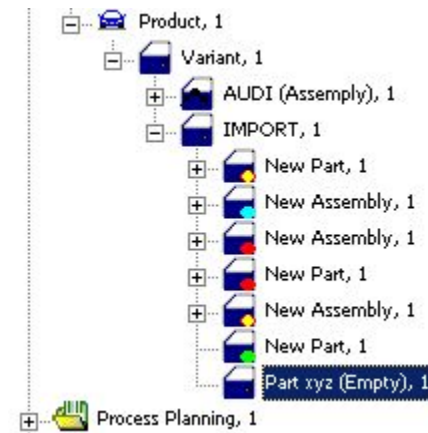


Figure 143: Result of the Configuration

Appendix C

Export Environment Variables

Table 4: Export Environment Variables

| Name | Default | Description |
|------------------------------|---------|--|
| ENOVIA_LCAIPD_COPS | off | Stop traversing a product structure when encountering a carry-over product specification (COPS) |
| ENOVIA_LCAIPD_MVA | on | Enable the transfer of multi-valued attributes (MVA) |
| ENOVIA_LCAIPD_CGR_TRANSFER | on | Enable the transfer of CGR files (same as above but with better wording and opposite meaning) |
| ENOVIA_LCAIPD_WP_TRANSFER | on | Enable the transfer of Work Package files |
| ENOVIA_LCAIPD_WIP | off | Enable in-work design (work in progress, WIP) effectivity handling |
| ENOVIA_LCAIPD_INSERT_ROOT_ID | off | Insert the ID of the root entity into the name of additionally exported files (CGR, WP reports). |
| ENOVIA_LCAIPD_RETHROW | off | Instead of ending with a return code (failure), rethrow an exception to make it visible on the top-level |

The following strings can be used to enable respectively disable a feature:

Table 5: Enable/Disable Feature

| Enable | Disable |
|--------|---------|
| 1 | 0 |
| On | off |
| Yes | no |
| True | false |

Strings and Enumerations

Table 6: Environment variables to (re-)set strings in the export code

| Name | Default | Description |
|--------------------------|-------------------------|--|
| ENOVIA_LCAIPD_OUTPUT_DIR | CATGetTempDirectory() | Absolute path to the directory which will contain the generated output |
| ENOVIA_LCAIPD_DOMAIN | Engineering Effectivity | Effectivity domain to be used. The default can also be overruled by implementing a user exit |
| ENOVIA_LCAIPD_EXPORT | | Restrict the export to only one part of the structure: REFERENCE INSTANCE The default is set to get both parts. |
| ENOVIA_LCAIPD_MAPPI | | Fine grained mapping: Colon-separated list of |

| Name | Default | Description |
|----------------------------------|-------------------|--|
| NG_ATTR | | attribute names (database names) which are checked for valid part types |
| ENOVIA_LCAIPD_MAPPING_FILTER | | Fine grained mapping: Colon-separated list of part types which are to be exported. This variable is only taken into account if mapping attributes are defined |
| ENOVIA_LCAIPD_WPTOOL_NAME | | Name of the tool to be used to export a work package if it's not already present through file introspection |
| ENOVIA_LCAIPD_WPTOOL_IN_PARAM | | Work-package tool argument which specifies the input file |
| ENOVIA_LCAIPD_WPTOOL_OUT_PARAM | | Work-package tool argument which specifies the output file |
| ENOVIA_LCAIPD_WPTOOL_PARAMLIST | | Additional parameters to be passed to the work-package tool |
| ENOVIA_LCAIPD_WPTYPE | REPORT | Type of work-package reports which is to be exported. Secondary XML files, CATIA reports, or both. Respective values are SECONDARY REPORT ALL |
| ENOVIA_LCAIPD_LOGLEVEL | NOTICE | Specify the lower boundary (priority) of messages to be placed into the log file. Valid values are METHOD DEBUG NOTICE WARN ALERT TIMES (profiling) FATAL |
| ENOVIA_LCAIPD_LOGFORMAT | %b %d %H:%M:%S | Format of the time stamps used by the logging facility (consult strftime(3) for valid formats) |
| ENOVIA_LCAIPD_CONFIGURATION_FILE | | "<Path to Customization file>".for EAR Filtering when creating partner files |

Deprecated

Environment settings that are still understood but should no longer be used.

Table 7: Environment Settings

| Name | Default | Replacement | Description |
|-------------------|---------|----------------------------|---|
| ENOVIA_LCAIPD_CGR | Off | ENOVIA_LCAIPD_CGR_TRANSFER | Disable the transfer of CGR files. Note the twist in meaning, you have to set it to on in order not to export CGRs. |

| Name | Default | Replacement | Description |
|--------------------|-----------------------|--------------------------|--|
| IPD_CGR_OUTPUT_DIR | CATGetTempDirectory() | ENOVIA_LCAIPD_OUTPUT_DIR | Absolute path to the directory which will contain the generated output |

A word on CGR export

Summarizing, we can say about the possibilities a user has with respect to the export of CGRs:

Table 8: CGR Export

| Release | CGR export |
|----------------|--|
| R14 and before | CGRs are always exported |
| R15 | user can suppress CGR export via <code>ENOVIA_LCAIPD_CGR=on</code> |
| R16 and later | user can suppress CGR export via <code>ENOVIA_LCAIPD_CGR_TRANSFER=off</code> |

Appendix D

XML-Based Configuration

With the new way of customization a more complex way of configuration is possible. The known mechanism via environment variables is still supported. If environment variables are defined they take precedence over settings in the configuration file.

The only environment variable necessary after this change is

ENOVIA_LCAIPD_CONFIGURATION_FILE.

Its value is the absolute path to the XML file containing the configuration, e.g.

```
export ENOVIA_LCAIPD_CONFIGURATION_FILE=/path/to/my/export-config.xml
```

Everything else can and should be placed in the XML file itself.

The configuration file must be well-formed. If an error is encountered by the software while parsing the configuration, the export will immediately stop the execution, even before any file will be created.

The error message issued by the parser is displayed in the command window along with the position in the configuration file. For example, a stray ampersand "&", which can be part of a PRC name, leads to the following message:

Fatal SAX Error: Expected entity name for reference, line 7, column 15 Internal configuration error.

The correctly encoded wording would be "&";

A DTD (resources/dtd/ENOVlpdIntegration.dtd) is provided along with the configuration file. Therefore, it should be possible to write **valid** configuration settings with an editor capable of evaluating a DTD and parsing an XML document accordingly.

