Agile Product Lifecycle Management

EC MCAD Connectors for Agile Engineering Collaboration

Installation and Administration Guide

v2.5.1

E16681-02
March 2010

Oracle Copyright

CONTENTS

Preface

The Agile PLM documentation set includes Adobe® Acrobat PDF files. The Oracle Technology Network (OTN) Web site http://www.oracle.com/technology/documentation/agile.html contains the latest versions of the Agile PLM PDF files. You can view or download these manuals from the Web site, or you can ask your Agile administrator if there is an Agile PLM Documentation folder available on your network from which you can access the Agile PLM documentation (PDF) files.

Note	To read the PDF files, you must use the free Adobe Acrobat Reader version 7.0 or later. This program can be downloaded from the Adobe Web site http://www.adobe.com.

The Oracle Technology Network (OTN) Web site http://www.oracle.com/technology/documentation/agile.html can be accessed through **Help** > **Manuals** in both Agile Web Client and Agile JavaClient. If you need additional assistance or information, please contact My Oracle Support (https://support.oracle.com) for assistance.

Note	Before calling Oracle Support about a problem with an Agile PLM manual, please have the full part number, which is located on the title page.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, 7 days a week. For TTY support, call 800.446.2398. Outside the United States, call +1.407.458.2479.

Readme

Any last-minute information about Agile PLM can be found in the Readme file on the Oracle Technology Network (OTN) Web site http://www.oracle.com/technology/documentation/agile.html

Agile Training Aids

Go to the Oracle University Web page http://www.oracle.com/education/chooser/selectcountry_new.html for more information on Agile Training offerings.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Chapter 1

Introduction to Engineering Collaboration

This chapter includes the following:

Overview

Agile Engineering Collaboration (EC) is an application that provides data and process integration between CAD applications and Agile PLM. The application consists of a core Engineering Collaboration Client (known as the "EC Client"), and individual MCAD Connectors to specific CAD systems (such as Pro/ENGINEER, SolidWorks, etc.). It allows CAD designers and engineers to capture and control the data representing a primary source of the product record. The EC Client provides a window into the Agile environment that is geared towards CAD designers and engineers. It supports searching and viewing of Agile data, and provides the user interface for all Connector operations such as saving and loading CAD data.

Each captured CAD file is stored in a PLM object. This object can be a Document, or as a new option available starting Agile PLM 9.2.2.4 release, a Design object. If using with Agile 9.2.2.4 or later, a configuration option allows setting the EC Client to work with either object class.

You can view CAD datasets created by the EC Connectors using AutoVue for Agile, allowing anyone with access to Agile PLM to be able to view, markup, and collaborate in real time on the 3D CAD designs across the web, without using the CAD tool.

Agile Engineering Collaboration requires Agile Product Collaboration as a prerequisite, and data created by Engineering Collaboration can be used in all other PLM solutions including Product Portfolio Management, Product Cost Management, Product Governance & Compliant, and Product Quality Management.

Key Features

The main features of the EC Connectors are:

- **Save** - Saves native CAD data from the current session into Agile
- **Load** - Loads native CAD data from Agile into the current CAD session
- **Manage Change** - Allows users to control checkout reservation and revisioning
- **Update Properties** - Updates property (attribute) values between the CAD files and Agile
- **Create Viewables** - Creates neutral format files, such as PDF, to be used for viewing, plotting, or manufacturing

