# Installation and Administration Guide

Siemens Xpedition Enterprise integration
for
SAP Engineering Control Center

Valid for product version: 1.1.7.0

Published: 19.05.2025 | Build: 887 | Revision: 3df4c5f02

---

XPLM Solution GmbH

Altmarkt-Galerie

Altmarkt 25

01067 Dresden

Office: +49 351 82658-0

Fax: +49 351 82658-88

Web: www.xplm.com

# Legal information

## Non-disclosure

The information in this document is to be treated confidentially.

## Licensing

XPLM Solution GmbH (XPLM) does not issue licenses for the software components to be integrated. These licenses must be obtained from the respective software manufacturer.

## Support

In the event of problems with the integration software, please check the knowledge base at https://support.sap.com, or create a support ticket.

## Handling personal data

Against the background of the EU General Data Protection Regulation (GDPR), XPLM informs about the processing of personal data in its contracts and privacy policies. The extent to which XPLM comes into contact with personal data from the customer environment while providing services depends on the customer's IT systems and the data transmitted to XPLM. XPLM is generally not interested in gaining access to personal data from the customer environment. If this cannot be avoided, the customer is responsible for such personal data and must enter into a GDPR-compliant data processing agreement with XPLM.

**Information exchange**: The integration software exchanges information between the systems involved in the form of attributes. What is contained in these attributes is left to the customer's definition. For ECAD integrations, the exchanged attribute information is saved in the file `metadata.xml` in the configured project directory or directly in the project files.
For other integrations, the exchanged attribute information is saved in files or objects of the source system.

**Log files**: To record activities of integration functions, the integration software stores log files on the computer on which it is installed. These log files can also contain information from Windows environment variables, for example the user and computer name. For ECAD integrations, log files are stored in the directory `%TEMP%\integrate` by default. This directory can be deleted, but new log files will be created when the integration software is used again. For other integrations, logging is deactivated by default. Activation, protocol level and storage location must be set up in advance. Directories with log files can be deleted, but new log files will be created when the integration software is used again.

**Other policies**: XPLM also refers to the SAP document Security Guide for Document Management (DMS), as well as to the privacy policies of the software manufacturers involved for the handling of personal data in their systems.


© 2025 XPLM Solution GmbH

# Table of Contents

**Legal information** ..... ii

**Glossary** ..... 7

**1 Introduction** ..... 10

1.1 About this document ..... 10

1.2 Naming conventions ..... 11

1.3 What's new ..... 12

**2 Setup overview** ..... 13

**3 Integration essentials** ..... 14

3.1 Basics ..... 14

3.1.1 Integration architecture ..... 14

3.1.2 Directory structure ..... 15

3.1.3 Integration variables ..... 16

3.1.4 Datamodel in SAP ECTR ..... 19

3.2 About configuration ..... 20

3.2.1 Important configuration files ..... 20

3.2.1.1 Other configuration files ..... 23

3.2.2 Restarting integration modules or IVS ..... 24

3.2.3 Working with different integration environments ..... 25

3.2.4 Encrypting passwords ..... 25

3.3 About parts synchronization ..... 26

3.3.1 Different ways for parts synchronization ..... 26

**4 System requirements** ..... 27

4.1 Required integration licenses ..... 29

4.2 Required installation media ..... 29

**5 Installation** ..... 30

5.1 Installing integration ..... 30

5.2 Siemens Xpedition Enterprise ..... 31

5.2.1 Installing integration menu in Xpedition Designer ..... 31

5.2.2 Installing PDFCreator printer ..... 31

5.3 SAP Engineering Control Center ..... 33

5.3.1 Activating BC set ..... 33


© 2025 XPLM Solution GmbH

**6 Initial setup ..... 34**
6.1 Siemens Xpedition Enterprise ..... 34
6.1.1 Setting up delimiter for Variant Manager ..... 34
6.1.2 Setting up local project and template directories ..... 34
6.1.3 PartSync 1: Setting up library connection ..... 34
6.1.4 PartSync 1: Setting up library database format ..... 36
6.1.5 PartSync 1: Setting up material number in library ..... 36
6.1.6 Creating templates ..... 38
6.2 SAP Engineering Control Center ..... 39
6.2.1 Setting up DTypes ..... 39
6.2.2 Updating DType settings ..... 39
6.2.3 Setting up status net ..... 39
6.2.4 Setting up material ..... 41
6.2.5 PartSync 1: Setting up classification for parts synchronization ..... 42
6.2.6 PartSync 2: Setting up SAP ECTR for alternative parts synchronization ..... 42
**7 Configuration ..... 44**
7.1 IVS & UI ..... 44
7.1.1 Configuring UI under Java 8 ..... 44
7.1.2 Configuring module flows ..... 44
7.1.3 Hiding design path in UI ..... 46
7.1.4 Hiding labels for buttons ..... 47
7.1.5 Hiding output file options ..... 47
7.1.6 Terminating IVS via batch file ..... 47
7.1.7 Configuring default text editor ..... 48
7.2 Datamodel objects & output files ..... 48
7.2.1 Configuring main design file ..... 48
7.2.2 Configuring object names ..... 48
7.2.3 Configuring new attributes for design objects ..... 49
7.2.4 Configuring appearance of object title ..... 50
7.2.5 Configuring attribute field lengths ..... 50
7.2.6 Configuring file names of output files ..... 50
7.2.7 Configuring extra files for upload ..... 51
7.2.8 Configuring own extractions in extra files ..... 51
7.3 BOM and assemblies ..... 53
7.3.1 Excluding or including BOM items ..... 53


© 2025 XPLM Solution GmbH

7.3.2 Configuring BOM report ..... 54
7.3.3 Configuring common variant ..... 55
7.3.4 Configuring columns and mappings in BOM comparison ..... 55
7.3.5 Configuring validation of BOM items ..... 56
7.3.6 Configuring BOM report ..... 57
7.3.7 Configuring BOM representation ..... 58
7.3.8 Configuring additional BOM from CSV ..... 59
7.4 Parts synchronization ..... 59
7.4.1 PartSync 1: Configuring new attributes for parts synchronization ..... 59
7.4.2 PartSync 1: Configuring additional attributes for quick search ..... 61
7.4.3 PartSync 1: Configuring synchronization direction ..... 62
7.4.4 PartSync 1: Configuring start dialog in Synchronize parts ..... 62
7.4.5 PartSync 1: Configuring ECAD/PLM labels in Synchronize parts ..... 62
7.5 Siemens Xpedition Enterprise ..... 64
7.5.1 Configuring colors in schematic PDF ..... 64
7.5.2 Configuring print options for fabrication PDFs ..... 64
7.5.3 Configuring PDFCreator printer ..... 64
7.5.4 Configuring extended print functionality ..... 65
7.5.5 Configuring Gerber data ..... 66
7.5.6 Configuring STEP data ..... 66
7.5.7 Configuring IPC-356 data ..... 68
7.5.8 Configuring CCZ/CCE data ..... 68
7.5.9 Configuring DXF data ..... 68
7.5.10 Configuring assembly data ..... 69
7.5.11 Configuring BOM export ..... 69
7.5.12 Configuring non-fitted components ..... 70
7.6 SAP Engineering Control Center ..... 71
7.6.1 Configuring creation mode for datamodel objects ..... 71
7.6.2 Saving templates ..... 71
7.6.3 Configuring templates ..... 72
7.6.4 Configuring alternative BOM items ..... 72
7.6.5 Configuring handling of BOM items in SAP ..... 73
7.6.6 Bypassing SAP license mechanism ..... 74
7.6.7 Distributing integration with Subversion ..... 75
7.6.8 Working with backend configuration support ..... 76


© 2025 XPLM Solution GmbH

7.6.9 Configuring ECN usage ..... 77
**8 Update & removal ..... 78**
8.1 Updating integration ..... 78
8.2 Uninstalling integration ..... 78
**9 Troubleshooting ..... 79**
9.1 Solving integration issues ..... 79
9.2 Solving blacklisted BAPIs in SAP S/4HANA ..... 80
**10 Reference information ..... 81**
10.1 Configuration options ..... 82


© 2025 XPLM Solution GmbH

# Glossary

**Application Programming Interface (API)**

Defines a set of routines, communication protocols and tools for building software. In general, they are clearly defined methods for communication between different components.

**Assembly**

Defines a unique identifier in the design to group components according to functional or other characteristics. Each assembly in Xpedition is represented in the datamodel in SAP ECTR as a combination of document and material. The document saves related output files for the placement of the PCB and the material saves the BOM.

**Assembly document**

Defines a datamodel object in SAP ECTR that saves the placement information for the assembly.

**Assembly part**

Defines a datamodel object in SAP ECTR with the BOM items of the assembly.

**Assembly schematic**

Defines an output file with the schematic derived from the assembly as a PDF file. Is saved in SAP ECTR for the defined datamodel object.

**Attribute**

Defines metadata in SAP ECTR that describes object properties, for example the object number or the object name.

**Base schematic**

Defines an output file with the base schematic as a PDF file. Is saved in SAP ECTR for the defined datamodel object.

**Bill Of Material (BOM)**

Defines the combined components or materials of an assembly in the design or in SAP ECTR.

**BOM position**

Defines a position in the BOM with unique identification, name, quantity and other characteristics.

**Component**

Defines an individual device in the design and in the library, for example a capacitor, resistor, or transistor.

**Connector**

Defines a central interface component of each XPLM integration. The integration uses connectors for each participating application to exchange data via their API.

**Datamodel**

Defines objects and their relationships in a PLM system that are managed by the integration to store data from an authoring application.

**Design**

Defines a project file in Xpedition containing schematics and PCB layouts, including related files.

**Design container**

Defines an output file with the native design data. Is saved in SAP ECTR for the defined datamodel object.

**Design document**

Defines a datamodel object in SAP ECTR that saves the design data from Xpedition.

**Drill drawing**

Defines an output file with the drill drawing for the PCB as a PDF file. Is saved in SAP ECTR for the defined datamodel object.


© 2025 XPLM Solution GmbH

**Dynamic Link Library (DLL)**

Defines a file with a library of functions and other information that can be accessed by a Windows program.

**Electronic Computer Aided Design (ECAD)**

Defines the software-supported development in electrical engineering and electronic design.

**Fabrication document**

Defines a datamodel object in SAP ECTR that saves the PCB fabrication information.

**Fabrication part**

Defines a datamodel object in SAP ECTR with the bare board as BOM item.

**Gerber data**

Defines an output file with the Gerber data for fabricating the PCB. Is saved in SAP ECTR for the defined datamodel object.

**IDF data**

Defines an output file with the proprietary design and fabrication data for the PCB. The IDF format is used to enable data exchange from ECAD to MCAD. Is saved in SAP ECTR for the defined datamodel object.

**IPC-2581 data**

Defines an output file with the proprietary design and fabrication data for the PCB. Is saved in SAP ECTR for the defined datamodel object.

**Mapping**

Defines the assignment of fields between Xpedition and SAP ECTR to enable a data exchange. For example, in Xpedition the component number can be defined by the parameter *Part Number*. In SAP ECTR, however, the object number is defined by the attribute *MATERIAL_EXT*. In order for the integration to know which information is written from which parameter in Xpedition to which attribute in SAP ECTR (or vice versa), the fields have to be connected in a mapping definition.

**Material**

Defines a material or product in SAP ECTR that is either produced or purchased.

**ODB++ data**

Defines an output file with the proprietary design and fabrication data for the PCB. Is saved in SAP ECTR for the defined datamodel object.

**Parameter**

Defines metadata in the design that describes properties for components or the design, for example the component number or the project name. Often, parameters from the design are used to fill the title block dynamically with values.

**Payload**

Defines the data contained within an API request. The description is borrowed from the transportation industry, where a truck carries its cargo (its payload) to a location. The truck, as with the API request, is always the same, but the payload changes with each request.

**Pick & place data**

Defines an output file with the placement data for the PCB derived from the assembly. Is saved in SAP ECTR for the defined datamodel object.

**Placement drawing**

Defines an output file with the placement drawing for the PCB derived from the assembly as a PDF file. Is saved in SAP ECTR for the defined datamodel object.


© 2025 XPLM Solution GmbH

**Printed Circuit Board (PCB)**

Defines a bare board for the electrical connection of electronic components. The PCB is represented in the datamodel in SAP ECTR as a combination of document and material. The document saves the output files for fabricating the PCB and the material defines the PCB as BOM item.

**Product Lifecycle Management (PLM)**

Defines systems and processes for managing data during the development of a product from creation through manufacturing to maintenance and disposal.

**Revision**

Defines a released object state in SAP ECTR that cannot be modified.

**Schematic**

Defines a diagram with electrical or electronic circuits and connections.

**STEP data**

Defines an output file with the proprietary design and fabrication data for the PCB. The STEP format is used to enable data exchange between ECAD and MCAD. Is saved in SAP ECTR for the defined datamodel object.

**User Interface (UI)**

Defines a (usually) graphical interface through which a user interacts with the computer.

**Version**

Defines an incremental counter of each object modification in SAP ECTR on check-in.

**x86/x64**

Defines the processor architecture in a computer and thus also the performance of applications. x86 corresponds to 32-bit and x64 corresponds to 64-bit.


© 2025 XPLM Solution GmbH

# 1 Introduction

## 1.1 About this document

You are reading the Installation and Administration Guide of the Siemens Xpedition Enterprise integration for SAP Engineering Control Center.

### Purpose

This document describes how to install, set up and configure the integration.

### Target audience

This document is intended for system administrators in the company who are responsible for the operation of the integration. In addition, system administrators should also read the User Guide to familiarize themselves with the user interface and integration functions.

### How to read this document

This document is structured chronologically and you should read it in the order of the chapters described. If you skip chapters, you will miss important information.

Where appropriate, cross-references to other chapters are listed. To quickly return to where you came from after clicking such a link, click the **Back** button in your PDF viewer. Try it right now with this link!

### Notes used



This note highlights additional information about the current content.



This note highlights important instructions.



This note highlights the integration version from which a particular feature or change was introduced or updated.

### Line breaks in code examples



In this document, long lines in code examples are sometimes manually wrapped to represent readable content. The blue arrow ← at the end of a line of code indicates such a manual line break. When you copy code from this document, remove the arrow and realign the lines.

### Searching for text strings

Long text strings, for example variables, are sometimes manually wrapped in this document to fit in a table cell. When you search for such terms in this document, Acrobat Reader and Acrobat cannot interpret a line break, which results in the entire term not being found. Therefore, always open the PDF in one of the above browsers and search there.


© 2025 XPLM Solution GmbH

### Copying code

You can read this document in all supported PDF viewers, including PDF browser plugins. Check the following recommendations about copying code and searching for text strings.

```
property('Name') {
  eda "INNO_NAME"
  plm "name"
}
```

Not all PDF viewers support proper copying of code. If you open this document in Firefox, Acrobat Reader, or Acrobat and copy the above code, the lines will be pasted without indents and even in one line only, depending on the version of the PDF viewer.

Browsers that copy code correctly are Chrome, Edge and Brave. Keep your browsers and Acrobat tools always up to date.

## 1.2 Naming conventions

In addition to the terms already listed in the glossary, the following naming conventions are used in this document. Familiarize yourself with these names. You will find them throughout the document.

| Form used | Generally refers to |
|---|---|
| SAP ECTR [1] | SAP Engineering Control Center |
| Xpedition | Siemens Xpedition Enterprise |
| Material (p. 8) | Material or product in SAP ECTR |
| Attribute (p. 7) | Object property in SAP ECTR |
| Component (p. 7) | Library part in Xpedition |
| Parameter (p. 8) | Component property in Xpedition |

[1] This name is also used to describe actions for managing data in SAP.


© 2025 XPLM Solution GmbH

## 1.3 What's new

Take a look at how the new integration version has affected documentation since the last release. See the official release notes for a complete list.