Here's a configuration file which contains all possible sections.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<configuration>
  <global>
    <properties>
      <cgr-transfer enabled="yes"/>
      <check-doc-iteration enabled="no"/>
      <idfile size="30" processes="2" keep-idf="no"/>
      <in-work-design enabled="no"/>
      <log level="method"/>
      <mva enabled="yes" separator="$$"/>
      <output directory="/tmp" insert-root-id="no"/>
      <rethrow enabled="no"/>
    
```

```
<rewrite-proc enabled="yes"/>
<separate-rl enabled="yes"/>
<stop-at-cops enabled="no"/>
<substitute-transfer enabled="no"/>
<user-exit-encryption enabled="no"/>
<wp-transfer enabled="yes"/>
</properties>

<effectivitydomains>
  <domain name="VALIDE" default="yes"/>
  <domain name="Maintenance Effectivity"/>
  <domain name="Manufacturing Effectivity"/>
  <domain name="Engineering Effectivity"/>
  <domain name="Proposed Effectivity"/>
</effectivitydomains>
</global>

<prcs>
  <prc id="prc1"/>

  <prc id="prc2">
    <exports>
      <token>export1</token>
      <token>export2</token>
    </exports>
  </prc>

  <prc id="prc3">
    <exports>
      <token>export1</token>
    </exports>
  </prc>
</prcs>

<permissions>
```

```

    <permission id="export1" description="Export permissions
export1">
        <token>ID90</token>
        <token>ID91</token>
        <token>ID92</token>
    </permission>

    <permission id="export2" description="Export permissions
export2">
        <token>ID90</token>
        <token>ID91</token>
        <token>ID92</token>
        <token>ID93</token>
    </permission>
</permissions>

<filters>
    <attributelist context="PartMaster">
        <blacklist>
            <token>V_description</token>
        </blacklist>
    </attributelist>

    <attributefilter context="PartVersion" attribute="ajz">
        <impact attribute="abc">
            <replacement>POTENTIAL SENSITIVE DATA|ASK JOE PUBLIC
FOR
            COMPLETE DEFINITION|FILLER TEXT|FILLER
TEXT</replacement>
        </impact>

        <impact attribute="def">
            <replacement>POTENTIAL SENSITIVE DATA|ASK JOE PUBLIC
FOR
            COMPLETE DEFINITION|FILLER TEXT</replacement>
        </impact>
    </attributefilter>

```

```

    <attributefilter context="PartMaster" attribute="a" de-
limiter=":"
      ignorecase="yes">
        <impact attribute="x">
          <replacement>censored text</replacement>
        </impact>

        <impact attribute="y">
          <replacement>must not see this</replacement>
        </impact>

        <impact attribute="z">
          <replacement>black bar</replacement>
        </impact>
      </attributefilter>

    <extendedfilter context="PartVersion" attribute="xx" in-
orecase="yes">
      <whitelist>
        <token>text</token>
      </whitelist>
    </extendedfilter>

    <extendedfilter context="PartInstance" attribute="yyy">
      <blacklist>
        <token>some text</token>
      </blacklist>
    </extendedfilter>
  </filters>

  <mappings>
    <attributeprefix context="PartMaster" prefix="PM::">
      <attributes>
        <token>V_user</token>
        <token>V_organization</token>
      </attributes>
    </attributeprefix>
  </mappings>

```

```

        </attributes>
    </attributeprefix>

    <attributeprefix context="PartVersion">
        <attributes>
            <token>V_organization</token>
        </attributes>
    </attributeprefix>

    <mapping context="PartVersion">
        <attributes>
            <token>abc</token>
            <token>def</token>
        </attributes>
    </mapping>

    <mapping context="ProductRoot">
        <attributes>
            <token>xyz</token>
        </attributes>
    </mapping>
</mappings>

<outputs>
    <docinfo postfix="-docinfo">
        <token>V_user</token>
        <token>V_extension</token>
        <token>C_created</token>
        <token>C_modified</token>
    </docinfo>
</outputs>
</configuration>

```

Allowed Contexts

- ProductClass
- ProductRoot
- PartMaster

- PartVersion
- PartInstance
- Document

Configuration sections which are not described in other sections:

- `<global><properties>`: This section contains most of the known environment settings. We have combined settings that belong in the same context, e.g. the MVA activation and the definition of the MVA separator.



Caution

If an environment variable is defined it overrides the settings in the configuration file!

- **rewrite-proc: Start** the post-processing of the generated XML files (rewriting) in two separate processes, reference and instance. Enabled by default.
- **separate-rl**: Export the links to the first level part instances (root links, rl) in separate processes. Enabled by default.
- `<filters><attributelist>`: It is possible to suppress the export of attributes. To this end define either a whitelist of attributes to be exported or a blacklist of attributes to be quelled within the given context. For example, in the above displayed configuration file the description attribute of a part master won't show up in the generated XML file(s).
It is not allowed to use a context more than once.

Exception: If an attribute filter is defined for this context, the filter attribute as well as all impacted attributes are exported, regardless of the attribute list definition.

- `<mappings><attributeprefix>`: It's possible to assign a prefix to a set of (specified) attributes within a given context. If no special prefix has been defined, the context name itself will be used as string. In this case a dot is appended to increase visibility.
Using the above example, instead of the unadorned names, the generated reference XML file would contain entries like

```
<Attribute Type="String" Name="PM::V_user">
  <Value Value="NDH"/>
</Attribute>
<Attribute Type="String" Name="PM::V_project">
  <Value Value=""/>
</Attribute>
<Attribute Type="String" Name="PartVersion.V_user">
  <Value Value="NDH"/>
</Attribute>
<Attribute Type="String" Name="PartVersion.V_organization">
  <Value Value="VPM"/>
```


</Attribute>

- **<outputs><docinfo>**: Since the generated XML files are in a binary format, it's no longer possible to gain information by parsing them. It is possible, though, to extract document vitals through this configuration. The given postfix is used in the generated file name which changes from PRC_ID-MODIFICATION_TIME.xml into PRC_ID-MODIFICATION_TIME**POSTFIX**.xml. Default postfix is "-doc". The given tokens are taken to be document attributes which are written into the additionally generated unencrypted XML file. For example, using the above configuration we may end up with a file containing the following contents:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Package    Uuid="00000000-0000-0000-0000000000000000"    Ver-
sion="0"    Type="OSM"
    Name="Root">
    <Package    Uuid="00000000000000d5443e86d8a000003c7"    Ver-
sion="1"
        Type="CATSpecContainer"    Name="ENOVIALCA.DATA">
        <Instance    Uuid="83a4637b00007270440e99eb000ce745"
            Alias="(DOCDIR.DOCUMENT)TstRewrite-document---
2006-03-08-09.47.18"
            Type="CATPDMObject"
            Name="(DOCDIR.DOCUMENT)TstRewrite-document---
2006-03-08-09.47.18">
            <Attribute    Type="String"    Name="C_created">
                <Value    Value="2006-03-08-09.46.35"/>
            </Attribute>
            <Attribute    Type="String"    Name="C_modified">
                <Value    Value="2006-03-08-09.47.18"/>
            </Attribute>
            <Attribute    Type="String"    Name="V_user">
                <Value    Value="MUTEST"/>
            </Attribute>
            <Attribute    Type="String"    Name="V_extension">
                <Value    Value="CATPart"/>
            </Attribute>
        </Instance>
    </Package>
</Package>
```

Configuration to support fine grained mapping for PRCs

On the EH-Server the user has to define an environment variable ENOVIA_LCAIPD_CONFIGURATION_FILE pointing to the Customization file. The Customization file contains a list of database names of the attributes used for type mapping. The order of the occurrence of the attributes does matter. A different order may lead to a different mapping on the import side. The search will stop on the first match. Match in this case means, the attribute exists on the current PRC and its value is not empty.