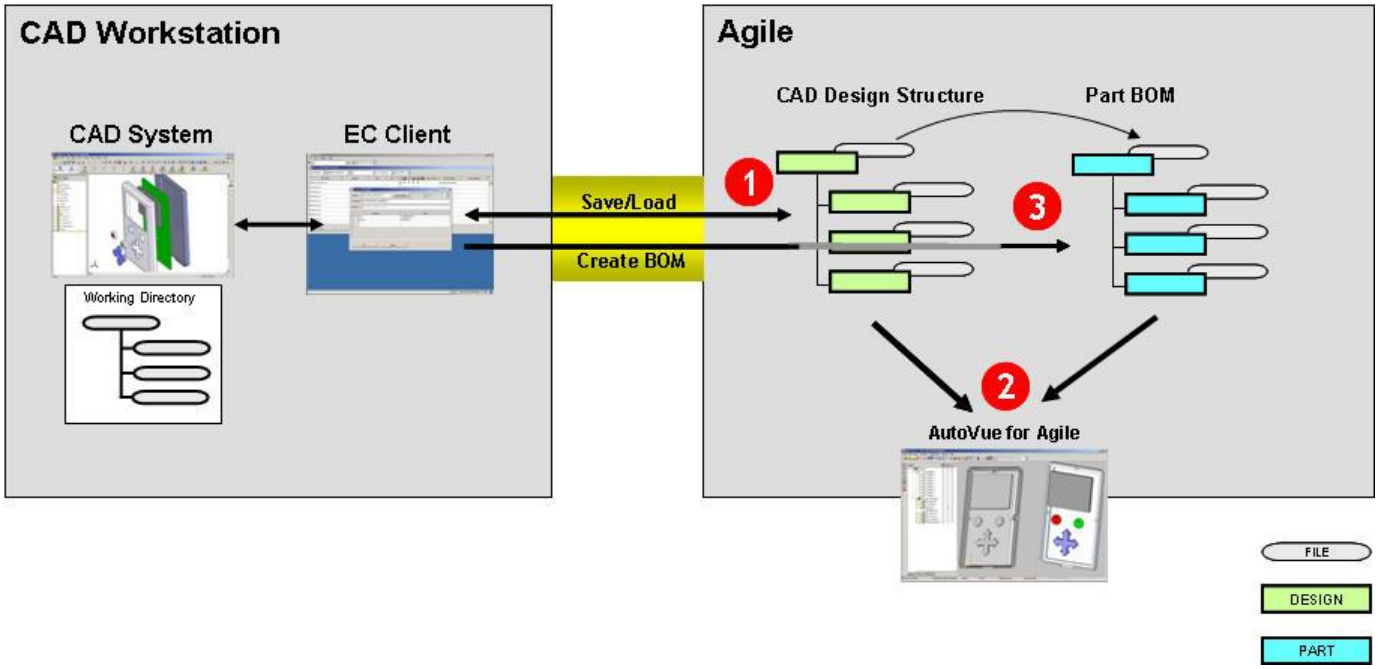
- **BOM Publication** - Automatically creates and updates the Part BOM based on the CAD design structure, with Agile change control.

Some of the characteristics of Agile Engineering Collaboration are:

- Multi platform, Java Client.
- Centralized configuration via XML.
- Dynamically loaded sessions support customization of the CAD connector's business logic.
- Simplified Common Search and View functionality for CAD designers and engineers.
- Manages both work in progress (WIP) and released CAD design data.
- Supports concurrent engineering, enabling multiple designers to work with a common assembly.
- Data access controlled by checkout reservation and/or timestamp.
- Automated BOM creation process.
- Bi-directional metadata exchange.
- Support for special CAD capabilities such as Pro/ENGINEER family tables, CATIA CGR fields and SolidWorks configurations.

Engineering Collaboration Process

The diagram below illustrates the main use cases supported by Agile Engineering Collaboration.



Save and Load CAD Files

CAD designs are created within the CAD system environment, with files in a working directory (which may be local or network attached). The designer saves into Agile, which creates a Document or Design structure that mimics the structure of the CAD assembly. You attach the native CAD files to this structure, and use them as the basis for loading and re-saving the CAD designs. Since Agile is a centralized repository, all CAD designers in the enterprise have access to these files, subject to the control of Agile roles and privileges. Individual designers can set checkout reservation in Agile when they load files into their CAD session, enabling concurrent engineering within CAD assemblies. You can attach additional viewable files (PDF, HPGL, etc.) to the Document or Design structure.

View CAD Files

One reason for managing the native CAD files within Agile are that the Agile viewer can be used to view and markup the files. This works across the web, and without having the native CAD system. Advanced functionality such as digital mockup, 3D comparison, interference checking, and real-time collaboration make this an important tool to support the overall design process.

Create BOM

When a design or design change is completed, the designer may use the Create Item/BOM command to create or update the Agile Part BOM, representing the true product structure. You use this function when there is a high correlation between the document/design structure and product structure, to avoid tedious manual entry of the Part BOM. This works in the context of an Agile change object, resulting in automated BOM redlining. You can also manually update the BOM, to add bulk items such as paint or glue. Further automated updates from CAD will not remove the manually added items.

Chapter 2

Installation and Configuration

This chapter includes the following:

Prerequisites

Prior to the installation of the Engineering Collaboration interface on a local system, you must verify the following items:

- Database is operational and running
- Install Agile PLM (see Introduction for supported versions) successfully on an accessible server (the prerequisites for Java Runtime Environment are the same as for Agile PLM server).