### Version 1.1.7.0

| Chapter | Description |
|---|---|
| * | General housekeeping on wording and consistency. |
| Setup overview (p. 13) | Renamed chapter *Quick start reference* to *Setup overview*.<br>Reworked whole chapter for an easier entry point to the integration setup. |
| Integration essentials (p. 14) | ■ Added chapter to combine important integration information in one place.<br>■ Updated the architecture diagram to make it clearer what protocols are used and in which programming language components are written.<br>See Integration architecture (p. 14) for more information. |
| System requirements (p. 27) | Updated versions. |
| Installation (p. 30) | Combined all main installation tasks in one place. This makes it easier to complete all required application-specific tasks before setting up the integration. |
| Initial setup (p. 34) | Restructured chapters to include only tasks that are relevant for an out-of-the-box scenario. |
| Configuration (p. 44) | ■ Restructured chapters for a better alignment with *Initial setup*.<br>■ OOTB-1545: Added new snippets to select and upload existing files based on different conditions.<br>See Configuring extra files for upload (p. 51) for more information.<br>■ OOTB-1262: Added the snippet `combine_bom_sources.snippet` to configure an additional BOM from a CSV for upload.<br>See Configuring additional BOM from CSV (p. 59) for more information.<br>■ OOTB-1464: Edit: Added the new flow option `:make_design_readonly` to define the read-only behavior when opening a design in view mode.<br>See Configuring module flows (p. 44) for more information<br>■ ECTR-2208: Improved working with ECNs in SAP ECTR. With the new setting `:change_number_source`, you can control how ECNs are used:<br>**1.** Use the active ECN in SAP ECTR (default).<br>**2.** Show a selection in the *Save* dialog.<br>**3.** Use the ECN already assigned to a document (if available).<br>See Configuring ECN usage (p. 77) for more information. |


© 2025 XPLM Solution GmbH

# 2 Setup overview

This overview serves as the starting point for the integration setup.

### Step 1: Preparation

The following chapters help you to understand key aspects of the integration, verify system requirements and to organize licenses and installation media.

- Integration essentials (p. 14)
- System requirements (p. 27)
- Required integration licenses (p. 29)
- Required installation media (p. 29)

### Step 2: Installation

Complete all installation tasks in chapter Installation (p. 30). This will ensure that the integration is installed correctly, menus for accessing integration functions are set up the applications are prepared for integration use.

### Step 3: Initial setup

Complete all setup tasks in chapter Initial setup (p. 34). This will ensure that the integration is prepared for data exchange between the applications.

### Step 4: Configuration

To fine-tune the integration, complete individual tasks in chapter Configuration (p. 44).

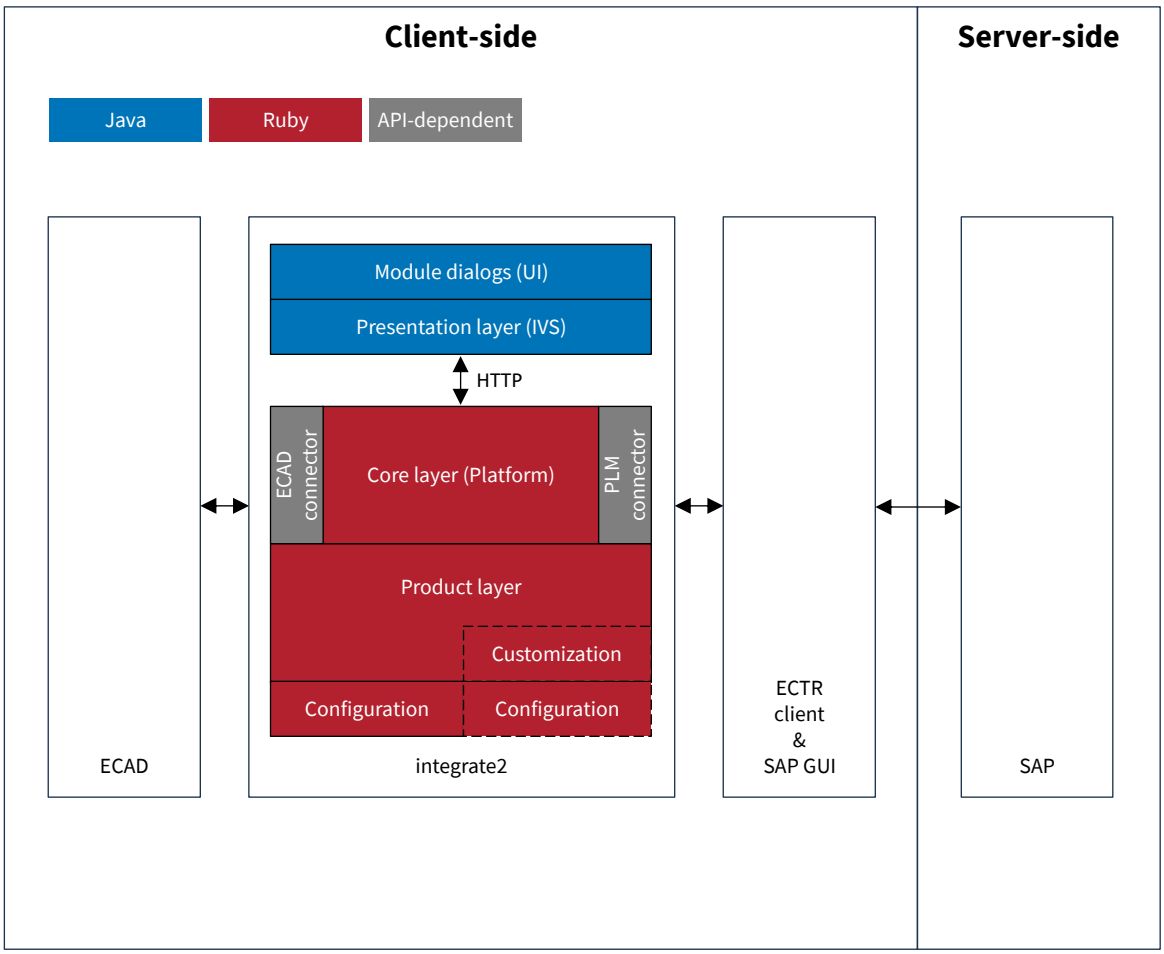

© 2025 XPLM Solution GmbH

# 3 Integration essentials

## 3.1 Basics

### 3.1.1 Integration architecture

The integration consists of several software components that interact with each other.




- **Module dialogs (GUI)**: The dialogs of the individual integration modules represent the user interface of the integration.
- **Presentation layer (Integration Visualization Server, IVS)**: IVS is responsible for presenting the module dialogs and storing general settings for all modules, for example the default log level. IVS also manages window sizes and positions between executions, as well as the history of executions with easy access to log files.
- **Core layer**: This layer includes the basic integration functionality and the connectors.
- **ECAD/PLM connector**: The connectors enable the data exchange with Xpedition and SAP ECTR.
- **Product layer**: This layer includes the product-specific features, configuration and customization.


© 2025 XPLM Solution GmbH

### 3.1.2 Directory structure

After installing the integration, the following directory structure will be created.

```
<ECTR DIR>
├──addons
│   └──ecad
└──applications
    └──xdx
        ├──appdata
        ├──basis
        │   └──dictionary
        ├──customize
        │   └──config
        │       └──environments
        ├──documentation
        ├──template-customize
        ├──template-templates
        └──templates
```

- `<ECTR DIR>`: Defines the default SAP ECTR installation directory, for example `C:\Program Files (x86)\SAP\ECTRS4`.
- `addons\ecad`: Contains core files for the integration platform.
- `applications\xdx`: Defines the product directory of the integration with specific configuration files. This directory is referenced as `<PRODUCT DIR>` from now on.
- `<PRODUCT DIR>\appdata`: Contains files to set up the integration.
- `<PRODUCT DIR>\basis`: Contains scripts for starting the integration modules.
- `<PRODUCT DIR>\basis\dictionary`: Contains language files for the localization of the user interface.
- `<PRODUCT DIR>\customize\config`: Contains the main configuration files for this integration.
- `<PRODUCT DIR>\customize\config\environments`: Contains specific integration settings for a defined environment.
- `<PRODUCT DIR>\documentation`: Contains the product documentation for this integration.
- `<PRODUCT DIR>\template-customize`: Defines the configuration directory as delivered. During an integration update, new configuration files are placed in this directory to not overwrite existing configuration in `xdx\customize\config`.
- `<PRODUCT DIR>\templates`: Defines the template directory to store local design templates.
- `<PRODUCT DIR>\template-templates`: Defines the template directory as delivered. During an integration update, new template files are placed in this directory to not overwrite existing templates in `<PRODUCT DIR>\templates`.


© 2025 XPLM Solution GmbH

### 3.1.3 Integration variables

The integration uses different variables for the execution.

#### Main integration variables

The installer has created the following integration variables under Windows:

| Variable name and definition |
| --- |
| `INTEGRATE_HOME`<br>Defines the path to the application directory `<ECTR DIR>\addons\ecad\basis\sys\win\bin`. |
| `INTEGRATE_SAP_ECTR_XPEDITION_PRODUCT_HOME`<br>Defines the path to the product directory `<PRODUCT DIR>\customize`. |

#### Other integration variables

You can manage the following integration variables in the batch file `init_env.bat`.

Alternatively, define them as environment variables under Windows.

How and where you want to manage the variables is up to you, but keep it the same everywhere.

If you set variables in a batch file, use the following syntax and always place the definition at the end of the file (if not already available):

```
SET <variable>=<value>
```

Example

```
SET INTEGRATE_TEMP_DIR=C:\temp
```

| Variable name and definition |
| --- |
| `INTEGRATE_SAP_ECTR_XPEDITION_ENVIRONMENT`<br>Defines the current environment and is set by default to `production`. An environment can for example describe a development or a production system with different settings. See Working with different integration environments (p. 25) for more information.<br>The batch file `init_env.bat` in the directory `<PRODUCT DIR>\config\environments` typically contains integration variables with paths to templates, the library or the project directory.<br>All files that refer to an environment are called environment files. |
| `INTEGRATE_JAVA_HOME`<br>The integration uses the JRE included in SAP ECTR. You can manually change the path with this variable if needed. |
| `LIBDB_CONNECTION_STRING` (in environment file)<br>`INTEGRATE_SAP_ECTR_XPEDITION_LIBRARY` (Windows)<br>Defines the connection to the library.<br>See PartSync 1: Setting up library connection (p. 34) for more information. |


© 2025 XPLM Solution GmbH

| Variable name and definition |
| --- |
| PROJECT_DIR<br><br>Defines the path to the project directory, for example C:\projects.<br><br>See Setting up local project and template directories (p. 34) for more information. |
| INTEGRATE_SCH_TEMPLATE_DIR<br><br>Defines the path to the template directory for schematics. Both drive letters and UNC paths are supported.<br><br>See Setting up local project and template directories (p. 34) for more information. |
| INTEGRATE_PCB_TEMPLATE_DIR<br><br>Defines the path to the template directory for layouts. Both drive letters and UNC paths are supported.<br><br>See Setting up local project and template directories (p. 34) for more information. |
| INTEGRATE_LANG<br><br>Defines the UI language in the integration. The following languages are supported:<br><br>■ English: Language code "en"<br>■ German: Language code "de"<br>■ French: Language code "fr"<br>■ Italian: Language code "it"<br>■ Spanish: Language code "es"<br>■ Portuguese: Language code "pt"<br>■ Japanese: Language code "ja"<br>■ Russian: Language code "ru"<br>■ Simplified Chinese: Language code "zh_CN"<br>■ Traditional Chinese: Language code "zh_TW"<br><br>By default, the language is set to English. For another language, set the variable to the exact value of the language code using the methods described.<br><br>You can find the localization files for the UI in <PRODUCT DIR>\basis\dictionary\<language code>. |
| INTEGRATE_TEMP_DIR<br><br>Defines the default directory %TEMP%\integrate with the temporary log files and other settings.<br><br>You can define a different temporary directory using the methods described. Restart IVS after changing this variable. |
| INTEGRATE_DEBUG \| INTEGRATE_DEEP_DEBUG<br><br>Defines the level of debugging output in the log files.<br><br>These two settings are defined in init_env.bat. The settings are disabled by default, but you can activate them there. INTEGRATE_DEBUG corresponds to log level **Debug** and INTEGRATE_DEEP_DEBUG corresponds to the highest level **Trace**.<br><br>As a preferred solution, you can control log levels also in the running integration module or in the IVS preferences. See User Guide for more information. |


© 2025 XPLM Solution GmbH

| Variable name and definition |
| --- |
| PRE_CLASSPATH<br>Defines the loading of third-party JAR classes before the integration classes. This can be used to meet certain technical requirements, for example in the SSO area. The variable is not set by default. |

### Application priorities of integration variables

Depending on where variables are set, they have priority in their application and override other locations.

At startup, the integration reads possible variables in this order:

1. Environment variables in Windows under *System*
2. Environment variables in Windows under *User*
3. `init_env.bat`

This results in the following application priority:

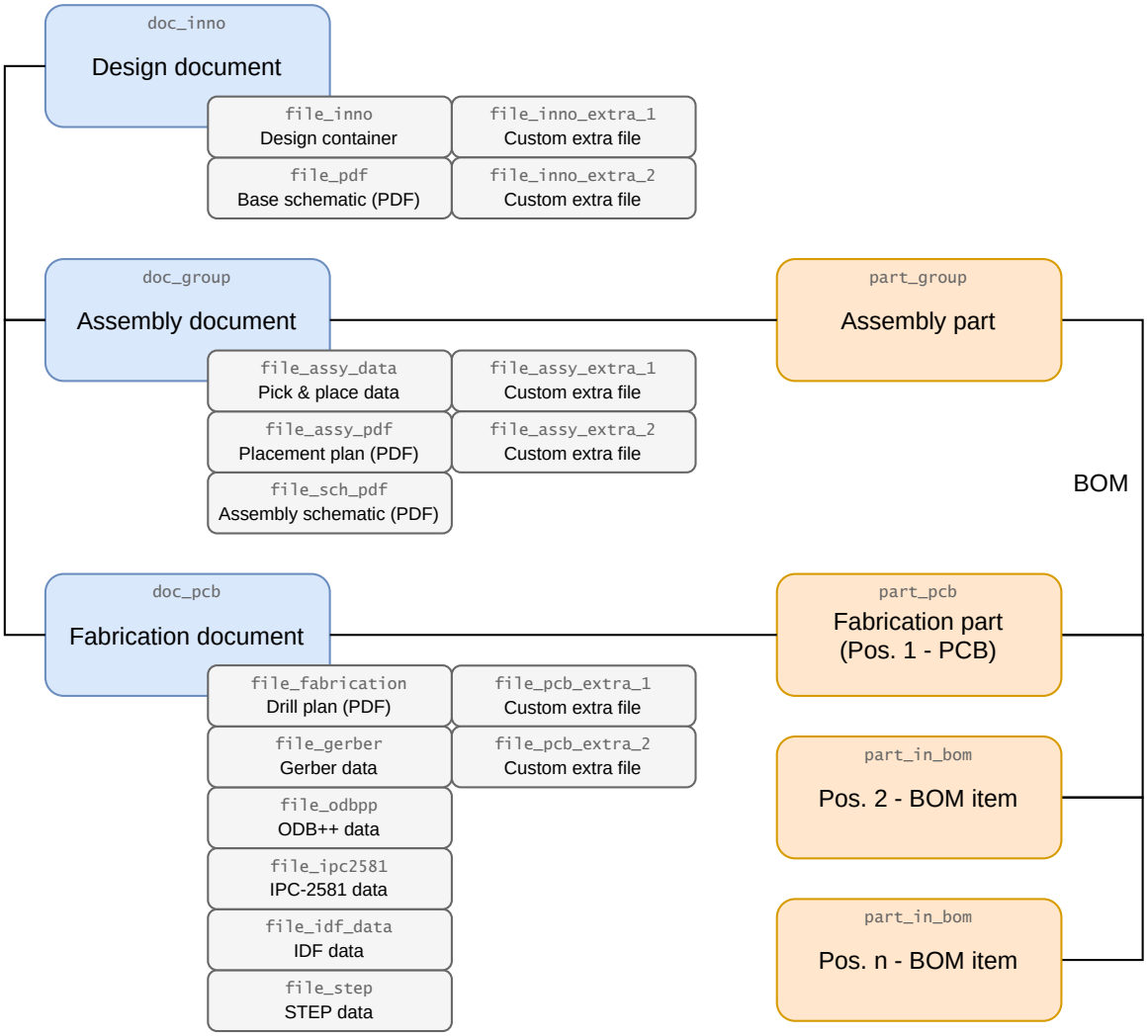
1. `init_env.bat`
2. Environment variables in Windows under *User*
3. Environment variables in Windows under *System*

**Related links**
Important configuration files (p. 20)
Restarting integration modules or IVS (p. 24)


© 2025 XPLM Solution GmbH

### 3.1.4 Datamodel in SAP ECTR

The datamodel defines the objects in SAP ECTR, which are used by the integration to manage design data and BOM(s).




- Blue boxes: Document
- Orange boxes: Material
- Grey boxes: Original (output file)
- Connection lines: Object relations

Extra files ( `file_*_extra_*` ) are disabled in the datamodel by default but can be enabled and configured for customer-specific outputs.