For example, the list may consist of mapping `mapping_attr_1` and `mapping_attr_2`, e.g.

```
<mappings>
  <mapping context="ProductRoot">
    <attributes>
      <token>mapping_attr_1</token>
      <token>mapping_attr_2</token>
    </attributes>
  </mapping>
</mappings>
```

Now, if the exporter comes across a PRC master or PRC version which contains an attribute `mapping_attr_1` with a non-empty value the following happens:

The PRC will be sent over the DELMIA – ENOVIA VPM V5 Connection and the value of `mapping_attr_1` is used as mapping information, no matter whether `mapping_attr_2` exists and has a value assigned to it.

If `mapping_attr_2` shall be used as mapping information, the attribute must contain a value. Additionally, `mapping_attr_1` may exist but must not have a value assigned to it or the order of the tokens in the attribute list has to be reversed.

The mapping attribute(s) customized in the Customization file have to be defined either on the PRC master or PRC version. PRC master attributes take precedence over PRC version attributes.

Currently the only way to customize the behavior of this feature is via the Export Customization File. There is no support for command line arguments or environment variables.

Configuration to support the export multiple effectivity domains

On the EH-Server the user has to define an environment variable ENOVIA_LCAIPD_CONFIGURATION_FILE pointing to the Export Configuration File. The Configuration file contains a list of effectivity domains which will be written into the DELMIA – ENOVIA VPM V5 Connection export data.

For example, the list may contain the two effectivity domains "Engineering Effectivity" and "Manufacturing Effectivity" ,

```
<effectivitydomains>
  <domain name="Engineering Effectivity" default="yes"/>
  <domain name="Manufacturing Effectivity"/>
</effectivitydomains>
```

Now, if the exporter comes across an assembly relation the following happens:

The assembly relation object will be queried for all customized domains and for each of those (if also customized) the WIP and the Approved effectivity will be extracted and written into the export data. The effectivities for multiple domains are extracted independently from each other. So a single mod statement may contain effectivity data for multiple domains. Still only the node effectivity is transferred, no single mod statements.

If no Export Configuration File is used, no effectivity domains are defined in the file, and no domain is set either by user exit or environment variable the "Engineering Effectivity" is transferred.

Configuration to support filtering of EAR data

Within the configuration section there is a section <prcs>: In this section the prc ids must be defined which should be used for export. Each PRC section may have a section <exports> which contains an entry for every respective export (a separate pair of files) wanted.

Each export name must correspond to exactly permission id in the permissions section. Within the section of one permission export tokens are specified. These can be understood as access keys to sensitive data.

In the section <filters> the attributefilters are defined which control the impacted attributes for each transferred entity (currently parts are supported). The key "attribute" of the attributefilter tag denotes the name of the attribute whose value contains the necessary set of access tokens. The separator between the tokens can be specified using the respective key. Whitespace is taken to be the default. It's possible to configure whether the token match should ignore the case. A respective key "ignorecase" is given, see above.

Within the attributefilter section impacted attributes are defined. The content of these attributes is replaced by the replacement text if not all the permission tokens read from the attributefilter attribute match the permission tokens for the export name as explained above.

An Example

```
<prcs>

  <prc id="787"/>

  <prc id="Sect41">
    <exports>
      <export>MHI</export>
      <export>FHI</export>
    </exports>
  </prc>

  <prc id="Wing">
    <exports>
      <export>MHI</export>
```

```

        </exports>
    </prc>
</prcs>

<permissions>
    <permission id="MHI" description="Mitsubishi Heavy In-
dustries Ltd.">
        <token>AAA90</token>
        <token>AAA91</token>
        <token>AAA92</token>
    </permission>

    <permission id="FHI" description="Fuji Heavy Industries
Ltd.">
        <token>AAA90</token>
        <token>AAA91</token>
        <token>AAA92</token>
        <token>AAA93</token>
    </permission>
</permissions>

<filters>
    <attributefilter context="PartMaster" attribute="ajz">
        <impact attribute="abc">
            <replacement>POTENTIAL EAR-SENSITIVE DATA|SEE THE
MBD FOR
                COMPLETE DEFINITION|FILLER TEXT|FILLER
TEXT</replacement>
        </impact>

        <impact attribute="def">
            <replacement>POTENTIAL EAR-SENSITIVE DATA|SEE THE
MBD FOR
                COMPLETE DEFINITION|FILLER TEXT</replacement>
        </impact>
    </attributefilter>
</filters>

```

Configuration to Support Extended Part Filtering

On the EH-Server the user has to edit the exporter's configuration file to define the extended filters. Empty token values are allowed and allow the transfer of parts which have no or an empty value for the filter attribute. The extended filter has to be defined in the filters section of the configuration file:

```
<filters>

  <extendedfilter context="PartMaster"
attribute="V_attributeName" ignorecase="yes"
    filtermissing="no">
    <whitelist>
      <token>Value 1</token>
      <token>Value 2</token>
      <token></token>
    </whitelist>
  </extendedfilter>

  <extendedfilter context="PartVersion"
attribute="V_another_attribute" ignorecase="no">
    <blacklist>
      <token>abc</token>
      <token>xyz</token>
    </blacklist>
  </extendedfilter>
</filters>
```

The **context** describes the object type, where the **attribute** is found. This can be ProductRoot, PartMaster, PartVersion or PartInstance. The attribute **ignorecase** is optional. When missing, comparisons will be done case sensitive. This filter definition would filter out all parts, whose PartMaster attribute "V_attributeName" and whose PartVersion attribute "V_another_attribute" have a value which is listed in the black list given.

Errors in the definition of the extended filters like an empty white list will be ignored that is the filter will not be applied. If it is impossible to parse a filter definition (i.e. unknown context, bad xml token, etc.) the exporter will abort the execution.

An unknown attribute in a white-list filter (typo) will lead to an empty export file, since no part matches the filter requirements. It is ignored in a black-list filter. This default behavior can be changed by setting the optional attribute "**filtermissing**". It can take the two values "yes" and "no". Thus, in the above example, if a Part does not have the attribute "V_attributeName" it will not be filtered.