Important If you are not working with a member of Agile's Solutions Delivery Organization, you are strongly encouraged to refer Oracle | Agile Product Lifecycle Management Documentation for installation procedures.

- Agile File Management Server is usable and accessible.
- A test environment is prepared
- Install a CAD system that the test user can launch from the home directory.
- Login name and password of the Agile PLM test user are known in Agile PLM.
- The test user can launch an Agile PLM client session.

Java Version Support

The EC Client runs on Java 1.6 (for Agile 9.2.2.4 and later and for Agile 9.3 and later).

Obtaining Software from Oracle E-Delivery

Oracle products are distributed as Media Packs on Obtaining Software from Oracle E-delivery (http://edelivery.oracle.com). A Media Pack is an electronic version of the software. Refer to the Media Pack description or the list of products that you purchased on your Oracle Ordering Document. Then, view the Quick Install Guide License List to help you decide which Product Pack you need to select in order to search for the appropriate Media Pack(s) to download. Prior to downloading, verify that the product you are looking for is in the License and Options section of the E-Pack README. Oracle recommends that you print the README for reference.

There will be an itemized part list within each of the packs and you will need to download all items in order to have the complete download for the desired Oracle Agile release.

All Oracle E-Delivery files have been archived using Info-ZIP's highly portable Zip utility. After downloading one or more of the archives, you will need the UnZip utility or the WinZip utility to extract the files. You must unzip the archive on the platform for which it was intended. Verify that the file size of your downloaded file matches the file size displayed on E-Delivery. Unzip each Zip file to its own temporary directory.

Implementation Checklist

The following process is recommended for successful implementation of the Agile EC MCAD connectors. This process should coincide with a customer workshop to determine how to set the various configuration options.

- Download latest MCAD connector software version from Oracle E-Delivery for your specific CAD tools.
- Download latest MCAD EC Client software version from Oracle E-Delivery.
- Review the Quick Install Guide and Readme documents that come with the software. The Quick Install Guide includes a platform support matrix showing the supported CAD tool versions for each connector
- Check for connector patch updates on My Oracle Support (Metalink); download any that apply and read the associated Readme file
- Read the EC MCAD Connector Installation and Administration Guide Introduction, Installation Requirements, and Getting Started sections to familiarize yourself with the basics of CAD connector operation and configuration.
- Install the connector, following the instructions in the Installating and Configuring section for that particular connector. This includes basic configuration instructions to get the connector working at a minimal level.
- Refer to the EC Client Configuration Options section to configure objects and attributes using the Agile Admin client.
- Test saving a single CAD part with the CAD Connector using the Save command.
- Review the object in Agile and the CaxClient.log file
- Correct any attribute or permission errors and re-test

- Follow the Administrator Guide Installation and Configuration section for the particular connector to perform further configuration to match customer requirements.
- Test creating a Part BOM using the Create Item/BOM command from the Connector
- Review the Agile Part Object and the CaxClient.log
- Correct any attribute or permission errors and re-test
- Conduct further tests of Save and Load using CAD assemblies and drawings, to make sure data is being saved properly

The problems most often seen during EC implementation are lack of testing, not reviewing the log files and not reading the Administrator Guide instructions.

Chapter 3

Getting Started with Engineering Collaboration

This chapter includes the following:

Attribute Exchange

Agile EC includes capability for bi-directional attribute exchange between CAD and PLM. In CAD these attributes are typically called "properties" or "parameters", and represent important textual information associated with the design file. The following capabilities are supported by EC:

- Setting PLM attributes for Design objects during the Save process
  - Based on existing CAD properties
  - Based on direct user input, which can also be captured in the CAD file properties
- Setting PLM attributes for Part objects during the Create Item/BOM process
  - Based on existing CAD properties
  - Based on direct user input, which can also be captured in the CAD file properties
- Updating CAD properties based on PLM attributes, on demand by the user
- Updating CAD properties based on PLM attributes, automatically during the Save process (some CAD connectors only)
- Updating CAD properties based on PLM attributes, automatically during the Load process (some CAD connectors only)

- Updating text within drawing title blocks, based on PLM attributes.