For each document, an additional `.system` file is saved in SAP ECTR. This file is used to support SAP ECTR system requirements, but is not relevant for the user.


© 2025 XPLM Solution GmbH

# 3.2 About configuration

## 3.2.1 Important configuration files

The main configuration is located in the directory `<PRODUCT DIR>\config` and defines the data exchange between the applications.

### Scope of datamodel configuration

In the datamodel configuration you typically maintain:

- General field definitions in Xpedition and SAP ECTR
- Field mapping between Xpedition and SAP ECTR
- Definitions of status actions in SAP ECTR
- Creation of output files
- Configuration of BOM processing
- Parts synchronization between Xpedition SAP ECTR

### Configuration files

| Processing order | Configuration file | Description |
|---|---|---|
| 1 | `release.rb` | This file contains basic settings of all datamodel objects.<br>This file is not a common configuration file and you should not make any changes to it unless specifically mentioned. |
| 2 | `datamodel.config` | This file contains product-specific enhancements to the datamodel objects. Entries in this file override or extend the file `release.rb`. |
| 3 | `classification.config` | This file contains product-specific mapping rules for parts synchronization. Entries in this file override or extend the files `datamodel.config` and `release.rb`. |


© 2025 XPLM Solution GmbH

### Configuration sections in datamodel.config

In the file `datamodel.config`, you can change or extend the default settings for the datamodel objects. The following table explains the available configuration sections.

| Configuration section | Description |
|---|---|
| bom_source | Defines settings and properties for the BOM attributes when exporting the BOM from Xpedition. |
| bom_target | Defines settings and properties for the BOM attributes when uploading the BOM to SAP ECTR, for example the initial position number, incrementation, sorting, etc. |
| doc_group | Defines settings and properties for the datamodel object Assembly document (p. 7). |
| doc_group_part_group_relation | This relationship type is not further defined. |
| doc_idf | Defines a datamodel object for importing MCAD data into an ECAD design. Consult with XPLM for implementation because not further documented. |
| doc_inno | Defines settings and properties for the datamodel object Design document (p. 7). |
| doc_inno_doc_group_relation | This relationship type is not further defined. |
| doc_inno_doc_pcb_relation | This relationship type is not further defined. |
| doc_pcb | Defines settings and properties for the datamodel object Fabrication document (p. 8). |
| doc_pcb_part_pcb_relation | This relationship type is not further defined. |
| doc_template_pcb | Defines settings and properties for the object that manages the layout template. This object is not part of the datamodel as such but is listed here for reference. |
| doc_template_sch | Defines settings and properties for the object that manages the schematic template. This object is not part of the datamodel as such but is listed here for reference. |
| file_assy_data | Defines settings and properties for the output file Pick & place data (p. 8). |
| file_assy_extra_1 | Defines settings and properties for a custom output file that is saved by default to Assembly document (p. 7). |
| file_assy_extra_2 | Defines settings and properties for a custom output file that is saved by default to Assembly document (p. 7). |
| file_assy_pdf | Defines settings and properties for the output file Placement drawing (p. 8). |
| file_emn | This file is part for the IDF format for importing MCAD data and is saved by default to *doc_idf*. |
| file_emp | This file is part for the IDF format for importing MCAD data and is saved by default to *doc_idf*. |
| file_fabrication | Defines settings and properties for the output file Drill drawing (p. 7). |
| file_gerber | Defines settings and properties for the output file Gerber data (p. 8). |


© 2025 XPLM Solution GmbH

| Configuration section | Description |
|---|---|
| file_idf_data | Defines settings and properties for the output file IDF data (p. 8). |
| file_inno | Defines settings and properties for the output file Design container (p. 7). |
| file_inno_extra_1 | Defines settings and properties for a custom output file that is saved by default to Design document (p. 7). |
| file_inno_extra_2 | Defines settings and properties for a custom output file that is saved by default to Design document (p. 7). |
| file_ipc2581 | Defines settings and properties for the output file IPC-2581 data (p. 8). |
| file_odbpp | Defines settings and properties for the output file ODB++ data (p. 8). |
| file_pcb_extra_1 | Defines settings and properties for a custom output file that is saved by default to Fabrication document (p. 8). |
| file_pcb_extra_2 | Defines settings and properties for a custom output file that is saved by default to Fabrication document (p. 8). |
| file_pdf | Defines settings and properties for the output file Base schematic (p. 7). |
| file_sch_pdf | Defines settings and properties for the output file Assembly schematic (p. 7). |
| file_template_pcb | Defines settings and properties for the PCB template file. |
| file_template_sch | Defines settings and properties for the schematic template file. |
| part_group | Defines settings and properties for the datamodel object Assembly part (p. 7). |
| part_in_bom | Defines settings and properties for attributes of components and materials in the *BOM comparison* dialog. |
| part_pcb | Defines settings and properties for the datamodel object Fabrication part (p. 8). |

## Configuration sections in classification.config

In the file `classification.config`, you can configure parts synchronization in the integration module *Synchronize parts*.

| Configuration section | Description |
|---|---|
| part_lib | Defines settings and properties for attributes of components and materials in the integration module *Synchronize parts*. |

**Related links**
Restarting integration modules or IVS (p. 24)
Integration variables (p. 16)


© 2025 XPLM Solution GmbH

#### 3.2.1.1 Other configuration files

In addition to the configuration files for the datamodel and classification, familiarize yourself with the following files.

### Snippets


Introduced or updated with product version 1.1.4.x.

Snippets are small configuration examples that describe a specific use case. Most of these snippets you can use in the file `datamodel.config`. How and where to use a snippet is described in the file itself and is not further detailed in this document.

You can find snippets with extension `.snippet` in the directory `<PRODUCT DIR>\config\examples`.

### Language files

Configuration file: `<PRODUCT DIR>\basis\dictionary\<language code>\ectr.txt`

The following files define the UI language in the module dialogs:

- `de\ectr.txt` (German)
- `en\ectr.txt` (English)
- ...

In general, it is common to search the language file for the text that appears in the module dialog and change it, for example:

```
dialogs.button.keep_reserved=Keep reserved
dialogs.button.upload_inno=Upload design
dialogs.button.upload_pdf=PDF
```

```
dialogs.button.keep_reserved=Keep checked-out
dialogs.button.upload_inno=Save native design
dialogs.button.upload_pdf=Schematic PDF
```

Note that this text may appear in several places, depending on how it is used.


© 2025 XPLM Solution GmbH

### IVS preferences file

Configuration file: `<PRODUCT DIR>\config\ivs.properties`

This file contains various settings for IVS, for example:

- Default values for module behavior and logging in the IVS menu

  These values are ignored if they are defined in the IVS menu. Values defined in the IVS menu are persistently saved in the registry under *HKEY_CURRENT_USER\Software\JavaSoft\Prefs\ivs\ui\preferences*. Values defined in the file only become active again when this registry key is deleted.
- Settings for the UI
- Settings for the creation of problem reports
- Settings for the integration module *Synchronize parts*

**Related links**
Restarting integration modules or IVS (p. 24)

## 3.2.2 Restarting integration modules or IVS

During configuration, it is often required to restart integration modules and sometimes also IVS or Xpedition for the changes to take effect. Familiarize yourself with common actions when modifying integration files, as they are not always mentioned later in this document.

| Modification to | Post action |
|---|---|
| ■ `datamodel.config`<br>■ `classification.config`<br>■ `release.rb` | ■ Restart the module.<br>■ Alternatively, press **F5** to dynamically reload all Ruby files and hotfixes in the current module and test the change in a new module run. This is very useful for quickly testing a configuration without having to restart the module each time.<br>Please note that changes to the UI always requires at least a module restart, or sometimes also IVS. See below. |
| `ivs.properties` | Restart IVS and the module. |
| Environment files (`*.bat`) | Restart the module. |
| Language files (`*.yml`) | Restart IVS and the module. |
| Java class path | Restart IVS and the module. |
| Integration variables in Windows | Restart Xpedition. |


© 2025 XPLM Solution GmbH

### 3.2.3 Working with different integration environments

You can start the integration with different settings which are adapted to specific PLM or ECAD environments.

Let's assume you configure the integration for a development and a production environment. While developing, you want to use a dedicated PLM instance, use test templates and a test library, and activate the highest logging level. For production, you want to use the live PLM system with approved templates, the production library, and the normal logging level.

1. Go to the directory `<PRODUCT DIR>\config\environments`.
2. Edit the file `init_env.bat` and define the settings for the production environment. See Initial setup (p. 34) for more information on what you can define in this file.
3. Copy the files `init_env.bat` and `production.rb`.
4. Rename the copies to `development.*`.
5. Edit the file `development.bat` and define the settings for the development environment.
6. Set the specific integration variable `INTEGRATE_SAP_ECTR_XPEDITION_ENVIRONMENT` with the value `development`.
7. As a result, the settings for the development environment are used.
8. To use the settings for the production environment again, set the environment variable either to `production` or clear its value. If the variable is not set, the environment is set to `production` by default.

### 3.2.4 Encrypting passwords

With this tool, you can encrypt passwords used by the integration. Encrypting passwords helps secure the integration and avoid connection problems caused by special characters in the password.

#### Procedure

1. Go to `<PRODUCT DIR>\customize\bin\win32` and double-click `encrypt_password.bat`.
   ➔ The *Encrypt Password* dialog opens.
2. In the field **Password**, enter the password in plain text and click **Generate**.

#### Result

The password is encrypted and can be copied from the field **Encrypted**. Example of an encrypted password:

```
CRYPT{AQUBFAsTAAhzQVJCHBIcQFEAWVdWHgENHUcMDVBeVg==}CRYPT
```

#### What to do next

You can use the encrypted password as-is in Ruby files (`.rb`) and batch files (`.bat`), or directly in environment variables under Windows.


© 2025 XPLM Solution GmbH

# 3.3 About parts synchronization

## 3.3.1 Different ways for parts synchronization

The integration supports two different methods for parts synchronization with Xpedition and SAP ECTR. Decide which approach fits your design process best.

### PartSync 1: Synchronization via integration module Synchronize parts

This is the preferred method when accessing the library is not a problem. This integration module was developed specifically for interaction with the library.

- **Advantage**: Flexible configuration for bidirectional parts synchronization and classification in SAP ECTR.
- **Disadvantage**: The setup can be complex depending on customer requirements.

This method is referred to as PartSync 1 in this document and appears in titles as well as in text.

### PartSync 2: Synchronization via BOM comparison dialog

This method is suitable for the following situations:

- If the ECAD application is the leading system for the part definition in PLM.
- If PLM part information cannot be written back to the library (especially problematic for libraries in the cloud).
- If library access is not possible.

For all these cases, you can create missing parts in SAP ECTR during the BOM upload in the *BOM comparison* dialog. During this approach, the library is by-passed and the part information is read from the exported BOM.

- **Advantage**: Easier setup for parts synchronization and convenient to use when uploading the BOM. The integration module *Synchronize parts* is not required and so is not the corresponding integration license.
- **Disadvantage**: Parts synchronization works only in direction SAP ECTR and no information from SAP ECTR can be written back to the library. You can define only one common class in SAP ECTR for all materials.

This method is referred to as PartSync 2 in this document and appears in titles as well as in text.


© 2025 XPLM Solution GmbH

# 4 System requirements

Before installing the integration, check that the following requirements are met.

## Client-side requirements

The integration supports Windows 10/11 (x64) operating systems. For installation, administrator rights are required. Hardware requirements depend on Xpedition. Check the client computer running the integration for the following setup:

- Xpedition in a supported version is installed and the listed special requirements are met. See Supported versions and special requirements (p. 27) for more information.
- SAP ECTR client application & SAP GUI in a supported version are installed. See Supported versions and special requirements (p. 27) for more information.
- Java JRE: The integration uses the JRE that comes bundled with the SAP ECTR installer.

## Server-side requirements

Make sure, SAP in a supported version is installed and the listed special requirements are met. See Supported versions and special requirements (p. 27) for more information.

## Supported versions and special requirements

The following information is current at the time of publishing this document and does not claim to be complete. XPLM reserves the right to change the information at any time.

The integration generally supports the two latest application versions. Older versions can be implemented on request, but this depends on the version. Please also note the special requirements that may be necessary.

- Official support: The version is supported by the latest integration. Issues are fixed immediately.
- Limited support: The version is generally supported by the latest integration, but has not been explicitly tested. Issues are fixed only if they occur in the official versions.

| Name | Official | Limited | Special requirements |
|---|---|---|---|
| SAP PLM | SAP R3/ECC/HANA (all on-premise only) | – | The SAP account used by the integration and SAP ECTR must have sufficient object management rights (create/read/update).<br>SAP ECTR backend component is required. |
| SAP Engineering Control Center | SAP R3/ECC:<br>■ SAP ECTR 5.2.6-7<br>SAP HANA:<br>■ SAP ECTR 1.1.6-7/1.2.0-1 | – | A license for SAP ECTR Professional is required from DSC Software. |


© 2025 XPLM Solution GmbH

| Name | Official | Limited | Special requirements |
|---|---|---|---|
| Siemens Xpedition Enterprise<br>■ Xpedition Designer<br>■ Xpedition Layout | VX.2.14.1<br>2409 | VX.2.13-14 | To use the integration module *Synchronize parts*, access to the component library is required. Depending on the underlying database, different connection methods are possible. Access via a connection string always requires x64 database drivers.<br>■ Access to library via OLE DB/ODBC connection string<br>□ MS Access: The driver Microsoft Access Database Engine is required. It provides OLE DB connections and generic ODBC.<br>□ MySQL: The MySQL ODBC driver is required.<br>□ Oracle: The Oracle ODBC driver is required.<br>□ Postgre: The Postgre ODBC driver is required.<br>□ SQL Server: The SQL Server ODBC driver or the SQL Server OLE DB driver is required.<br>□ SQLite: A SQLite ODBC driver is required. Commercial or open-source versions are available.<br>■ Access to library via file path<br>□ MS Access MDB file: Nothing else is required.<br>To create variant-specific BOMs and PDFs, an additional license for the Variant Manager is required from Siemens.<br>To export IDF, an additional license is required from Siemens.<br>To export STEP, the 3D plugin must be installed in Xpedition Layout. |


© 2025 XPLM Solution GmbH

## 4.1 Required integration licenses

The integration functions and connectors require corresponding licenses. Make sure you have them available.

### Used license mechanism

The integration uses the license mechanism from SAP. Organize the required licenses directly via SAP or an authorized reseller.

| ECAD | SAP license ID |
|---|---|
| Siemens Xpedition Enterprise | ■ 7588 (Xpedition Designer)<br>■ 7587 (Xpedition Layout) |

See https://support.sap.com/infopages/swdc/plm/ectr_ecad10.html for more information.

## 4.2 Required installation media

Prior to installation, make sure that you have all installation media available.

Get the required installation media from the SAP download portal.


© 2025 XPLM Solution GmbH

# 5 Installation

## 5.1 Installing integration

Make sure that all installation media and licenses are available, and then start the installation.

### Before you start

Perform the following manual actions in SAP ECTR to make sure that the integration can create and modify objects using the designated account:

- Create objects and record mandatory fields during object creation. You will need this information later when you configure the integration.
- Change attribute values for the created objects.
- Reserve objects, and clear the reservation.
- Change the object status. Also record all names or IDs of the different statuses. You will need this information later when you configure the integration.
- Delete the created objects.

The following procedure applies to a new installation. To update an existing installation, see Updating integration (p. 78) for more information.

### Procedure

1. Close all running instances of Xpedition.
2. Start the installer with administrator rights.
3. Select the integration for your ECAD application and complete the steps of the wizard to the end.

### Result

The integration is installed.


© 2025 XPLM Solution GmbH

# 5.2 Siemens Xpedition Enterprise

## 5.2.1 Installing integration menu in Xpedition Designer

To use the integration functions, you must install a new menu in Xpedition Designer.

### Procedure

1. Go to the directory `<PRODUCT DIR>\appdata\dxdesigner`.
2. Copy the file `IntegrateMenu_sap_ectr_xpedition.vbs` to the working directory.

   The working directory is set during the installation of Xpedition in an environment variable that starts with `WDIR_EEVX*`. See the variable for the correct path.
3. In the working directory, create the file `scripts.ini` as follows:

   ```
   [Viewdraw]
   Script#0=IntegrateMenu_sap_ectr_xpedition.vbs
   ```

   Use the first free number for `Script#<number>`, if the file already references other scripts. Otherwise, it has to start with 0.
4. Save the file.
5. Start Xpedition Designer.

### Result

The integration menu is installed. You can now access the integration functions from the menu **SAP ECTR** in the menu bar.

