creo® illustrate

Creo® Illustrate Installation and Configuration Guide

7.1.0.0

PTC Inc., 121 Seaport Blvd, Boston, MA 02210 USA

Contents

1

About the Installation and Configuration Guide

Creo Illustrate is one of PTC's technical illustration applications. It creates 3D technical illustrations from CAD files and their associated bills of material. Creo Illustrate runs standalone or in the context of a Windchill server. This guide describes how to generate and configure licenses, and how to install and maintain Creo Illustrate. A section for system administrators on deploying Creo Illustrate describes performing a silent installation and customizing the installer.

System administrators who are performing advanced deployment tasks for Creo Illustrate must be experienced in application installations and must have a good understanding of operating systems.

How to Use This Guide

This guide supplements the descriptions in the installer and provides installation prerequisites, instructions, and reference information. To generate or update licenses and then install or reconfigure Creo Illustrate, see the following chapters:

Chapter	Role	Information
Overview	All users	Describes the materials in your software shipment and lists the prerequisites for installation.
Quick-Start Installation	Existing users	Describes a quick-start approach for users who have previously installed license management and product software.
Obtaining and Configuring Licenses	System administrators	Describes how to generate or update licenses for Creo Illustrate and how to install PTC License Server.
Installing Creo Illustrate	All users	Describes a simple installation process for Creo Illustrate.
Starting Creo Illustrate and Modifying an Installation	All users	Describes the steps to start Creo Illustrate and the procedure to uninstall the software.
Deploying Creo Illustrate and Customizing the Installation	System administrators	Describes methods for deploying Creo Illustrate and outlines strategies for silent installation.
Updating an Installation	All users	Describes how to update Creo Illustrate on your system.

Documentation Conventions

PTC documentation uses the following conventions:

Convention	Item	Example
Bold	Menu paths, dialog box options, buttons, and other selectable elements from the user interface	Click **File** ▸ **New**. Click **OK**.
Courier	User input, system messages, directories, and file names	`Processing completed.`
Courier with less-than and greater-than symbols (< >)	Variables for which an appropriate value is substituted	`output= <LOADPOINT>`

Note

Examples of command-line arguments may contain hidden line breaks to fit on the page.

Related Documentation

The following documents on the Reference Documents page may be helpful as you proceed with the installation:

- *FlexNet Publisher License Administration Guide* that discusses the third-party license management software for distributing Creo licenses
- *Installing and Configuring the Standalone License Server* that discusses the installation of the lmadmin-based PTC License Server

Technical Support

Contact PTC Technical Support via the PTC website, phone, fax, or e-mail if you encounter problems using your software. You can log a new case or track an existing case or SPR (Software Performance Report) using the PTC webpage at www.ptc.com/en/support.

You must have a Service Contract Number (SCN) before you can receive technical support. If you do not have a SCN, contact PTC License Management using the instructions in the *Customer Support Guide*.

Documentation for PTC Products

PTC provides documentation for download at PTC.com on the Reference Documents page. The following forms of documentation are available:

- Context-sensitive Help with a search facility and quick links to helpful information.
- *Creo Illustrate Installation and Configuration Guide* and other books as PDF files. To view and print the books, you must have Adobe Acrobat Reader installed.

Press F1 or click ? on a user interface item to get context-sensitive Help.

To access all PTC Documentation from PTC.com, you must have a valid user account. Go to the Create New PTC eSupport Account page to request a user account or call Customer Support. For worldwide phone numbers click **Contact** from*Customer Support Guide*.

Feedback to Documentation

PTC welcomes your suggestions and comments on its documentation—send feedback to the following address:

`mcad-documentation@ptc.com`

Please include the name of the application and its release with your comments. For online books, provide the book title.

Additionally, you can report any documentation issues using the online Case Logger tool. Select **Help Center / Documentation** from the **Technical Area** list when prompted for this detail. Upon submission of all information, a case number is returned immediately.

2

Overview

This chapter describes the materials in your software shipment. It also outlines the prerequisites for installing and running the software.

What You Receive

PTC (Parametric Technology Corporation) sends the following materials related to your software order:

- Software Order Confirmation e–mail—Before the receipt of your PTC software, you will receive an e-mail containing all the details of your order.
- Software DVD—Your order contains one or more DVDs for each product purchased. These programs are included on the Creo Illustrate DVD:
  - Creo Illustrate 64-bit

About Creo Illustrate Licensing

The Creo Illustrate license enables all technical illustration authoring functions and interoperability with Windchill. The optional Schematic Illustrator and Massive Assembly modules are licensed separately.

Depending on the Creo Illustrate license configuration set up by the system administrator, you can retrieve license options for your workstation according to your tasks. For example, the Massive Assembly license can be set to load at startup or when required. This flexibility provides efficient license sharing. To work offline, you must go online and borrow temporary Creo Illustrate licenses.

Massive Assembly Requirements

The Massive Assembly option (FlexNet license feature `280`) is required to perform operations which consumes large memory.

Note

Creo Illustrate Professional comes with Massive Assemblies capability.

The option is required if addressed memory exceeds the following thresholds while opening large data sets in Creo Illustrate:

Release	Creo Illustrate
Creo Illustrate 1.0	2.0 Gb
Creo Illustrate 2.0	3.5 Gb
Creo Illustrate 3.0 F000 to M020	8.0 Gb
Creo Illustrate 3.0 M030 or later	16.0 Gb

Before You Begin

Before you install Creo Illustrate, make sure that all the prerequisites are met:

- You have opened an online account at PTC.com (Administrators of Creo Illustrate only).
- You have received the Software Confirmation Order e-mail with the product code or the Sales Order Number for your shipment. Alternatively, you must have received your License Pack via e-mail from PTC License Management (Administrators of Creo Illustrate only).
- You have received the product DVD in your software shipment. If not, follow the link on your Software Confirmation Order e-mail to download the installation package to a folder on the local or the network computer. Alternatively, download Creo Illustrate from PTC.com. See Downloading Creo Illustrate.
- You have checked the following locations for information on support of various platforms, languages, and toolkits:
  - The *Technical Graphics Compendium*.
  - The System Prerequisites.

You must install PTC License Server Manager before installing Creo Illustrate unless you have purchased node-locked licenses.

See Obtaining and Configuring Licenses.

Opening a PTC Online Account

You must have a PTC online account to generate a license and install Creo Illustrate. Use the following instructions to create an online account:

1. On a computer connected to the Internet, go to the Create New PTC eSupport Account page.
2. Fill in any empty boxes.
3. Click **Create Account**. A confirmation page indicates a successful account creation.
4. Review and print this confirmation for your record. A confirmation of your account is sent to your e-mail address.

Resetting Your Password

If at any time you do not remember your PTC.com password, follow these instructions:

1. Open the Reset Your Password page.
2. Type your user name.
3. Click **Continue**. A URL to reset your password is sent to your e-mail address.

System Prerequisites

This section describes the prerequisites for hardware, operating systems, graphics cards, and software.

Hardware and Operating System Requirements

You create illustrations from imported CAD files. The hardware requirements partly depend on the type and size of the CAD files you import. Hardware drivers not updated by the manufacturer in the last four years may not meet the needs of Creo Illustrate (see CS254885 for more information).

The minimum requirements are sufficient for small-sized or average-sized files. For optimum performance, meet these recommended specifications:

Item	Minimum Requirement	Recommended Specification
Random access memory (RAM)	2 GB	3 GB to 8 GB
Central processing unit (CPU) or Core	Single-core, 1-GHz x86 Intel compatible	Dual-core or greater, 2-GHz x86 64-bit Intel compatible
Disk space	1 GB for a 64-bit platform	2 GB for the installation Additional disk space for data
Operating system	Windows 10	Windows 10 64-bit

The Platform Support page contains information on requirements for hardware, graphics drivers, and operating systems for PTC products, including Creo Illustrate.

Graphics Requirements

You must have OpenGL library version 1.1 or later installed on your machine. Make sure you have the latest drivers for your graphics hardware. You can download the latest drivers from the manufacturer’s website. The graphics requirements depend on the size and complexity of your data sets. If you plan to import large or complex CAD files, check PTC’s list of supported hardware for a list of appropriate graphics cards at Platform Support.

Applying special configurations to your graphics card is not recommended.

Platform-specific Requirements

Creo Illustrate supports 64-bit Windows operating systems. You must have administrative privileges on Windows to install Creo Illustrate. You must also have the latest proprietary graphics drivers from your graphics card vendor. Your operating system’s graphics drivers are not sufficient.

Software Requirements and Helper Applications

All of the software requirements are automatically met when you install from one of these locations:

- Product DVD
- `CreoIllustrate_64.exe` executable file

If you install the application as an administrator directly from the MSI file, the software prerequisites are not automatically met. See Deploying Creo Illustrate and Customizing the Installation for more information.

Microsoft C++ Runtimes are required for installing and running Creo Illustrate. The required C++ runtimes are shipped with Creo Illustrate and installed automatically if needed. If you already have C++ Runtime, the installation is not modified.

Other helper applications are shipped and automatically installed with Creo Illustrate:

- Creo View Files Tools—Used for transforming or optimizing Creo View data.

Browser Support Requirements

You may have to set special options for your Web browser.

Google Chrome Support for Creo Illustrate

- The Version Checker plug-in does not work with the Chrome and Firefox browser.
- Before using Creo Illustrate with the Chrome browser, you must install Creo Illustrate from the DVD or from the downloaded DVD image (use the installer script `setup.vbs`). For more information, see Performing an Installation from the DVD on page .

Adobe Version Verification

If you attempt to open a PDF document, but it does not open, perform the following steps:

1. Ensure that a supported version of Adobe Acrobat or Reader is installed. If both Adobe Acrobat and Reader are installed, ensure that both are supported

versions. Look for the latest *Creo View Clients and Toolkits Software Matrix* for detailed version information.

2. If a supported version of Adobe Acrobat or Reader is already installed, check that the Windows Registry software `InstallPath` values for Adobe Acrobat or Reader, located under the HKEY_LOCAL_MACHINE key, point to valid installation directories.
3. If the Windows Registry is correct, uninstall and reinstall Adobe Acrobat or Reader (or both, if both are installed), and then uninstall and reinstall Creo View.

For more details, refer to Technical Support article CS235004.

PTC Customer Agreement

Before you install software, you must accept the PTC Customer Agreement: To do so, follow these steps:

- You must accept the license agreement to proceed even if a license agreement already exists. The license agreement shown supersedes the previous license agreement.
- If you decline the license agreement you cannot proceed.

See Performing an Installation from the DVD or Performing an Installation Using the Executable File.

3

Quick-Start Installation

This chapter provides a quick-start approach for experienced Creo Illustrate users who have previously installed license management and product software on license server, license client, and node-locked machines.

For step-by-step installation instructions, read the subsequent chapters in this guide. New users can refer to the chapter Installing Creo Illustrate.

Updating PTC License Server

You must have PTC License server installed before you install Creo Illustrate. If you are updating your software to a later release, you must update your PTC licenses. See Updating an Installation for details. You need not install PTC License server if you purchased uncounted node-locked licenses.

Performing a Quick-Start Installation

Install Creo Illustrate to a client machine and to the license server machine using the following workflow:

1. Start the installer.
2. Accept the PTC Customer Agreement.
3. Accept the default installation path or set a new path.
4. Click **Install**. The Creo Illustrate clients are installed.

   See Performing an Installation Using the Executable File for details on the installation process.
5. Click **Finish**.

Where Your Software Is Installed

The software is installed in a default directory. You can modify the path during installation. The default path follows:

```
C:\Program Files\PTC\Creo <release_number>\Illustrate\
```

PTC License Server, Creo Illustrate, and any related components are installed by default into separate subdirectories within the `PTC` directory. You cannot change the installation location for Creo Illustrate.

4

Obtaining and Configuring Licenses

This chapter explains how to obtain or update licenses. It also provides instructions to install PTC License Server, a third-party license management software for Creo Illustrate.You must install PTC License Server before installing Creo Illustrate unless you have purchased uncounted node-locked licenses. Refer to your sales documents for your license type.

The license server need not be installed on the same machine as Creo Illustrate.

PTC uses FlexNet Publisher from Flexera Software, Inc. as its license server. See License Management Software for an overview and benefits of the license management software. The *FlexNet Publisher License Administration Guide* may be helpful as you install PTC License Server.

Every time you install or update Creo Illustrate, make sure you have the latest FlexNet software released by PTC.

Using the License Server Manager

The license server manager handles the initial contact with your PTC software that uses FlexNet licensing. It passes the connection to the appropriate vendor daemon. A license server manager serves the following purposes:

- Starts and maintains vendor daemons for serving license rights from different software.
- Transfers specific software requests to the correct vendor daemon.

There are two versions of the license server manager for license administration:

- `lmadmin`—Uses a graphical user interface (GUI)
- `lmgrd`—Uses a command-line interface

PTC currently ships with its products a license server manager based on `lmadmin` for the Windows platform and `lmgrd` for all other platforms. If you are using an existing license server with `ptc_d` vendor daemon of version 11.13. x.x or earlier, you will need to upgrade to the latest version.

Verifying System Requirements

Installation requirements for the license server manager software, follow:

- On all Microsoft Windows platforms, you must have administrative privileges to install the PTC License Server.
- You must have TCP/IP (Transmission Control Protocol/Internet Protocol) installed and configured correctly on your Windows system before installing the software.

Before You Proceed

Gather all the necessary information that follows about the installation of the license server manager:

- Check the FlexNet License Hardware Notes – PTC License Server for license server installation requirements and to determine from where to download the license server software for your server machine.
- Make note of your product code that arrives via e-mail. If you have received a license file via e-mail, store it at a secure location on your disk.
- If you have already installed PTC License Server based on `lmgrd` and choose to migrate to `lmadmin`, complete these steps before the migration:

  1. Shut down any `lmgrd` service or processes running on the system.
  2. Save the `license.dat` file from your existing license server installation to a secure location.
  3. Uninstall the previous installation completely.

  See “Migrating from lmgrd to lmadmin” in the *FlexNet Publisher License Administration Guide* for more information.

Obtaining a License

You must generate or update existing licenses to use the latest software for Creo Illustrate. Licenses determine the optional modules that you are authorized to run on your computer. You can get a license for Creo Illustrate in two ways:

- New customers—Use the PIM installer (PTC Install Manager). See the next section.
- Existing customers—Use existing tools (required). Skip to the section Existing Customer Licensing.

Creo Illustrate uses a PTC License Server powered by FLEXnet Publisher 11.10.x or later from Flexera Software Inc. If you are already running that license server for another Creo application, you need only update your license file to support the latest version of Creo Illustrate. If you are a new user, or if your license server is outdated, you must install the PTC License Server when using floating licenses.

New Customer Licensing

The PTC Install Manager streamlines the license installation process for new customers. Use the link on the Shipping Confirmation Letter. You need your Sales Order Number (SON) and an upgraded PTC account associated with the customer number. To begin, update your account with a user name and password at the following link:

http://www.ptc.com/appserver/common/account/secure/premiumAccount.jsp

If you do not have a PTC Support account, you can open one here:

https://www.ptc.com/appserver/common/account/create.jsp

Perform the following steps to obtain the license using the PIM installer, which also has on-screen instructions:

1. Start the PIM installer from the link on the Shipping Confirmation Letter.

> **Note**
>
> You must run the installer on the machine to be licensed. This is true for both a floating license for a license server or a node-locked license for a single client machine.

2. Type your user account name (usually your e-mail address), and then type your SON (Sales Order Number). Your license file is downloaded. For floating licenses, PTC License Server is also installed.
3. Click **Next**. The License Agreement appears.
4. Accept the agreement and click **Next**. A message appears.
5. Verify the network card for your license file configuration, and then click **Next**. This step determines automatically the Host ID of the license. A message appears.
6. Supply your license file. The installer downloads the licenses and installs the License Server if applicable.

Node-locked licenses are saved locally in a path noted on the screen. A copy of the license type will be e-mailed to the address on file for the account used in step 2.

Existing Customer Licensing

Existing customers should continue to use the existing licensing tools. Visit the *Customer Support Guide* page, click **Licensing**, and then follow the instructions to manage your licenses.

Note

For Creo Illustrate, select PTC Creo View & ProductView.

Installing the PTC License Server

After you have received an e-mail from PTC with the product codes for your software order, install the Standalone PTC License Server. Follow the installation instructions in the *Installing and Configuring the Standalone License Server* guide.

Borrowing and Returning Licenses

To work independently of a server, for instance from your laptop at home or at a customer site, you must borrow a temporary license. After 14 days this license expires, but it is good practice to borrow it only as long as needed so it is available for other users.

Borrow and return licenses using an executable file that comes with the software. Run it using arguments to borrow licenses for the versions and options you need. Instructions follow.

1. Open a command prompt on a computer with a server connection.
2. To read a short help message, type `<Creo Illustrate installation location>\bin\illustratelicense_borrow.exe`.
3. Add arguments to the command line as follows:
   - To borrow a license, type `-borrow` and then the features and options to borrow, and the license end date. See Example: Selecting License Options to Borrow.
   - To return a license, type `-return` and then the features or options to return.
   - To view a list of currently borrowed licenses and their expiry dates, type `—list`.
4. Press ENTER.
5. To verify that your borrowed license works, disconnect your computer from the network and run Creo Illustrate.

The general structure of the command-line syntax for borrowing a license consists of the following elements: `illustratelicense_borrow [-server LicenseServerInfo] dd-mmm-yyyy[:hh:mm] LicenseFeature [+LicenseOptions]`.

A typical license borrowing request looks like this: `D:\Users\myname>"C:\Program Files\PTC\Creo <release_number>\Illustrate\bin\illustratelicense_borrow.exe" -borrow 31-dec-2012 illustrate +largeaddress`.

Example: Selecting License Options to Borrow

If you purchased Creo View MCAD, these options are available to borrow:

Creo View MCAD Options	Description
`+largeaddress`	Massive assembly—Opens very large assemblies. For more information, see Massive Assembly Requirements in the Overview chapter.

5

Downloading Creo Illustrate

Follow the next procedure to download the DVD image of Creo Illustrate from PTC.com.

1. Visit the **Order or Download Software Updates** page:

   http://www.ptc.com/appserver/cs/software_update/swupdate.jsp

   You may have to type your user name and password and click **Log In**.
2. Click **Order or Download Software Updates**. The **Authorized Use Only** page opens.
3. Under **Customer Search**, type your **Customer Number** or **Customer Name** and click **Next**. The **PTC Software Download** page opens to **Step 1: Select the Product Family**.
4. Select **CREO ILLUSTRATE**. The **PTC Software Download** page opens to **Step 2: Choose Release & Download**.
5. Find the release to download and click ⊞ to expand it.
6. Click ⊞ next to Creo Illustrate.
7. To download the most recent datecode of the release, click ⊞ to expand **Most Recent Datecode**, and then next to **Download now**, click **HTTP**. The download begins.
8. To download another datecode, follow these steps:

   a. Click ⊞ to expand **Show all Other Available Datecodes**.

   b. Click ⊞ next to the datecode to download.

   c. Next to **Download now**, click **HTTP**. The download begins.