Appendix E

Table 9: Configuration Settings - ProductDataGen

| Element | Attribute | Value (default) | Value (option) | Command Line | Description |
|---|-----------|-----------------|---|--------------|---|
| whitelist | token | | | | Specifies list of ENOVIA environments. If the part does not have a matching value, it should be filtered out. |
| blacklist | token | | | | Specifies list of ENOVIA environments. If the part has a matching value, it should be filtered out. |
| <code><mappings></code> <code><mapping></code> | | | | | |
| | context | | PartInstance PartMaster PartVersion ProductClass ProductRoot ProductSpec | | It's possible to assign a ENOVIA attributes as discriminators for fine-grain mapping to DPE Plantypes. Context specifies which ENOVIA types should be used for this mapping. |
| attributes | token | | | | Specifies the ENOVIA attribute for a given context whose value will act as a discriminator for fine-grain mapping |
| <code><mappings></code> <code><attributeprefix></code> | | | | | |
| | context | | PartInstance PartMaster PartVersion ProductClass ProductRoot ProductSpec Document | | It's possible to assign a prefix to a set of (specified) attributes within a given context. If no special prefix has been defined, the context name itself will be used as string. In this case a dot is appended to increase visibility. |
| | prefix | | | | Specifies the prefix to be assigned to each attribute within the given context. |

| | | | | | |
|-----------------|--------|--|--|--|---|
| attributes | token | | | | Each token specifies the ENOVIA attribute that should have a prefix for the given context |
| <markings> | | | | | |
| marking | | | | | Specifies a header information to go into export xmls |
| <outputs> | | | | | |
| docinfo | | | | | It is possible, though, to extract document vitals through this configuration. The given postfix is used in the generated file name which changes from PRC_ID-MODIFICATION_TIM E.xml into PRC_ID-MODIFCATION_TIM EPOSTFIX.xml. Default postfix is "-doc". The given tokens are taken to be document attributes which are written into the additionally generated unencrypted XML file |
| <prcs> <prc> | | | | | |
| | Id | | | | Enables split-bridge export for the list of prcs identified by id |
| exports | export | | | | Specifies the permission ids that will have an EAR filtered split-bridge export for the given prcs. |

| Element | Attribute | Value (default) | Value (option) | Command Line | Description |
|----------------------------------|-----------|-------------------------|----------------|--------------|---|
| <global> <effectivitydomains> | | | | | |
| domain | name | Engineering Effectivity | | | Allows user to export multiple effectivity domains in a single export |
| domain | default | yes (Engineering) | yes/no | | Mutually exclusive default domain to be imported to Manufacturing Hub. Can be |

| | | | | | |
|---|------------|--------------|---|--|--|
| | | Effectivity) | | | overridden on import using +domain |
| <code><filters></code> <code><attributefilter></code> | | | | | |
| | Attribute | | | | The key “attribute” of the attributefilter tag denotes the name of the attribute whose value contains the necessary set of impact attributes |
| | context | | | | It is possible to replace sensitive attribute values with a predefined text. This can be done within a given context |
| | Delimiter | | | | |
| | Ignorecase | | yes/no | | Ignore the case of the text when applying filter |
| impact | Attribute | | | | impacted attributes are defined. The content of these attributes is replaced by the replacement text if not all the permission tokens read from the attributefilter attribute match the permission tokens for the export name as explained above |
| <code><filters></code> <code><attributefilter></code> <code><impact></code> | | | | | |
| replacement | | | | | Specifies text to replace the impacted attribute |
| <code><filters></code> <code><attributelist></code> | | | | | |
| | context | | Document PartInstance PartMaster PartVersion ProductRoot ProductSpec | | It is possible to specify attributes that should always be transferred in incremental mode. This can be done within a given context |
| blacklist | token | | | | Specifies list of attributes for a given context. Attributes that are on this list should not be transferred. |
| mandatory-list | token | | | | Specifies list of attributes for a given context. This list specifies the attributes which should always be transferred in incremental mode |
| whitelist | token | | | | Specifies list of attributes for a given context. Only attributes that are on this list and mandatory list should be transferred |

| <filters> | | | | | |
|--|----------------|--|---|--|--|
| <extendedfilter> | | | | | |
| r m a n e n t L i m i t a t i o n s | Attribute | | | | The key "attribute" of the extendedfilter tag denotes the name of the attribute whose value contains the necessary set of access tokens |
| | Context | | PartInstance PartMaster PartVersion | | It is possible to suppress the export parts and their sub structures based on a given attribute filter. This can be done within a given context |
| | filter-missing | | | | An unknown attribute in a white-list filter (typo) will lead to an empty export file, since no part matches the filter requirements. It is ignored in a black-list filter. This default behaviour can be changed by setting the optional attribute "filtermissing" |
| | ignore-case | | | | Ignore the case of the text when applying filter |
| ▪ C O P S white list | token | | | | Specifies list of attributes values for a given context. If the part does not have a matching value for a given context, it should be filtered out. |
| i n s t black list | token | | | | Specifies list of attributes values for a given context. If the part has a matching value for a given context, it should be filtered out. |
| <filters> | | | | | |
| <environmentfilter> | | | | | |

e dir

- Situations in which a product component is linked to different parents components in the EBOM in DPE is not supported by the DPE data model.
- Parts with ambiguous part instance IDs cannot be used as root components for Connection transfer (neither interactive nor batch).
- Unsynchronized ENOVIA VPM V5 instance structures cannot be transferred correctly to Manufacturing Hub; the instances may point to different part versions but the structure itself must be identical; this also accounts for the existence of substitute parts, e.g. if one instance in an assembly owns a substitute part and the assembly is reused, a corresponding substitute part has to be created in the reused assembly as well. Otherwise it is not defined which version of the assembly is transferred.

Appendix F

Table 10: Registry Settings

| Key | Value (default) | Value (option) | Description |
|---|-----------------------|-------------------------|--|
| [HKEY_CURRENT_USER\Software\Delmia\ergoplan\PPRLoader] | | | |
| "username" | "" | | Look at documentation, chapter ".PPRLoader Configuration Settings" . Enter username of DPE to use the pprloader. |
| "password" | "" | | Look at documentation, chapter PPRLoader Configuration Settings ". Enter password of DPE to use the pprloader. |
| "resetupdates-tate" | "" | "1 true yes 0 false no" | Controls the reset of the update states for all links and objects before updating the project (special interest for incremental update) |
| "propagatedeletes-tate" | "" | "1 true yes 0 false no" | Specifies, that the update state "delete" is propagated from an object to the entire sub-structure beneath |
| "operationmode" | "" | "synchronized" | Optimizes the PCS behavior of PPRLoader for massive parallel operation on multiprocessor machines, and provides lower memory consumption when compared to the default mode. |
| [HKEY_CURRENT_USER\Software\Delmia\ergoplan\PPRLoader\VPM] | | | |
| "backbonesender" | "<YourMachines-Name>" | | Look at documentation, chapter "Configure the PPRDaemon" . Enter the name of the machine where the PPRDaemon/PPRLoader is running. Comment: Enter "LOCALHOST" (currently backbonesender and receiver have to be the same machine) |
| "backbonereceiver" | "<YourMachines-Name>" | | Look at documentation, chapter "Configure the PPRDaemon" . Enter the name of the machine where the ENOVIA VPM V5 Client is running. Comment: Enter "LOCALHOST" (currently backbone sender and receiver have to be the same machine) |
| ;"xmlftphost" | "NO_LONGER_IN_USE!!!" | | obsolete |