## 5.2.2 Installing PDFCreator printer

PDFCreator is the native PDF printer for the integration. You can also use Mentor's extended print functionality, or use any other PDF printer that supports printing to a predefined directory with automatic file name generation.

### About this task

The following procedure has been verified with PDFCreator 5.

### Procedure

1. Download PDFCreator from the website https://www.pdfforge.org and install it.
2. Start PDFCreator and click **Profiles**.
3. Click **Add** to create a new profile and name it *Integrate*.
4. In the section **Save**, define settings as follows:
   - **Interactive/Automatic**: Change to **Automatic**.
   - **Filename**: Enter the variable `<InputFilename>`.
   - **Target directory**: Enter the path `C:\temp\integrate_pdf`.
   - **Only save file temporarily**: Disable this option.
   - **Behaviour of existing file**: Change to **Overwrite**.
5. In the section **Actions** > **Send/Open**, define settings as follows:
   - **Open File**: If you want the PDF file to be opened after it has been created, change to **Windows default viewer**.
   - If you want the PDF file not to be opened, delete the line **Open File**.


© 2025 XPLM Solution GmbH

6. In the section **Output Format**, define a landscape format. The auto-detect mode may not always produce the correct results.
7. Click **Save**.
8. In the section **Printers**, click **Add Printer**.
9. Name the printer *IntegratePrinter*, assign the profile *Integrate* to it, and make it the default printer.
10. Exit PDFCreator.

### Result

PDFCreator is installed and configured.

**Related links**
Configuring extended print functionality (p. 65)
Configuring PDFCreator printer (p. 64)


© 2025 XPLM Solution GmbH

# 5.3 SAP Engineering Control Center

## 5.3.1 Activating BC set

When installing SAP ECTR, a Business Configuration set (BC set) is usually installed in SAP that contains the SAP ECTR-specific object definitions. Before you use the integration, you must first activate the BC set.

### Before you start

If for some reasons the BC set was not installed in SAP, you must request it from DSC, the vendor of SAP ECTR (https://www.dscsag.com), and upload/install it manually using the transaction **SCPR3**. Please also contact DSC if you have any questions about installation.

Sample extract from a BC set:

```
...
Define document type (dtype)
    XEC
Define document type language-specific
    DE – XEC - ECAD document
    EN – XEC - ECAD document
Maintain the text of the object link
    DE – MARA - Materialstamm
    EN – MARA – Material
Define Workstation-Application in the network
    PC – XEC
...
```

### Procedure

1. Start the SAP client and log in.
2. Execute the transaction **SCPR20** and enter the name **/DSCSAG/ECTR_ECAD** for the BC set.
3. Click **Activate** or select the corresponding action from the menu bar.
4. Select additional activation options and complete the activation.

### Result

The BC set is activated and related data can be saved in SAP.


© 2025 XPLM Solution GmbH

# 6 Initial setup

## 6.1 Siemens Xpedition Enterprise

### 6.1.1 Setting up delimiter for Variant Manager

When uploading the BOM, the integration uses the BOM export functionality from Variant Manager. Make sure the delimiter in the exported BOM text file matches with the integration configuration.

The default delimiter in the integration configuration is `\t`, which corresponds to tabs. Change it for example to comma, if this is your delimiter.

#### Example

Configuration file: `<PRODUCT DIR>\config\datamodel.config`

```
bom_source {
...
  eda :bom_report_delimiter => ","
...
}
```

### 6.1.2 Setting up local project and template directories

The integration uses local directories in which designs are saved and from which templates are read.

#### Example

Environment file: `<PRODUCT DIR>\config\environments\init_env.bat`

```
IF "%PROJECT_DIR%" == "" SET PROJECT_DIR=<some path>
IF "%INTEGRATE_SCH_TEMPLATE_DIR%" == "" SET INTEGRATE_SCH_TEMPLATE_DIR=<some path>
IF "%INTEGRATE_PCB_TEMPLATE_DIR%" == "" SET INTEGRATE_PCB_TEMPLATE_DIR=<some path>
```

Alternatively, you can also define the project and template directories as Windows environment variables and thus do not have to change the file `init_env.bat`.

**Related links**
Integration variables (p. 16)

### 6.1.3 PartSync 1: Setting up library connection

For parts synchronization, the integration requires full access to the component library.

A working setup for the component library in Xpedition is mandatory. Use only x64 database drivers and valid connection strings. See System requirements (p. 27) and https://www.connectionstrings.com for more information.

#### Managing the library connection

Add the library connection by using one of the following options:

- Define the connection in `init_env.bat`, for example `IF "%LIBDB_CONNECTION_STRING%" == "" SET LIBDB_CONNECTION_STRING=<MY CONNECTION STRING>`.
- Set the integration variable `INTEGRATE_SAP_ECTR_XPEDITION_LIBRARY` under Windows.

#### Example: Library connection via file path

MS Access MDB file: `<path to MDB file>`


© 2025 XPLM Solution GmbH

### Example: Common OLE DB and ODBC connection strings

- MS Access: `Provider=MICROSOFT.ACE.OLEDB.12.0; Data Source=<path to MDB file>; Persist Security Info=False;`
- SQL Server

  Server authentication: `Provider=SQLOLEDB; Server=myServer; Database=myDatabase; UID=myUsername; PWD=myPassword;`

  Windows authentication: `Provider=SQLOLEDB; Server=myServer; Database=myDatabase; Integrated Security=SSPI;`
- ODBC

  You can define ODBC connections with a datasource name (DSN), or without.
  For connections with a DSN, use the application *ODBC Data Sources (64-bit)* and define the database connection by setting up a specific driver. Then use the generic ODBC connection string to refer to this DSN.
  For connections without a DSN, you define the corresponding driver in the connection string. Check the driver name in the tab **Drivers** of the *ODBC Data Source Administrator* window, and use this name in the section `Driver` of the connection string. Setting up a DSN is therefore not required.

  - Generic ODBC with DSN to a SQL-capable database (default for MS Access)

    No authentication: `Provider=MSDASQL; DSN=myDatasource;`

    Server authentication: `Provider=MSDASQL; DSN=myDatasource; UID=myUsername; PWD=myPassword;`

    Windows authentication: `Provider=MSDASQL; DSN=myDatasource; Trusted_Connection=Yes;`
  - MySQL (without DSN): `Driver=MySQL ODBC 9.0 Driver; Server=myServer; Database=myDatabase; User=myUsername; Password=myPassword; Option=3;`
  - SQL Server (without DSN)

    Server authentication: `Driver=ODBC Driver 18 for SQL Server; Server=myServer; Database=myDatabase; UID=myUsername; PWD=myPassword;`

    Windows authentication: `Driver=ODBC Driver 18 for SQL Server; Server=myServer; Database=myDatabase; Trusted_Connection=Yes;`
  - SQLite (without DSN)

    The following connection string is from the open-source driver and may look different for commercial drivers.

    `Driver=SQLite3 ODBC Driver; Database=<path to DB file>; LongNames=0; Timeout=1000; NoTXN=0; SyncPragma=NORMAL; StepAPI=0;`
- You can also use any other valid OLE DB connection string, provided a compatible database driver is installed.

For security reasons, you should always encrypt passwords and don't use them in plain text. To do this, use the supplied encryption tool `encrypt_password.bat` in directory `<PRODUCT DIR>\customize\bin\win32`. Encrypted passwords also eliminate problems that can occur with special characters in passwords. Note that such problems are related to Windows and database connection strings, not the integration.


© 2025 XPLM Solution GmbH

However, there may be situations where you use passwords with special characters in plain text. Pay attention to the following rules when setting such passwords in `init_env.bat` or in environment variables under Windows:

- **PLM connection**: Precede the following special characters `<`, `>`, `|`, `&`, or `^` with the escape character `^`. For example, to resolve the password `56^trE!GH&33<` correctly, enter it as `56^^trE!GH^&33^<`.
- **Library connection string**: If the password contains in particular the special character `;` (semicolon) or other characters that are not letters or numbers, surround the whole password string with double quotes, for example `Password="zZ&897;rr%";`. By using this method, other special characters don't need to be escaped in addition. Exception to this method is if the password contains other double quotes. In this case, escape double quotes as `""`.
- For more information, look up password syntax in environment variables and connection strings on the internet.

**Related links**
Integration variables (p. 16)
Working with backend configuration support (p. 76)

### 6.1.4 PartSync 1: Setting up library database format

For libraries in common database formats, the integration tries to detect the database type automatically. This can sometimes fail and you get connection or query errors. In this case it can be helpful to define the database explicitly.

Introduced or updated with product version 1.1.0.x.

Uncomment the line `#runtime[:ecad_library].db_type` and define the database in use.

#### Example

Environment file: `<PRODUCT DIR>\config\environments\production.rb`

```
runtime[:ecad_library].db_type = :access
```

### 6.1.5 PartSync 1: Setting up material number in library

During synchronization, new material numbers are created in SAP ECTR. Depending on the setup, the new number may be created from the number generator in SAP ECTR or derived directly from the component number in the library. For the first case, the number returned from SAP ECTR is usually written into an additional field in the corresponding component table of the library.

#### Procedure

1. Adding new field for material number in database:
   a) Open the database for editing.
   b) Add a new text field in each component table, for example **PLMNumber**.
   c) Save the database.
      ➔ The new field exists for all component tables in the database.
2. Setting up databook:
   a) Start Xpedition Designer and create a new design.
   b) Open the window *Databook*.


© 2025 XPLM Solution GmbH

c) Right-click somewhere in the window and select **Configure** > **Edit Configuration**.

d) Click **Update All** to load the field **PLMNumber** into the configuration.

e) In each component table, mark the field **PLMNumber** green, and enable the options **Load** and **Annotate**.

f) Click **OK** to close the configuration.

g) Close the window *Databook*. This will prompt you to save the databook configuration.

➔ The settings are saved to the file `*.dbc`.

**3.** Checking variants:

a) Open the window *Variants* and select **Settings** > **Library Query Setup**.

b) Make sure, the field **Data source configuration file** contains the same path to the file `*.dbc` as used under Databook in the project settings.

c) Make sure, the field **Part number attribute** is set to the value **Part Number**.

d) Click **OK** to close the window, and **Save** to save the settings.

➔ The settings are saved to the file `VmgrLdataGrid.cfg` in the project directory.

**4.** Setting up central library:

a) Start Library Manager and open the file `*.lmc` for the central library.

b) Select **Tools** > **Property Definition Editor**.

c) Create a new user property **PLMNumber** and enable it with the front checkbox. Leave the default settings for all other options.

d) Click **OK** and close Library Manager.

➔ The settings are saved to the file `*.lmc`.

**5.** Testing setup:

a) Open the database for editing.

b) In a component table, add a value to the field **PLMNumber** for some components.

c) In the window *Databook* of Xpedition Designer, search for these components and place them in the schematic.

d) In the window *Variants*, define an assembly variant.

e) Right-click somewhere in the window and select **Report** > **BOM reports**.

f) Select type **Ref Des List** and in **xDX Databook properties** the field **PLMNumber**.

g) Create the BOM report and check if the correct value from the field **PLMNumber** was exported.

### Result

The library is set up with the material number from SAP ECTR.


© 2025 XPLM Solution GmbH

### 6.1.6 Creating templates

In the integration module *New ECAD project*, you can select templates saved in the file system or, if supported, in SAP ECTR. The following procedure explains how to create templates and where to save them.

#### Procedure

1. Create a new design or open an existing one.
2. Complete the design and save it.
3. Creating schematic template:
   a) In Xpedition Designer, select **Tools** > **Archiver**.
   b) Select **Export** and click **Next**.
   c) Define the target directory and complete the export process.
   d) Go to the project directory and copy the file `VmgrLdataGrid.cfg` to the target directory.

      This file is not exported by the *Archiver* and you must add the file manually. Without this file, the BOM for the variants are not exported correctly. Make sure the variants are configured correctly. See PartSync 1: Setting up material number in library (p. 36) for more information.

   e) Go to the target directory.
   f) Select everything except the directory `PCB` and create a ZIP file.
4. Creating layout template:
   a) Go to the project directory and there to the directory `PCB`.
   b) Select everything and create a ZIP file.
5. Rename the template files for easy identification and move them to corresponding template directory.
6. **Optional**: Save templates in SAP ECTR. See Saving templates (p. 71) for more information.

#### Result

The template is created and can be used with the integration.


© 2025 XPLM Solution GmbH

# 6.2 SAP Engineering Control Center

## 6.2.1 Setting up DTypes

If you use other DTypes than those that are preconfigured in the integration, you must adjust the configuration.

In the sections `doc_*`, change the value from `plm :dtype` to the DType definition used in SAP ECTR. Exception: The sections `doc_*_relation` require no definition.

### Example

Configuration file: `<PRODUCT DIR>\config\datamodel.config`

```
doc_inno {
...
  plm :dtype => "xdxI1"
...
}
```

`plm :dtype`: Defines the DType in SAP ECTR.

## 6.2.2 Updating DType settings

The SAP ECTR client reads the `dtype.xml` files from various sources and compares the settings with SAP. If these settings between SAP ECTR client and SAP are different, you must update the DType settings in SAP.

### Before you start

Install the integration first.

### Procedure

1. In SAP ECTR, select **Administrator** > **System Setup** > **Upload Settings to SAP System**.
2. Click **Yes** to confirm the upload.

### Result

DType settings in SAP are updated from the SAP ECTR client.

## 6.2.3 Setting up status net

You must define the valid object statuses as used in SAP ECTR. The integration can only reserve and change objects in work. Released objects are locked for modification.

Search the configuration files `datamodel.config` and `classification.config` for the following status definitions:

- Released: `plm :status_released`
- In work: `plm :status_in_work`
- Obsolete: `plm :status_obsolete` (only if supported)

These definitions occur in the sections `doc_*` and `part_*`. Adjust the definitions as needed if they do not match SAP ECTR.


© 2025 XPLM Solution GmbH

### Example

Configuration file: <PRODUCT DIR>\config\datamodel.config

```
doc_inno {
...
  plm :status_in_work => "WR", "IW", "AC", "IA", "AA", "AE"
  plm :status_released => "RE", "FR"
...
}
```

## Status actions

Some actions cannot be allowed by workflow rules in SAP ECTR. The following tables show only integration actions depending on workflow status for documents or materials. Ask your administrator about the defined rules in SAP ECTR.

Table 1: Innovation document (doc_inno)

| Status | Default values | Status actions |
|---|---|---|
| Released | RE | **Save:** It is not possible to save design data in SAP ECTR.<br>**Edit:** It is not possible to reserve the design for the current user in SAP ECTR. |
| In work | WR, IW | **Save:** It is possible to save design data in SAP ECTR.<br>**Edit:** It is possible to reserve the design for the current user in SAP ECTR. |
| Other | - | **Edit:** It is not possible to reserve the design for the current user in SAP ECTR. |

Table 2: Assembly document (doc_group)

| Status | Default values | Status actions |
|---|---|---|
| Released | RE | **Save:** It is not possible to save assembly data in SAP ECTR. |
| In work | WR, IW | **Save:** It is possible to save assembly data in SAP ECTR. |
| Other | - | **Save:** It is not possible to save assembly data in SAP ECTR. |

Table 3: PCB document (doc_pcb)

| Status | Default values | Status actions |
|---|---|---|
| Released | RE | **Save:** It is not possible to save fabrication data in SAP ECTR. |
| In work | WR, IW | **Save:** It is possible to save fabrication data in SAP ECTR. |
| Other | - | **Save:** It is not possible to save fabrication data in SAP ECTR. |


© 2025 XPLM Solution GmbH

Table 4: PCB material (part_pcb)

| Status | Default values | Status actions |
|---|---|---|
| Released | - | - |
| In work | - | - |
| Other | - | - |

Table 5: Assembly material (part_group)

| Status | Default values | Status actions |
|---|---|---|
| Released | - | **Save:** It is not possible to save the BOM in SAP ECTR. |
| In work | - | **Save:** It is possible to save the BOM in SAP ECTR. |
| Other | - | **Save:** It is not possible to save the BOM in SAP ECTR. |

Table 6: Purchased part (part_lib)

| Status | Default values | Status actions |
|---|---|---|
| Released | - | - |
| In work | - | **Save:** When uploading the BOM to SAP ECTR, the component is listed in the BOM comparison dialog for unreleased parts. |
| Other | - | **Save:** When uploading the BOM to SAP ECTR, the component is listed in the BOM comparison dialog for unreleased parts. |

### 6.2.4 Setting up material

You must define the valid material properties for the material objects used by the integration.

In the sections `part_pcb`, `part_group` and `part_lib`, change all `plm :matl_*` definitions to the values used in SAP ECTR.