Setting Up EC Attributes and Attribute Mapping

Before setting up any attribute mapping you must first configure attributes in the Agile Admin client. There are two sets of attributes to be concerned with:

1. Attributes required for the proper operation of EC CAD connectors
2. Additional customer-specific attributes to be mapped between CAD and PLM using EC

Required Attribute Configuration

The effort involved for setting up the required (type 1) attributes depends on the data model you are using. The required attribute values are shown in Appendix B.

- Using DocuBOM data model
  - All EC Document attributes must be manually configured in the Admin client
  - All EC Part attributes must be manually configured in the Admin client
- Using Design data model
  - All EC Design attribute are pre-configured on the Design class (see note below)
  - All EC Part attributes must be manually configured in the Admin client

Note	In Agile PLM 9.2.2.4 (and later 9.2.2.x versions), the Model Name attribute on the Design Structure tab is improperly set as a Text attribute. See Appendix B for details on how to fix this.

To insure that all required attributes can be access by EC correctly, it is necessary to check all privileges associated with the specific object classes to make sure that all the required attributes are selected in the "Applied To" list of each privilege.

Attribute Mapping Configuration Files

Attribute mapping involves both the XML (CaxClient_Designs.xml or CaxClient_Documents.xml) configuration file and the specific connector's INI configuration file, as follows:

XML file:

- Defines a symbolic attribute name which maps to an attribute with a specific PLM base ID
- Determines whether the attribute will be visible for interactive entry within the EC Client

INI file:

- Defines the actual mapping between the specific CAD property and specific PLM attribute, and under what conditions the mapping occurs
- Defines additional string manipulation to modify the attribute value during the mapping (CAD connector dependent)

XML File Attribute Definition

The definition of symbolic attribute names used by EC is performed in the <objectProperties> section of the XML file (basically the bottom half of the file). There are three object classes defined in this section as follows:

```
<subclass name="FILEFOLDER" type="55" id="subclass id">  -- Design
subclass
<subclass name="DOCUMENT" type="2" id="subclass id"> -- Document
subclass
<subclass name="ITEM" type="2" id="subclass id"> -- Part subclass
```

Either the Design or Document subclass is used in any particular installation, depending on which data model is in use. The unused section can simply be ignored. The Part subclass is always used.

The way these sections are configured is that for "subclass id" you substitute the desired subclass ID you want to use. For example, for the standard Designs subclass you use 2000008310, and for the standard Parts subclass you use 9141. Whichever subclass is identified here will be the default in all EC Client dialogs.

Note	Note: Do not change the subclass name identifiers in these sections.
Note	Note: Object subclass IDs are not accessible from the Admin client. The CaxClient.log file lists all pertinent class IDs for easy access.

Within the three subclass sections you will see various "table name" sections which define the tabs for each object class, and within those sections are "attribute name" definitions. For example:

```
<attribute name="CAX_CRE_SYSTEM" id="2007" set="0" get="1"/>
```

The name "CAX_CRE_SYSTEM" is called a symbolic name, and is used by the CAD connector INI file to define attribute mappings (see next section). This is NOT the name the user sees in the PLM user interface. For example, in this case the UI name is "Design System", which is base ID 2007.

For further details see the EC Client Configuration Options chapter in this document.

INI File Attribute Definition

Each CAD connector has an INI configuration file that is used partially for defining attribute mapping:

- SolidWorks, Solid Edge – 3DCADMapping.ini
- Pro/ENGINEER – AcpCustomer9.ini
- CATIA – AccCustomer9.ini
- Unigraphics NX – Ecu.ini

These files include sections for defining the following types of mappings:

- Mapping from CAD to PLM
  - Attributes to be set for Designs or Documents in PLM, during the Save process
    - This is set separately for initial Saves and update Saves

- Attributes to be set for Parts in PLM, during the Create Item/BOM process

- Mapping from PLM to CAD
  - Attributes to be updated from PLM Designs, Documents or Parts to CAD file properties
    - Based on using the Update Properties command, or during the Save or Load commands (not supported in all CAD systems)
  - Attributes to be updated from PLM Designs, Documents or Parts to CAD drawing title blocks

Here are examples of each of the main types of mapping, using SolidWorks:

Mapping from CAD to PLM, to a Design object during the Save process

```
[Agile9CreateDocument]
CAX_CRE_SYSTEM    = 3DCADTable.ModelVersion
```

This says that the symbolic name CAX_CRE_SYSTEM, representing a specific PLM base ID per the XML file mapping, is going to be set equal to the value on the right side (which happens to be a system variable representing the version of SolidWorks)

Mapping from CAD to PLM, to a Part object during the Create Item/BOM process

```
[Agile9UpdateItem]
DESCRIPTION       = 3DCADTable.Property.Description
```

This says that the symbolic name DESCRIPTION, representing a specific PLM base ID per the XML file mapping, is going to be set equal to the value on the right side (which is the CAD property "Description")

Mapping from PLM to CAD, from a Design object using the Update Properties command

```
[Agile9UpdateProperties]
Part_Number       = CAX_PART
```

This says that the CAD property Part_Number will be set to the value of the symbolic name CAX_PART, representing a specific PLM base ID per the XML file mapping.