You have now downloaded Creo Illustrate. See Performing an Installation from the DVD for the next steps.

6

Installing Creo Illustrate

This chapter describes the process for installing Creo Illustrate using the installer on a single machine. Administrators who want to deploy Creo Illustrate on multiple machines should skip to the next chapter, Deploying Creo Illustrate and Customizing the Installation.

Downward Compatibility

Creo Illustrate is downward compatible. You can retrieve Creo Illustrate illustration files created in a previous release of Creo Illustrate using the current version of Creo Illustrate.

Installers for Creo Illustrate

There are two locations from which you can access a Creo Illustrate installer:

- DVD or downloaded DVD image.
- `CreoIllustrate_64.exe` downloaded executable files.

Instructions follow for using these installers.

Performing an Installation Using the Executable File

If you download the installer, you can use the executable file to perform a simple installation or an advanced installation. A simple installation automatically uses one of the default installation locations below:

- 64-bit installation—`C:\Program Files\PTC\Creo <release_number>\Illustrate\`.

These features are also installed:

- **Import Filters**—All
- **Language support**—English only

To add languages or remove features, skip to the section below, Performing an Advanced Installation.

Performing a Simple Installation

To install Creo Illustrate and the default components to the default location, follow the steps below.

1. Double-click `CreoIllustrate_64.exe`, or `Setup.exe`. The **PTC Creo Illustrate 7.1.0.0 Setup** dialog box opens to the **PTC Customer Agreement** page.
2. Read the Terms and Conditions, and then select **I accept the terms in the License Agreement**.
3. Click **Install**. Creo Illustrate is installed.
4. Click **Finish**.

Performing an Advanced Installation

To modify the location or contents of the installation, including language support, follow the steps below. You can add or remove entire features, or selected components of a feature. For example, you can remove all import filters, or you can set each filter's status individually.

1. Double-click `CreoIllustrate_64.exe`, or `Setup.exe`. The **PTC Creo Illustrate 7.1.0.0 Setup** dialog box opens to the **PTC Customer Agreement** page.
2. Read the Terms and Conditions, and then select **I accept the terms in the License Agreement**.
3. Click **Advanced**. The **Destination Folder** page opens.
4. Accept the default path for the installation, or set a new path:
   a. Click **Change**. The **Change destination folder** page opens.
   b. Next to **Look in**, browse to a new location.
   c. Click **OK**. The path is changed.
5. Click **Next**. The **Product Features** page opens.
6. To add or remove one or more features, such as language support or file import filters, follow these steps:
   a. Next to a feature to remove, click ☒, and then select ☒ **Entire feature will be unavailable**.
   b. Next to a feature to add, click 🖴, and then select one of these options:
      - 🖴 **Will be installed on local hard drive**—Installs the selected feature or component.
      - 🖴 **Entire feature will be installed on local hard drive**—Installs all components of the selected feature.
7. Click **Install**. Creo Illustrate is installed.
8. Click **Finish**.

You have now installed Creo Illustrate. To start the program, continue to the next chapter, Starting Creo Illustrate and Modifying an Installation.

7

Starting Creo Illustrate and Modifying an Installation

This chapter explains how to start Creo Illustrate and configure the startup. Proceed to the end of the chapter for the procedure to uninstall Creo Illustrate.

Starting Creo Illustrate

After your license management and product software have been installed, start Creo Illustrate. You can run the application using the **Start** menu. Click **Programs** ▸ **PTC Creo** ▸ **PTC Creo Illustrate** <release_number>. The first time you start the software, the **Creo Illustrate Startup** dialog box opens. Instructions follow for setting license options. To set a language for the Creo Illustrate user interface, see Setting the Language below.

Setting the Edition and License Options

Use the **Creo Illustrate Startup** dialog box to set the options for starting Creo Illustrate as follows:

1. To make the startup options the default, clear the **Choose edition at startup** check box.
2. For each license option that you purchased, select a loading option:
    - **Load at startup**
    - **Load when required**
    - **Never** (optional modules only)
3. To update your license, under **License Information** in the **License Server** box, type the name of the server or the path to the node-locked license file.
4. Click **Apply**. Click **OK**. Creo Illustrate starts.

Setting the Language

By default, the language of the Creo Illustrate user interface is determined according to your operating system's settings. To set a different language for Creo Illustrate from the operating system language, use the environment variables described below.

The generic environment variable, `LANG`, sets one language for all programs on your machine. You can use PTC-specific variables to designate another language for one or more programs:

- `PVIEW_LANG`—Creo Illustrate
- `PRO_LANG`—Creo Parametric and its associated applications. When you set this variable, you must also set `PVIEW_LANG`. This is true even if you want the same value for `PRO_LANG` and `PVIEW_LANG`.

Modifying an Installation

If you installed Creo Illustrate from the DVD, all languages and features are already installed. If you installed the software from the downloaded `CreoIllustrate_64.exe`, or `Setup.exe` installers, it is good practice to use the installer to modify your installation as follows:

1. Start the installer. The **Creo Illustrate 7.1.0.0 Setup** dialog box opens to the welcome page.
2. Click **Next**. The **Change, repair, or remove installation** page opens.
3. Click **Change**. The **Product Features** page opens.
4. Add or remove features. See Performing an Advanced Installation for more information.
5. Click **Change**. The installation is changed.

Uninstalling Creo Illustrate

You can remove an installation of Creo Illustrate through the Control Panel as follows:

1. From the **Start** menu, click **Settings** ▸ **Control Panel**.
2. Double-click an icon according to your version of Windows:
   - **Programs and Features**—Windows 10 or later
   - **Add/Remove Programs**—Earlier versions
3. In the program application list, select the application to remove.
4. Right-click and choose **Uninstall** from the shortcut menu. A confirmation appears.
5. Click **Yes** to proceed. Creo Illustrate is removed.

Note

You must separately remove the helper applications, such as Creo Illustrate Files Tools, using the previous procedure.

8

Deploying Creo Illustrate and Customizing the Installation

This chapter is for system administrators. It provides information on deployment, including example procedures for customizing the automated installation of the program.

Deploying Creo Illustrate Using a Silent Installation

Perform a silent installation in either of these ways:

- Deploy the program from the downloaded `Setup.exe`, or `CreoIllustrate_64.exe`—See Deploying from a Downloaded Executable File for an example.
- Extract the executable file and directly call `msiexec.exe` on some or all the required installers—See Customizing the Installer for examples.

In the Command Prompt you can enter `msiexec /?` to read an explanation of all the generic MSI options. Some common examples of options for a silent installation follow:

- `/qn`—Installs the software with no installer user interface.
- `/qb`—Provides a progress bar with a **Cancel** button.
- `/qb!`—Provides only a progress bar. A user cannot stop the installation.

Generic Features for Creo Illustrate

Feature Name	Feature Description
`ALL`	Use this feature to install all the features for Creo Illustrate. It does not require any other features to be specified. It is highly recommended to use this feature unless you are required to deploy a specific subset of features for Creo Illustrate.
`main`	Core Creo Illustrate runtime. This feature is required.
`en`	Basic language resources. This feature is required.
`doc`	CHM Help file. This feature is required.
	Sets the license server variable. Must be used with the `LICENSESERVER` property.

Language Features

Language	Feature for User Interface	Feature for Help
English—This feature is required.	en	doc
German	de	docde
Spanish	es	doces
French	fr	docfr
Italian	it	docit
Japanese	ja	docja
Korean	ko	docko
Russian	ru	docru
Portuguese—Brazil	pb	docpb
Chinese—Simplified	cs	doccs
Chinese—Traditional	ct	docct

File Import Filters Features-Prerequisites

The features listed below are the pre-requisite features required for installing specific file import filters.

Feature Name	Import Filter
dwg	3D DWG support. This feature is a prerequisite for pvifdwg
pvimportall	Forces the installation of all the pvif* features listed in the individual import filters table. Works in conjunction with the ALLIMPORTFILTERS=1 property. Using this feature on its own prevents the need for using the pvimport and pvif* individual features.
pvimport	Basic set of files required for installing the import filters listed below. This is a prerequisite to any pvif* features and to the pvaroptimizer feature. This feature should be used in conjunction with at least one of the pvif* or pvaroptimizer features.

Individual File Import Filters Features

The features listed below are the prerequisite features required for installing specific file import filters.

Feature Name	Import Filter
`pvifcatiadirect`	Import filter for CATIA model format
`pvifdae`	Import filter for Collada model format
`pvifdgn`	Import filter for DGN model format
`pviffbx`	Import filter for FBX model format 64–bit only
`pvifgbf`	Import filter for GAV and GBF model formats
`pvifiges`	Import filter for IGES model format
`pvifinvdirect`	Direct import filter for Inventor 3D model formats (IPT and IAM) 64–bit only
`pvifobj`	Import filter for OBJ 3D model formats 64–bit only
`pvifpgl`	Import filter for PGL model format
`pvifsedirect`	Direct import filter for Solid Edge 3D model formats (PAR, ASM, PSM 64–bit only
`pvifstep`	Import filter for STEP model format
`pvifstl`	Import filter for STL model format
`pvifvrml`	Import filter for VRML model format

Deploying from a Downloaded Executable File

To call a silent installation directly from `CreoIllustrate_64.exe`, add the `/v` argument to the program call immediately followed by all the parameters for the underlying MSI package. The command is prefixed with `start /w` to force one process to finish before another one is started. This is useful for scripting installations.

The following examples illustrate three different scenarios:

Example 1

Includes the feature `licvar` in the metafeature “`ALL`”. Therefore, the property `LICENSESERVER` is sufficient.

```
start /w CreoIllustrate_64.exe /vADDLOCAL="ALL"
         LICENSESERVER=”7788@licsrv.example.com” /qn
```

Example 2

Controls the installation of some features, the STEP ImportFilter is excluded, and the Russian language is forced:

```
start /w CreoIllustrate_64.exe /v
         ADDLOCAL="main,dwg,plugin_acrobat,msvcrt8,en,ru,doc,
docru,
         pvimport,pvifiges,pvifstl,pvifgbf,pvifvrml,pvifdgn,
         pviffbx,pvifinvdirect,pvifobj,pvifsedirect" /qb
```

Example 3

Controls the installation of some features and uses “pvimportall” to deploy “all import filters” without a detailed list.

```
start /w CreoIllustrate_64.exe /v
         ADDLOCAL="main,dwg,plugin_acrobat,msvcrt8,
         en,ru,doc,docru,pvimportall"
         ALLIMPORTFILTERS="1"/qb
```

⚠ **Caution**

To uninstall or upgrade, if you used the `ALLIMPORTFILTERS=1` property alongside the `pvimportall` feature to install the product, you must include `ALLIMPORTFILTERS=1` as an argument in the `msiexec.exe` command line.

Customizing the Installer

`CreoIllustrate_64.exe` is self-extracting archives that each run `CreoSetup.exe`. `CreoSetup.exe` is a wrapper application for chaining the installations of several installers in the archive, including `CreoIllustrate_64.msi`. Running the MSI file alone does not constitute a complete installation of Creo Illustrate. However, you can customize the installer in these ways:

- Remove applications installed by default
- Bundle other applications with the Creo Illustrate installation

Use 7-Zip to extract the archive to a folder, and then modify its contents. Continue to Creating and Deploying a Customized Installer for an example.

Creating and Deploying a Customized Installer

You can add and remove applications from the Creo Illustrate installation. For instance, the users in your organization have the following needs:

Required	Not Required
Using Creo Illustrate client	Microsoft Visual Studio C++ 2017 Runtime Redistributable
Working with JT (*.jt) files	

In this case, you may want to customize the installer in these ways:

- Bundle the JT Import Filter with the Creo Illustrate installation
- Remove Microsoft Visual Studio C++ 2017 Runtime Redistributable because you have already deployed it. We will still deploy 2012 Runtime.

Accomplish the customization by performing two basic tasks:

1. Prepare the directory structure by adding and removing installer files.
2. Modify `CreoSetup.ini` to run the installers correctly.

After you complete these tasks, you can deploy the software. The procedures in the next sections contain instructions for completing the tasks above, and for and the deployment.

Preparing the Directory Structure

Follow the steps below to customize the installation directory.

1. Create an empty directory (`<directory>`).
2. Copy `CreoIllustrate_64.exe` into `<directory>`, and then right-click the file and choose **7–Zip ▸ Extract Here**. The setup files are extracted from the wrapper.
3. Delete `CreoIllustrate_64.exe`. These subdirectories and files remain:
   - `\prereqs` subdirectory
   - `\illustrate` subdirectory
   - `CreoSetup.exe`
   - `CreoSetup.ini`
4. Create a new subdirectory called `\jtimport`.
5. Navigate to the installation subdirectory on the JT DVD image: `cdimages/jtadapter/installers/`

   Copy `CreoView_JT_Import_64.msi`, and then paste them into the `\jtimport` subdirectory you created in step 4.

6. Under the `\prereqs` subdirectory, remove the two Microsoft Visual Studio 2017 C++ Redistributable x86 and Microsoft Visual Studio 2017 C++ Redistributable x64. They are called `vcredist_x64_VS2017.5u8exe` and `vcredist_x64_VS2015u3.exe`.

The directory is now ready. Continue to the next section to customize the INI file.

Modifying `CreoSetup.ini`

You have added an installer file to bundle the JT Import Filter with the Creo Illustrate installation. You must now modify `CreoSetup.ini` to make these changes to the default installation directives. You can delete unnecessary lines and sections from the file, or you can modify them. In this example, they are modified.

- Add new directives to run the JT installer
- Remove or disable the `vcredist_x64_VS2017u8.exe` directives.

Follow the next steps to update the INI file accordingly.

1. Open `CreoSetup.ini` in a text editor.
2. In the `[Other Applications]` section, find the line `VC2017_X86=YES` and replace it with `VC2017_X86=NO`, or `VC2017_X64=YES` and replace it with `VC2017_X64=NO`.
3. At the end of the `[Other Applications]` section, add these lines:
```
JTIMPORT_X86=YES
JTIMPORT_X64=YES
```
4. At the end of the file, add these sections with information for the JT Import Filter, as shown in these examples:
```
[JTIMPORT_X86]
PROGRAM=CreoView_JT_Import_Filter_32.msi
COMMANDLINE=ADDLOCAL="ALL" REBOOT="ReallySuppress" /l*v
    "[TempFolder]pvinstjt.log" /qb!
LOCATION=jtimport
PRODUCTCODE={7D39690A-FB6B-4559-B1FD-26F7735162B3}
PLATFORM=X86
[JTIMPORT_X64]
PROGRAM=CreoView_JT_Import_Filter_64.msi
COMMANDLINE=ADDLOCAL="ALL" REBOOT="ReallySuppress" /l*v
   "[TempFolder]pvinstjt.log" /qb!
LOCATION=jtimport
PRODUCTCODE={E6438BA9-7C4D-4D54-A87D-65272F3169A6}
PLATFORM=X64
```
5. Save `CreoSetup.ini` and close the text editor.

You have now defined the installation in the INI file:

- For more information on locating the code for an application, continue to the next section, *Finding the Product Code*.

- For more information on the INI file, skip to the section *Understanding the* `CreoSetup.ini` *File*.
- To begin deployment, skip to the section *Deploying with the Customized Installer* below.

Finding the Product Code

This example uses the JT `ProductCode` for Creo 2.0 M020. Each release has a unique `ProductCode`. An application also has a permanent `UpgradeCode`. The `UpgradeCode` is not used by `CreoSetup.exe` because it does not identify the version of the application. You can find the product code for the applications to install by using the Registry Editor. On a computer where the application is already installed, navigate to the registry key.

⚠ **Caution**

Do not modify the registry. Doing so can severely damage your computer.

1. Open a command prompt, and then type `regedit`.
2. Press ENTER. The **Registry Editor** opens.
3. On the left, navigate to this key:
   `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall`
4. Find the application to install, and copy the key, including the curly brackets. These are examples of the registry keys for Creo Illustrate 2.0 M020 JT Import Filters:
   - 64–bit—`{E6438BA9-7C4D-4D54-A87D-65272F3169A6}`

Understanding the `CreoSetup.ini` File

Each application in the file's `[Other Applications]` section has a `YES` or `NO` value. A `YES` value does not necessarily mean the application's installer will always be executed. Instead, each application with a `YES` value is a potential installation. If your system has an installation that matches the product code line, then nothing is done. If your system does not have a matching installation, the application is installed.

The `PRODUCTCODE` line for each application is optional. In all cases, the application is installed automatically when your system does not have the same version installed. Providing a product code minimizes the time the installation takes to complete.

The value for the `PROGRAM` line must be a `*.exe` file or a `*.msi` file.

Deploying Creo Illustrate with the Customized Installer

You have now prepared the directory and modified the INI file. In this example, deploy the applications with a passive installation. You may use one or both of these strategies:

- Run `CreoSetup.exe` directly from the network location.
- Create a new, self-extracting archive from the modified directory. Make sure the self-extractor calls `CreoSetup.exe` to chain all the installations to deploy.

The call to `CreoSetup.exe` must include, in the command line, all MSI arguments for the base Creo Illustrate installer. At this time, you cannot embed the Creo Illustrate MSI arguments directly in `CreoSetup.ini`.

To perform the passive installation, open a command shell and enter the following command:

```
CreoSetup.exe /vADDLOCAL="ALL" APPLICATIONFOLDER="C:\
ptc\Creo Illustrate" LICENSESERVER=
”7788@licsrv.example.com” /qb!
```

Note

Make sure there is no space between `/v` and the next character in the command.

Creo Illustrate and the JT Import Filter are installed with a progress bar, but no **Cancel** button. For more information on common examples of options for a silent installation, see Deploying Creo Illustrate Using a Silent Installation.

Performing a Silent Uninstallation

MSI installers automatically uninstall a matching product as part of an upgrade. Therefore, for upgrades it is unnecessary to perform a silent uninstallation. However, if you do want to manually and silently uninstall Creo Illustrate, you must first obtain the ProductCode for each version of the application to uninstall. New ProductCodes are generated for each release and patch build of Creo Illustrate. This section contains instructions for finding the ProductCodes and for using them to perform the silent uninstallation.