Sections `part_pcb` and `part_group` are found in file `datamodel.config`. Section `part_lib` is found in file `classification.config`.

#### Example

Configuration file: `<PRODUCT DIR>\config\classification.config`

```
part_lib {
...
  plm :matl_type => "HALB"
  plm :matl_group => "01"
...
 }
```

- `plm :matl_type`: Defines the material type.
- `plm :matl_group`: Defines the material group.
- See comments with definitions for other `plm :matl_*` entries in the configuration file.


© 2025 XPLM Solution GmbH

### 6.2.5 PartSync 1: Setting up classification for parts synchronization

For parts synchronization, you can configure the integration to create materials in SAP ECTR in a class that matches the library structure.

#### Example (Classification for materials based on library structure)

Configuration file: `<PRODUCT DIR>\config\classification.config`

```
part_lib {
  template(:attr_to_plm) {...}
  template(:attr_to_eda) {...}

  subitem(:<internal class name>, :title => '<ui class name>') {
    eda :table => '<library component table>'
    plm :class => '<class name>
    attr_to_plm('<internal attribute name>', '<ecad field>', '<plm field>')
  }
}
```

- `template(...) {...}`: Defines templates for new attributes, which can then be easily called and filled with values.
- `subitem(...) {...}`: Defines the characteristics of the described component type for this class. Contains as options the internal name of the class and the name shown in the integration module *Synchronize parts*.
  - `eda :table`: Defines the component table in the library (only for libraries in common database formats).

    Even if you don't use a classification in SAP ECTR, but your library consists of different tables where the components are managed, you must at least set up the configuration up to this point. Otherwise, the integration cannot read the library properly.
  - `plm :class`: Defines the class in SAP ECTR to which the materials are assigned.
  - `attr_to_plm(...)`: Defines the template call of the attribute to be synchronized.

### 6.2.6 PartSync 2: Setting up SAP ECTR for alternative parts synchronization

To create missing parts via the *BOM comparison* dialog, you must enable this feature in the integration.

#### About this task

When creating missing parts in SAP ECTR, the integration needs to link the SAP ECTR number with the Xpedition component number. Only with this combination, the integration can identify if a component already exists in SAP ECTR when uploading the BOM.

#### Procedure

1. Create a new attribute on material level in your SAP ECTR datamodel to save the component number from Xpedition. See the SAP ECTR documentation for more information. If such an attribute already exists, use this one.
2. Set up the integration:
   a) Edit the file `<PRODUCT DIR>\config\datamodel.config`.
   b) Go to section `part_in_bom`.
   c) In the section `property('ID')`, set the option `eda :none`.
   d) In the section `property('EcadPartNumber')`, set the option `plm "<attribute name>"`.


© 2025 XPLM Solution GmbH

e) Go to section `bom_target` and set the option `common :create_missing_parts => :interactive`.

f) Save the file.

### Result

The feature for creating missing parts via the *BOM comparison* dialog is set up. You can access it in the *BOM comparison* dialog and tab **Missing in PLM**.


© 2025 XPLM Solution GmbH

# 7 Configuration

## 7.1 IVS & UI

### 7.1.1 Configuring UI under Java 8

By default, the UI of the integration is shown in a Windows look-and-feel and works with Java 8 and higher. If you still use Java 8 and want to use the UI in the XPLM look-and-feel, you can configure this easily. Please note that this UI style only works under Java 8.

Introduced in a previous product version.

Comment out the following lines as shown. Restart IVS and the module.

#### Example

Configuration file: `<PRODUCT DIR>\config\ivs.properties`

```
...
# ivs.ui.look_and_feel=com.sun.java.swing.plaf.windows.WindowsLookAndFeel
# ivs.ui.font.default=SansSerif; PLAIN; 12
# ivs.ui.control.I2ItemActionButton.size=58x58
...
```

### 7.1.2 Configuring module flows

Module flows describe a certain behavior when executing an integration module from the menu. You can configure this behavior to a certain extent.

Introduced or updated with product version 1.1.7.x.

Configuration file: `<PRODUCT DIR>\config\datamodel.config`

#### Common flow with dialog: New ECAD project

Section: `flow(:new)`

| Option | Description |
|---|---|
| overwrite_existing_design | Defines whether to overwrite the design if it already exists in the target directory.<br>■ `:always`: Always overwrite without a warning.<br>■ `:warning` (default): Show a warning that the design already exists with the option to overwrite.<br>■ `:error`: Show an error if the design already exists and stop. |

#### Common flow with dialog: Edit

Section: `flow(:open)`


© 2025 XPLM Solution GmbH

| Option | Description |
|---|---|
| overwrite_existing_design | Defines whether to overwrite the design if it already exists in the target directory.<br>■ `:always`: Always overwrite without a warning.<br>■ `:warning` (default): Show a warning that the design already exists with the option to overwrite.<br>■ `:error`: Show an error if the design already exists and stop. |
| :make_design_readonly | Defines the read-only behaviour when opening a design in view mode.<br>■ `:auto` (default): This is the standard behaviour, read-only on view and editable on checkout.<br>■ `:interactive`: Select interactively with additional checkbox in the dialog.<br>■ `true\|false`: If true, the design will always be read-only. If false, the design will always be editable. |

### Common flow with dialog: Save

Section: `flow(:save)`

| Option | Description |
|---|---|
| :delete_design_after_upload | Defines whether to delete the local design data after saving to SAP ECTR.<br>■ `false` (default): Never delete the design data.<br>■ `:force`: Always delete the design data.<br>■ `:yesno`: Show a dialog with options to delete or not. |
| :upload_design_when_not_reserved | Defines whether to save the design if the object is not reserved by another user.<br>■ `false`: Never save the design.<br>■ `:force`: Always save the design.<br>■ `:yesno` (default): Show a dialog with options to save or not. |
| :auto_accept_comparison | Defines whether to show the *BOM comparison* dialog when saving the BOM.<br>■ `false` (default): Always show the *BOM comparison* dialog. This value also corresponds to the default global setting **Skip BOM Comparison Dialog (accept)** in the dialog *Settings*. See User Guide for more information.<br>■ `true`: Never show the *BOM comparison* dialog. |

### Quick-save flows without dialogs

The flows in the following sections allow save actions without showing the dialog *Save* in order to quickly save only the relevant data in SAP ECTR:

- `flow(:save, :save_design, ...)`: Save the design file.
- `flow(:save, :save_fabrication, ...)`: Save all output files for PCB fabrication.
- `flow(:save, :save_bom, ...)`: Save the BOM(s).


© 2025 XPLM Solution GmbH

- `flow(:save, :save_all, ...)`: Save the design file, all output files and BOM(s).

The default settings in each section are configurable in a macro-like structure to control the options of the dialog in the background, for example:

```
...
  dm.deselect_all
  dm.select_option(:inno)
  dm.ok
...
```

First, all options are deselected. Then the option `:inno` is selected. Finally, the action is executed with **OK**. See the examples in the following table for a description of the common options.

| Option | Description |
|---|---|
| event_log_only | Defines whether to show only the event log or the entire dialog.<br>■ `true` (default): Never show the module dialog but only the event log.<br>■ `false`: Always show the module dialog with event log. |
| :auto_accept_comparison | In the flows `:save_bom` and `:save_all`, the *BOM comparison* dialog is actively suppressed with the option `force_value: true`. |
| dm.validate_in_macro | Defines whether to show warnings or errors during the action.<br>■ `:ignore` (default): Never show warnings or errors.<br>■ `:display`: Always show warnings or errors. |
| dm.select_all | Enables all options in the dialog and saves this data in SAP ECTR. |
| dm.select_option(...) | Enables a specific option and saves this data in SAP ECTR. In the configuration file `release.rb`, search for `:extract =>` and use the referenced values as option, for example:<br>■ `:inno`: Save the design file.<br>■ `:pdf`: Save a PDF file with the base schematic.<br>■ `:assy_data`: Save the pick and place data for the assembly.<br>■ : Save a thumbnail with the base schematic.<br>To create a sequence of extractions and uploads, put each `dm.select_option(...)` entry on its own line. |

### 7.1.3 Hiding design path in UI

In the integration module *Save*, the design path is shown at the top, with the option to open this directory. You can now hide this information in the integration module if required.

Introduced or updated with product version 1.1.5.x.

Remove the comment at the following line and save the file. Restart IVS and the module.


© 2025 XPLM Solution GmbH

### Example

Configuration file: `<PRODUCT DIR>\config\ivs.properties`

```
...
ivs.ui.hide_components=active_design_panel
...
```

## 7.1.4 Hiding labels for buttons

In the integration modules, you can hide the labels shown below most of the action buttons. Labels for **OK**, **Apply** and **Cancel** are always shown.


Introduced in a previous product version.

Remove the comment for the following line and change the value from `false` to `true`. Restart IVS and the module.

### Example

Configuration file: `<PRODUCT DIR>\config\ivs.properties`

```
...
ivs.ui.control.I2ItemActionButton.hide_text=true
...
```

## 7.1.5 Hiding output file options

In the integration module *Save*, you can easily hide the options for creating unwanted output files in the user interface, and prevent their creation.

### Example

Configuration file: `<PRODUCT DIR>\config\datamodel.config`

```
file_pdf {
...
  common :exclude => true
...
}
```

Add the statement `common :exclude => true` to any section that begins with `file_` and restart the integration module. The corresponding option is no longer shown in the user interface, nor will the output file be created.

## 7.1.6 Terminating IVS via batch file

For example, you want to be able to terminate IVS and all integration modules when closing the ECAD application to release integration licenses.


Introduced in a previous product version.

In this case you would implement a custom script that is called when you close the ECAD application. You then can execute the batch file `ivs_stop.bat` in the directory `<PRODUCT DIR>\customize\bin\win32` from the script.

The same directory also contains batch files for starting the integration modules. This can be useful for further process automation.


© 2025 XPLM Solution GmbH

### 7.1.7 Configuring default text editor

You can open log files directly from the event log in Notepad++. If this is not your default text editor, you can configure it here.



Introduced or updated with product version 1.1.4.x.

Replace `notepad++.exe` with the executable of your text editor.

#### Example

Configuration file: `<PRODUCT DIR>\config\ivs.properties`

```
ivs.module_log.navigate_from_event_log.command= ←
cmd /c "start notepad++.exe  \"<FILEPATH>\" -p<OFFSET>"
```

## 7.2 Datamodel objects & output files

### 7.2.1 Configuring main design file

Creating or opening a design from SAP ECTR requires that the correct design file in the container is selected. If the container contains several design files for whatever reasons, you can configure selection rules to select the correct one.



Introduced or updated with product version 1.1.6.x.

Add the option `common :project_file_ambigious_resolver` at the top of the configuration file in the general section, directly after `# encoding: utf-8`. Then define a selection rule for the design file in the container.

#### Example

Configuration file: `<PRODUCT DIR>\config\datamodel.config`

```
[1] common :project_file_ambigious_resolver => :ui
[2] common :project_file_ambigious_resolver => /final/i
[3] common :project_file_ambigious_resolver => imod_lambda {|design_dir, candidates|
candidates.first}
[4] common :project_file_ambigious_resolver => imod_lambda {|design_dir, candidates|
candidates.last}
```

1. Shows a selection dialog to select the design file manually.
2. This RegEx searches for the pattern *final* in the file name. Upper and lower case is ignored. Adapt the rule to match your requirements. See https://www.regular-expressions.info for more information.
3. Searches for the first design file in the container.
4. Searches for the last design file in the container.

### 7.2.2 Configuring object names

To some extent, you can configure the object names that are used in the integration module *Save* when creating objects in SAP ECTR.



Introduced in a previous product version.

Object names are derived as follows:

- Design document (p. 7) (`doc_inno`): Name of the design.


© 2025 XPLM Solution GmbH

- Fabrication part (p. 8) (`part_pcb`): Name of the design plus "PCB".
- Fabrication document (p. 8) (`doc_pcb`): Derived automatically from `part_pcb`.
- Assembly part (p. 7) (`part_group`): Name of the design plus name of the assembly.
- Assembly document (p. 7) (`doc_group`): Derived automatically from `part_group`.

To maintain consistency, you should only change the following names and work with either prefixes or suffixes

- `doc_inno`
- `part_pcb`
- `part_group`

The object name is created in `property('Name')` and uses the function `common :auto_value_volatile`. The following example shows how you can add a prefix to `doc_inno`. Similarly, you can do the same for other object names.

### Example

Configuration file: `<PRODUCT DIR>\config\datamodel.config`

```
doc_inno {
...
  property('Name') {
    common :auto_value_volatile => imod_lambda ←
    { (respond_to?(:design) && design) ? "<prefix> #{design.name}" : '' }
  }
...
}
```

## 7.2.3 Configuring new attributes for design objects

In addition to the default attributes in design objects, you can add new attributes and define the mapping of the fields between Xpedition and SAP ECTR.

### Procedure

1. Add the new attribute in SAP ECTR.
2. Add the new parameter in Xpedition.
3. Go to the directory `<PRODUCT DIR>\config`.
4. Edit the file `datamodel.config`.
5. Add the new definitions `attr()` and `property()` in the section `doc_inno`:

   ```
   doc_inno {
     attr('<attribute name>', :blank_equal => true)
     property('<attribute name>')
   ```

   The definition `attr()` instantiates the new attribute, while `property()` extends it with additional options. Keep `<attribute name>` consistent in both definitions.


© 2025 XPLM Solution GmbH

6. Add `eda` and `plm` statements to define the mapping in Xpedition and SAP ECTR:

```
doc_inno {
  attr('<attribute name>', :blank_equal => true)
  property('<attribute name>') {
    eda "<parameter in Xpedition>", :create => true
    plm "<attribute in SAP ECTR>", :create => true
  }
}
```

7. Add other options, for example `direction` and `merge`. See Configuration options (p. 82) for more information.
8. Save the file and test the new attribute with the integration module *Save*.

### Result

The new attribute is defined and can be synchronized between the two systems.

## 7.2.4 Configuring appearance of object title

In the integration module *Save*, you can now more easily configure how the object title is shown, for example *ID: 000526* | *Revision: A* | *Status: In work*. See the snippet `datamodel.item_title.snippet` for more information.

Introduced or updated with product version 1.1.5.x.

## 7.2.5 Configuring attribute field lengths

Attribute fields in one or another application may have different lengths (in number of characters) defined, which may cause errors when updating. You can define that attribute lengths are truncated during updating if the length exceeds the specified limit.

### Example

Configuration file: `<PRODUCT DIR>\config\datamodel.config`

```
doc_inno {
...
  property('Name') {
    eda :external_length => 20
    plm :external_length => 10
  }
...
}
```

- `eda :external_length`: Defines the maximum attribute length in Xpedition.
- `plm :external_length`: Defines the maximum attribute length in SAP ECTR.

## 7.2.6 Configuring file names of output files

You can now better define how file names of output files are composed. See the snippet `datamodel.filename.snippet` for more information.

Introduced or updated with product version 1.1.5.x.


© 2025 XPLM Solution GmbH

### 7.2.7 Configuring extra files for upload

The integration configuration comes with additional, and free configurable output files you can select in the integration module *Save*. Ideally, such files are not created by the integration but by manual or other methods. If configured accordingly, the integration can select these files from the project directory, or another location, and upload them to SAP ECTR.

Introduced or updated with product version 1.1.7.x.

Extra output files are pre-configured for the following sections:

- **Design**
  - file_inno_extra_1
  - file_inno_extra_2
- **PCB fabrication**
  - file_pcb_extra_1
  - file_pcb_extra_2
- **PCB assembly**
  - file_assy_extra_1
  - file_assy_extra_2

By default, the options are disabled and hidden in the integration module. To enable and show, set the option `common :exclude => false`.

#### Examples

Configuration file: `<PRODUCT DIR>\config\datamodel.config`

The following snippets contain examples for selecting and uploading existing files in different ways.

- Upload one file: `file_existing_uploadOneFile.snippet`
- Upload multiple files: `file_existing_uploadMultipleFiles.snippet`
- Zip and upload files: `file_existing_uploadZIP.snippet`

When using a snippet in `datamodel.config`, replace its name with the extra file name. For example, the snippet `file_all_from_folder {...}` would become `file_inno_extra_1 {...}`.

Review also existing output file configurations in `datamodel.config` for PLM-specific settings and include them in the extra file configuration.

**Related links**
Other configuration files (p. 23)

### 7.2.8 Configuring own extractions in extra files

If you already have a script or an application to create a specific output file you can use this extraction method instead of the available integration options. A basic function was already provided in an earlier product version and has now been extended. See also snippet `file_extractions.snippet` for more information.