| Key | Value (default) | Value (option) | Description |
|-------------------|--|----------------|-------------|
| ;"xmlftpscript" | "NO_LONG ER_IN_USE !!!____<Your FTPScriptsD ir>\ftpgetvp mxml.bat" | | obsolete |
| ;"xmlftpusername" | "NO_LONG ER_IN_USE !!!" | | obsolete |
| ;"xmlftppassword" | "NO_LONG ER_IN_USE !!!" | | obsolete |
| ;"xmldir" | "NO_LONG ER_IN_USE !!!____<DEL MIAProces- sEnginee- rInstall- Dir\PPRClie nt\data>" | | obsolete |
| ;"xmlsrvdir" | "NO_LONG ER_IN_USE !!!____<VPM ServerEx- portDirecto- ry>" | | obsolete |
| ;"lxcftphost" | "NO_LONG ER_IN_USE !!!" | | obsolete |
| ;"lxcftpscript" | "NO_LONG ER_IN_USE !!!____<Your FTPScriptsD ir>\ftpgetvp mlxc.bat" | | obsolete |
| ;"lxcftpusername" | "NO_LONG ER_IN_USE !!!" | | obsolete |
| ;"lxcftppassword" | "NO_LONG ER_IN_USE !!!" | | obsolete |
| ;"lxcdir" | "NO_LONG ER_IN_USE !!!____<DEL MIAProces- sEnginee- rInstall- Dir\PPRClie nt\data>" | | obsolete |
| ;"lxcsrvdir" | "NO_LONG ER_IN_USE !!!____<XCA DServerEx- portDirecto- | | obsolete |

| Key | Value (default) | Value (option) | Description |
|-------------------|---|-------------------------------|---|
| | ry>" | | |
| ;"cgrftpghost" | "NO_LONG ER_IN_USE !!!" | | obsolete |
| ;"cgrftpscript" | "NO_LONG ER_IN_USE !!!____<Your FTPScriptsD ir>\ftpgetvp mcgr.bat" | | obsolete |
| ;"cgrftpusername" | "NO_LONG ER_IN_USE !!!" | | obsolete |
| ;"cgrftppassword" | "NO_LONG ER_IN_USE !!!" | | obsolete |
| ;"cgrdir" | "NO_LONG ER_IN_USE !!!____<DEL MIAProces- sEnginee- rInstall- Dir\PPRClie nt\product_ cadpath>" | | obsolete |
| ;"cgrsrvdir" | "NO_LONG ER_IN_USE !!!____<XCA DServerEx- portDirecto- ry>" | | obsolete |
| "cfgcheck" | "1" | "1 true yes 0 false no" | Enable/disable update protocol check for configuration attributes (effectivity.begin, effectivity.end, tailnumber, coderulestring, exten- dedeffectivity) |
| "cfgdate" | "link+comp" for VPM V4 "link" for VPM V5 | "all link comp link+co mp" | Specifies, if date effectivity is stored on relationship_nodes (link), on ergocompbase (comp) or on both |
| "cfgoption" | "link+comp" for VPM V4 "link" for VPM V5 | "all link comp link+co mp" | Specifies, if option effectivity is stored on relationship_nodes (link), on ergocompbase (comp) or on both |
| "cfgrange" | "link+comp" for VPM V4 "link" for VPM V5 | "all link comp link+co mp" | Specifies, if range effectivity is stored on relationship_nodes (link), on ergocompbase (comp) or on both |
| "cfgexteff" | "link+comp" for VPM V4 "link" for VPM V5 | "all link comp link+co mp" | Specifies, if extended effectivity is stored on relationship_nodes (link), on ergocompbase (comp) or on both |

| Key | Value (default) | Value (option) | Description |
|-----------------------------|-----------------------------------|--|---|
| "cfgnormal" | "1" for VPM V4 "0" for VPM V5 | | Specifies, whether effectivity is stored in the « old » configuration attributes |
| "cfgextended" | "0" for VPM V4 "1" for VPM V5 | | Specifies, whether effectivity is stored in the extendedeffectivity attribute |
| "cgratfile" | "" | | obsolete |
| "cgrftpdelay" | "" | | obsolete |
| "longextid" | "" | | obsolete |
| "prodroot" | "" | | obsolete |
| "prodnodet" | "" | | obsolete |
| "prodleaf" | "" | | obsolete |
| "prodmodel" | "" | | obsolete |
| "proddoc" | "" | | obsolete |
| "prodmodelattr" | "" | | obsolete |
| "proddocattr" | "" | | obsolete |
| "resroot" | "" | | obsolete |
| "resnode" | "" | | obsolete |
| "resleaf" | "" | | obsolete |
| "resmodel" | "" | | obsolete |
| "resdoc" | "" | | obsolete |
| "resmodelattr" | "" | | obsolete |
| "resdocattr" | "" | | obsolete |
| "cfgfile" | "<De- faultVPMCo nfigFile>" | | Look at documentation, chapter "Customized/Extended Attribute Mapping". Enter location of Customized/Extended Attribute Mapping configuration file. |
| "deletemode" | "notify" | "notify attachment auto none" | Look at documentation, chapter "Two modes for deleted objects". |
| "filterfile" | | | |
| "updatestateattrlink" | "updates- tate" | | obsolete |
| "updatestateattrrobject" | "updates- tate" | | ergocompbase attribute for update protocol |
| "updatestateattrrelation" | "updates- tate" | | For future use |
| "updatestateattrattachment" | "updates- tate" | | relationship_plant_provides_prod attribute for update protocol |
| "multiinst" | "" | | obsolete |