Obtaining ProductCodes

To query the MSI inventory and retrieve the ProductCodes, you can use any of these methods:

- Visual Basic script

- Powershell script
- Free tool such as msiinv.exe

For example, execute the msiinv.exe with these arguments:

```
msiinv.exe -p > %TEMP%\msiinv_output.txt
```

You now have a list of all MSI packages and their ProductCodes in your %TEMP% directory, in the file msiinv_output.txt. You can uninstall the applications. Continue below.

Uninstalling the Applications

Use the msiexec.exe tool to uninstall one or more Creo Illustrate applications. For each application to uninstall, you must run msiexec.exe with the corresponding ProductCode. Instructions follow.

1. Note the ProductCode you retrieved in the previous procedure.
2. Open a command window, and type the following command:
   ```
   msiexec.exe /x {<ProductCode>}
   ```
3. Press ENTER. The command is executed.

You have now manually and silently uninstalled the application. Repeat the procedure above for each application to uninstall.

9

Updating an Installation

This chapter explains how to update your copy of Creo Illustrate. Each time you update your software to a new release, you must update your license file. New license codes are not required if you are updating to a maintenance release of Creo Illustrate within the same release. In some cases you may have to update the current installation of PTC License Server. See the next sections for details.

Updating the License Server

You can update the current installation of PTC License Server with your new license information when updating your PTC software. An installation update of PTC License Server is typically required in one of these scenarios:

- Adding license information from your new Sales Order
- Updating the existing license features in your license file with those of a maintenance release

You may have to update the license server software itself depending on the version installed. In such cases, you must uninstall PTC License Server using administrative privileges and then follow the installation instructions in the *Installing and Configuring the Standalone License Server* guide.

Update your licenses using this workflow:

1. Make note of the product code that you have received via e-mail. Alternatively, use the PTC License Management Web tools to request for a license file via e-mail. Save the license file in an ASCII format to a secure location on your disk.
2. Navigate to the `\bin` subfolder of your license server installation. The default path follows:

   ```
   C:\Program Files\FLEXnet Admin License Server\bin
   ```

3. Right-click `ptcsetup.bat` and choose **Run as Administrator** from the shortcut menu. The **PTC Setup — PTC License Server** dialog box opens.
4. Click 📂. The **Open** dialog box opens.
5. Browse to your license file, and then click **Open**. The path appears under **License File**.
6. Click **Configure**. The **Installation Progress** page opens with a progress bar.
7. A warning opens. Click **OK**.
8. To check the status of your updated license, follow these steps:

   a. Click **Program Files ▸ PTC ▸ PTC FLEXnet Admin License Server**. The **FLEXnet License Administrator** webpage opens.

   b. Near the top of the page, click **Administration**. The **Administration** tab opens.

   c. On the left, click **Vendor Daemon Configuration**. The **Vendor Daemon Configuration** tab opens with your license.

   d. Under the **Status** column, verify that the status is **RUNNING**.

PTC License Server restarts. For Triad configurations, two of the three partner machines must be running before licenses can be distributed.

Updating Creo Illustrate

After you install a particular release of Creo Illustrate, you can install a new maintenance release. The previous release is overwritten.

10

License Management Software

This chapter discusses license management software and describes the benefits of PTC License Server. License simplification is explained for new and existing users.

Overview of PTC Licensing

PTC software including PTC optional applications must be licensed for use. Licensing authorizes the installed software to run. Creo Illustrate licenses are not time-sensitive.

License Types

Depending on the PTC product, a license can be one of the following types.

- Node-locked licenses—Restricts the use of the software to a particular machine (independent workstation).
- Floating licenses—Served by the license server and can be used on any machine connected over the network. There are two different types of floating licenses:
    - Single server licenses—Configured for a single machine as the server.
    - Triad licenses—Configured for a set of three machines on the same network that work together to manage a pool of floating licenses. This configuration provides redundant backup in case of a server outage. An identical license file is used for each Triad partner.
- Extended license—An extended license makes floating licenses available for locked modules.
- Borrowed license—A borrowed license allows you to work temporarily on your machine without being connected to the license server. Refer to Obtaining and Configuring Licenses for details.

For more information on the additional license types that are supported for your product, refer to the product documentation.

PTC License Server

Flexera Software, Inc.'s FlexNet Publisher license management software is integrated with PTC software. For more information, visit www.flexerasoftware.com.

Benefits of Using FlexNet Publisher

Using FlexNet Publisher to control usage of licenses offers the following key advantages:

- Single-Server Solution—FlexNet Publisher can manage PTC software and other vendor applications without conflict. Note that the PTC license file cannot be combined with the license files of other vendors.
- Immediate License Recovery—If there is a premature exit of the licensed software (for example, the system shuts down), the FlexNet Publisher license server automatically reclaims the PTC license.
- Increased Flexibility—System administrators can reserve or deny licenses based on user name, host name, display name, or IP address. For more information see Managing the Options File in the *FlexNet Publisher License Administration Guide*.
- Centralized License Storage—PTC customers can store all PTC licenses for all PTC products in a single file for each license server.
- Multiple Licenses for a Single Command—One command can be used to execute multiple licenses of PTC software based on availability.
- License Borrowing—Using FlexNet Publisher 8.6 or later, you can borrow licenses from a license server and run a licensed application on a remote client without being connected to the license server.

Downward License Compatibility Using FlexNet Publisher

FlexNet Publisher license servers have downward compatibility with PTC applications using FLEX*lm* license servers. For example, a Creo Illustrate 2.0 license can run Creo Parametric 1.0 software, because both releases of the software use the same PTC License Server.

Note

To run the current release of Creo Illustrate, you must install FlexNet Publisher version 11.x or later.

Running FlexNet Publisher with Other Applications

You can use FlexNet Publisher to run PTC products as well as other applications. Each application that uses FLEXlm or FlexNet Publisher has a corresponding license server manager (`lmgrd` or `lmadmin`) and a vendor daemon. The license server manager starts the vendor daemon (for example, `ptc_d`) that controls the usage of licenses.

You cannot combine a PTC license file with a license file of another vendor. Do not, for example, combine PTC feature lines with those of other vendors in a single license file. This action causes the licenses to be invalid.

If all applications and vendor daemons are FLEX*lm* 6.1 or later, `lmgrd` or `lmadmin` can process multiple license files. This is true even when the Host IDs are different (as long as the license files refer to the same node). For more information, see Managing Licenses from Multiple Software Publishers in the *FlexNet Publisher License Administration Guide*.

Understanding Timeout Parameters

Timeout parameters enable the license client and the license server to communicate with one another so that licenses are released and available upon request.

You can reconfigure the `TIMEOUTALL` parameter within a specified range, as described in the next section, Changing the Inactivity Timeout Parameter.

Timeout Parameter	Value	Description
Inactivity timeout (TIMEOUTALL)	120 minutes (default)	This parameter prevents a license from remaining idle. If the license client is inactive for a specific period, the license can be reclaimed by the license server and used by another license client. For the Creo applications, the inactivity timeout default is 120 minutes. Activity is measured as active menu selections. You can change the default of 120 minutes so that the parameter ranges from 20 minutes (minimum value) to infinity (maximum value).
License refresh	1 minute	A license refresh occurs at intervals of 1 minute. When you select a command after such an interval, the license client communicates with the license server. The license client and the license server both

Timeout Parameter	Value	Description
		must be working. If the license server is not found, the license client loses its license to run. If the license client is not found, the server reclaims the license for use by another license client.
Validation retries	None	The first time a license client cannot validate its license the application's graphical user interface will freeze. You are immediately given the following three options: Click **Retry** to request a license from an active license server. Click **Save File(s)**. Click **Exit** to close the application without saving the file or files.
Startup	10 seconds	Upon starting a session, the license client requests a license and has 10 seconds in which to have the request validated.

Changing the Inactivity Timeout Parameter

You can set the `TIMEOUTALL` parameter that determines how long the system allows a license to remain idle before it is reclaimed by the license server. To change the default inactivity timeout parameter, you must update the FlexNet Publisher option file, `ptc.opt` in `<ptc License Server loadpoint>/FLEXnet Publisher/licensing`. The default is 120 minutes. Edit the default parameter:

```
TIMEOUTALL 7200
```

Change 7200 seconds (120 minutes) to another value in seconds.

The minimum value is 20 minutes (or 1200 seconds) and the maximum value is infinity. To make infinity the default, remove the `TIMEOUTALL` parameter from the `ptc.opt` file. If you set a minimum value to less than 20 minutes, the system defaults to 20 minutes.

11

lmadmin License Server Manager

This chapter contains basic information on `lmadmin`, a Web-based license server manager. It also provides information on migrating from `lmgrd` to `lmadmin`.

Overview of lmadmin as a GUI-Client

The `lmadmin` license server manager supports a Graphical User Interface (GUI) client with connection over HTTP. It provides a Web-based administrative interface. It is compatible with license files and vendor daemons created with FlexNet Publisher 9.2 and later. Use `lmadmin` for the following functions:

- Perform server configurations and administration functions
- Start the license server manager without any configuration options
- Directly configure vendor daemon and the license server manager without editing the license files
- Import existing license files
- Support multiple vendor daemons with a single `lmadmin` process
- Display license rights status
- Display alerts and monitor status of various processes, such as license expiration or unavailable licenses

See the chapter lmadmin − GUI−based License Server Manager in the *FlexNet Publisher License Administration Guide* for more information.

Differences between lmgrd and lmadmin

The `lmadmin` license server manager includes all the features of the `lmgrd` license server manager. The differences between the two license server managers follow:

lmgrd	lmadmin
Uses a command-line interface.	Supports a GUI client with connection over HTTP.
Configuration settings are retained for all the sessions.	Configuration settings are session-based.

See the chapter Migrating from lmgrd to lmadmin in the *FlexNet Publisher License Administration Guide* for more information.

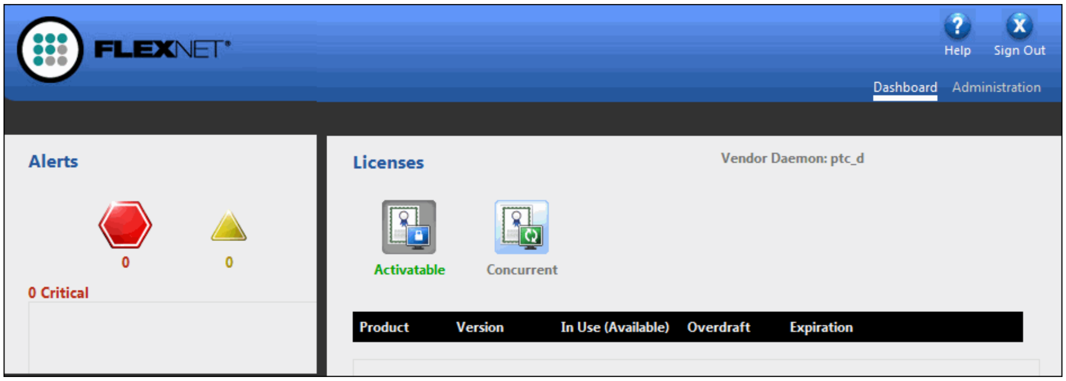
Installing PTC License Server Based on lmadmin

Follow the instructions given in the section Obtaining and Configuring Licenses to install PTC License Server based on lmadmin. After you install the license server, you can configure the license server administration settings using the FLEXnet License Administrator Web interface.

Working with the FLEXnet License Administrator Web Interface

The `lmadmin`-based license server uses the FLEXnet License Administrator Web interface. This interface replaces the `lmtools` utility used by the `lmgrd`-based license server. You can launch the interface by clicking **Yes** at the end of the PTC License Server installation. Alternatively, from the Windows Start menu click **Program Files** ▸ **PTC** ▸ **PTC FLEXnet Admin License Server** ▸ **PTC FLEXnet Admin License Server Web Interface** to open the interface. Using the FLEXnet License Administrator, you can check the server status, start and stop the server, or reconfigure the server.

The FLEXnet License Administrator Web interface has two main pages: the **Dashboard** and the **Administration** pages. Click the **Help** button for information on the FLEXnet License Administrator interface elements.



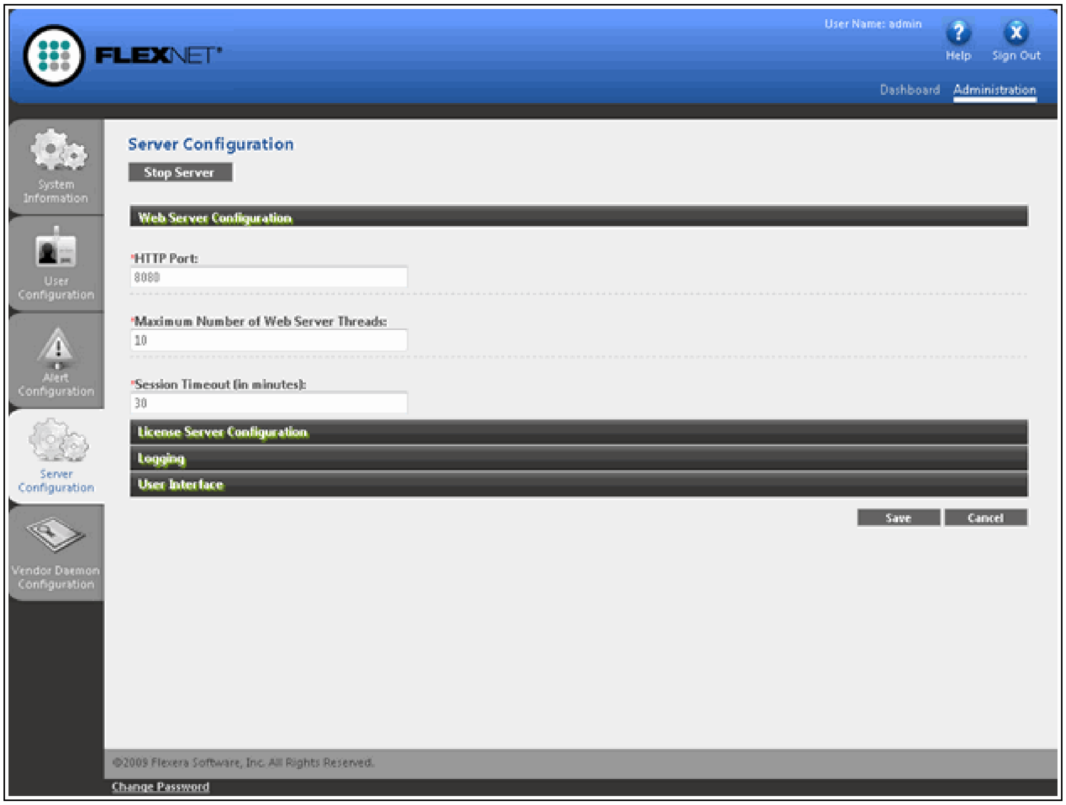
Product	Version	In Use (Available)	Overdraft	Expiration

- **Dashboard**—Displays any alerts and the current activity of the license server manager.
- **Administration**—Provides configuration tools for the license server management system. The default username/password combination is `admin/admin` for this password-protected page. You are prompted to change these

credentials the first time you log in. Only the overview information for the **Server Configuration** and the **Vendor Deamon Configuration** tabs on the **Administration** page is included in this document. Click a tab to open the corresponding pages.

Controlling the License Server Manager Settings

A server administrator can use the options on the **Server Configuration** page to control the settings for the License Server Manager.



Server Configuration

*HTTP Port: 8080

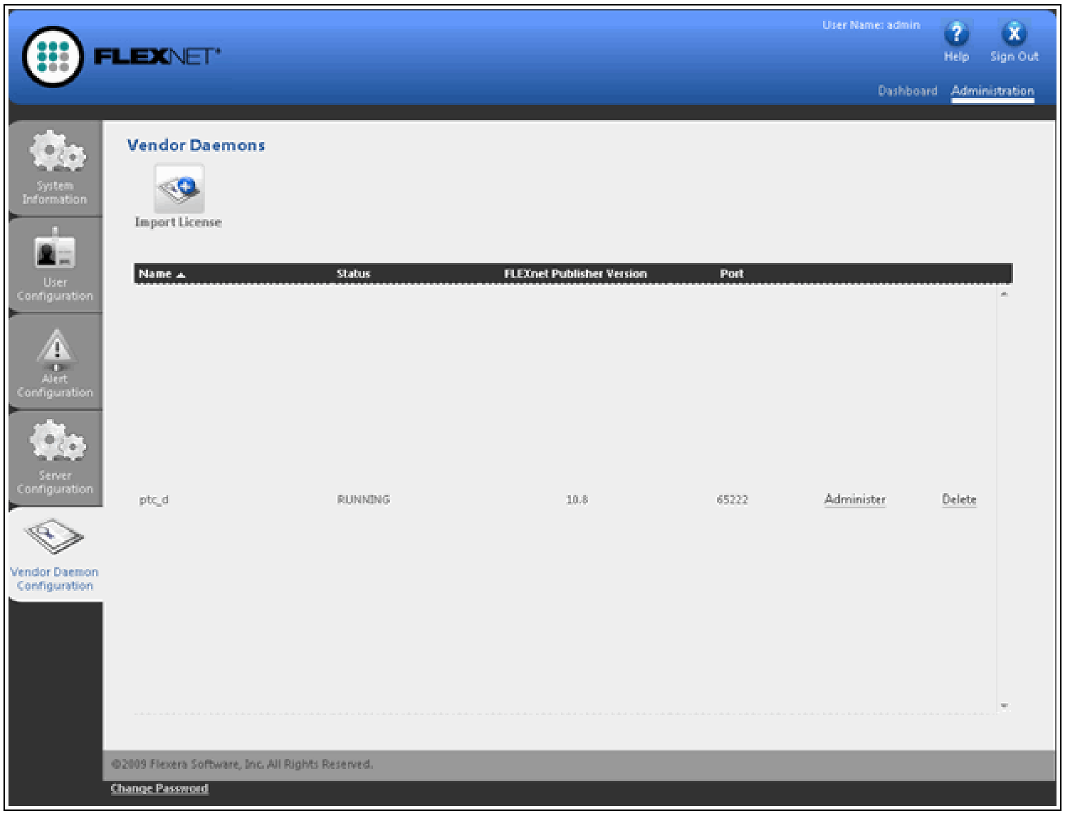
*Maximum Number of Web Server Threads: 10

*Session Timeout (in minutes): 30

©2009 Flexera Software, Inc. All Rights Reserved.

Reconfiguring the Vendor Daemon

A server administrator can use the **Vendor Daemon Configuration** tab to verify whether the server is running and to reconfigure the vendor daemon.



Vendor Daemons

Name	Status	FLEXnet Publisher Version	Port		
ptc_d	RUNNING	10.8	65222	Administer	Delete

©2009 Flexera Software, Inc. All Rights Reserved.

If PTC License Server has been successfully installed, `Running` appears under the **Status** column.