Introduced or updated with product version 1.1.5.x.

The best way to configure your own extractions is to use one of the additional outputs `file_*_extra_1` or `file_*_extra_2`. Set `common :exclude => false` to show this output option in the integration module *Save*.


© 2025 XPLM Solution GmbH

### Example 1

In this example, a script with parametrized input is used for extraction.

Configuration file: `<PRODUCT DIR>\config\datamodel.config`

```
file_assy_extra_1 {
  filename imod_lambda { |doc| doc.format('<ID>_<Revision>_extra1.<subkey>.txt')}
  attr('Name') { plm :filter => true, :filter_default => "%_%_extra1.%.txt" }
  common :exclude => false
  eda :type => :customer_extract
  eda :customer_extract => ['customer_file1.bat', :output, '--variant', :subkey, ←
  '--design_dir', :dir, '--prj_file', :projectFile, '--pcb', :pcb_design_path]
  eda :use_pcb => false
  ...
}
```

- `eda :type`: Defines with the option `:customer_extract` an own extraction method.
- `eda :customer_extract`: Defines the specific command line call for the extraction.
  - `customer_file1.bat`: Defines the script or the application. You can also use absolute paths here. If you define only the file name, the integration expects this file in the current project directory.
  - `:output`: Defines the parameter for the temporary directory in which the output file is stored. You can also define another, fixed directory here instead of this parameter.
  - `--variant`: Defines the parameter for passing the variant. Use the option `:subkey` to select the name of the current variant.
  - `--design_dir`: Defines the parameter for passing the project directory. Use the option `:dir` to select the current directory.
  - `--prj_file`: Defines the parameter for passing the project file. Use the option `:projectFile` to select the current file. Depending on the setting of `eda :use_pcb`, this is either the project file/schematic, or the associated layout.
  - `--pcb`: Defines the parameter for passing the path for the layout. Use the option `:pcb_design_path` to select the current path.
- `eda :use_pcb`: Uses the schematic (`false`) or the layout (`true`) for the extraction.

### Example 2

In this example, no additional parameters are used for extraction. The custom extraction method creates the output by itself. However, the output directory is known and the integration can upload the file from there to SAP ECTR. This is useful when the extraction method has no way to change the output directory.


© 2025 XPLM Solution GmbH

Configuration file: <PRODUCT DIR>\config\datamodel.config

```
file_assy_extra_1 {
  filename imod_lambda { |doc| doc.format('<ID>_<Revision>_extra1.<subkey>.txt')}
  attr('Name') { plm :filter => true, :filter_default => "%_%_extra1.%.txt" }
  common :exclude => false
  eda :type => :customer_extract
  eda :customer_extract => ['customer_file2.bat']
  eda :clean_output_dir => false
  eda :output_dir => '<design.dir>/custom_output2'
  eda :zip => true
  ...
}
```

- `eda :clean_output_dir`: Defines if the output directory should be deleted each time (`true`) or not (`false`).
- `eda :output_dir`: Defines the output directory. The path variable `<design.dir>` corresponds to the current project directory.
- `eda :zip`: Zips the output (`true`) or not (`false`).

**Related links**
Other configuration files (p. 23)

## 7.3 BOM and assemblies

### 7.3.1 Excluding or including BOM items

You can exclude BOM items whose attributes match certain criteria from the BOM or systematically include them in the BOM.

#### Examples

Configuration file: <PRODUCT DIR>\config\datamodel.config

**Excluding**

Exclude BOM items with the attribute `NOLOAD` and value `true` from the BOM:

```
bom_source {
...
  eda :ignore => ['NOLOAD', true]
...
}
```

Exclude BOM items whose reference designators start with `TP` or `MH` from the BOM:

```
bom_source {
...
  eda :ignore => lambda {|bl| bl['TAG'].to_s =~ /^(TP)|(MH)/}
...
}
```

Only one `ignore` parameter per BOM is supported.


© 2025 XPLM Solution GmbH

**Including**

Only include BOM items whose reference designators start with `R` in the BOM:

```
bom_source {
...
  eda :cae_flag => lambda {|bl| sbl['TAG'] =~ /^R/}
...
}
```

Only one `:cae_flag` option per BOM is supported.

## 7.3.2 Configuring BOM report

In the *BOM comparison* dialog, you can configure the BOM report to a certain extend.

Introduced in a previous product version.

### Example: Defining template and email settings

Configuration file: `<PRODUCT DIR>\config\datamodel.config`

```
bom_target {
...
  common :bom_report => {
    :template => 'bom_report/bom_diff_template.xls',
    :properties_range => 'A1:D4',
    :row_range => 'A5:N5',
    #:attr_ids => ['POS_NO','ID','TAG','Name','QUANTITY','Length','Unit']
  }
...
  common :send_mail_command => proc { |ctx|
    ctx[:address] ||= "mail@address.com"
    ctx[:subject] ||= "BOM Report"
    ctx[:body] ||= "Attached BOM Report"
  ...
  }
...
}
```

- `common :bom_report`: Defines settings for the creation of the Excel and ASCII report.
  - `:template`: Defines the Excel template in the directory `<PRODUCT DIR>\appdata\bom_report\bom_diff_template.xls`.
  - `:properties_range`: Defines a range in the Excel template in which to search for the placeholder for identifying the assembly.
  - `:row_range`: Defines a range in the Excel template in which to search for placeholders for the attributes in the BOM rows.
  - `:attr_ids`: Defines a list of attributes to be included in the ASCII report. By default all attributes are included. Remove the comment and define individual attributes if required.
- `common :send_mail_command`: Defines settings for composing an email and attaching the report.
  - `ctx[:address]`: Defines the email address.
  - `ctx[:subject]`: Defines the subject of the email.


© 2025 XPLM Solution GmbH

- `ctx[:body]`: Defines the content of the email.

### Example: Defining target directory for BOM report

You can define the target directory for the BOM report with the environment variable `REPORT_DIR`. Set it in the file `init_env.bat`, for example `SET REPORT_DIR=C:\temp\bomreport`. Alternatively, define the variable under Windows.

## 7.3.3 Configuring common variant

You can configure the integration to add a common variant to the design and define the name of this variant. However, naming the variant is not supported by all ECAD applications. See also snippet `design_variant_settings.snippet` for more information.


Introduced or updated with product version 1.1.4.x.

### Example

Configuration file: `<PRODUCT DIR>\config\datamodel.config`

```
# encoding: utf-8
[1] eda :common_variant => "Total"
[2] eda :add_common_variant => true
...
```

1. Defines the name of the common variant (not supported by Xpedition)
2. Adds a common variant, if none is defined: `true` (always add, default) | `false` (don't add) | `:only_if_no_variants` (add only if no variant exists)

## 7.3.4 Configuring columns and mappings in BOM comparison

You can now configure additional component properties from Xpedition to be shown as columns in the *BOM comparison* dialog. If you define a mapping to a BOM attribute in SAP ECTR, the two values can also be compared directly via a new method.


Introduced or updated with product version 1.1.5.x.

Complete the following example to make this work. The additional column should automatically appear in both tabs **Refdes BOM** and **Quantity BOM**. If it's not visible, click  and select the column from the list. If it's not in the list either, check your configuration for errors.


© 2025 XPLM Solution GmbH

### Example

Configuration file: <PRODUCT DIR>\config\datamodel.config

```
bom_source {
  ...
  attr('MYCOMPPROP')
  property('MYCOMPPROP') {
    eda "<property name>"
  }
  ...
}

bom_target {
  ...
  common :identical_comparison => ←
  [:childEntity,"POS_NO","QUANTITY","X","Y","ROTATION","SIDE","MYCOMPPROP"]
  ...
  common :quantity_identical_comparison => ←
  [:childEntity,"POS_NO","SUM_QUANTITY","SUM_TAG","MYCOMPPROP"]
  ...
  attr('MYCOMPPROP')
  property('MYCOMPPROP') {
    common :instance_attr => true
    plm "<attribute name>"
  }
  ...
}
```

- `common :instance_attr`: Defines this attribute an instance attribute and shows it in the tab **Refdes BOM**.
- `common :identical_comparison`: Enables comparison in the tab **Refdes BOM**.
- `common :quantity_identical_comparison`: Enables comparison in in the tab **Quantity BOM**.

Do not change any of the options `common :comparison` and `common :quantity_comparison`.

## 7.3.5 Configuring validation of BOM items

Before uploading the BOM in the *BOM comparison* dialog, the BOM items are validated against certain criteria and shown in a final results window. Depending on the configured validation level, you can upload the BOM or not.

Introduced in a previous product version.

You can define the validation level as follows:

- `:warning`: The BOM items are reported as warning in the result window and the BOM can be uploaded. This is the default setting for all items.
- `:error`: The BOM items are reported as error in the result window and the BOM cannot be uploaded. You must fix the issues first.
- `:ignore`: The BOM items are not shown in the result window but are only logged in the event log.
- `false`: The BOM items are not validated.


© 2025 XPLM Solution GmbH

### Example

Configuration file: `<PRODUCT DIR>\config\datamodel.config`

```
bom_target {
...
  common :validate_missing_parts => :error
  common :validate_unreleased_parts => :warning
  common :validate_unassigned_parts => :error
  common :validate_obsolete_parts => :warning
...
}
```

- `:validate_missing_parts`: Defines the validation level for materials in the tab **Missing in PLM**. In the above example you would have to create the missing materials in SAP ECTR first before you can upload the BOM.
- `:validate_unreleased_parts`: Defines the validation level for materials in the tab **Unreleased in PLM**.
- `:validate_unassigned_parts`: Defines the validation level for components in the tab **Unassigned parts**. In the above example you would have to add the missing components to the library first before you can upload the BOM.
- `:validate_obsolete_parts`: Defines the validation level for materials in the tab **Obsolete in PLM**.

## 7.3.6 Configuring BOM report

In the *BOM comparison* dialog, you can configure the BOM report to a certain extend.

Introduced in a previous product version.

### Example: Defining template and email settings

Configuration file: `<PRODUCT DIR>\config\datamodel.config`

```
bom_target {
...
  common :bom_report => {
    :template => 'bom_report/bom_diff_template.xls',
    :properties_range => 'A1:D4',
    :row_range => 'A5:N5',
    #:attr_ids => ['POS_NO','ID','TAG','Name','QUANTITY','Length','Unit']
  }
...
  common :send_mail_command => proc { |ctx|
    ctx[:address] ||= "mail@address.com"
    ctx[:subject] ||= "BOM Report"
    ctx[:body] ||= "Attached BOM Report"
  ...
  }
...
}
```

- `common :bom_report`: Defines settings for the creation of the Excel and ASCII report.


© 2025 XPLM Solution GmbH

- `:template`: Defines the Excel template in the directory `<PRODUCT DIR>\appdata\bom_report\bom_diff_template.xls`.
- `:properties_range`: Defines a range in the Excel template in which to search for the placeholder for identifying the assembly.
- `:row_range`: Defines a range in the Excel template in which to search for placeholders for the attributes in the BOM rows.
- `:attr_ids`: Defines a list of attributes to be included in the ASCII report. By default all attributes are included. Remove the comment and define individual attributes if required.

- `common :send_mail_command`: Defines settings for composing an email and attaching the report.
  - `ctx[:address]`: Defines the email address.
  - `ctx[:subject]`: Defines the subject of the email.
  - `ctx[:body]`: Defines the content of the email.

#### Example: Defining target directory for BOM report

You can define the target directory for the BOM report with the environment variable `REPORT_DIR`. Set it in the file `init_env.bat`, for example `SET REPORT_DIR=C:\temp\bomreport`. Alternatively, define the variable under Windows.

### 7.3.7 Configuring BOM representation

You can define how the BOM is represented in SAP ECTR, for example the initial position number, the incremental steps or the sorting.

#### Example

Configuration file: `<PRODUCT DIR>\config\datamodel.config`

```
bom_target {
...
  common :initial_posno => 100
  common :posno_step => 5
  common :posno_sort_by_attributes => ['TAG', 'ID']
  plm :alternative => "01"
  plm :bom_usage => "2"
  plm :plant => ""
...
}
```

- `common :initial_posno`: Defines the value of the first BOM position number.
- `common :posno_step`: Defines the increment of the BOM position number.
- `common :posno_sort_by_attributes`: Defines the sorting of BOM positions. One or more attributes are possible. The positions are sorted according to the attribute sequence.
- `plm :alternative`: Defines the BOM alternative.
- `plm :bom_usage`: Defines the BOM usage for different areas in the company.
- `plm :plant`: Defines the BOM allocation for different plants in the company.


© 2025 XPLM Solution GmbH

### 7.3.8 Configuring additional BOM from CSV

You can configure an additional BOM from a CSV file that is processed with the exported BOM from the design. See the snippet `combine_bom_sources.snippet` for more information.


Introduced or updated with product version 1.1.7.x.

## 7.4 Parts synchronization

### 7.4.1 PartSync 1: Configuring new attributes for parts synchronization

Adding new attributes for parts synchronization is a common use case and is described here in an example.

#### Procedure

1. Add the new attribute in SAP ECTR.
2. Add the new parameter in Xpedition.
3. Go to the directory `<PRODUCT DIR>\config`.
4. Edit the file `classification.config`.


© 2025 XPLM Solution GmbH

5. Below the section `part_lib`, add two new template definitions with configuration options:

```
part_lib {
  template(:attr_to_eda) { |name, eda_name, plm_name, opts={}, &block|
    opts *= {:classification => true}
    attr(name) {}
    property(name) {
      common :blank_equal => true, :compare => true, :ui_visible => true
      common :type => opts[:type] if opts[:type]
      common :classification => opts[:classification] if opts[:classification]
      direction [:eda]
      merge [:plm, :eda]
      eda eda_name, :create => true, :write => true, :read => true
      eda :external_type => opts[:eda_external_type] if opts[:eda_external_type]
      plm plm_name, :create => false, :write => false, :read => true
      plm :external_type => opts[:plm_external_type] if opts[:plm_external_type]
      ergo(&block)
    }
  }

  template(:attr_to_plm) { |name, eda_name, plm_name, opts={}, &block|
    opts *= {:classification => true}
    attr(name) {}
    property(name) {
      common :blank_equal => true, :compare => true, :ui_visible => true
      common :type => opts[:type] if opts[:type]
      common :classification => opts[:classification] if opts[:classification]
      direction [:plm]
      merge [:eda, :plm]
      eda eda_name, :create => false, :write => false, :read => true
      eda :external_type => opts[:eda_external_type] if opts[:eda_external_type]
      plm plm_name, :create => true, :write => true
      plm :external_type => opts[:plm_external_type] if opts[:plm_external_type]
      ergo(&block)
    }
  }
}
```



Templates are a good way to simplify the configuration effort. Everything related to the attribute definition is kept in one place, for example placeholders for the internal attribute name (`name`), the field mapping (`eda_name` | `plm_name`), the synchronization direction (`direction` | `merge`) and other settings (`opts`).


© 2025 XPLM Solution GmbH

6. Depending on the synchronization direction, call the template with `attr_to_plm(...)` or `attr_to_eda(...)`, and pass the required values:

```
part_lib {
  template(:attr_to_plm) {...}
  template(:attr_to_eda) {...}
  attr_to_plm('<internal attribute name>', '<ecad field>', '<plm field>')
}
```



- By default, the synchronization direction is defined from Xpedition to SAP ECTR and you must use the template `attr_to_plm`. If you want to synchronize from SAP ECTR to Xpedition, you have to configure this explicitly, see PartSync 1: Configuring synchronization direction (p. 62) for more information. If you decide for this direction, use the template `attr_to_eda`.
- If you call the template directly under `part_lib`, the new attribute is treated a general attribute. If you call it under `subitem`, the attribute is treated a classification attribute. Both templates have this declaration enabled by default with the option `(opts *= {:classification => true})`. If you want to specifically disable this option for non-classification attributes, use the notation `attr_to_plm(..., :classification => false)`.

7. Save the file.
8. Start the integration module *Synchronize parts* and test the configuration.

### Result

The new attribute is defined and can be synchronized between the two systems.

## 7.4.2 PartSync 1: Configuring additional attributes for quick search

By default, you can only search for the attributes `ID` or `EcadPartNumber` in the quick search field. If required, you can configure other attributes to be searched for.



Introduced or updated with product version 1.1.0.x.

Add the option `common :quick_search => true` to the attribute you want to additionally search for.

### Example

Configuration file: `<PRODUCT DIR>\config\classification.config`

```
part_lib {
...
  property('Name') {
    common :quick_search => true
  }
...
}
```


© 2025 XPLM Solution GmbH

### 7.4.3 PartSync 1: Configuring synchronization direction

By default, the synchronization direction is defined from Xpedition to SAP ECTR. If you want it the other way, you have to configure this.