| Key | Value (default) | Value (option) | Description |
|---|------------------|---|---|
| "enuminst" | "order-number" | | ergocompbase attribute for enumeration of objects |
| "versions" | "1" | "1 0" | Enables/disables support for VPM V4 versions |
| "dump" | "" | "comp link cgr cad attr progress config env id all" | Allows racing of additional information during import |
| "recalcabsmatrix" | "1" | "1 true yes 0 false no" | Controls the recalculation of the absolute position matrices by the Manufacturing Hub server |
| "mvamultiline" | "" | "1 none" | Look at documentation "EH-MH Transfer of Multi Value Attributes". If the MVA value is long and requires more than one line then enter 1 to get several lines. |
| "mvaseparator" | "" | Any separator except "@@" because this is used as default by program. | Look at documentation "EH-MH Transfer of Multi Value Attributes". To use other separators than the default. |
| "mvalinebreak" | "" | "\r\n". | Look at documentation "EH-MH Transfer of Multi Value Attributes". The default MVA line break character is "\r\n". Without any changes every value of the attribute is shown in a separate line. |
| ;"useclientcache" | "" | | |
| [HKEY_CURRENT_USER\Software\Delmia\ergoplan\PPRLoader\VPM\Environments] | | | |
| [HKEY_CURRENT_USER\Software\Delmia\ergoplan\PPRLoader\VPM\Aliases] | | | |
| [HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings] | | | |
| "securetransfer-mode" | "0" | "0 <0 .. ftp 1 .. https>" | Enter 0 for transfer via FTP and enter 1 for transfer via https. |
| [HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings\ABF Settings] | | | |
| [HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings\CGR Settings] | | | |
| [HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings\LXC Settings] | | | |
| [HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings\WPK Settings] | | | |
| [HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings\XML Settings] | | | |
| "xmlftphost" | "<FTPHost-Name>" | | Look at documentation, chapter "Configure XML Settings". Enter |

| Key | Value (default) | Value (option) | Description |
|--|--|-----------------|--|
| | | | the name of the ENOVIA VPM V5 database Server. |
| "xmlftpusername" | "<FTPUser-Name>" | | Look at documentation, chapter "Configure XML Settings". Enter your username. |
| "xmlftppassword" | "<FTPPass word>" | | Look at documentation, chapter "Configure XML Settings". Enter your UNIX password |
| "xmldir" | "<DELMIA-ProcessEngineerInstall-Dir\PPRClient\data>" | | Look at documentation, chapter "Configure XML Settings". Enter the pathname of the directory in which you want the XML files stored. |
| "xmlsrvdir" | "<VPMServerExportDirectory>" | | Look at documentation, chapter "Configure XML Settings". Enter the path to find XML on the ENOVIA VPM V5 server. |
| "xmllogdir" | "C:\Temp" | | Look at documentation, chapter "Configure XML Settings". Enter the path to find XML. |
| "xmlhttpsbaseurl" | "" | | Look at documentation, chapter "Configure XML Settings". |
| [HKEY_CURRENT_USER\Software\Delmia\ergoplan\PPRLoader\VPM\ABF Fastener Mapping] | | | |
| "fastener" | "" | | |
| "joint" | "" | | |
| "jointelement" | "" | | |
| "jointbody" | "" | | |
| "mode" | "internal external" | | |
| "fileextension" | "" | | |
| "cfgfile" | "<DefaultABFConfig-File>" | | |
| "relationship" | "" | | |
| [HKEY_CURRENT_USER\Software\Delmia\ergoplan\PPRLoader\VPM\Product Structure Mapping] | | | |
| "Resource Environments" | "RESENV" | "ENV1+ENV2+..." | Specifies the environment(s) on the VPM side for which relations between a product component and a resource component shall be created |
| [HKEY_CURRENT_USER\Software\Delmia\ergoplan\PPRLoader\VPM\Product Structure Mapping\PRODUCT] | | | |
| ("PRODUCT" serves as an example for other product structure mappings; for any defined environment the keys must be existing and not empty) | | | |

| Key | Value (default) | Value (option) | Description |
|---|-----------------|----------------------------|--|
| [HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\Product Structure Mapping\PRODUCT\CATIA_MODEL] | | | |
| "Plantype" | "Part" | Any valid product plantype | "Valid" also includes parent-child constraints resulting from the used plan type set |
| "CAD File Attribute" | "cadfile" | | Do not change |
| [HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\Product Structure Mapping\PRODUCT\DOCUMENT] | | | |
| "Plantype" | "Part" | Any valid product plantype | "Valid" also includes parent-child constraints resulting from the used plan type set |
| "CAD File Attribute" | "cadfile" | | Do not change |
| "Document Attribute" | "nameshort" | | Do not change |
| [HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\Product Structure Mapping\PRODUCT\PART_LIST] | | | |
| "Plantype" | "Subassembly" | Any valid product plantype | "Valid" also includes parent-child constraints resulting from the used plan type set |
| [HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\Product Structure Mapping\PRODUCT\Root] | | | |
| "Plantype" | "Product" | Any valid product plantype | "Valid" also includes parent-child constraints resulting from the used plan type set |
| [HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\PCS] | | | |
| "doqueries" | "" | "1 true yes 0 false no" | Specifies, whether objects are pre-read into Manufacturing Hub server by using a query |
| "committhreshold" | "" | | Look at documentation, chapter "Configure the PPRDaemon". Enter the name of the machine where the ENOVIAVPM V5 client is running. Comment: Enter "LOCALHOST" |
| "threadsleep" | "" | | Number of milliseconds to wait before starting the next thread |
| "newstructuretraversal" | "" | "1 true yes 0 false no" | Enables alternative structure traversal |
| "parallelizeparseandfetch" | "" | "1 true yes 0 false no" | Enables parallel parse and fetch (threads are started in parallel to parsing the XML file once the root object is read) |
| "preloadattributes" | "" | "1 true yes 0 false no" | Disables the pre-loading of the attributes |
| "threads" | "" | | Number of child threads to pre-load the structure |

| Key | Value (default) | Value (option) | Description |
|--------------------|-----------------|-------------------------|---|
| "threadpriority" | "" | | Thread priority |
| "threadsreadahead" | "1" | "1 true yes 0 false no" | Controls the read-ahead behavior of PPRLoader |

List of Figures

| | |
|---|----|
| Figure 1: Login Screen | 12 |
| Figure 2: Logon | 12 |
| Figure 3: ENOVIA VPM V5 Client | 13 |
| Figure 4: Manufacturing Hub | 14 |
| Figure 5: User Authorization | 14 |
| Figure 6: ENOVIA VPM V5..... | 15 |
| Figure 7: Product Structure | 17 |
| Figure 8: Reference Tree | 21 |
| Figure 9: Instance Attributes..... | 22 |
| Figure 10: Secure Connection..... | 24 |
| Figure 11: DCOM Setting – Step 1 | 26 |
| Figure 12: DCOM Setting – Step 2..... | 27 |
| Figure 13: DCOM Setting – Step 3..... | 28 |
| Figure 14: ServerTools client – PoolingServer Lock Status | 29 |
| Figure 15: Set Property Exclude from MCM in Configuration..... | 31 |
| Figure 16: Set Property Exclude from MCM and CCz in Configuration..... | 31 |
| Figure 17: Set Proprty Exclude from MCM in Configuration..... | 32 |
| Figure 18: Backbonesender | 33 |
| Figure 19: Enter the Path to Find CGR Files..... | 34 |
| Figure 20: Share Drive | 35 |
| Figure 21: LXC Settings | 36 |
| Figure 22: Configure XML Settings | 36 |
| Figure 23: Share the Drive | 37 |
| Figure 24: WRK Settings | 38 |
| Figure 25: CLG File | 39 |
| Figure 26: Config Manager..... | 40 |
| Figure 27: Config Tool | 40 |
| Figure 28: Config Tool | 41 |
| Figure 29: Deletemode | 41 |
| Figure 30: Interaction and Data Flow between the ENOVIA VPM V5 Client, PPRDaemon and PPRLoader | 44 |
| Figure 31: Registry Keys | 45 |
| Figure 32: Product Structure Mapping..... | 46 |