12

Troubleshooting Tips for Licensing and Creo Illustrate Runtime

This chapter documents common problems that occur when you install Creo Illustrate and PTC License Server. This appendix also provides general debugging techniques and cites other sources of information available from the PTC website.

Troubleshooting List

Skim through the following list of problems to find any that appear to be the same as the one you are experiencing. The information is presented in the following format.

Symptom: Describes the problem.

Cause: Lists the cause of the problem.

Solution: Provides steps to resolve the problem.

Cannot Connect to the License Server

Symptom: When you start Creo Illustrate, you receive an error message in the **Startup** dialog box.

Cause: The license server is not running or the license client cannot reach the server.

Solution: Verify that the license server manager and `ptc_d` daemons are running. A network problem exists if a license client attempts to ping the server by host name and the ping fails.

Invalid Licenses

Symptom: You receive the error message `Invalid license`.

Cause: Licensing information is inaccurate.

Solution: Return to the **FLEXnet license editor** and verify that the information entered is exactly as it appears in your License Pack. If no licenses are listed, return to the **FLEXnet license editor** and ensure no text is highlighted.

If all licenses are listed as `invalid`, verify that the PTC `Host_ID` in the License Pack corresponds with what you see in the **FLEXnet license editor**. For example, one server line and one daemon line represent a single server. Three server lines and one daemon line represent a fault-tolerant or Triad configuration. Remove all the lines that do not pertain to the PTC `HOST_ID`.

Your incremental lines must have no blank lines. Verify that all continuation characters (\) are at the end of each line, except for the last line. If some licenses are valid while others are invalid, find the invalid feature name lines in the License File window and correct the text.

If you received your license codes via e-mail, remove any extraneous text such as the header and footer. Another option is to delete the invalid license in the **FLEXnet license editor** window.

FlexNet Publisher Fails to Start (Triad Configurations)

Symptom: PTC License Server does not start after a Triad server is installed and configured.

Cause: The following requirement has not been met: two of the three partners of the Triad configuration must be running (Quorum) before licenses can be distributed.

Solution: Go to a shell window or a command prompt and change into the `<FLEXnet_Installation_Directory>\bin`. Type in ptcstartserver.

General Debugging Hints

The `ptc.log` file records license server activities and can be found in "`\FLEXnet Admin License Server\logs\`". This file has useful information that you should check when you have a problem.

Setting Variables and Installing Tools for Troubleshooting

The Creo Illustrate installation includes software for troubleshooting. Install the software and set troubleshooting environment variables as follows.

Installing Debug Symbols

Debug Symbols is a bundle of files for troubleshooting Creo Illustrate.If Creo Illustrate crashes, the detailed information is gathered by the configured trace file instead of by the Windows default mechanism. Instructions for installing Debug Symbols follow.

1. Navigate to `<installer_directory>\debug_symbols`.

2. Double-click the installer for your platform: `CreoView_Client_Debug_Symbols_64.exe`.

   The **Debug Symbols Setup** dialog box opens to the **PTC Customer Agreement** page.
3. Read the Terms and Conditions, and then select **I accept the terms in the License Agreement**.
4. Click **Install**. Debug Symbols is installed.
5. To read additional instructions required to start using Debug Symbols, select **View the Creo View Debug Symbols usage instructions**.
6. Click **Finish**.

You must set the corresponding environment variable to activate Debug Symbols. See the next section, Using Variables in Troubleshooting.

Using Variables in Troubleshooting

To aid in troubleshooting the runtime of Creo Illustrate, use these environment variables:

- `PVIEW_PROC_TRACE=<full or relative file path>\pvtrace.log`—Generates a unique log file for each Creo Illustrate process running. The log file saved to the path you set is called `pvtrace_<process number>.log`.
- `PVIEW_CATCH_EXCEPTIONS=1`—Works with Debug Symbols to gather troubleshooting information. The information is stored in the trace file defined immediately above. When you install Debug Symbols, you must set this variable.

Online Information

See www.ptc.com/en/support for a wealth of how-to information for new and experienced users, including order and license support. The *Customer Support Guide* provides online tools and other support services. You can search the Knowledge Base of over 14,000 technical documents or download reference documents.

Customer Support Guide and **Contact Support** are available at the My eSupport page. For information on FlexNet Publisher diagnostic environment variables, consult the *FlexNet Publisher License Administration Guide*.

13

Managing Preferences

This chapter provides an overview of Creo Illustrate preferences, including disabling default commands. It describes the locations for preference files and their hierarchy. Instructions for locking preferences are also included.

Overview of Creo Illustrate Preferences

This section contains an overview of the types of preferences you can set for Creo Illustrate. It also provides information for locking preferences.

Types of Preferences

There four types of preferences. Each type is read from an XML file:

Type	File Name	Location
Default	`Illustrate_prefs.xml`	Read from `<Creo Illustrate>\preferences\Illustrate\` subdirectory of the installation directory. **Note** A preference sample file is located in`<Creo Illustrate>\resources\Illustrate\preferences\illustrate_pref.xml`
Shared	`shared_pref.xml`	Downloaded from a shared location, to `%APPDATA%/ptc/Illustrate/shared_prefs`. Once enabled, the shared preference file is updated automatically.

Type	File Name	Location
Administrator	admin_prefs.xml	Read from <Creo Illustrate>\ preferences\ illustrate\ subdirectory of the installation directory
User	user_prefs.xml	Read from the user’s profile:%APPDATA%/ ptc/Illustrate/ user_prefs.xml

The files are processed in the order listed above. You can override a value in a preference file with a different value in a subsequent file:

- A user preference replaces the setting for a preference in the administrative file.
- If a shared preference file exists and is active, the administrative preference replaces the shared preference.

For example, the administrator disables the color highlighting command by defining the highlight_using_color preference in the admin_prefs.xml file, as shown:

```
<?xml version="1.0" encoding="utf-8"?>
<preferences>
     <category name="General">
          <subcategory name="Navigation" >
               <preference1 name="highlight_using_color" value=
"False"/>
          </subcategory>
     </category>
</preferences>
```

However, if the user explicitly enables the same preference in the user interface, the setting is saved to the user_prefs.xml file. As a result, the administrator preference is overridden and the color highlighting command is enabled. To prevent a user from overriding the administrator preferences, lock the administrator preferences. Continue to the next section, *Locking Preferences*, for more information.

Locking Preferences

At any level, you can lock a preference. A locked preference cannot be modified or overridden by a file on a lower level. Using the color highlighting example from the previous section, you can add the attribute locked=”True” to prevent the user from changing the preference:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<preferences>
     <category name="General">
          <subcategory name="Navigation" >
               <preference1 name="highlight_using_color" value=
"False" locked="True"/>
          </subcategory>
     </category>
</preferences
```

The color highlighting command is now disabled and locked.

Setting Preferences

Set the user preferences using the **Creo Illustrate Options** dialog box in the Creo Illustrate client, and then click **OK**. For more information, see the Creo Illustrate Help.

Set the administrator preferences by basing them on one of these files:

- User preferences
- Default preferences

Instructions for both methods follow.

Basing the Administrator Preferences on the User Preference File

1. In the Creo Illustrate client, click **File ▸ Options**. The **Creo Illustrate Options** dialog box opens.
2. Set one or more preferences, click **OK**, and then close Creo Illustrate. The changes are saved to the `user_prefs.xml` file.
3. Open these files in a text editor:
   - `user_prefs.xml`
   - `admin_prefs.xml`
4. In `user_prefs.xml`, find and copy the entries for the preferences you set in step 2.
5. Paste the entries in `admin_prefs.xml`.

   Note

   Make sure the XML structure hierarchy is maintained.

6. To lock one or more preferences, add the attribute `locked=”True”` to the definition of those preferences.

Note

The locked attribute is not supported in the user_prefs.xml.

7. Save and close admin_prefs.xml.

Basing the Administrator Preferences on the Default Preference File

1. Make a copy of the default preferences file, Illustrate_prefs.xml.
2. Open Illustrate_prefs.xml in a text editor.
3. Modify one or more preferences.
4. To lock one or more preferences, add the attribute locked="True" to the definition of those preferences.
5. Save the file as admin_prefs.xml

You have now set and locked the administrator preferences. For information about using the preference file to disable default commands, continue to the next section, Disabling Commands.

Disabling Commands

Each control in the user interface has a unique command. For example, the command for the **Print** button is PrintCmd. You can use a preference file to disable access from the user interface to one or more commands on buttons or ribbons. You cannot disable commands on shortcut menus. Instructions for finding and disabling a command follow.

Finding a Command

1. To find a command for a button, set the following environment variable:
   ```
   set PVIEW_DISPLAY_COMMANDS=1
   ```
2. Open Creo Illustrate, and then place the pointer over the button. A tooltip appears with the command. For example, PrintCmd.

Using the Preference File to Disable the Command

Follow the instructions in this procedure to disable the **Print** button.

1. Open admin_prefs.xml in a text editor.
2. At the end of the file, add the following section:
   ```
   <preferences>
      <category name="General">
         <subcategory name="UI_Config">
   ```

```
          <preference name="disabled_commands" value="PrintCmd"/
>
       </subcategory>
    </category>
</preferences>
```

3. To disable more than one command, list them, separated by commas, as the value for `disabled_commands`.
4. Save `admin_prefs.xml` and close the file.

The command is now disabled. When all of the preference files are read, the values in each `disabled_commands` section are cumulatively disabled.

Loading Preferences from a Shared Location

You can centrally manage Creo Illustrate preferences. The preferences are stored in a shared location, and they are copied to the user's computer. Shared preferences are available even when the user disconnects from the shared location. The shared preferences are saved as a ZIP file that contains `shared_prefs.xml` and any other required files. Users can load the shared preferences from one of these locations. The locations are checked in the order below, and the preferences are loaded only from the first location where they are found:

1. The **HKEY_LOCAL_MACHINE** registry entry
2. The **HKEY_CURRENT_USER** registry entry
3. The `ILLUSTRATE_SHARED_PREFERENCES` environment variable

Configuring Shared Preferences for Cadence Allegro and APD

This section describes how you can open Cadence Allegro and APD design files with no local Allegro installation.

To configure shared preferences for Cadence Allegro and APD follow the steps below:

1. In the Creo Illustrate client, click **File ▸ Options**. The Creo Illustrate **Options** dialog box opens.
2. Set one or more preferences and then click **OK**. The changes are saved to the `user_prefs.xml` file.
3. Open the `user_prefs.xml` file in a text editor. Locate the entries for the preferences you set in step 2 and copy them into a new file.
4. Save this file as `admin_prefs.xml`, `server_prefs.xml`, and `shared_prefs.xml`.

Note

The Cadence Allegro extracta file (.acceptedAllegroExtracta), must be kept in the same location as the admin_prefs.xml, server_prefs.xml and shared_prefs.xml.

5. To lock one or more preferences, add the attribute locked="True" to the definition of those preferences.

Note

The locked attribute is not supported in the user_prefs.xml.

6. Save and close the user_prefs.xml and server_prefs.xml files. The saved file is copied to the APPDATA% directory and automatically loaded into the shared_prefs.xml folder.

Shared Location Types

These are examples of the two types of shared locations:

- File system—s:\shared_prefs\shared_prefs.zip
- Shared host—\\sharedhost\shared_prefs\shared_prefs.zip

Defining the Location of Shared Preferences Using the Registry

This section contains instructions on how to define in the registry the location for shared preferences.

To define the location using an environment variable instead, skip ahead to the next section, Defining the Location of Shared Preferences Using an Environment Variable.

Click **Start** and in the search box, type Regedit.exe and press **ENTER**. The **Registry Editor** page opens. Then, perform one of the operations below:

- List the shared preference file in the local machine registry:
  1. Go to **HKEY_LOCAL_MACHINE ▸ SOFTWARE ▸ PTC ▸ Creo Illustrate**.
  2. Right-click and choose **New ▸ Key**. A new key is added under **Creo Illustrate**.
  3. On the left pane, under **Creo View**, rename the new key to shared_preferences and press ENTER. The **shared_preferences** key expands.
  4. Right-click and choose **New ▸ String Value**.

5. Name the new value `file`.
6. Double-click **file** and under **Value data** type the name of the shared preference file, for example `\\sharedhost\shared_prefs\shared_prefs.zip`.
7. Click **OK**.

- List the shared preference file in the current user registry:

1. Go to **HKEY_CURRENT_USER ▸ SOFTWARE ▸ PTC ▸ Creo Illustrate**.
2. Right-click and choose **New ▸ Key**. A new key is added under **Creo Illustrate**.
3. On the left pane, under **Creo Illustrate**, rename the new key to `shared_preferences` and press ENTER. The **shared_preferences** key expands.
4. Right-click and choose **New ▸ String Value**.
5. Name the new value `file`.
6. Double-click **file** and under **Value data**, type the name of the shared preference file. For example, `s:\shared_prefs\shared_prefs.zip`.
7. Click **OK**.

You have now defined a registry key to point to the shared preference file. Skip to Loading the Shared Preference File for more information about the next steps.

Defining the Location of Shared Preferences Using an Environment Variable

This section contains the procedure for creating an environment variable that points to the shared preference file.

1. Click **Start**, and in the search box, type `environment` and press **ENTER**. The **Environment Variables** page opens.
2. Under **User variables for <your user name>**, click **New**. The **New User Variable** dialog box opens.
3. Under **Variable name**, type `ILLUSTRATE_SHARED_PREFERENCES`.
4. Under **Variable value**, type the name of the shared preference file, for example `\\sharedhost\shared_prefs\shared_prefs.zip`, and click **OK**.
5. In the **Environment Variables** dialog box, click **OK** to close the dialog box.

You have now set an environment variable to point to the shared preference file. Continue to the next section, Loading the Shared Preference File.

Loading the Shared Preference File

After you configure shared preferences, the preferences are automatically loaded and updated in Creo Illustrate. The shared file is copied to the user's `%APPDATA%` directory. It is placed in the designated folder, `shared_prefs`. When the user starts Creo Illustrate, the local copy and the shared file are compared and the local file is updated. For example, if a user has manually modified the XML file for the preferences, these changes are overwritten by the shared preferences file.

If the shared file is missing from the `%APPDATA%` directory, and the shared location is not found, the user cannot start Creo Illustrate at all. To fix the problem, do one of these operations:

- Reconnect to the shared location
- Remove the registry setting or environment variable that points to the shared location

Disabling Shared Preferences

To disable shared preferences, delete the environment variable or the registry entries you added in "Defining the Location of Shared Preferences Using the Registry" or "Defining the Location of Shared Preferences Using an Environment Variable".

Customizing Shared Preferences

To share a set of symbols, stamps, insets, recipes and callouts, the administrator needs to edit the standard and then re-import it using the **Creo Illustrate Options** dialog.

Note

It is recommended that creating shared preferences is done only by users with a good understanding of xml files/structures.

Creating Shared Preferences

To define a shared set of symbols, stamps, insets, recipes, and callouts follow this procedure:

1. In Creo Illustrate modify the illustration level preferences (the standard) as required and save as a new standard (Export as standard).
2. Add that standard zip file (single or multiple) to the standards list in global preferences.
3. Modify the global preferences according to your needs.

4. Close Creo Illustrate.
5. Navigate to `%APPDATA%\ptc\Illustrate`.
6. Copy `user_prefs.xml` and `StandardFiles` folder to a suitable location.
7. Rename `user_prefs.xml` to `shared_prefs.xml`
8. Edit the file `shared_prefs.xml` in a text editor, only keeping the relevant sections for the settings you previously modified (make sure to preserve the nested tag structure those appear in).
9. Create a zip archive of `shared_prefs.xml` and `StandardFiles` folder with a suitable name.

Using Shared Preferences

There are two methods for shared preferences:

1. Environment Variable
    - Create an environment variable `ILLUSTRATE_SHARED_PREFERENCES` and for value enter path to the previously created zip file.
    - Clear previously generated preferences `(%APPDATA%\ptc\Illustrate)` and launch Creo Illustrate.
2. Registry Key
    - In the Registry Editor open `HKEY_LOCAL_MACHINE` and navigate to `SOFTWARE\PTC\CREO ILLUSTRATE`.
    - Create a new key with name `shared_preferences`
    - On the new key create a `String Value` with the name `file`
    - Edit the string value and for `Value data` enter path to the previously created zip file.
    - Close Registry Editor
    - Clear previously generated preferences `(%APPDATA%\ptc\Illustrate)` and launch Creo Illustrate.

Note

Include the name of zip file while adding the path to the environmental variable/registry key.

14

Customizing sBOM XML Files

This chapter describes the sBOM XML document structure, elements, and attributes. You can use this information to customize sBOM XML files for use in Creo Illustrate. (For the sBOM XML document schema, see the separate appendix, sBOM XML Document Schema on page 82.)

About Customizing sBOM XML Files

In Creo Illustrate, you can export or import an sBOM XML file that contains a valid sBOM XML document. This XML document encodes the sBOM tree, figures, and Item Lists for an illustration.

If you export an sBOM XML file, you can edit it and then import it to update your illustration. You can also create a new sBOM XML file and then import it as a template for new illustrations. When you import an sBOM XML file, Creo Illustrate creates or modifies the illustration sBOM tree. It also generates any figures and Item Lists defined in the sBOM XML document markup.

Note

In this appendix the term "part" is used for both single parts and assemblies.

sBOM XML Document Description

The sBOM XML document for Creo Illustrate has three main elements:

<sbom> (Required) Contains the illustration sBOM tree defined by the hierarchy of nested `<instance>` elements. Each `<instance>` element corresponds to a part in the sBOM. (See <sbom> Element on page 66.)

<figures> (Optional) Contains a `<figure>` element for each figure in the illustration. Each `<figure>` contains an optional `<itemlist>`. (See <figures> Element on page 73.)

<itemlist> (Optional) Contains the Item List for a figure. It includes elements that describe the columns and items in the Item List. (See <itemlist> Element on page 74.)

Note

In this appendix, sBOM parts are called "items" when they are referenced or "itemized" in a figure. Itemized parts can appear in the figure's Item List.

sBOM XML Document Structure

```
<import3di>
  <sbom />
  <figures>
    <figure>
    <itemlist />
  </figure>
 </figures>
</import3di>
```

<sbom> Element

The <sbom> element defines the illustration sBOM tree. It can be empty, but most often it contains a nested hierarchy of <instance> elements that form the sBOM tree structure. Each <instance> is a node in the sBOM tree. The <sbom> element has no attributes.

Note

The illustration eBOM is a collection of imported CAD structures.

<sbom> Syntax

```
<sbom>
  <name />
  <instance>
    <instance />
    <instance />
     ...
    <instance />
  </instance>
  <instance />
   ...
  <instance />
</sbom>
```

<sbom> Child Elements

<name>	(Optional) text string name for the root node of the sBOM tree.
<instance>	See <instance> Element on page 67.

<sbom> Example

Use the <name> element to specify a name for the root node of the sBOM tree.