#### Procedure

1. Go to the directory `<PRODUCT DIR>\basis\sys\win\bin` and edit the file `sync_part_actions.rb`.
2. Check the definition in section `create_in_eda?` if it matches the following content:

   ```
   def create_in_eda?(model_item)
     !model_item.eda
   end
   ```

   Change the definition as shown, if it doesn't match and save the file.
3. Go to the directory `<PRODUCT DIR>\config` and edit the file `release.rb`.
4. In the section `libGui`, change the line `next true if mi.eda.nil?` to `next false if mi.eda.nil?` and save the file.
5. Restart the integration module *Synchronize parts* and search for materials in SAP ECTR.
6. Select an entry and verify the button **Synchronize** is active.

#### Result

The synchronization direction now works from SAP ECTR to Xpedition. Additional configuration may be required for the synchronization to work to full satisfaction, see Configuration options (p. 82) for more information.

### 7.4.4 PartSync 1: Configuring start dialog in Synchronize parts

When you start the integration module *Synchronize parts*, you can define whether the dialog for loading components with missing materials is shown or the dialog for adding components from the library. By default, nothing is shown.

#### Example

Configuration file: `<PRODUCT DIR>\config\ivs.properties`

```
ivs.ui.sync_parts.startup_wizard = no
```

- `no`: Nothing is shown (default)
- `hierarchy`: The dialog for adding components from the library is shown.
- `missing`: The dialog for loading components with missing materials is shown.

### 7.4.5 PartSync 1: Configuring ECAD/PLM labels in Synchronize parts

In the integration module Synchronize parts, you can configure the labels of the columns **ECAD** and **PLM** in the detail view.


Introduced in a previous product version.


© 2025 XPLM Solution GmbH

### Example

Configuration file: <PRODUCT DIR>\basis\dictionary\<language code>\ectr.txt

```
leftName: ECAD
rightName: PLM
```


© 2025 XPLM Solution GmbH

# 7.5 Siemens Xpedition Enterprise

## 7.5.1 Configuring colors in schematic PDF

You can define the colors used in schematic PDF files.

### Example

Configuration file: `<PRODUCT DIR>\config\datamodel.config`

```
file_pdf {
...
  eda :type => :plot, :color => :black_on_white
...
}
```

- `:color => :blank_on_white`: Black on white, print black text (default)
- `:color => :black_on_white_suppress_black_text`: Black on white, suppress black text
- `:color => :color_on_white`: Color on white
- `:color => :color_on_black`: Color on black

## 7.5.2 Configuring print options for fabrication PDFs

Define basic print options for creating fabrication PDFs directly in Xpedition Layout.

### Procedure

1. Start Xpedition Layout.
2. Select **File** > **Printing** > **Print Setup**.
3. Enable the options **Print entire design**, **Fit paper**, **No Footer** and **B&W only**.
4. Save the settings.

### Result

The print options are set.

### What to do next

You can define additional print options in the configuration file `<PRODUCT DIR>\config\datamodel.config` and the section `file_fabrication`.

## 7.5.3 Configuring PDFCreator printer

By default, the integration uses the installed PDFCreator printer to create fabrication and assembly PDFs (drill and placement plans).

Fabrication PDFs are configured in the section `file_fabrication`. Assembly PDFs are configured in the section `file_assy_pdf`.


© 2025 XPLM Solution GmbH

### Example

Configuration file: <PRODUCT DIR>\config\datamodel.config

```
file_fabrication {
...
  eda_pcb :printer_name => "IntegratePrinter"
  eda_pcb :print_to_folder => "C:\\temp\\integrate_pdf"
  eda_pcb :plot_pages => ←
  [{:display_scheme=>"Loc: FAB_DWG_SH1", :orientation=>"landscape", ←
  :size=>"A3"}, ←
  {:display_scheme=>"Loc: FAB_DWG_SH2", :orientation=>"landscape", ←
  :size=>"A3"}, ←
  {:display_scheme=>"Loc: DRILL_VIEW", :orientation=>"landscape", ←
  :size=>"A3"}]
...
}
```

- `:printer_name`: Defines the configured PDF printer.
- `:print_to_folder`: Defines the temporary spooling directory.
- `:plot_pages`: Defines a PDF file with defined pages, orientation and size.

## 7.5.4 Configuring extended print functionality

You can create fabrication PDFs or assembly PDFs with Mentor's extended print functionality. This method requires a separate license.

The extended print function is commented out by default and you must activate it.

Locate the sections `file_fabrication` for creation of fabrication PDFs and `file_assy_pdf` for creation of assembly PDFs.

To use the extended print functionality, uncomment the lines `extended_plot_setup` and `extended_plot_settings_file` and comment out the lines `eda_pcb :printer_name`, `eda_pcb :print_to_folder` and `eda_pcb :plot_pages` for the native print settings as shown.


© 2025 XPLM Solution GmbH

### Example

Configuration file: <PRODUCT DIR>\config\datamodel.config

```
file_assy_pdf {
...
  eda_pcb :extended_plot_setup => ←
  File.join(Integrate2::PRODUCT_CUSTOM_HOME, ←
  'expedition_pcb', 'assembly.pcf')
  eda_pcb :extended_plot_settings_file => File.join(ENV['APPDATA'], ←
  'mgc\ezprint\defaultEzPrintConfig.xml')
  #eda_pcb :printer_name => "IntegratePrinter"
  #eda_pcb :print_to_folder => "C:\\temp\\integrate_pdf"
  #eda_pcb :plot_pages => [{:display_scheme=>"Loc: ASSEMBLY_TOP", ←
  :orientation=>"landscape", ←
  :size=>"A3"}, {:display_scheme=>"Loc: ASSEMBLY_BOTTOM", ←
  :orientation=>"landscape", :size=>"A3"}]
...
}
```

## 7.5.5 Configuring Gerber data

To create Gerber data, the integration uses the VX Gerber Output Utility.

### Procedure

1. Start Xpedition Layout and open a design.
2. Select **Output** > **Gerber**.
3. Adjust settings and output files.
4. Click **OK**.

### Result

The Gerber data is configured.

## 7.5.6 Configuring STEP data

For STEP export, the integration simulates clickable export actions in the menu **3D** of Xpedition Layout. You can also export STEP manually and have the integration upload the file from a defined path within the project directory.

Introduced or updated with product version 1.1.3.x.

For the export via the menu **3D**, the 3D plugin must be installed and the 3D view available when the layout is open.

### Example

Configuration file: <PRODUCT DIR>\config\datamodel.config

#### Automatic export 1

The following configuration is for automated STEP export with user-defined options from the menu **3D** (default setting):


© 2025 XPLM Solution GmbH

Export STEP from the menu **3D** once with defined settings manually. The next time you export STEP via the integration, these settings are used.

```
file_step {
...
  # common :find => imod_lambda { File.join(design.dir, design.name + '.step') }
  eda_pcb :type => :step
  #  eda_pcb :step_export_config => proc { |exporter|
  #  exporter.ExportMechanicalModels = true
  #  exporter.ExportCopperPads = true
  #  exporter.ExportCopperOuter = true
  #  exporter.ExportCopperInternal = true
  #  }
...
```

**Automatic export 2**

The following configuration is for automated STEP export with fix-defined options from the menu **3D** (all other options in the menu are ignored):

These are the only options you can set. Contact https://support.sap.com if you need more options.

```
file_step {
...
  # common :find => imod_lambda { File.join(design.dir, design.name + '.step') }
  eda_pcb :type => :step
  eda_pcb :step_export_config => proc { |exporter|
  exporter.ExportMechanicalModels = true
  exporter.ExportCopperPads = true
  exporter.ExportCopperOuter = true
  exporter.ExportCopperInternal = true
  }
...
```

**Manual export**

The following configuration expects the manually exported STEP file in the directory `PCB\Output` within the current project directory. Comment/uncomment the lines as shown.


© 2025 XPLM Solution GmbH

The method `File.join ()` assembles a path to the STEP file. For the current project directory you can use the variable `design.dir` and for the design name the variable `design.name`. Make sure to encapsulate folders and files with 'single quotes'.

```
file_step {
...
  common :find => imod_lambda { File.join(design.dir, 'PCB', 'Output', design.name +
 '.step') }
  # eda_pcb :type => :step
  # eda_pcb :step_export_config => proc { |exporter|
  # exporter.ExportMechanicalModels = true
  # exporter.ExportCopperPads = true
  # exporter.ExportCopperOuter = true
  # exporter.ExportCopperInternal = true
  # }
...
```

### 7.5.7 Configuring IPC-356 data

To create IPC-356 data, you can use an additional output file and configure it as described in snippet `export_ipc356.snippet`.

Introduced or updated with product version 1.1.4.x.

**Related links**
Configuring extra files for upload (p. 51)
Other configuration files (p. 23)

### 7.5.8 Configuring CCZ/CCE data

To create CCZ/CCE data, you can use an additional output file and configure it as described in snippet `export_ccz.snippet`. The data is created with the last scheme or setting defined in the export dialog of Xpedition Layout.

Introduced or updated with product version 1.1.4.x.

**Related links**
Configuring extra files for upload (p. 51)
Other configuration files (p. 23)

### 7.5.9 Configuring DXF data

To create DXF data, you can use an additional output file and configure it as described in snippet `export_dxf.snippet`.

Introduced or updated with product version 1.1.4.x.

**Related links**
Configuring extra files for upload (p. 51)
Other configuration files (p. 23)


© 2025 XPLM Solution GmbH

### 7.5.10 Configuring assembly data

You can define which types of assembly data the integration saves in SAP ECTR.

The default configuration exports a Midtron `.gencad` output, creates a ZIP file and saves the result in SAP ECTR. You can also create and save output files for `.ais` and `.ate`. Set `true|false` for the output files as required.

#### Example

Configuration file: `<PRODUCT DIR>\config\datamodel.config`

```
file_assy_data {
...
  eda_pcb :ais => false
  eda_pcb :ate => false
  eda_pcb :gencad => true
...
}
```

- `:gencad => true|false`: Defines Midtron `.gencad` output (default)
- `:ais => true|false`: Defines generic `.ais` output
- `:ate => true|false`: Defines generic `.ate` output

### 7.5.11 Configuring BOM export

By default, the integration uses Variant Manager for exporting the BOM, but Variant Manager requires a license. If you don't have such a license, you can use the alternative export method with Part Lister.

When using Variant Manager, check for the correct delimiter in your settings (tab or comma separated). By default, the integration is configured for tabs.

When using Part Lister, the parameters in the BOM report are different compared to Variant Manager. Change as shown.



In previous Xpedition versions, Part Lister had the limitation that parts that used a replacement number in a variant were not exported correctly. Instead of the replacement number, the original component number was still exported and was therefore incorrect. This limitation is no longer present in versions VX2.8+. To summarize, if you use Xpedition older than VX2.8, the use of Part Lister as alternative to Variant Manager is not recommended. For VX2.8+, you can use Part Lister without limitation.

#### Example (using Variant Manager and delimiter comma)

```
bom_source {
...
  eda :bom_report_delimiter => ","
...
}
```


© 2025 XPLM Solution GmbH

### Example (using Part Lister)

Configuration file: <PRODUCT DIR>\config\datamodel.config

```
bom_source {
...
  eda :bom_extract_method => :partlister
...
  property('TAG') {
    eda "Ref Designator"
    eda_pcb "REFDES"
  }
...
}
```

```
part_in_bom {
...
  property('Name') {
    eda "Description"
    ...
  }
...
}
```

To get the correct parameters, make a variant active in Variant Manager, select **Tools** > **Part Lister** and run the BOM report, or get the parameter names directly from the tab **Columns**.

## 7.5.12 Configuring non-fitted components

By default, the integration processes also non-fitted components and shows them in the *BOM comparison* dialog. If you don't need this information, you can disable processing of non-fitted components.

Introduced or updated with product version 1.1.7.x.

### Example

Configuration file: <PRODUCT DIR>\config\datamodel.config

```
bom_source {
...
  eda :skip_load_not_fitted => true
...
}
```


© 2025 XPLM Solution GmbH

# 7.6 SAP Engineering Control Center

## 7.6.1 Configuring creation mode for datamodel objects

The integration can create documents and materials in SAP ECTR with or without user interaction. You can define the specific creation mode for each datamodel object.

In the interactive mode, the integration opens the object creation dialog in SAP ECTR and you can change values and parameters.

In the batch mode, objects are created automatically with the configured settings.

By default, the batch mode is used. To activate the interactive mode on the datamodel object, use the option `plm :create_interactive => true`.

### Example

```
doc_inno {
...
  plm :create_interactive => true
...
}
```

## 7.6.2 Saving templates

In the integration module *New ECAD project*, you can select templates saved in SAP ECTR as a basis for new designs.

### Before you start

A design template for Xpedition is available. If not, create one first.

### Procedure

1. In SAP ECTR, create a new document to store the schematic template.
2. In the list **DType Group**, select the corresponding ECAD application.
3. In the list **DType**, select the entry **Schematic Template**.
4. Click **Browse** and attach the template file created earlier.
5. In the field **Description**, enter a meaningful name to identify this template later.
6. Release the document to status *Final Release (FR)*.
7. Click **OK**.
8. Repeat the previous steps for the layout template and use as **DType** the entry **Layout Template**.

### Result

In the integration module *New ECAD project* and tab **PLM template**, click **Load**. The integration searches for released templates in SAP ECTR and shows the results.

**Related links**
Creating templates (p. 38)
Setting up status net (p. 39)


© 2025 XPLM Solution GmbH

### 7.6.3 Configuring templates

In the integration module *New ECAD project*, you can select templates saved in SAP ECTR as basis for new designs. By default, the integration filters no templates in the selection list. You can add a filter to show only templates matching a given name pattern or a certain status.

Locate the sections `doc_template_sch` for schematic templates and `doc_template_pcb` for layout templates. Change `plm :search_default` to add a filter.

#### Example

Configuration file: `<PRODUCT DIR>\config\datamodel.config`

```
doc_template_sch {
...
  property('Name') {
  plm "SAP_DESCRIPTION"
  plm :search_default => '*production*'
  }
...
  property('Status') {
    plm "SAP_DOCSTATUS"
    plm :search_default => 'DE;EX;FR'
  }
...
}
```

- `plm :search_default => '*production*'`: Defines a filter for the attribute `Name` in SAP ECTR. If activated, only templates with the defined pattern are shown in the selection list of the dialog.
- `plm :search_default => 'FR'`: Defines a filter for the attribute `Status` in SAP ECTR. If activated, only templates with the defined statuses are shown in the selection list of the dialog.

### 7.6.4 Configuring alternative BOM items

With the integration, you can create alternative BOM items in SAP.

Components that create alternative BOM items in SAP require additional properties to be defined in the ECAD application. You can add these properties in the library as general information or on the component instance, for example *ALTERNATE_1*, *ALTERNATE_2*, *ALTERNATE_3*. The value of these properties must match an existing SAP material number of the alternative material to be used. See also https://blogs.sap.com/2014/02/03/concept-of-alternative-item-group-functionality-in-bom/.

In SAP, the following fields are filled in the original and alternative BOM item:

- AltItemGroup
- Priority
- Strategy
- Usage Probability

BOM items with alternatives are gathered in groups (**AltItemGroup**), starting for example with the value *AA*. You can configure this initial group. The rest of the fields are filled automatically and should not be changed.

In the *BOM comparison* dialog, these fields are hidden by default. You can show the corresponding columns by clicking the button to the right and select the following columns:

- AG_NAME


© 2025 XPLM Solution GmbH

- AG_PRIO
- AG_STRATEGY
- AG_PROBABILITY

On the ECAD side, you can configure the property prefix to be used.

Alternative BOM items are not enabled by default. You can enable them as shown in the example.

### Example

Configuration file: `<PRODUCT DIR>\config\datamodel.config`

In the section `bom_source`, uncomment the line `eda :alternate_property` and change the regular expression `/^ALTERNATE_/` to match the property prefix on the ECAD side according to your requirements:

```
bom_source {
...
  eda :alternate_property => /^ALTERNATE_/
...
}
```

In the section `bom_target`, uncomment the line `plm :alternate_initial_group` and change the value `AA` for the initial material group in SAP according to your requirements:

```
bom_target {
...
  plm :alternate_initial_group => 'AA'
...
}
```

## 7.6.5 Configuring handling of BOM items in SAP

By default, the integration only reads and updates its own BOM items of a specific BOM in SAP. Manually entered BOM items are ignored, as well as BOM items created by other applications in the same BOM, for example MCAD. You can configure the integration to respect this specific use cases.