| | |
|---|-----|
| Figure 33: Set Attribute Description..... | 48 |
| Figure 34: Import | 49 |
| Figure 35: Product Structure | 56 |
| Figure 36: Integrate State..... | 59 |
| Figure 37: Removed State..... | 59 |
| Figure 38: Undefined State..... | 60 |
| Figure 39: Untouched State..... | 60 |
| Figure 40: Substitute Parts | 68 |
| Figure 41: Settings | 73 |
| Figure 42: Customization of the Database | 73 |
| Figure 43: Notify Mode | 77 |
| Figure 44: Library Subcomponentets | 78 |
| Figure 45: Deep Delete Mode | 78 |
| Figure 46: Customization..... | 79 |
| Figure 47: Set Corresponding Parent Object to Geometry..... | 81 |
| Figure 48: PRC in DPE..... | 82 |
| Figure 49: Select Edit from the Context Menu..... | 83 |
| Figure 50: Config Tool | 83 |
| Figure 51: Data Transfer | 84 |
| Figure 52: Partial Product Transfer | 85 |
| Figure 53: Project Structure..... | 86 |
| Figure 54: Product Transfer..... | 87 |
| Figure 55: Data Transfer | 90 |
| Figure 56: ENOVIA VPM V5 – PRC/Product Structure | 91 |
| Figure 57: Expoer Mode | 92 |
| Figure 58: Product Structure with COPS..... | 95 |
| Figure 59: PRC Structure | 96 |
| Figure 60: Product Class View | 97 |
| Figure 61: Child PRC1 Configured | 98 |
| Figure 62: ParentPRC Configured..... | 98 |
| Figure 63: ChildPRC1-COPS | 99 |
| Figure 64: Import of ParentPRC | 99 |
| Figure 65: Import of ChildPRC1 | 100 |
| Figure 66: Export of a PC Structure | 105 |
| Figure 67: New ENOVIA VPM V5 Root Node | 106 |

| | |
|--|-----|
| Figure 68: PRC Nodes within the PC Structure..... | 106 |
| Figure 69: Sub-PC Nodes within the Top-Level PC | 107 |
| Figure 70: PRC Import..... | 108 |
| Figure 71: PCs Transfer | 111 |
| Figure 72: Transfer of a PC Structure | 112 |
| Figure 73: MH Importing Side..... | 113 |
| Figure 74: Project EXAMPLE1_TRANSFER_OF_A_PC_STRUCTURE | 114 |
| Figure 75: Attributes “vpm_instanceid1 | 114 |
| Figure 76: Project EXAMPLE1_TRANSFER_OF_A_PC_STRUCTURE | 115 |
| Figure 77: Check the Creation of a top-Level Root Node..... | 116 |
| Figure 78: Project EXAMPLE3_PC_STRUCTURE | 117 |
| Figure 79: Project EXAMPLE1_TRANSFE_OF_ A_PC_STRUCTURE | 119 |
| Figure 80: New Line VPM Resource | 119 |
| Figure 81: Batch Transfer of a PC Structure | 120 |
| Figure 82: MH Importing Side..... | 120 |
| Figure 83: Project EXAMPLE5_BATCH_TRANSFER..... | 121 |
| Figure 84: vpm_instanceid1 | 122 |
| Figure 85: Siblings” in the Project’s Library | 123 |
| Figure 86: Secondary Formats | 124 |
| Figure 87: CATIA Reports | 124 |
| Figure 88: V5 Work Package document in the ENOVIA VPM V5 Client | 125 |
| Figure 89: Send To Work Package Report Generator | 126 |
| Figure 90: Comapre Contents | 128 |
| Figure 91: PRC with a COPS and two PS..... | 130 |
| Figure 92: Change Order with two affected Product Specifications | 130 |
| Figure 93: Config Manager..... | 131 |
| Figure 94: Log File..... | 132 |
| Figure 95: Relation between COs and CMs | 133 |
| Figure 96: COs and their Children COs..... | 133 |
| Figure 97: New Product Structure | 136 |
| Figure 98: Attribute Description of Part_ R17GPL375_1..... | 137 |
| Figure 99: Attribute Description of Part_ R17GPL375_3..... | 137 |
| Figure 100: Navigate in Project Library | 139 |
| Figure 101: Resulting Structure 1 | 140 |
| Figure 102: Resulting Structure 2..... | 140 |

| | |
|--|-----|
| Figure 103: Category with two Specifications and an Defined Alias | 142 |
| Figure 104: Visible Alias | 143 |
| Figure 105: Specification Alias used in Product Specifications and Filters | 143 |
| Figure 106: Changes Visible in the MH | 143 |
| Figure 107: Model Configuration of a Filter | 144 |
| Figure 108: Transfer Settings | 147 |
| Figure 109: XML Settings | 147 |
| Figure 110: Registry Keys | 148 |
| Figure 111: Transfer Settings | 150 |
| Figure 112: Reference Import..... | 152 |
| Figure 113: Effectivity Term in EH..... | 154 |
| Figure 114: Effectivity Representation in MH as of R15 | 154 |
| Figure 115: Effectivity Representation in MH as of R16 | 155 |
| Figure 116: Exposure of Configuration in both Products after the Transfer | 156 |
| Figure 117: Configuration File | 157 |
| Figure 118: ENOVIA Portal | 160 |
| Figure 119: Imported Product Structure | 161 |
| Figure 120: Engineering Effectivities | 161 |
| Figure 121: Imported Product Structure Unchanged Structure | 162 |
| Figure 122: Manufacturing Effectivities | 162 |
| Figure 123: Resulting Structure..... | 164 |
| Figure 124: Parts Filtering | 166 |
| Figure 125: Filter File..... | 167 |
| Figure 126: Config Template | 168 |
| Figure 127: New Instance..... | 177 |
| Figure 128: Alternate Parts..... | 177 |
| Figure 129: Add a Supplementary Document | 178 |
| Figure 130: Product Structure transferred over the Bridge into the Manufacturing Hub..... | 179 |
| Figure 131: Correct Handling of the Versions | 179 |
| Figure 132: Multiline Effectivity Attribute Value | 181 |
| Figure 133: Tokens are generated, special characters are kept | 184 |
| Figure 134: Tokens are generated, but special characters are removed | 185 |
| Figure 135: Tokens are generated, but special characters are removed. I.e. setting “cfgnormal” rules over “cfgextended” | 185 |
| Figure 136: Tokens are not generated | 185 |
| Figure 137: Tokens are generated, but special characters are removed | 185 |

| | |
|--|-----|
| Figure 138: Tokens are generated, but special characters are removed | 186 |
| Figure 139: Import and Selection of the Requested Bitmap | 189 |
| Figure 140: Change the Control Type | 189 |
| Figure 141: Special Attributes | 190 |
| Figure 142: Set entry Depends on values to Yes | 190 |
| Figure 143: Result of the Configuration | 190 |