```
<sbom>
  <name>245-9776</name>
```

<instance> Element

The <instance> element is a child of the <sbom> element. The <sbom> element can contain any number of <instance> elements, or it can be empty. An sBOM <instance> can reference a source item in the eBOM, or it can exist in the sBOM only.

Each node in the sBOM tree has an <instance> element. Each <instance> represents one or more parts in the sBOM in a particular service state. They can be hierarchically nested to create the sBOM tree structure.

The attributes and children of <instance> encode service information for parts in the sBOM. They can, for example, convey the following information about sBOM parts:

- Part names
- Part numbers
- Whether or not sBOM parts are referenced in the eBOM
- If parts are itemized and replaceable
- If parts ship preassembled or in a kit

When service information changes for parts in the sBOM, you can the edit the sBOM's <instance> elements. For example, you can add, remove (hide), or change attribute values in the <instance> element itself, or change the values of its child elements.

<instance> Syntax

```
<instance type="FOLDER" qty="1">
  <name />
  <ebom />
  <attribute name="" />
  <attribute name="" />
   ...
  <attribute name="" />
  <instance />
  <instance />
   ...
  <instance />
</instance>
```

<instance> Attributes

type Specifies the type of instance:

FOLDER Default instance type.

REPLACEABLE The instance is a valid, usable, itemizable part.

COLLAPSED The instance is combined with its parent node. It is not itemizable.

PREASSEMBLED The instance is a set of parts that are serviced as single unit. The parent node of this instance is `type="REPLACEABLE"`. All children of this instance are `type= "COLLAPSED"` by default.

qty Specifies the number (quantity) of instances to be created. The default value (if omitted) is 1.

<instance> Child Elements

<name> Optional free form text name for this instance.

<ebom> See <ebom> Element on page 70.

<attribute> An `<instance>` can contain zero or more child `attribute` elements. The values of attributes in `attribute` elements override values in any referenced eBOM node. They can also add new attributes to the sBOM instance.

<instance> An `<instance>` can contain zero or more child `<instance>` elements. For example, an `<instance>` element for an assembly node can contain a child `<instance>` element for each part in the assembly.

<instance> Example 1

This example shows how to use the `<name>` and `attribute` child elements to assign a display name to an instance, assign a new sBOM-only attribute, and to hide (delete) an attribute.

```
<sbom>
 <instance>

  <!-- assign a display name -->
  <name>my sbom instance</name>

  <!-- include a new attribute on the sBOM instance -->
  <attribute name="PartNo" type="symbol" category=“custom”>123-
4567</attribute>

  <!-- hide (delete) an attribute from an instance -->
  <attribute name="Designer" type="symbol" category="PROE
Parameters" hidden="true"/>
```

<instance> Example 2

This example shows how to define instances hierarchically. An example of an assembly instance is shown below. If the instance references an eBOM item which is also an assembly, the entire contents of that assembly are brought across as child `<instance>` elements.

```
<instance qty="1" type="PREASSEMBLED">
 <name>tree </name>

 <!-- The copyasm=true flag specifies that the entire ebom
instance
    and all children should be copied across -->
 <ebom copyasm="true">
  <refpart name=“CustomId” category=" id-db " type="symbol">321-
7654</refpart>
 </ebom>

 <!-- Any instances declared inside this assembly instance (above)
     override the items that were copied over as part of the
parent
     operation. In this case, this specified reference instance
has
     been given a specific display name and assigned the
     REPLACEABLE type.-->
 <instance qty="1" type="REPLACEABLE">
  <name>replacable child </name>
  <ebom>
   <refpart name=“CustomId” category=" id-db " type="symbol">432-
8765</refpart>
  </ebom>
 </instance>
</instance>
```

As shown above, a `type="PREASSEMBLED"` item specifies an assembly which is copied from the eBOM with the structure intact. All of its children are marked as combined (i.e., `type="COLLAPSED"`). Any child definition in the XML will change the sBOM that was created in the parent action. In the example above we find one of the items by property matching. We change its `<name>` and also change its `type` to `REPLACEABLE` (i.e., not-`COLLAPSED`).

<instance> Example 3

This example shows the default instance `type="FOLDER"`. A folder instance named “bolts” is created. It contains a child instance named BOLT representing 12 bolts (`qty="12"`).

```
<instance type=”FOLDER”>
 <name>bolts</name>
 <instance qty="12" type="REPLACEABLE">
  <name>BOLT</name>
  <ebom>
   <refpart name=“CustomId” category=" id-db " type="string">ab-
3344</refpart>
  </ebom>
  <property name="PartNo" type="symbol" category=“custom”> 789-
3344</property>
</instance>
```

<ebom> Element

The `ebom` element is an optional child of an sBOM `<instance>` element. If present, it specifies which eBOM part the sBOM instance is linked to. For example, it provides base attribute and visual settings.

An `<instance>` can contain one child `ebom` element or none. If it contains one, the sBOM instance is linked to a part in the eBOM specified by the `ebom` element. If it contains none, the sBOM instance is not linked to a part in the eBOM. It represents a service-only part that only exists in the sBOM.

<ebom> Syntax

```
<ebom copyasm="" pvcidpath="">
  <refpart name="" category=""></refpart>
</ebom>
```

<ebom> Attributes

copyasm Boolean value (default=`false`). If `true`, the referenced eBOM item and all of its children are copied to the sBOM. This creates a structure in the sBOM that is identical to the referenced eBOM structure.

pvcidpath (Optional) specifies the internal ID path to an eBOM item in the eBOM structure so it can be referenced in the sBOM. If `pvcidpath` is specified, a child `refpart` element is not required.

<ebom> Child Elements

<refpart> See <refpart> Element on page 71.

<refpart> Element

The refpart element is child of the ebom element. It is required if pvcidpath is not specified in the parent ebom element.

Like pvcidpath, the refpart element specifies the eBOM part that an sBOM instance is linked to. However, instead of using a path to specify the link source, refpart uses the values of its attributes, such as <name>, category, and type, to identify the referenced eBOM part.

<refpart> Syntax

```
<refpart name="" category="">CDATA</refpart>
```

<refpart> Attributes

name Specifies the name of an attribute that is used to find a match in the eBOM.

category Specifies the category for the name attribute above.

type (Optional) specifies the type for the name attribute above. If type is omitted, the default is "symbol".

Note

The type attribute is often omitted because most Creo View files are published with attributes of type= "symbol".

<refpart> Child Elements

CDATA Text string value of the specified <name> attribute above. For example, if name="PartNo", then CDATA value is the actual part number text string, such as 123ABC.

<refpart> Example

This example shows an sBOM instance referencing an eBOM part using refpart.

The reference is made to any eBOM item that has the specified property name, category, type and CDATA value. The reference is made on the first unused match found. Once referenced ("used") the eBOM item will not be chosen again.

```
<instance qty="1" type="REPLACEABLE">
  <name>reuse from ebom </name>
  <ebom>
```

```
    <refpart name="CustomId" category=”id-db" type=
"symbol">1234567</refpart>
  </ebom>
</instance>
```

The `<instance>` attribute `qty="1"` means one sBOM item is created and it references one eBOM item. You can use the same syntax to create “n” sBOM items from the referenced eBOM item by setting the instance `qty="n"`.

<figures> Element

The optional `<figures>` element contains one or more `<figure>` elements. Each `<figure>` element corresponds to a figure in the illustration.

<figures> Syntax

```
<figures>
  <figure />
  <figure />
   ...
  <figure />
</figures>
```

<figure> Element

Each `<figure>` element is a placeholder for a figure in the illustration. Each figure has a name and an optional associated Item List. This information is encoded in the `<figure>` element's child elements, `<name>` and `<itemlist>`. The `<figure>` element has no attributes.

<figure> Syntax

```
<figure>
  <name />
  <itemlist />
</figure>
```

<figure> Child Elements

<name>	Text string representing the name of the figure.
<itemlist>	(Optional) Item List associated with this figure. (See <itemlist> Element on page 74.)

<figure> Example

This example shows two figures named `print` and `exploded`. The `print` figure has an Item List. The `exploded` figure does not.

```
</sbom>
<figures>
  <figure>
    <name>print</name>
    <itemlist />
  </figure>
  <figure>
    <name>exploded</name>
  </figure>
</figures>
```

<itemlist> Element

Each figure in an illustration can have an associated Item List described by the `<itemlist>` element. The optional `<itemlist>` element is a child of the `<figure>` element. The `<itemlist>` element has no attributes.

An Item List can have one or more columns and it can contain one or more items. The columns and items in an Item List are encoded in the `<itemlist>` child elements, `<columns>` and `item`.

<itemlist> Syntax

```
<itemlist>
  <columns>
    <column name="" category="" />
    <column name="" category="" />
     ...
    <column name="" category="" />
  </columns>
  <item />
  <item />
   ...
  <item />
</itemlist>
```

<itemlist> Child Elements

<columns> Specifies which columns are shown in the Item List. Contains one or more child `<column>` elements, one for each column in the Item List.

<item> Specifies an item in the Item List. An `<itemlist>` element can contain one or more child `<item>` elements. (See <item> Element on page 75.)

<itemlist> Example

This example shows an `<itemlist>` element for a 4-column Item List.

The first three `<column>` elements have these `name` attributes: `ITEM_INDEX`, `_NAME` and `_COUNT`. Their `category` attribute value is `SYSTEM_VARIABLES`, indicating that these columns are system-provided.

The fourth `<column>` element's `name` attribute is `PartNo`. Its `category` attribute value is `custom`, indicating that this column is custom-defined.

```
<itemlist>
  <columns>
    <!—The ITEM_INDEX, _NAME and _COUNT columns are provided by
the system -->
    <column name="ITEM_INDEX" category="SYSTEM_VARIABLES" />
    <column name="ITEM_NAME" category="SYSTEM_VARIABLES" />
    <column name="ITEM_COUNT" category="SYSTEM_VARIABLES" />
    <!—This custom column will display the PartNo attribute -->
    <column name=" PartNo " category="custom" />
  </columns>
```

<item> Element

The <item> element defines one item in an Item List that is associated with a figure. It is a child of <itemlist>. An <itemlist> can contain one or more <item> elements.

An <item> element can contain one or more child <item> elements. When one item is nested inside another, the child item appears hierarchically indented with respect to its parent.

<item> Syntax

```
<item itemised="">
  <itmlabel />
  <itmtag />
  <itmgroup />
    <refitm />
  <item itemised="" />
  <item itemised="" />
   ...
  <item itemised="" />
</item>
```

<item> Attributes

itemised Boolean value; controls the numbering of the part.

- **true** (Default if omitted.) The callout ID (ITM number) is visible if it exists. A callout can be assigned to this item.
- **false** The callout ID (ITM number) is not visible. No callout can be assigned to this item.

<item> Child Elements

<itmlabel> Specifies the callout ID (ITM number).

<itmtag> Specifies the name (tag) of the item as it appears in the Creo Illustrate user interface **Item List**.

<itmgroup> Specifies the number of items grouped, and what those items are.

<item> An <item> element can contain one or more child <item> elements. Nested <item> elements represent the item hierarchy (or indentation) in an Item List.

<item> Example 1

This example shows an single item comprised of a group of six instances of a bolt, `BOLT (3/8)`. All instances of the bolt have the same specified `name`, `category`, and <refitm> value, `789-3344`. (See <refitm> Element on page 77.)

This item will appear on the **Item List** as ITM 2 and QTY = 6.

```
<item itemised="true">
  <itmlabel>2</itmlabel>
  <itmtag>BOLT (3/8) </itmtag>
  <itmgroup qty="6">
    <refitm name="PartNo" category=“custom” type="symbol"> 789-
3344</refitm>
  </itmgroup>
</item>
```

<item> Example 2

This example shows a hierarchical item; i.e., an item which includes another. Each item is as simple as all the others shown in Example 1, however, the child item is itemized (its `itemised` attribute is true because it is omitted). It also has a label (`<itmlabel>`) assigned.

```
<item>
  <itmlabel>14</itmlabel>
  <itmtag>Parent</itmtag>
  <itmgroup qty="1">
    <refitm name="PartNo" category=“custom” type="symbol">123-
4567</refitm>
  </itmgroup>

  <!-- This is the child item of the parent item.
       The child is itemized and has a label assigned -->
  <item>
    <itmlabel>14A</itmlabel>
    <itmtag>Child</itmtag>
    <itmgroup qty="1">
      <refitm name="PartNo" category=“custom” type="symbol">89a-
bcde</refitm>
    </itmgroup>
  </item>
</item>
```

<itmgroup> Element

The `<itmgroup>` element specifies how many items are grouped and what those items are. It is a child of the `<item>` element.

<itmgroup> Syntax

```
<itmgroup qty="">
  <refitm name="" category="" type="" />
</itmgroup>
```

<itmgroup> Attributes

qty Specifies the number of parts that are collected under this item.

<itmgroup> Child Elements

<refitm> Provides a reference to one or more sBOM items.

<itmgroup> Example

This example references six sBOM parts (`qty="6"`) that match a specific attribute value. A default quantity of 1 (`qty="1"`) is assumed.

```
<itmgroup qty="6">
  <refitm name="PartNo" category=“custom” type="symbol">789-3344</
refitm>
```

<refitm> Element

The `<refitm>` element provides a reference to one or more sBOM items.

<refitm> Syntax

```
<refitm name="" category="" type="">CDATA</refitm>
```

<refitm> Attributes

name Specifies the name of an attribute that is used to find a match in the sBOM.

category Specifies the element category.

type (Optional) specifies the element type. If omitted, `symbol` is the default (`type="symbol"`).

<refitm> Child Elements

CDATA	Text string value of the specified `name` attribute above. For example, if `name="PartNo"`, then CDATA value is the actual part number text string, such as `123ABC`.

How To Import and Export an sBOM XML File

To Import and Apply an sBOM XML File

Use one of these methods to import an sBOM XML file into Creo Illustrate and apply it to the illustration sBOM.

- Import a PVZ file and an sBOM XML file with same name, in same folder. For example, if the folder contains `rocket.pvz`, Creo Illustrate looks for `rocket.xml` in the same folder. If it finds a valid `rocket.xml` file , it applies it to the illustration sBOM.
- If Creo Illustrate is connected to Windchill, import a Windchill GDD with primary PVZ that has an “XML companion file” attachment. If the XML companion file is a valid, Creo Illustrate applies it to the illustration sBOM.
- In Creo Illustrate sBOM Edit mode, on the **Home** tab, click **Import**. In the **Import File** dialog box, type or select the sBOM XML file to import and click **Open**.

To Export an sBOM XML File

In Creo Illustrate sBOM Edit mode, on the **Home** tab, click **Export**. In the **Export sBOM Structure** dialog box, type or select an XML file name and then click **Save**. You can edit the exported sBOM XML file and reimport it.

15

Updating sBOM Files

This chapter describes the elements involved when updating a structure XML file. Export the structure update files from Creo Illustrate, then use these files (current sBOM and updated eBOM) to help create a single structure update file to import back into Creo Illustrate.

See "Importing and Updating sBOM XML Files" in the *Creo Illustrate Help Center* for details about the export and import procedures.

Update structure XML file

The update requires these elements

<update>

(Required) contains one only and only one <sbom> tag

<sbom>

(Required) Defines the start, under this tag will appear <delete> and or <replace> tags.)

<delete>

(Optional) Contains a list of <oldsbom> tags, each tag defines an sBOM instance that the user wants to delete.

<replace>

(Optional) Contains a list of <pair> tags, each pair defines a part replacement (like the input for **Edit Structure ▸ Replace Part** in Creo Illustrate).

<delete> Section

Contains a list of <oldsbom> tags, each tag defines an sBOM instance that the user wants to delete.

<oldsbom>—Holds an sbomidpath attribute which defines the idpath to the sBOM instance.

Example of <delete> Section:

```
<delete>
      <oldsbom sbomidpath="/0/0"/>
      <oldsbom sbomidpath="/9/2/10"/>
      <oldsbom sbomidpath="/3/0/0"/>
</delete>
```

<replace> Section

Contains a list of <pair> tags, each <pair> defines a part replacement.

Each <pair> includes exactly one <oldsbom> tag followed by exactly one <newebom> tag.

<oldsbom>

Contains an sbomidpath attribute that defines the idpath to the sBOM instance to replace its eBOM part.

Example:

```
<oldsbom sbomidpath="/3/0/10"/>
```

<newebom>

Includes a pvcidpath parameter that defines the idpath to the new eBOM part to attach to the sBOM instance defined in <oldsbom> tag. The data of <newebom> is either empty (in which case, the name of the eBOM instance is used), or has a new name to apply to the sBOM instance after the replacement.

Examples

```
<newebom pvcidpath=":0/75"/>
<newebom pvcidpath=":0/43">NewName for the part</
newebom>
<newebom pvcidpath=":3/@@PV-AUTO-ID@@000/@@PV-
AUTO-ID@@002"/>
<newebom pvcidpath=":3/@@PV-AUTO-ID@@000/@@PV-
AUTO-ID@@002">
                NewName for the part</newebom>
```

Example of <replace>

```
<replace>
       <pair>
              <oldsbom sbomidpath="/0/1"/>
              <newebom pvcidpath=":0/43">NewName for the part</
newebom>
       </pair>
       <pair>
              <oldsbom sbomidpath="/3/0/5"/>
              <newebom pvcidpath=":0/75"/>
        </pair>
        <pair>
              <oldsbom sbomidpath="/3/0/3"/>
              <newebom pvcidpath=":3/@@PV-AUTO-ID@@000/@@PV-AUTO-
ID@@002">
```