Introduced in a previous product version.

### Examples

Configuration file: `<PRODUCT DIR>\config\datamodel.config`

Integration reads and updates only its own BOM items in SAP (default, not to be set and shown just for information):

```
bom_target {
...
  plm :cad_ind => true
...
}
```


© 2025 XPLM Solution GmbH

Integration reads and updates all BOM items in SAP, removing others:

```
bom_target {
...
  plm :cad_ind => false
...
}
```

Integration reads an updates only BOM items flagged with *ECAD* in SAP:

```
bom_target {
...
  plm :cad_ind_text2 => 'ECAD'
...
}
```

Integration reads and updates only BOM items flagged with *ECAD* in SAP, removing others that are not explicitly flagged:

```
bom_source {
...
  plm :cad_ind => false
  plm :cad_ind_text2 => 'ECAD'
...
}
```

## 7.6.6 Bypassing SAP license mechanism

By default, the integration uses the license mechanism in SAP. There may be situations where you want to use the XPLM license mechanism instead, for example during testing or setting up the integration. Please note that for productive use, you must obtain and use the official licenses from SAP.

### Before you start

Collect installation media:

- Contact support under support@xplm.com and request a new license file based on your needs:

| Type | License | Description |
|---|---|---|
| Module | eda_logic | Use integration module *Save* to manage the design in SAP ECTR, create and upload schematic-related data, and extract and upload the BOM. |
| Module | eda_layout | Use integration module *Save* to create and upload fabrication and assembly data, and add layout information to the BOM. |
| Module | eda_library | Use integration module *Synchronize parts* to manage library parts in SAP ECTR and synchronize attributes.<br>If you create missing parts in SAP ECTR via the *BOM comparison* dialog, this license is not required, as no library parts are being modified. |
| Connector | dxdesigner | Interact with Xpedition Designer functions via API. |
| Connector | expedition_pcb | Interact with Xpedition Layout functions via API. |
| Connector | dxdatabook\|mentor_lmc | Interact with the library via API. |


© 2025 XPLM Solution GmbH

| Type | License | Description |
|---|---|---|
| Connector | ectr5 | Interact with SAP ECTR functions via API. |

The use of a license depends on which action a user performs in an integration module and from which ECAD tool it is started.

- Login to the **XPLM Support Portal** at https://support.xplm.com.
- Go to the tab **Download Portal**.
- Download License Client and documentation from `Products/Licensing`.
- Download the license file from `Customer/<Company name>/License`.

### Procedure

1. Installing License Client:
   a) Install License Client normally or in silent mode. See the supplied installation guide for more information.
   b) Restart the client computer.
2. Installing the license file:
   a) After the restart, rename the downloaded license file to `*.lic`.
   b) Copy the license file to the directory `<ECTR DIR>\xml`, overwriting the existing file.
3. Go to the directory `<PRODUCT DIR>\config\environments\plm` and edit the file `production.rb`.
4. Add `:license => :xplm` as follows:

```
...
runtime[:plm].options.merge!(
  :license => :xplm,
  :app_type => 'xdx'
  )
...
```

5. Save the file and start an integration module.
6. Check that all integration features work.

### Result

The integration now uses the XPLM license mechanism.

## 7.6.7 Distributing integration with Subversion

It is common to manage and distribute the ECTR and integration code with a version control system like Subversion. If this method is used to distribute the software to users, you must configure the Subversion client to include certain Ruby files. Otherwise, not everything will be synchronized with the repository and the integration fails to work. Then you must register a specific DLL on the client computer to make the integration work.

### About this task

By default, the Subversion client ignores certain files. One of these ignored file extension is `.so`, a shared Ruby library that is not normally saved in the repository. The Ruby distribution included in the integration contains many of these `.so` binary files, and they would all be lost unless configured otherwise.


© 2025 XPLM Solution GmbH

The DLL `ectr_wrapper.dll` is a .NET assembly. To register the DLL, use the tool `RegAsm.exe` from the .NET 4 framework. Do not use `regsv32.exe` to register this DLL.

### Procedure

1. Configuring Subversion:
   a) On Windows systems, open the Subversion configuration file `%APPDATA%\Subversion\config` in a text editor.
   b) Locate the line `# global-ignores = *.o *.lo *.la *.al .libs *.so *.so.[0-9]* *.a *.pyc *.pyo __pycache__`.
   c) Uncomment the line, change it to `global-ignores = xxx` and save the file.
   d) Add the ECTR and integration code with `svn add --force` for inclusion into the repository.
   e) Commit the files.
      ➔ The ECTR and integration code was correctly synchronized with the repository and no files are missing.
2. On the client computer, update the ECTR and integration code from the repository or check out the repository.
3. Registering the DLL `ectr_wrapper.dll`:
   a) Make sure the DLL `plm_conn.dll` in `<ECTR DIR>\basis\sys\win\lib64` is registered.
   b) Copy the DLL `plm_conn.dll` from `<ECTR DIR>\basis\sys\win\lib64` to `<ECTR DIR>\addons\ecad\basis\sys\win\bin`.
   c) Start a CMD window with administrative rights and navigate within this window to the directory `<ECTR DIR>\addons\ecad\basis\sys\win\bin`.
   d) Execute the command `C:\Windows\Microsoft.NET\Framework64\v4.0.30319\RegAsm.exe ectr_wrapper.dll /codebase /verbose /tlb`.
      ➔ The DLL `ectr_wrapper.dll` is registered.

### Result

Integration code that is manually distributed via Subversion works on the client computers being used.

## 7.6.8 Working with backend configuration support

The backend configuration support for SAP ECTR is enabled by default in the integration. If backend support is enabled and set up in SAP ECTR, the integration will use this configuration directory. See the following information for working with backend support.

Introduced or updated with product version 1.1.7.x.

### Disabling backend support

If backend support is disabled in SAP ECTR, the integration will use the default configuration directory in the SAP ECTR installation directory. You can also disable backend support for the integration conveniently via an environment variable in `init_env.bat` or under Windows.


© 2025 XPLM Solution GmbH

Environment file: `<PRODUCT DIR>\config\environments\init_env.bat`

```
...
SET INTEGRATE_ECTR5_BACKEND_SUPPORT = FALSE
...
```

If set `FALSE`, the integration will not honour a backend support directory and will use the default configuration directory in the SAP ECTR installation directory – even if backend support is still enabled in SAP ECTR.

### Setting correct library connection

When starting an integration module, a batch file is started which in turn starts `init_env.bat` and then starts Ruby, which reads `production.rb` and other files.

Only after this, the backend configuration is evaluated and re-read. Because this is Ruby process now, only Ruby files can be read and no batch files. This is why you can't define the library connection in `init_env.bat` when using backend configuration.

Instead, if you should define the connection string in `<PRODUCT DIR>\config\environments\production.rb` as follows:

```
ENV['LIBDB_CONNECTION_STRING']="<MY CONNECTION STRING>"
load File.join(File.dirname(__FILE__),'altiumdesigner_ectr5.rb')
```

Alternatively, define the library connection via an environment variable under Windows. See PartSync 1: Setting up library connection (p. 34) for more information.

## 7.6.9 Configuring ECN usage

By default, the integration uses the active ECN defined in SAP ECTR. You can configure the integration to show a selection in the *Save* dialog, or use an already assigned ECN of the current document.

Introduced or updated with product version 1.1.7.x.

### Example

Configuration file: `<PRODUCT DIR>\config\datamodel.config`

```
...
plm :change_number_source => :plm
...
```

- `:plm`: Uses the active ECN in SAP ECTR (default).
- `:ui`: Shows a selection in the *Save* dialog.
- `:document`: Uses an assigned ECN of the current document (if available).
  - On object creation, the active ECN in SAP ECTR is always used.
  - On document update, the ECN assigned to the current document is used.
  - On material and BOM update, the ECN of the corresponding document is used, for example an update of *part_pcb* metadata uses the ECN assigned to *doc_pcb*. Or, an update of *part_group* metadata and the uploaded BOM uses the ECN assigned to *doc_group*.

**Related links**
Important configuration files (p. 20)


© 2025 XPLM Solution GmbH

# 8 Update & removal

## 8.1 Updating integration

You can update the integration code to a new version without loosing your configuration. After the update, you must compare old and new configuration files and merge changes.

### Before you start

Backup the installation directory of the integration.

### Procedure

1. Install the new integration version in the existing installation directory.
   Existing configuration and template files are not overwritten. New configuration and template files are saved in the directories `template-customize` and `template-templates` as follows:

   **New configuration files**

   - `<ECTR DIR>\addons\ecad\template-customize`
   - `<ECTR DIR>\applications\x00\template-customize`
   - `<ECTR DIR>\applications\xdx\template-customize`

   **New template files**

   `<ECTR DIR>\applications\xdx\template-templates`

2. To activate new functions of the new integration version, compare the existing configuration files in the `config` directories under the respective path with the new files in the `template-customize` directories. Merge the changes with the existing files.

   **Existing configuration directories**

   - `<ECTR DIR>\addons\ecad\customize\config`
   - `<ECTR DIR>\applications\x00\customize\config`
   - `<ECTR DIR>\applications\xdx\customize\config`

3. **Optional**: Copy new template files (if available) from the directory `template-templates` to the directory `templates`.

### Result

The integration is updated and the existing configuration is merged with the new one.

## 8.2 Uninstalling integration

An uninstall program is provided during the installation of the integration.

### Procedure

1. Select **Control Panel** > **All Control Panel Items** > **Programs and Features**.
2. Select **SAP Engineering Control Center Interface to ECAD** and click **Uninstall/Change**.

### Result

The integration is uninstalled.


© 2025 XPLM Solution GmbH

# 9 Troubleshooting

## 9.1 Solving integration issues

There are many ways why the integration does not work as expected. The following non-exhaustive list gives you some hints for solving integration issues.

| Issue | Solution |
|---|---|
| I have installed the integration but don't see the integration menu. | Re-check menu setup.<br>See Installing integration menu in Xpedition Designer (p. 31) for more information. |
| How can I check the available integration variables? | To check the available integration variables, open a CMD window, type `set int` and press **Enter**. The variables start with `INTEGRATE_*`.<br>See Integration variables (p. 16) for more information. |
| I don't know which mandatory fields are needed when creating objects in SAP ECTR. | In SAP ECTR, create objects manually and check the mandatory fields and their default values.<br>See Installation (p. 30) for more information. |
| I have set up up the correct library path, but *Synchronize parts* does not work. | Try the following:If you can, update to integration version 1.1.7.0<br>■ Check the system requirements.<br>■ Check the mapping of attributes.<br>■ Check the library setup in Xpedition, especially the connection to the database.<br>See System requirements (p. 27) and PartSync 1: Setting up library connection (p. 34) for more information. |
| I have increased the log level in the IVS preferences and in the integration module, but it is not recognized. | Check if the variable `INTEGRATE_DEBUG` or `INTEGRATE_DEEP_DEBUG` is set in one of the locations described and overrides the settings.<br>See Integration variables (p. 16) for more information. |
| IVS fails to start. | When you start an integration module, IVS is started in the background (if not already active). Sometimes, IVS fails to start and shows the error message: `Address already in use: JVM_Bind (Port = 51346)`.<br>Try the following:<br>**1.** Check the task bar for multiple IVS icons. Close all IVS icons in the task bar and try again.<br>**2.** If (1) does not solve the issue, check the task manager for hanging `java.exe` processes. Stop them and try again.<br>**3.** If (2) does not solve the issue, delete the file `%USERPROFILE%\.ivs_product_ports` and try again.<br>**4.** If (3) does not solve the issue, restart computer and try again. |
| I've tried everything. What else can I do? | Contact the support. See the chapter **Troubleshooting** in the User Guide for more information. |


© 2025 XPLM Solution GmbH

## 9.2 Solving blacklisted BAPIs in SAP S/4HANA

There is a blacklist in Hana systems, in which all BAPIs (RFC calls) are initially marked as blocked. If a server is restarted, the blacklist has to be switched off manually. See also SAP Note 2295840 for detailed information.

### About this task

By default, the integration does not use any blacklisted RFCs, but these can be used in customisations, for example the RFC *CSAP_BOM_MAINTAIN*. If your customisations require blacklisted RFCs, perform the following procedure after a restart of the Hana system.

### Procedure

1. Start SAP GUI and log in as administrator.
2. Open transaction **RZ11** and enter the parameter **ucon/rfc/check_blacklist**.
   ➔ The dialog **Display Profile Parameter Details** appears.
3. Click **Change Value**.
   ➔ The dialog **Change Parameter Value** appears.
4. If **Current Value** is 0, the blacklist is disabled and all BAPIs are active. You can close the dialog and log off from SAP GUI.
5. If **Current Value** is 1, the blacklist is enabled and all BAPIs are inactive. You probably will get a lot of errors when using the integration with SAP. Proceed as follows:
   a) In **New Value**, enter 0 and click **Save**.
   b) Restart the integration and try the SAP-related function that caused the error.
   c) Verify that everything works as designed.

### Result

Blacklisted BAPIs are enabled again and the integration works.


© 2025 XPLM Solution GmbH

# 10 Reference information


© 2025 XPLM Solution GmbH

## 10.1 Configuration options

Table 7: Options for attr() and property() definitions

| Option | Values | Description | Validity |
|---|---|---|---|
| `:type` | ■ `Boolean`<br>■ `String`<br>■ `Integer`<br>■ `Float`<br>■ `Time` | Defines how field values are parsed and compared. For example "2.12" is equal to "2.120" if set to `Float` and is not equal if set to `String`. | Valid for all datamodel objects and integration modules. |
| `direction` | ■ `[:eda]`<br>■ `[:plm]` | Defines the synchronization direction of attributes:<br>■ `[:eda]`: From SAP ECTR to Xpedition<br>■ `[:plm]`: From Xpedition to SAP ECTR<br>This parameter is only evaluated if attributes on both sides have a value and is relevant for update only. | |
| `merge` | ■ `[:eda]`<br>■ `[:plm]`<br>■ `[:user]`<br>■ `[:auto]` | If the attribute values in Xpedition and SAP ECTR are different, this parameter defines, how the integration selects the winning value to be synchronized. The parameter accepts a list of values, of which the first non-empty value is used.<br>For example, `merge [:eda, :plm]` uses the value in Xpedition as a first result. If this value is empty, the value in SAP ECTR is used next. The resulting value will then be written to the target side defined in the parameter `direction`.<br>`:user` defines a value entered by the user.<br>`:auto` defines an auto-generated value. This option is always added as a last merge statement. | |
| `:blank_equal` | ■ `true`<br>■ `false` | If set to `true`, NULL and empty string values are considered as equal.<br>Recommended is `true`. | |


© 2025 XPLM Solution GmbH

| Option | Values | Description | Validity |
|---|---|---|---|
| `:compare` | ■ `true`<br>■ `false` | If set to `true`, integration compares the attribute values in Xpedition and SAP ECTR, shows the differences and tries to synchronize them.<br>If set to `false`, the integration just shows the attribute. | Valid for datamodel object `part_lib` in the integration module *Synchronize parts* only. |
| `:ui_visible` | ■ `true`<br>■ `false` | If set to `true`, the attribute is shown in the UI.<br>If set to `false`, the attribute is hidden from the UI but is still synchronized. | |
| `:classification` | ■ `true`<br>■ `false` | If set to `true`, the attribute is considered a classification attribute. Required for certain PLM systems. To be used on the `plm` statement. | |

Table 8: Options for common `eda` and `plm` statements

| Option | Values | Description | Validity |
|---|---|---|---|
| `:create` | ■ `true`<br>■ `false` | Transfers the attribute at object creation to Xpedition or SAP ECTR. | Valid for all datamodel objects and integration modules. |
| `:write` | ■ `true`<br>■ `false` | Transfers the attribute at object update to Xpedition or SAP ECTR. | |
| `:read` | ■ `true`<br>■ `false` | Reads the attribute from Xpedition or SAP ECTR. | |
| `:none` | | Sets `:create`, `:write` and `:read` to `false`. Additionally some other values are set.<br>This parameter is evaluated for `property()` definitions only. | |
| `:external_type` | See `:type` | Defines how field values are parsed and compared. Can be applied separately to `eda` or `plm` statements, for example `eda <component ID>, :external_type => Float` | |

Table 9: Options for `eda` and `eda_pcb` statements used in output definitions

| Option | Values | Description | Validity |
|---|---|---|---|
| `:zip` | ■ `true`<br>■ `false` | If required, zips the output. | Valid for all sections that start with `file_`. |


© 2025 XPLM Solution GmbH