List of Tables

| | |
|---|-----|
| Table 1: Customization File's Permission Table..... | 134 |
| Table 2: Assign Effectivities..... | 159 |
| Table 3: DELMIA – ENOVIA VPM V5 Connection Update Protocol Values | 188 |
| Table 4: Export Environment Variables | 191 |
| Table 5: Enable/Disable Feature | 191 |
| Table 6: Environment variables to (re-)set strings in the export code | 191 |
| Table 7: Environment Settings | 192 |
| Table 8: CGR Export | 193 |
| Table 9: Configuration Settings - ProductDataGen | 205 |
| Table 10: Registry Settings | 210 |

Index

A

| | |
|---------------------------|-----|
| Advanced Properties | 187 |
| approved | 68 |
| Attribute Length | 64 |

B

| | |
|-----------------------------|-----|
| Batch mode | 7 |
| pprloader –complete..... | 10 |
| pprloader –incremental..... | 10 |
| pprloader –partial | 10 |
| pprloader –project | 9 |
| pprloader –reference | 10 |
| Start PPRLoader | 7 |
| Best-So-Far (BSF) | 100 |
| Bitmap | 187 |

C

| | |
|--------------------------------------|----------|
| Calculation Model | 183 |
| CGR Settings | 34 |
| Code Rules..... | 152 |
| config file | 165 |
| filter config file | 165 |
| Config Handler | 183 |
| Config Matrices | 183 |
| Config Rules..... | 183 |
| Configuration | |
| CGR Settings | 34 |
| LXC Settings | 35 |
| XML Settings..... | 37 |
| Configuration File | 65 |
| COPS | 93 |
| Customer Specific File Markings..... | 140 |
| Customization file | 134, 135 |

D

| | |
|-----------------------------------|-----|
| Database Classes and Attributes | |
| Advanced Properties | 187 |
| Attribute & Bitmap Selection..... | 187 |
| Properties | 187 |
| Special Attributes | 187 |
| Deep Delete | 77 |
| Limitations | 78 |
| Deleted objects..... | 75 |
| auto | 75 |
| deep | 75 |
| notify..... | 75 |
| Depends on Values | 189 |
| dictionary object | 183 |

Document

| | |
|---------------------------------|-----|
| Attachment Mapping..... | 70 |
| Object Mapping..... | 70 |
| DPE Configuration Manager | 186 |

E

| | |
|---|-----|
| EAR Filter | |
| Limitations | 135 |
| EAR Filter | |
| Creating/Editing the Customization file | 135 |
| Defining the path to the Customization file | 134 |
| ECCN | 133 |
| EAR Filter..... | 133 |
| Effectivity..... | 181 |
| ENOVIA Filter for EAR Data | 133 |
| ENOVIA VPM V5 | |
| dictionary object..... | 183 |
| ENOVIA Work Packages | 123 |
| ergocompproductdefault | 188 |
| Export of Multiple PRCs in One Call | 101 |
| Limitations | 102 |
| Export of Product Classes..... | 103 |
| Limitations | 110 |
| Extended Filtering on Export..... | 162 |
| Extended Part Filtering..... | 51 |

F

| | |
|---|-----|
| File Transfer via HTTPS, HTTP or FTP | 144 |
| filter config | |
| template | 167 |
| filter config file | 165 |
| registry settings..... | 166 |

G

| | |
|----------------------------------|----|
| General Mapping Information..... | 21 |
|----------------------------------|----|

I

| | |
|--------------------------|----|
| Incremental Update | 89 |
| Initial Import..... | 81 |

L

| | |
|---------------------|-----|
| List of Values..... | 187 |
|---------------------|-----|

M

| | |
|-------------------------|----|
| Multi Environment | 45 |
|-------------------------|----|

| | |
|---|--------|
| Multi Instancing | 55 |
| Multi Value Attributes | 71 |
| Multiple effectivity domains | 156 |
| Multiple Parts Filtering on Import..... | 165 |
| MVA..... | 71, 72 |

N

| | |
|--------------------------|----|
| New Functions | |
| General Information..... | 3 |
| Nonliability | ii |

O

| | |
|----------------------|----|
| Object ID (OID)..... | 55 |
| ORBIX | 7 |

P

| | |
|------------------------------------|---------------|
| Partial Product Data Transfer..... | 83 |
| PPRDaemon | 7, 33, 43, 44 |
| backbonereceiver..... | 33 |
| backbonesender..... | 33 |
| PPRListener | 43 |
| PPRLoader | |
| Configuration Settings | 33 |
| Configure the PPRDaemon..... | 33 |
| Settings | 38 |
| Set Attribute Mappings | 38 |
| PRC | 81 |
| Prerequisites | 6 |
| Client and Server Side | 6 |
| Client Side | 6 |
| Server Side | 6 |
| Priority of Update Status | 74 |
| Product Root Class | 81 |
| Product Structure | 22 |
| Product Structure Mapping..... | 42 |

R

| | |
|--|-----|
| Recalculation of absolute instance positions | 78 |
| Reference Import..... | 151 |

S

| | |
|--------------------------------|-----|
| Secure Registry Settings | 148 |
| TRANSFER Settings | 148 |

| | |
|--------------------------------|-----|
| Significant attributes | 74 |
| Spares Configuration Data..... | 175 |
| Special attribute | 187 |
| Special Attribute | |
| Updatestate | 187 |
| Substitute parts | 67 |
| PrimeUUID..... | 67 |
| V_issubstitute | 67 |
| V_nbsubstitute | 67 |

T

| | |
|--------------------------------|-----|
| Tailnumber | 181 |
| Token Group | 183 |
| Top Level | 81 |
| Top Level Product Folder | 81 |
| TRANSFER Settings..... | 148 |
| Type Mapping | |
| FAQ | 50 |
| Import | 49 |
| Limitations | 49 |

U

| | |
|------------------------------------|------------|
| Update protocol..... | 39, 40, 73 |
| Update Protocol | 79 |
| updatestate | 75 |
| Updatestate | 73 |
| Update Protocol for Geometry | 79 |
| Update Status | 74 |
| Update Status for links | 74 |
| updatestate | 188 |

V

| | |
|----------------|----|
| V_status | 51 |
|----------------|----|

W

| | |
|------------------------|-----|
| WIP | 68 |
| work in progress | 68 |
| work in progress..... | 68 |
| Work Packages | 123 |

X

| | |
|-----------------------------------|-----|
| XML Product Structure Files | 144 |
| Binary Format | 144 |