```
               NewName for the part</newebom>
        </pair>
</replace>
```

16

sBOM XML Document Schema

This chapter lists the XML schema for the sBOM XML document used in Creo Illustrate.

About the XML Schema

The XML schema listed below describes the following properties of the sBOM XML document:

- Elements and attributes that can appear in the document, along with their data types and values
- Which elements are child elements, and the order and number of child elements
- Whether an element is empty or can include text

sBOM XML files exported from Creo Illustrate follow this schema. Any sBOM XML files you create or edit for import into Creo Illustrate must also follow this schema.

Note

For more information on the sBOM XML Document, see Customizing sBOM XML Files on page 64.

XML Schema Listing

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="column">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="columnSimpleType">
          <xs:attribute name="name" type="xs:string" use=
"required"/>
          <xs:attribute name="category" type="xs:string" use=
"required"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:simpleType name="columnSimpleType">
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
  <xs:element name="columns">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="column" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="ebom">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="refpart" type="xpropertyref" minOccurs=
"0"/>
      </xs:sequence>
      <xs:attribute name="copyasm" type="xs:boolean"/>
      <xs:attribute name="pvcidpath" type="xs:string"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="figure">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="name"/>
        <xs:element ref="itemlist" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:string" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="figures">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="figure" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="import3di">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="sbom"/>
        <xs:element ref="figures" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="instance">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="name" minOccurs="0"/>
        <xs:element ref="ebom" minOccurs="0"/>
        <xs:element ref="attribute" minOccurs="0" maxOccurs=
"unbounded"/>
        <xs:element ref="instance" minOccurs="0" maxOccurs=
"unbounded"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:string"/>
      <xs:attribute name="qty" type="xs:integer" default="1"/>
      <xs:attribute name="type" use="optional" default="FOLDER">
        <xs:simpleType>
          <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="PREASSEMBLED"/>
            <xs:enumeration value="REPLACEABLE"/>
            <xs:enumeration value="FOLDER"/>
            <xs:enumeration value="COLLAPSED"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
  <xs:element name="item">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="itmlabel" minOccurs="0"/>
        <xs:element ref="itmtag"/>
        <xs:element ref="itmgroup"/>
        <xs:element ref="item" minOccurs="0" maxOccurs=
"unbounded"/>
      </xs:sequence>
      <xs:attribute name="itemised" type="xs:boolean" default=
"true"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="itemlist">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="columns" minOccurs="0"/>
        <xs:element ref="item" minOccurs="0" maxOccurs=
"unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="itmgroup">
    <xs:complexType>
      <xs:choice>
        <xs:element ref="refitm" maxOccurs="unbounded"/>
        <xs:element ref="itmpart"/>
      </xs:choice>
      <xs:attribute name="qty" type="xs:integer"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="itmlabel" type="xs:string"/>
  <xs:element name="itmpart" type="xs:string"/>
  <xs:element name="itmtag">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="auto" type="xs:boolean"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="listtag" type="xs:string"/>
  <xs:element name="name" type="xs:string"/>
  <xs:element name="attribute">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="name" type="xs:string" use=
"required"/>
          <xs:attribute name="type" type="xs:string" default=
"symbol"/>
          <xs:attribute name="category" type="xs:string" use=
"optional"/>
          <xs:attribute name="hidden" type="xs:boolean" default=
"false"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="refitm">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xpropertyref">
          <xs:attribute name="sbomidpath" type="xs:string"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="property">
    <xs:complexType>
      <xs:attribute name="name" type="xs:string" use="required"/>
      <xs:attribute name="category" type="xs:string" use=
"optional"/>
      <xs:attribute name="type" type="xs:string" default="symbol"/
>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="xpropertyref">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="name" type="xs:string" use="required"/
>
        <xs:attribute name="category" type="xs:string" use=
"optional"/>
        <xs:attribute name="type" type="xs:string" default=
"symbol"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:element name="sbom">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="name" minOccurs="0"/>
        <xs:element ref="instance" minOccurs="0" maxOccurs=
"unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

17

Creo Illustrate Symbol Library File: Tag Editing

Use a text editor to edit the Creo Illustrate Symbol Library template XML file:

```
%APPDATA%\ptc\Illustrate\SymbolsLibrary\
PartSymbolLibrary.xml
```

This file contains the XML markup for library items and their tags.

About Library Item Elements

`<symbol>` and `<shape>` elements in the Symbol Library XML file represent 2D and 3D library items.

- `<symbol>` elements are 2D library items, such as signs or arrows.
- `<shape>` elements are 3D library items such as 3D models of tools and consumable supply containers.

About Library Item Tag Elements

You can tag library items in the Symbol Library XML file with different categories of tags, such as resource or message tags. To tag a library item, do the following:

1. Add a `<tags>` element to a `<symbol>` or `<shape>` element.
2. Add one or more `<tag>` elements to the `<tags>` element. The tag type is defined by the `type` attribute value of the `<tag>` element. Valid `type` attribute values are listed below.
3. (Optional) Add `<description>` and `<quantity>` elements to each `<tag>` element.

<tag> Elements for Resources

In addition to the tag icon, these tag types appear in the Step Editor **Resources** tab for the step.

type Attribute	Description
consumable	Item has variable consumption, such as 3 liters of oil. Its child element, <quantity>, includes a unit attribute.

<tag> Elements for Messaging

In addition to the tag icon, these tag types display a message in the Step Editor **Description** tab for the step.

type Attribute	Description
safety	Displays a warning tag icon and message for this step.
quality	Displays a quality tag icon and message for this step.
skill	Displays a skill tag icon and message for this step.
environmental	Displays an environmental tag icon and message for this step.
procedure	Displays a procedure tag icon and message, such as inspection instructions, for this step.

Syntax for Tagged 2D <symbol> Elements

Use the syntax below to create tags for 2D <symbol> elements.

```
<symbol id="[n]" categoryId="[n]" billboard="[true|false]">
  <name>[Item Name]</name>
  <description>[item description]</description>
  <tooltip>[tooltip text]</tooltip>
  <gallery_thumbnail url="[path/filename.64x64.jpg]"/>
  <image url="[path/filename.jpg]"/>
  <scale>[0-100]</scale>
  <flip horizontal="[true|false]" vertical="[true|false]"/>
  <rotate right="[0-360]" left="[0-360]"/>
  <tags>
    <tag type="[type_name]" category=”[category_name]”>
      <description>[description text]</description>
      <quantity unit="[unit_symbol">[value]</quantity>
    </tag>
    <tag>[]</tag>
```

```
      …
    <tag>[]</tag>
  </tags>
</symbol>
```

Syntax for Tagged 3D <shape> Elements

Use the syntax below to create tags for 3D symbol <shape> elements.

```
<shape id="[n]" categoryId="[n]">
  <name>[Item Name]</name>
  <description>[item description]</description>
  <tooltip>[tooltip text]</tooltip>
  <gallery_thumbnail url="[path/filename.64x64.jpg]"/>
  <graphic url="[path/filename.ol]"/>
  <scale>[0-100]</scale>
  <orientation>[0-10]</orientation>
  <tags>
    <tag type="[type_name]" category=”[category_name]”>
      <description>[description text]</description>
      <quantity unit="[unit_symbol">[value]</quantity>
    </tag>
    <tag>[]</tag>
      …
    <tag>[]</tag>
  </tags>
</shape>
```

Example

The following markup in the Symbol Library XML file creates a library item named Glue Tube, a 3D symbol <shape> element representing a metal bonding adhesive container.

consumable tag—Indicates that the item symbolized (glue) is consumed in this sequence. It contains two child elements, <description>, for glue application instructions, and <quantity>, to specify the container volume, 177 milliliters (unit="ml").

```
<shape id="2" categoryId="4">
  <name>Glue Tube</name>
  <description>Metal bonding adhesive</description>
  <gallery_thumbnail url="SymbolsLibrary\Shape\Glue_tube64x64.
jpg"/>
  <graphic url="SymbolsLibrary/Shape/Glue_tube.ol"/>
  <tags>
    <tag type="consumable" category=”resource”>
      <description>Apply generous coating to both joining
surfaces</description>
      <quantity unit="ml">177</quantity>
    </tag>
  </tags>
</shape>
```

18

Customizing Page Size

This chapter provides instructions on how to add Page Size configuration for 2D publishing and export in Creo Illustrate.

To Add New Page Sizes

In the `user_prefs.xml` file, locate the category called `<category name= "Publish">`.

Add a new sub-category called `<category name="Preview"></ category`.

Page Size Elements Description

The following elements define the `Preview` category:

<unit>	Unit Values: • 5 = mm • 11 = inch • 12 = point
<SizeName>	Name displayed in menu
<height>	
<width>	

Syntax for Page Size Elements

This example shows two Page Size options.

```
<category name="Publish">
 <category name="Preview">
  <preferenceList name="DisplaySizeTable">
   <preferenceItem>
```

```
    <preference name="height" value="5.5"/>
    <preference name="width" value="7"/>
    <preference name="unit" value="11"/>
    <preference name="SizeName" value="New Page 1"/>
   </preferenceItem>
   <preferenceItem>
    <preference name="height" value="75"/>
    <preference name="width" value="100"/>
    <preference name="unit" value="5"/>
    <preference name="SizeName" value="New Page 2"/>
   </preferenceItem>
  </preferenceList>
 </category>
</category>
```

19

Customizing Lighting Setups

This chapter describes the structure, elements, and attributes of the Light Scenes XML document. (For the XML document schema, see the separate appendix, Lighting Setups XML Document Schema.)

About Customizing Lighting Setups

Lighting setups in the Creo Illustrate **Lights** list are encoded the Light Scenes XML document. Creo Illustrate uses the Light Scenes XML document stored in this XML text file,
`LightScenes.xml`, which is in the `LightScenes` folder that is in the exported standard zip file.

See “Working with Standards” In the *Creo Illustrate Help Center*.

You can edit this file to change or delete lighting setups in the **Lighting Setup** list. (To open the list in Creo Illustrate, click the ribbon **Figure** tab, and then click the **Lighting Setup** box in the **Lights** group.) Lighting setups include properties for light type, color, and direction. To update the **Lighting Setup** list after you edit the XML file, place a copy of the updated file in
`LightScenes.xml`
and then restart Creo Illustrate.

About the Light Scenes XML Document

The main elements in the Light Scenes XML document are summarized below. All elements and attributes in the XML document are specified separately in the topics that follow.

About Element and Attribute Types in the Light Scenes XML Document

All elements and attributes are the String data type unless their descriptions in this appendix starts with Mixed, Boolean, or Restricted in parentheses as shown below.

- (Mixed) elements can contain both text and other elements.
- (Boolean) attributes can only be set `true` or `false`.
- (Restricted) attributes can only be set to the values enumerated in their description.

In addition, all attributes are required unless their description in this appendix starts with (Optional).

Main Elements in the Light Scenes XML Document

<lightscenes> Lighting setup list element. Contains one `<lightscene>` element for each lighting setup in the list. (See <lightscenes> Element.)

<lightscene> Lighting setup element. Contains the properties for one lighting setup. (See <lightscene> Element.)

<name> Lighting setup name element. Can contain one of three localized predefined names or a custom name. (See <name> Element.)

<light> Lighting element. Defines the type, color, and direction for one light in a lighting setup. A lighting setup (`<lightscene>` element) contains one or more lights (`<light>` elements). (See <light> Element.)

Structure of the Light Scenes XML Document

```
<?xml version="1.0" encoding="UTF-8"?>
<lightscenes>
  <lightscene>
    <name />
    <light>
      <color />
      <position />
      <direction />
      <angle></angle>
      <exponent></exponent>
    </light>
     …
    <light />
    <light />
  </lightscene>
  <lightscene />
   …
  </lightscene />
</lightscenes>
```

<lightscenes> Element

The <lightscenes> element defines all lighting setups in the Creo Illustrate **Lighting Setup** list. It contains one or more child <lightscene> elements. The <lightscenes> element has no attributes. An XML file can contain only one <lightscenes> element.

<lightscenes> Syntax

```
<lightscenes>
  <lightscene />
  <lightscene />
   …
  <lightscene />
</lightscenes>
```

<lightscenes> Child Elements

<lightscene> Lighting setup element. Contains the properties for one lighting setup. (See <lightscene> Element.)

<lightscenes> Example

The <lightscenes> element below creates two lighting setups in the **Lighting Setup** list:

- DEFSingle—A single-light setup with one key light and a predefined localized name.
- TwoLights—A two-light setup with one key light and one fill light and a custom non-localized name.

```
<lightscenes>
  <lightscene>
    <name locid="DEFSingle"/>
    <light name="key" type="direct" relative="view" />
      <color />
      <direction />
  </lightscene>
  <lightscene>
    <name>TwoLights</name>
    <light name="key" type="direct" relative="view" />
      <color />
      <direction />
    <light name="fill" type="direct" relative="view" />
      <color />
      <direction />
  </lightscene>
</lightscenes>
```

<lightscene> Element

The `<lightscene>` element is a child of the `<lightscenes>` element. Each `<lightscene>` element encodes the properties for one lighting setup containing one or more lights (`<light>` elements).

Each `<lightscene>` element has a unique name defined in its `<name>` element. The `<name>` element can contain an optional predefined localized lighting setup name or a custom name.

The `<lightscenes>` element has no attributes.

<lightscene> Syntax

Example: Lighting setup with a predefined localized name:

```
<!-- Lighting setup with a predefined localized name -->
<lightscene>
  <name locid="DEFlocName"/>
  <light />
  <light />
   …
  <light />
</lightscene>
```

Example: Lighting setup with a custom non-localized name:

```
<!-- Lighting setup with a custom non-localized name -->
<lightscene>
  <name>Lighting Setup Name</name>
  <light />
  <light />
   …
  <light />
</lightscene>
```

<lightscene> Child Elements

<name> (Mixed) Specifies the lighting setup name shown in the **Lighting Setup** list. (See <name> Element.)

<light> Defines the type, color, and direction for one light in a lighting setup. A `<lightscene>` element contains one or more `<light>` elements. (See <light> Element.)

<lightscene> Example

This example shows the predefined lighting setup, `DEFTwoLights`, encoded in a `<lightscene>` element.

```
<lightscene>
  <name locid="DEFTwoLights"/>
  <light name="key" type="direct" relative="view">
```

```
    <color ambient="#141414" diffuse="#BEBEBE" specular="#CCCCCC"/
>
    <direction x="-60.00" y="15.00" z="5.00"/>
  </light>
  <light name="fill" type="direct" relative="view">
    <color ambient="#050505" diffuse="#969696" specular="#AFAFAF"/
>
    <direction x="10.00" y="60.00" z="0.00"/>
  </light>
</lightscene>
```

<name> Element

The <name> element is a child of the <lightscene> element. It contains either a locid attribute that defines a predefined localized name, or, it contains a custom name. The <name> element has no child elements.

Note

The Creo Illustrate installation includes predefined and localized locid attribute values for the <name /> child element. These locid values have the prefix DEF and are reserved by Creo Illustrate. Do not edit locid values. Rather, if you want to change the lighting setup name, replace the self-closing element with an element of the same name that has a start and an end tag; for example, <name>My Name</name>. The content of the <name> element, My Name, overrides the locid value (if present).

<name> Syntax

```
<!-- Predefined localized name -->
<name locid="DEFlocName"/>

<!-- Custom name -->
<name>Lighting Setup Name</name>

<!-- Custom name overrides predefined name -->
<name locid="DEFlocName">Lighting Setup Name</name>
```

<name> Attributes

locid

(String) Optional, unique predefined and localized text for the lighting setup name. If the <name> element contains custom name text, the custom name overrides the predefined name in the locid attribute.

<name> Example

This example shows a `<name>` element with a custom name, `Key and Fill Lighting`, that overrides the predefined name in the `locid` attribute, `DEFTwoLights`. The **Lighting Setup** list in Creo Illustrate will show **Key and Fill Lighting**.

```
<name locid="DEFTwoLights">Key and Fill Lighting</name>
```

<light> Element

The `<light>` element is a child of the `<lightscene>` element. The `<light>` element defines the type, color, and direction for one light in a lighting setup.

<light> Syntax

```
<light name="Light Name" type="[direct|spot|point]" relative=
"[view|scene]">
  <color />
  <position />
  <direction />
  <angle>
  <exponent>
</light>
```

<light> Attributes

name (String) Specifies the name of the light.

type (String) Specifies the light type as `direct` (default), `spot`, or `point`.

relative (String) Specifies whether the light is relative to the `view` (default) or the `scene`.

> **Note**
>
> For orthographic projections, if you set the `<light>` attribute `type="spot"` or `type="point"`, you should only set the attribute `relative="scene"`. Setting `relative="view"` can produce unpredictable results.

<light> Child Elements

<color /> Specifies the color intensity of the ambient, diffuse, and specular light settings. (See <color> Element .)

<position /> For `spot` and `point` lights only—Specifies the XYZ coordinates for the location of the point or spot light source, assuming it not infinitely far from the scene. (See <position> Element .)

<direction /> For `direct` and `spot` lights only—Specifies the XYZ vector for the direction of the direct or spot light source. (See <direction> Element .)

<angle> For `spot` lights only—Specifies the cutoff angle for the spot light source. (See <angle> Element .)

<exponent> For `spot` lights only—Specifies the exponent for the spot light source. (See <exponent> Element .)

<light> Example

This example shows the light element for a spot light source that has a position in the scene.

```
<light name="back" type="spot" relative="scene">
  <color ambient="#0A0A0A" diffuse="#FFB669" specular="#E0E0E0"/>
  <position x="71.00" y="66.00" z="24.00"/>
  <direction x="-60.00" y="60.00" z="-20.00"/>
  <angle>10.00</angle>
  <exponent>128</exponent>
</light>
```

<color> Element

The `<color>` element is a child element of `<light>` that specifies the intensity of the ambient, diffuse, and specular settings for the RGB hexadecimal color codes. (The `<color>` element is self-closing. It has no content or child elements.)

<color> Syntax

```
<color ambient="#hexColor" diffuse="#hexColor" specular=
"#hexColor" />
```

<color> Attributes

ambient Specifies the ambient light setting as a RGB hexadecimal color code. Valid values are `#000000` to `#FFFFFF`. This attribute value corresponds to the **Ambient** setting in the **Custom** lighting setup dialog box.

diffuse Specifies the diffuse light setting as a RGB hexadecimal color code. Valid values are `#000000` to `#FFFFFF`. This attribute value corresponds to the **Diffuse** setting in the **Custom** lighting setup dialog box.

specular Specifies the specular light setting as a RGB hexadecimal color code. Valid values are `#000000` to `#FFFFFF`. This attribute value corresponds to the **Specular** setting in the **Custom** lighting setup dialog box.

<color> Example

```
<color ambient="#333333" diffuse="#CCCCCC" specular="#CCCCCC"/>
```

<position> Element

The `<position>` element is a child element of `<light>` for `spot` and `point` lights only. It specifies the XYZ coordinates for the location of the point or spot light source that is not infinitely far from the scene. If the `<position>` element is omitted, the point or spot light source is considered infinitely far away; i.e., rays of light from the source are considered parallel.

The `<position>` element is self-closing. It has no content or child elements.

<position> Syntax

```
<position x="xValue" y="yValue" z="zValue" />
```

<position> Attributes

x y z (Float) Specifies the ±(X, Y, or Z) coordinate for the point or spot light source location.

<position> Example

This example shows a `<position>` element for the XYZ location (-150, 150, 600).

```
<position x="-150.00" y="150.00" z="600.00" />
```

<direction> Element

The <direction> element is a child element of <light> for direct and spot lights only. It determines the axis of the cone of light for the light source. The XYZ vector coordinates are taken relative to the origin at the vertex of the light cone.

(The <direction> element is self-closing. It has no content or child elements.)

<direction> Syntax

```
<direction x="xValue" y="yValue" z="zValue" />
```

<direction> Attributes

x y z	(Float) Specifies the ±(X, Y, or Z) coordinate for the direction vector along the axis of the direct or spot light.

<direction> Example

This example shows a <direction> element with the XYZ vector coordinates (-60, 60, -20).

```
<direction x="-60.00" y="60.00" z="-20.00"/>
```

<angle> Element

The <angle> element is a child element of <light> for spot lights only. It specifies the cutoff angle in positive degrees from 0.00 to 180.00 as Float. The cutoff angle is the angle between the axis of the light cone and a ray along the edge of the cone.

<angle> Syntax

```
<angle>degValue</angle>
```

<angle> Example

This example shows an <angle> element with a cutoff angle of 10 degrees.

```
<angle>10.00</angle>
```

<exponent> Element

The <exponent> element is a child element of <light> for spot lights only. It specifies the exponent for the spot light source as a positive whole number from 0 (default) to 128 as Integer. Increase the exponent value to increase the focus or to concentrate the spot light source.

<exponent> Syntax

```
<exponent>expValue</exponent>
```

<exponent> Example

This example shows an `<exponent>` element with a exponent value of 128.

```
<exponent>128</exponent>
```

Light Scenes XML Document Examples

Light Scenes XML Document As Installed

The installed version of the Light Scenes XML document in `LightScenes.xml` is listed below. It contains three predefined lighting setups that appear on the **Lighting Setup** list in Creo Illustrate: `DEFSingle` (**Single Light**), `DEFTwoLights` (**2 Lights**), and `DEFThreeLights` (**3 Lights**).

```
<?xml version="1.0" encoding="UTF-8"?>
<lightscenes>
  <lightscene>
    <name locid="DEFSingle"/>
    <light name="key" type="direct" relative="view">
      <color ambient="#3C3C3C" diffuse="#BABABA" specular=
"#DCDCDC"/>
      <direction x="-60.00" y="15.00" z="5.00"/>
    </light>
  </lightscene>
  <lightscene>
    <name locid="DEFTwoLights"/>
    <light name="key" type="direct" relative="view">
      <color ambient="#141414" diffuse="#BEBEBE" specular=
"#CCCCCC"/>
      <direction x="-60.00" y="15.00" z="5.00"/>
    </light>
    <light name="fill" type="direct" relative="view">
      <color ambient="#050505" diffuse="#969696" specular=
"#AFAFAF"/>
      <direction x="10.00" y="60.00" z="0.00"/>
    </light>
  </lightscene>
  <lightscene>
    <name locid="DEFThreeLights"/>
    <light name="key" type="direct" relative="view">
      <color ambient="#0A0A0A" diffuse="#B4B4B4" specular=
"#E0E0E0"/>
      <direction x="-60.00" y="15.00" z="5.00"/>
    </light>
    <light name="fill" type="direct" relative="view">
      <color ambient="#0A0A0A" diffuse="#CCCCCC" specular=
"#CCCCCC"/>
      <direction x="10.00" y="60.00" z="0.00"/>
    </light>
    <light name="back" type="direct" relative="view">
      <color ambient="#0A0A0A" diffuse="#CCCCCC" specular=
"#CCCCCC"/>
      <direction x="320.00" y="190.00" z="125.00"/>
    </light>
  </lightscene>
</lightscenes>
```

Light Scenes XML Document with All Light Types

The following XML document shows a 3-light lighting setup with definitions for all light types; `direct`, `point`, `spot`

```
<?xml version="1.0" encoding="UTF-8"?>
<lightscenes>
  <lightscene>
    <name locid="DEFThreeLights">Three Light Setup</name>
    <light name="key" type="direct" relative="view">
      <color ambient="#333333" diffuse="#CCCCCC" specular=
"#CCCCCC"/>
      <direction x="-60.00" y="-16.10" z="9.34"/>
    </light>
    <light name="fill" type="point" relative="scene">
      <color ambient="#333333" diffuse="#CCCCCC" specular=
"#CCCCCC"/>
      <position x="-150.00" y="150.00" z="600.00"/>
    </light>
    <light name="back" type="spot" relative="scene">
      <color ambient="#0A0A0A" diffuse="#ffb669" specular=
"#E0E0E0"/>
      <position x="71.00" y="66.00" z="24.00"/>
      <direction x="-60.00" y="60.00" z="-20.00"/>
      <angle>10.00</angle>
      <exponent>128</exponent>
    </light>
  </lightscene>
</lightscenes>
```

20

Lighting Setups XML Document Schema

This chapter lists the XML schema for the lighting setups XML document used in Creo Illustrate.

About the XML Schema

The XML schema listed below defines the following characteristics of the Light Scenes XML document:

- Elements and attributes that can appear in the document, along with their data types and values
- Which elements are child elements, and the order and number of child elements
- Whether an element is empty or can include text

Any custom Light Scenes XML files that you intend to use with Creo Illustrate must follow this schema.

Note

For more information on the Light Scenes XML Document, see Customizing Lighting Setups.

XML Schema Listing

```
<?xml version="1.0" encoding="utf-16"?>
<xsd:schema attributeFormDefault="unqualified" elementFormDefault=
"qualified"
           version="1.0" xmlns:xsd="http://www.w3.org/2001/
XMLSchema">
```

```
  <xsd:element name="lightscenes">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element maxOccurs="unbounded" name="lightscene">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="name">
                <xsd:complexType>
                  <xsd:attribute name="locid" type="xsd:string" />
                </xsd:complexType>
              </xsd:element>
              <xsd:element name="light">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="color">
                      <xsd:complexType>
                        <xsd:attribute name="ambient" type="xsd:
string" />
                        <xsd:attribute name="diffuse" type="xsd:
string" />
                        <xsd:attribute name="specular" type="xsd:
string" />
                      </xsd:complexType>
                    </xsd:element>
                    <xsd:element name="direction">
                      <xsd:complexType>
                        <xsd:attribute name="x" type="xsd:decimal"
/>
                        <xsd:attribute name="y" type="xsd:decimal"
/>
                        <xsd:attribute name="z" type="xsd:decimal"
/>
                      </xsd:complexType>
                    </xsd:element>
                  </xsd:sequence>
                  <xsd:attribute name="name" type="xsd:string" />
                  <xsd:attribute name="type" type="xsd:string" />
                  <xsd:attribute name="relative" type="xsd:string"
/>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

21

3D Publishing Options

A recipe is a set of tool-specific default rules used to configure the publishing of Creo View 3D model data from Creo Illustrate. The Creo View Adapters recipes control the level of information to publish. Sample recipe files are located at `C:\Program Files (x86)\PTC\Creo 7.1.0.0\View Files Tools\recipe`. The recipe files are annotated text files that contain details about the purpose of certain settings, and groups of related settings. You can manually edit the contents to adjust the level of information that is published.

These recipe files are included with Creo Illustrate:

- `pvsoptimize_for_illustration.rcp`
- `pvsoptimize_for_illustration_with_view_state.rcp`

Note

- Most data are converted efficiently using these default recipes, but some workflows require modifications to improve the detail and quality of the viewable, to remove intellectual property, or reduce file size.
- Editing recipe settings can have subtle and unexpected effects on the output. Avoid editing unless instructed to do so by PTC.

See “Defining Additional Viewable Compression Recipes” in the *Creo View MCAD Adapters Installation and Configuration Guide* for more information on these recipes.

See “Settings for Publishing Figures to 3D” in the *Creo Illustrate Help Center* for information on setting recipe options.

22

Configuring 3D and Schematic Standards

This chapter provides an overview of company standards, formats, and profiles for 3D and Schematic illustrations. For user information and help, see the Creo Illustrate Help.

Creo Illustrate is delivered with a default standard. Changes to CAD standard entities can be applied using the Creo Illustrate UI **Options**. To change schematic standard content, follow this workflow:

1. Unzip the standard zip file.
2. Modify the `standard.xml` file in an editor.
3. Zip the folder with the edited XML file and all files and folders that you previously unzipped.
4. In the Creo Illustrate **Options**, select **Global** ▸ **Standard** ▸ **Add** to browse to the modified standard to add it to the **Standard** library.

See “About Standards” in the Creo Illustrate Help for more information about Standards.

About Standards for the 3D Illustrator

A `<standard>` defines industry and company standards for 3D figures in Creo Illustrate. The standard is a toolbox or palette that contains all the editing and appearance options for your enterprise. The user must select a standard before creating a new schematic illustration. You can define multiple standards for the user to choose from.

The standard is a prepared XML document that is saved with the file.

The following entities are defined in a standard:

- Standard ID—Each standard must have a unique ID.
- Name—Specify a name for each illustration standard. This name does not have to be unique.
- `length_units`—Defines the length units for all numeric values of the standard. Default value is millimeters (optional).

Note

Standard ID and name are shared by 3D and Schematic.

This is an example of a script containing the ID, a name, and a length unit:

```
<standard id="4e93c9c8-4cc4-4d0b-a067-818364c35a8b"/>
name="New full Illustrate Standard" length_units="cm">
```

A standard may contain the following child tags in any order. They are all optional.

```
<formats>
<attr_rules>
<profiles>
```

A 3D standard may contain `<profiles>`. For 3D Standards, continue to “Defining Profiles” below.

Defining Formats

The format defines the page setup of the figure. It includes definitions of the following features:

- Name
- Border
- Navigation grid
- Title block

Note

Only title block is an optional feature. The others are mandatory.

Name

The format’s name identifies it. If two formats have the same name, both are listed.

This is an example of a format name:

```
<format name="format A">
```

`<border>`

The border is a single line running around the edge of the figure that defines the usable area. It includes the following features:

Rule	Description
`width`	Defines the width of the format border.
`height`	Defines the height of the format border.
`line_color`	Specifies the color of the border.
`line_weight`	Specifies the thickness of the border.

This is an example of a border script:

```
<border width="20.0" height="10.0" line_color="#bf8230" line_weight="0.1"/>
```

`<navgrid>`

The navigation grid divides the format vertically and horizontally into segments. The user controls the navigation grid’s visibility. Define these details of the grid:

Rule	Description
`start_point`	Designates the corner of the border where the reference numbers and letters (grid labels) start (A1).
`line_spacing`	Defines the spacing between grid lines (vertical and horizontal).
`line_color`	Specifies the color of the navigation grid when visible.
`font_size`	Controls font size of the grid.
`offset`	Defines the distance of the grid labels from the border.

This is an example of a navigation grid script:

```
<navgrid start_point="bottom left" line_spacing="2.0" line_color=
"#a65f00"
 font_size="10" offset="1.0"/>
```

`<titleblock>`

The title block is a rectangular block with an outline and one or more sections of text. It contains a title or description and can contain references to the illustrated items and additional notes and references.

- A list of title blocks can be empty. It can also have one or more `<titleblock>` tags.
- A list of text sections can be empty. It can also have one or more `<section>` tags.
- The title block contains a title section and other sections for text.
- The font in all sections is constant, but of different sizes and styles (bold, italics).
- Each section inherits the settings from the previous section, unless otherwise defined.
- If a line of text is too long, it wraps onto a new line.
- The height of a title block is the sum of all lines of text, including line spacing, and the margin.

The definition for the title block outline includes the following details:

Rule	Description
`name`	The name must be unique.
`min_width`	Sets the minimum width of the title block when re-sized.
`min_height`	Sets the minimum height of the title block when re-sized.
`position`	Sets the corner of the border that the title block is aligned with. The stretch direction away from the border
`line_color`	Defines the color of the title block border.
`line_weight`	Defines the width of the title block border.
`bgcolor`	Defines the color of title block background.
`margin`	Defines the minimal distance of the text sections from the title block borders.

Rule	Description
`offset_x`	Defines horizontal distance from the border. Maximum `offset_x` = border width minus `min_width`
`offset_y`	Defines vertical distance from the border. Maximum `offset_y` = border height minus `min_height`

Note

If no offsets are defined, the two sides to which the title block is aligned, touch the border.

The definition of text for a section of a title block includes the following details:

Rule		Description
`sections` There are three types of text in a title block.	`const_text`	User cannot delete or change.
	`edit_text`	User can edit.
	`dynamic_text`	The text is interpreted and replaced by the Creo Illustrate application in real time, but the user cannot delete or change.
`font_size`		Sets font size for the text included in this section.
`bold`		Sets style for the text included in this section.
`italic`		Sets style for the text included in this section.
`font_color`		Sets font color for the text included in this section.
`alignment`		Aligns the text section to the left, middle, or right.
`section_spacing`		Defines the space between the current section and the following section.

This is an example of text for a section of a title block outline and script:

```
<titleblocks>
 <titleblock name="Title block - Text types" min_width="90.0"
offset_x="1"
     offset_y="1"line_color="#000000" line_weight="0.3" bgcolor=
"#ffffff"
     margin="1"position="bottom right">
  <sections>
   <section font_size="5" bold="true" italic="false" underline=
"false"
     strikeout="false"font_color="#000000" alignment="left"
     section_spacing="0.1">
    <const_text>This is constant text, </const_text><edit_text>
     This text can be edited </edit_text><const_text>
     the following are all dynamic text:</const_text>
    <const_text>&#xA;author = </const_text><dynamic_text>author
         </dynamic_text>
    <const_text>&#xA;created = </const_text><dynamic_text>created
         </dynamic_text>
    <const_text>&#xA;figure name = </const_text><dynamic_
text>figure name
         </dynamic_text>
    <const_text>&#xA;file name = </const_text><dynamic_text>file
name
         </dynamic_text>
    <const_text>&#xA;last modified = </const_text><dynamic_text>
         last modified</dynamic_text>
    <const_text>&#xA;last modified by = </const_text><dynamic_
text>
                                     last modified by</dynamic_
text>
   </section>
   <section font_size="4" bold="false" italic="true" underline=
"false"
          strikeout="false" font_color="#4a69bd" alignment=
"middle"
          section_spacing="0.1">
    <const_text>Multiple sections can be defined.
                    Each section can have its own styling.</const_
text>
   </section>
  </section>
  </titleblock>
```

Continue to the next section for information on defining attribute rules.

Defining Attribute Rules

Attribute Rules (<attr_rules>) are optional rules that define when an attribute appears in the attributes list. Use attribute rules to hide or to lock attributes, and to enable localization of attributes in illustrations. A standard can contain rules to control the visibility of attributes in the source data as well as to lock attributes at different levels. A list of attributes rules can be empty.

- type—Sets the attribute type. Currently schematics is the only supported type.
- <if>—The existence of an <if> statement is optional. A statement can contain one or no <if> and exactly one <then> argument. When <if> isn't defined, <then> appears to all items.
  - ○ and|or|not—You can combine conditions with boolean tags.
  - ○ item—A condition that applies to items only.

item arguments	**Description**
type	The type of the item. You can use regexpr if regexpr=true (optional).
name	The name of the item. You can use regexpr if regexpr=true (optional).
regexpr	Use regular expressions in the condition.
equal	The values of arguments (type/ name) must be equal for the condition to be true.
caseins	Case sensitive when the statement is true.

This is an example of an attribute applied to an item:

  - ○ attribute—A condition that is applied to the attributes of items.

attribute arguments	**Description**
name	The name of the attribute. You can use regexpr if regexpr=true (optional).
value	The value of the attribute. You can use regexpr if regexpr=true (optional).

`attribute` arguments	Description
`regexpr`	Use regular expressions in the condition.
`caseins`	Case sensitive when the statement is `true`.

- `then`—A rule must have a `then` argument. Only add the argument when you want to apply the rule. It contains everything that the rule applies to.

`then` arguments	Description
`attribute`	Add any number of effects to attributes (optional).
`lock`	Set the lock attribute to `true` to lock it.
`visible`	Set the attribute to `false` to hide it.
`localize`	Enables localization of attributes.

This is an example of a `then` argument:

```
<attr_rules>
  <rule type="schematics">
    <if>
      <item type="co*" name="*1*" regexpr="true"/>
    </if>
    <then>
      <attribute visible="false"/>
    </then>
  </rule>
  <rule type="schematics">
    <if>
      <attribute name="*name*" regexpr="true"/>
    </if>
    <then>
      <attribute lock="true"/>
    </then>
  </rule>
  <rule type="schematics">
    <if>
      <and>
        <attribute name="name"/>
        <attribute value="*2*" regexpr="true"/>
      </and>
    </if>
    <then>
      <attribute localize="true" lock="false" visible="true"/>
    </then>
  </rule>
</attr_rules>
```

Continue to the next section for information on defining profiles.

Defining Profiles

Profiles (`<profile>`) are optional sets of rules defined within a standard to control the display of items in a figure. You can define multiple profiles within the standard for users to apply to a figure. Profiles control the appearance of data, such as color and line weight, as well as the visibility of user-added content. Rules are applied in the order in which they appear in the profile.

The table below displays items that are definable by profile rules, and characteristics that need to be defined:

Item	Rules included
Components	• Line style • Line weight • Line color • Fill color
Pins	• Line style • Line weight • Line color • Fill color
Wires	• Line style • Line weight • Line pattern or color
Highways	• Line style • Line weight • Line color
Cables	• Line style • Line weight • Line color

You can define these characteristics in a Schematic profile:

- Profile name (mandatory)
- `rules`—Define the set of rules contained in a profile. Schematics rules can contain one or no `<if>` and exactly one `<then>` arguments.

  ```
  <rule type="schematics">
  ```

  The existence of an `<if>` statement is optional. When `<if>` is not defined, then `<then>` applies to all items. `<if>` conditions can be combined with boolean `<and|or|not>` tags. Not using any of these will be treated as `<and>` for all conditions inside `<if>`.

  You can include the following condition types in an argument:

arguments	Description
`item`	Sets the item to which the condition is applied.
`type`	Defines the type of the item. You can use `regexpr` if `regexpr=true` (optional).
`name`	Defines the name of the item. You can use `regexpr` if `regexpr=true` (optional).
`regexpr`	When set to `true`, you can use regular expressions in conditions.
`equal`	When set to `true`, the values of arguments (type/name) must be equal for condition to be true.
`caseins`	When set to `true`, the condition is case sensitive.

This is an example of a condition applied to items:

```
<item type="co*" name="*1*" regexpr="true" equal="true"
caseins="true"/>
```

- `<attribute>`—Sets the condition which is applied to attributes of items

`attribute` **arguments**	Description
`name`	The name of the attribute. You can use `regexpr` if `regexpr=true` (optional).
`value`	The value of the attribute. You can use `regexpr` if `regexpr=true` (optional).
`equal`	When set to `true`, the values of arguments (type/name) must be equal for condition to be true.
`regexpr`	Use regular expressions in the condition.
`caseins`	Case sensitive when the statement is `true`.

This is an example of a condition applied to the attributes of items:

```
<attribute name="name" value="value" regexpr="true"
equal="true" caseins="true"/>
```

- `then`—A rule must have a `then` argument. Only add the argument when you want to apply the rule. It contains everything that the rule applies to.

- `bgcolor`—Sets the background color of the figure. Use it only with an empty condition.
  `<bgcolor></bgcolor>`
- `line`—Sets the line style (optional).

line style arguments	Description
`style`	Selects the style of the line. For instance, multicolored, solid, dashed, and dotted. Default value is `solid`.
`color`	Sets the color of the line.
`weight`	Sets the width of the line.
`spacing`	Sets the space between dots of the dotted style. Default value is `0.5` mm.
`pattern`	Describes the pattern of the dashed line style. Default value is `0.5` mm.
`text_color`	Sets the color of the line text.
`border_color`	Sets the color of the line border. Default value is `black`.
`border_weight`	Sets the width of the line border in percentage where 100% is `1` and 40% is `0.4`. Default value is `0.1`.
`colorseg` Includes: ◆ `color`—Sets the color of the segment. ◆ `length`—Sets the length of the segment.	Applies only when `multicolor` is selected. Each `colorseg` tag adds a color to the style. If length is not defined then last length is used. If there are no lengths, or all lengths are `0`, then the border weight is used as length, in which case all color segments are squares.

Note

Border is applied only if the following line style parameters are defined:

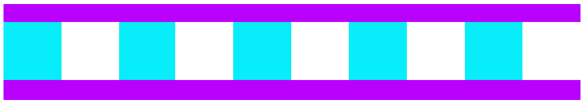
- `border_color` or `border_weight`
- `solid` or `multicolor`
- `color` or `colorseg`

Examples of `line` arguments

Single color solid line:

```
<then>
    <line color="#00ff00" weight="1"/>
</then>
```

Stripe with border:

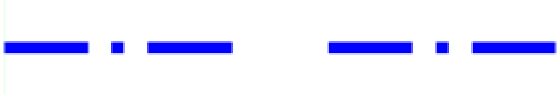


```
<then>
    <line style="multicolor" text_color="#0000ff" border_color=
"B900FC" border_weight="0.4" color="#07EDF9" weight="1">
        <colorseg color="#07EDF9" length="0.6"/>
        <colorseg color="#FFFFFF"/>
    </line>
</then>
```

Dotted:

```
<then>
    <line style="dotted" spacing="2" color="#ff0000"/>
</then>
```

Dashed:

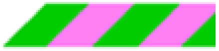


```
<then>
```

Examples of `line` arguments

```
    <line style="dashed" pattern="3.5;1;0.5;1.0;3.5;4;" weight=
"0.6"/>
</then>
```

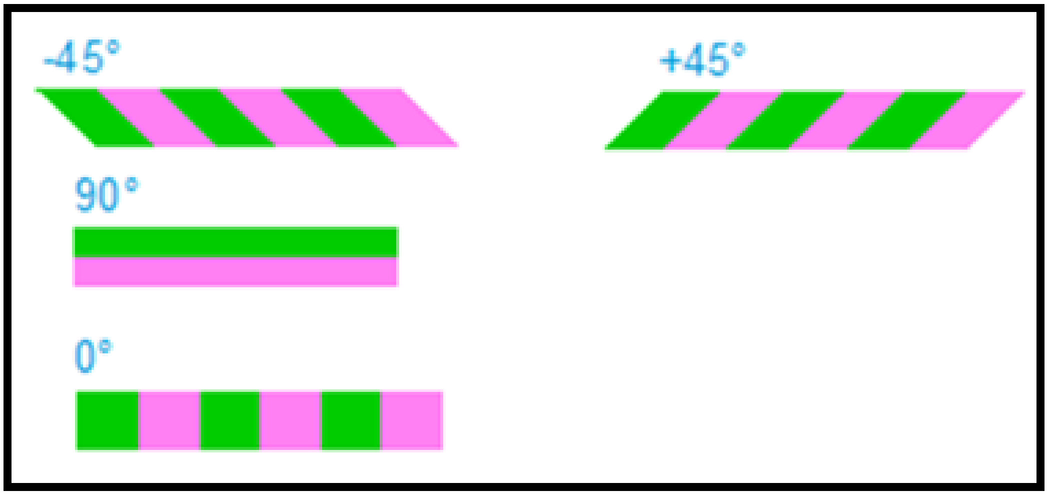
Angled stripe:



```
<then>
    <line style="multicolor" angle="30" color="#888888" weight=
"0.6">
        <colorseg color="#ff0000" length="0.2"/>
        <colorseg color="#00ff00" length="0.1"/>
        <colorseg color="#0000ff" length="0.3"/>
    </line>
</then>
```

Note

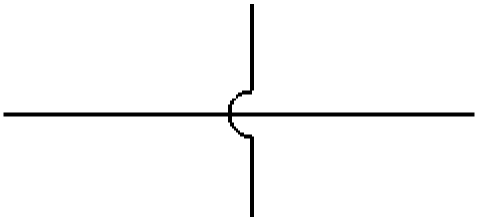
You can set any angle between -45 and +45 degrees, and also 90 degrees.



Examples of Crossing Point Styles

This example shows the arguments of four styles in which lines can cross.

Arc with vertical on top:

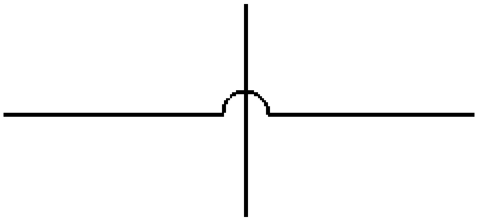


```
   <profile name="Vertical Arc connection crossing style with
```

Examples of Crossing Point Styles

```
horizontal segments on top">
    <rules>
     <rule type="schematics">
      <then>
                 <setting crossing_point_style="arc"/>
      </then>
     </rule>
    </rules>
    <rules>
     <rule type="schematics">
      <then>
                 <setting top_segment_vertical="true"/>
      </then>
     </rule>
    </rules>
   </profile>
```

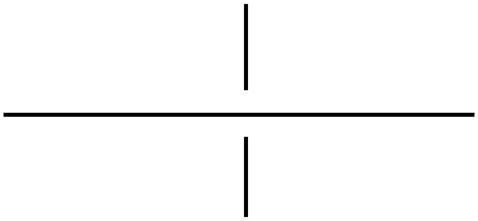
Arc with horizontal on top:



```
   <profile name="Arc connection crossing style with horizontal
segments on top">
    <rules>
     <rule type="schematics">
      <then>
                 <setting crossing_point_style="arc"/>
      </then>
     </rule>
    </rules>
    <rules>
     <rule type="schematics">
      <then>
                 <setting top_segment_vertical="false"/>
      </then>
     </rule>
    </rules>
   </profile>
```

Gap with horizontal on top:

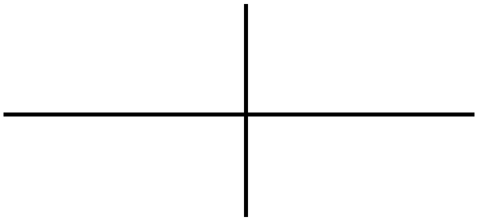
Examples of Crossing Point Styles



```
   <profile name="Gap connection crossing style with horizontal
segments on top">
    <rules>
     <rule type="schematics">
      <then>
                 <setting crossing_point_style="gap"/>
      </then>
     </rule>
    </rules>
    <rules>
     <rule type="schematics">
      <then>
                <setting top_segment_vertical="false"/>
      </then>
     </rule>
    </rules>
   </profile>
```

Examples of Crossing Point Styles

No connection crossing style:



```
   <profile name="None connection crossing style with horizontal
segments on top">
    <rules>
     <rule type="schematics">
      <then>
                 <setting crossing_point_style="none"/>
      </then>
     </rule>
    </rules>
    <rules>
     <rule type="schematics">
      <then>
                <setting top_segment_vertical="false"/>
      </then>
     </rule>
    </rules>
   </profile>
```

You can define these characteristics in a 3D profile:

- Profile name (mandatory), for example,
  `<profile name=`
  `"Yellow color to all figure items">- <start tag>`
- `rules`—Define the set of rules contained in a profile. 3D figure rules can contain one or no `<if>` and exactly one `<then>` arguments.
  `<rule type="3D_Figures">`

  The existence of an `<if>` statement is optional. When `<if>` is not defined, then `<then>` applies to all items. `<if>` conditions can be combined with boolean `<and|or|not>` tags. Not using any of these will be treated as `<and>` for all conditions inside `<if>`.

  You can include the following condition types in an argument:

Arguments	Description
`item`	Sets the item to which the condition is applied.
`type`	Defines the type of the item. You can use `regexpr` if `regexpr=true`

Arguments	Description
	(optional).
`name`	Defines the name of the item. You can use `regexpr` if `regexpr= true` (optional).
`regexpr`	When set to `true`, you can use regular expressions in conditions.
`equal`	When set to `true`, the values of arguments (type/name) must be equal for condition to be true.
`caseins`	When set to `true`, the condition is case sensitive.

This is an example of a condition applied to items:

```
<item type="co*" name="*1*" regexpr="true" equal="true"
caseins="true"/>
```

- `<attribute>`—Sets the condition which is applied to attributes of items

`attribute` **arguments**	**Description**
`name`	The name of the attribute. You can use `regexpr` if `regexpr=true` (optional).
`value`	The value of the attribute. You can use `regexpr` if `regexpr=true` (optional).
`equal`	When set to `true`, the values of arguments (type/name) must be equal for condition to be true.
`regexpr`	Use regular expressions in the condition.
`caseins`	Case sensitive when the statement is `true`.

This is an example of a condition applied to the attributes of items:

```
<attribute name="name" value="value" regexpr="true"
equal="true" caseins="true"/>
```

- `then`—A rule must have a `then` argument. Only add the argument when you want to apply the rule. It contains everything that the rule applies to.
  - `item`—Sets the override to an item.

`item` arguments	Description
`Phantom`	Sets the phantom value of the item.
`transparency`	Sets the transparency value of the item.
`color`	Sets the color value of the item.

- `figure`—Sets an override to a figure argument:

`figure` arguments	Description
`background`	Sets the background color value of the figure.
`background-gradient`	Sets the background gradient color value of the figure.

Example

```
<profile name="Profile A">
  <rules>
    <rule type="3D_Figure">
      <if>
        <and>
           <attribute regexpr="true" name="sBOM Name" value=
"49130031*"/>
           <attribute name="Source_file_name" value="49130031.
prt.2"/>
        </and>
      </if>
      <then>
           <item color="#00ff00" transparency="25.0"/>
      </then>
  </rule>
   <rule type="3D_Figure">
      <if>
        <and>
          <attribute regexpr="true" name="sBOM Name" value=
"49130032*"/>
        </and>
      </if>
      <then>
          <item color="#ff00ff"/>
      </then>
    </rule>
  </rules>
</profile>

<profile name="Profile B">
  <rules>
    <rule type="3D_Figure">
      <if>
```

```
        <and>
          <attribute regexpr="true" name="sBOM Name" value="DIN*"/
>
        </and>
      </if>
    <then>
          <item phantom="true"/>
    </then>
  </rule>
  <rule type="3D_Figure">
    <if>
      <and>
         <attribute regexpr="true" name="sBOM Name" value=
"Brake*"/>
     </and>
    </if>
      <then>
         <item transparency="20.0"/>
      </then>
    </rule>
  </rules>
</profile>

<profile name="Profile C">
  <rules>
    <rule type="3D_Figure">
      <then>
          <figure background="#ff00ff" background-gradient=
"#00ff00"/>
      </then>
    </rule>
  </rules>
</profile>
<profile name="Profile D">
  <rules>
    <rule type="3D_Figure">
      <if>
        <and>
          <attribute name="Feature_Id" value="1337"/>
        </and>
    </if>
        <then>
          <item color="#f0b823"/>
        </then>
    </rule>
  </rules>
</profile>
```

23

Configuring Creo Illustrate for Translation Management

You can use the Creo Illustrate client together with Windchill Service Information Manager to manage your multi-language illustrations. Make sure your system meets these prerequisites:

- Windchill Service Information Manager is installed
- Add the following to `admin_prefs.xml`:

```
<category name="Windchill">
    <preference name="Windchill_localization" value="true"/>
</category>
```

Use this workflow to manage localized content:

1. In Creo Illustrate, create figures and annotate them.
2. Select the content to localize and the languages. For more information, see these topics in the Creo Illustrate Help:
   - About Localizing Text
   - To Set Localization Options for the Current Illustration

   The content you select is exported to a separate XLIFF (`*.xlf`) file when the illustration (C3DI file) is checked in.
3. Click **Windchill** ▸ **Auto-Check In**. The illustration, including the XLIFF file, is checked in to Windchill.
4. In Windchill, check out and check in XLIFF files.
5. In Creo Illustrate open the illustration file and accept updates, if necessary. The updates to the XLIFF file are applied automatically. For more information, in the Creo Illustrate Help, see the topic About Managing Localized Text in Windchill.

24

Formatting PDF Templates

This chapter provides the XML structure, elements, and attributes of the layout template that you can use when publishing a 3D figure to a 3D or image PDF in Creo Illustrate. You can use this information to customize the layout template.

Alternatively, use the `Sample_PDF_Template.zip` that is provided with Creo Illustrate. The sample PDF template can be found in the following directory: `%AppData%\ptc\illustrate`.

PDF templates contain the following attributes:

- Page attributes
- Header attributes
- Content attributes
- Footer attributes

Example

The attributes in this example are described below.

```
<page width="8.5" height="11.0" length_unit="in" margin="0.3"
bottom_margin="0.5"
   top_margin="0.5" font="Courier" font_size="7">
  <header width="7" height="1.0" font="Helvetica" font_size="7">
    <row>
      <cell width="70" align="center">
        <text>Figure Name:</text>
        <dynamic_text>FIGURE_NAME</dynamic_text>
      </cell>
      <cell width="30">
        <image path="ptc_logo.jpg" URL="www.ptc.com" />
      </cell>
    </row>
  </header>
  <content split="60" graphic="true" margin="0.3"
 table_padding ="0.01" table_left_padding="0.02" text_wrap=
"false">
  </content>
  <footer width="7.0" height="0.5">
      <text>This is static text </text>
      <text>and this is dynamic text: </text>
      <dynamic_text>ILLUSTRATION_NAME</dynamic_text>
  </footer>
</page>
```

Page Attributes

```
<page width="8.5" height="11.0" length_unit="in" margin="0.3"
bottom_margin="0.5"
 top_margin="0.5" font="Courier" font_size="7">
```

Attribute Name	Default Setting and Requirement	Description
width	Mandatory	Specify the width of the page.
height	Mandatory	Specify the height of the page.
length_unit	Mandatory	Specify what length unit is used. All 15 units of Creo Illustrate are supported.
margin	Optional 0*	Specify the margin on all 4 sides.
left_margin right_margin bottom_margin top_margin	Optional 0* for each margin	Specify the specific margin. margin will be overwritten where applied.
font	Optional Helvetica*	Enter the name of the font to use in the page.
font_size	Optional 8*	Specify the size of the font in points.

Header Attributes

```
<header width="7" height="1.0" font="Helvetica" font_size="7">
  <row>
    <cell width="70" align="center">
      <text>Figure Name:</text>
      <dynamic_text>FIGURE_NAME</dynamic_text>
    </cell>
    <cell width="30">
      <image path="ptc_logo.jpg" URL="www.ptc.com" />
    </cell>
  </row>
</header>
```

Attribute Name	Default Setting and Requirement	Description
width	Mandatory	Specify the width of the header cell in percentage.
height	Mandatory	Specify the height of the header cell in percentage.
align	Mandatory	Specify the alignment of the text.
font	Optional	Enter the name of the font that is used in the header. Overrides the font of the page when applied.
font_size	Optional	Set the size of the font in points.
row	Optional	Adds a row to the header. Only one row is presently supported.
text	Mandatory	Entter static text that is displayed in the header
dynamic_text	Optional	Enter text that is replaced according to the content. This can be ILLUSTRATION_NAME, FIGURE_NAME, or AUTHOR.
cell width	Optional	Specify the width of the cell in percentage
image path	Mandatory	Enter a path to a raster image for logo to be insert in the header.
URL	Optional	Adds an URL to the image. Click on the image to open the URL in the user's web browser.

Note

The dynamic text tag can contain one of the following options:

- ILLUSTRATION_NAME—The name of the illustration
- FIGURE_NAME—The name of the figure
- AUTHOR_NAME—The name of author

Content Attributes

```
<content split="60" graphic="true" itemslist="true" sort_by="ITM"
ascending="false"
margin="0.3" table_padding ="0.01" table_left_padding="0.02" text_
wrap="false">
</content>
```

Attribute Name	Default Setting and Requirement	Description
split	Optional 100*	Specify how the content area is distributed between the graphic window and the items list table.
graphic	Optional true*	A flag that determines if the graphical window should be shown.
margin	Optional 0*	Specify the margin of the content for all 4 sides.
left_margin right_margin bottom_margin top_margin	Optional 0* for each margin	Specify the specific margin attribute. margin will be overwritten where applied.
table_padding	Optional 0*	Specify the table padding of the items list table for all 4 sides. Padding is the space between the text and the border of the cell.
table_left_padding	Optional 0* for each padding	Specify the specific table padding.

Attribute Name	Default Setting and Requirement	Description
table_right_padding table_bottom_padding table_top_padding		table_padding will be overwritten where applied.
text_wrap	Optional false*	A flag for text wrap.
itemslist 3D figures only	Optional false*	A flag that determines if the items list table should be shown
sort_by 3D figures only	Optional sort_order*	Specifies the columns presented in the item list table.
ascending 3D figures only	Optional true*	Determines whether the table is sorted in ascending or descending order.

Footer Attributes

Dimensions of footer and text that is displayed in the footer

```
<footer width="7.0" height="0.5">
  <text>This is static text </text>
  <text>and this is dynamic text: </text>
  <dynamic_text>ILLUSTRATION_NAME</dynamic_text>
</footer>
```

Text Attributes

Attribute Name	Default Setting and Requirement	Description
width	Mandatory	Specify the width of the footer
height	Mandatory	Specify the height of the footer
text_align	Optional left*	Set the alignment of the text within the footer

Attribute Name	Default Setting and Requirement	Description
`text`	Optional	Enter static text
`dynamic_text`	Optional	Enter text that is replaced according to the content (figure name, illustration name, author name, etc.)