For further details pertaining to each CAD connector, see the appropriate Connector Administration section in this document.

Methods for Mapping from CAD to PLM

As previously mentioned there are two use cases for mapping attributes from CAD to PLM:

1. Designs, during initial creation or update using the Save command
2. Parts, during initial creation or update using the Create Item/BOM command

The chart below summarizes the capabilities of all connectors and in which section of the INI file the settings are made:

CAD Tool	Designs - Initial	Designs - Update	Parts
SolidWorks	[Agile9CreateDocument]	[Agile9UpdateDocument]	[Agile9UpdateItem] – For regular parts

			[Agile9UpdateItemConfigured] – For configured parts
Solid Edge	[Agile9CreateDocument]	[Agile9UpdateDocument]	[Agile9UpdateItem]
Pro/ENGINEER	[ProEToAgile.Create_DOCUMENT]	[ProEToAgile.Update_DOCUMENT]	[ProEToAgile.Update_ITEM]
UG NX	[SaveProperties], using Create.Doc. prefix	[SaveProperties], using Update.Doc. prefix	[SaveProperties], using Create.Item. or Update.Item. prefix
CATIA V5	[CatiaToAgile.DOCUMENT], for both initial and update		[CatiaToAgile.ITEM]

Adding Customer-specific Attributes

To add customer-specific attributes and map them with the CAD connectors, follow this process:

1. Configure the attribute in the Agile Admin client per the standard process. Turn on an available attribute of the desired type, and set the desired name.
2. Add the attribute to the appropriate section in the XML configuration file. The symbolic name can be anything you want, and it does not have to start with "CAX".
3. Set up the desired mappings in the INI file, using the symbolic name you defined in step 2.

For example, say that you want to add a "Material" attribute of type List to the Design object, and allow it to be set with a Material property from the CAD file. This is what you need to do:

- In the Agile Admin client, go to the Design class and to the attribute definition list for Page Two.
- Double-click on an unused List-type attribute, for example "List11".
- Set Visible to Yes, and enter the Name "Material"
- Because this is a List-type attribute, you need to define the valid list values
- In the XML file, create an entry in the FILEFOLDER section which defines the new attribute, such as this:
- <attribute name="MATERIAL" id="1271" set="1" get="1"/>
- If you want the Material attribute in PLM to be set by the value from the CAD property, then in the INI file you would make an entry like this (this example uses SolidWorks):

```
[Agile9CreateDocument]
MATERIAL       = 3DCADTable.Property.Material
```

Changing an Attribute to a Different Base ID

The standard EC attributes are pre-defined in the XML file to use certain base IDs. Sometimes you may have a conflict where one of these base IDs is already being used by the customer for some other attribute. This happens most commonly with the Part object. It is not necessary to change

the existing customer attribute, you can simply change the base ID used by EC for the standard attributes.

As an example, let's say that base ID 1313 on Page Two of the Part subclass is already being used by the customer. This base ID is defined by default in EC for the CAX_PUBLISHED attribute (which is "Published From" in the UI and contains the value of the Design version used to publish the Part). Here is the standard definition in the XML file:

```
<attribute name="CAX_PUBLISHED" id="1313" set="0" get="1"/>
```

In order to change this to a different base ID, follow this process:

1. In the Agile Admin client, go to the Part class and to the attribute definition list for Page Two.
2. Verify that the attribute in question, base ID 1313, is a Text type attribute
3. Double-click on an unused attribute of the same type (in this case Text), for example "Text05".
4. Set Visible to Yes, and enter the Name "Published From" (or whatever the attribute name is supposed to be)
5. In the XML file, change the attribute definition entry by putting in the new base ID:

   ```
   <attribute name="CAX_PUBLISHED" id="2011" set="0" get="1"/>
   ```

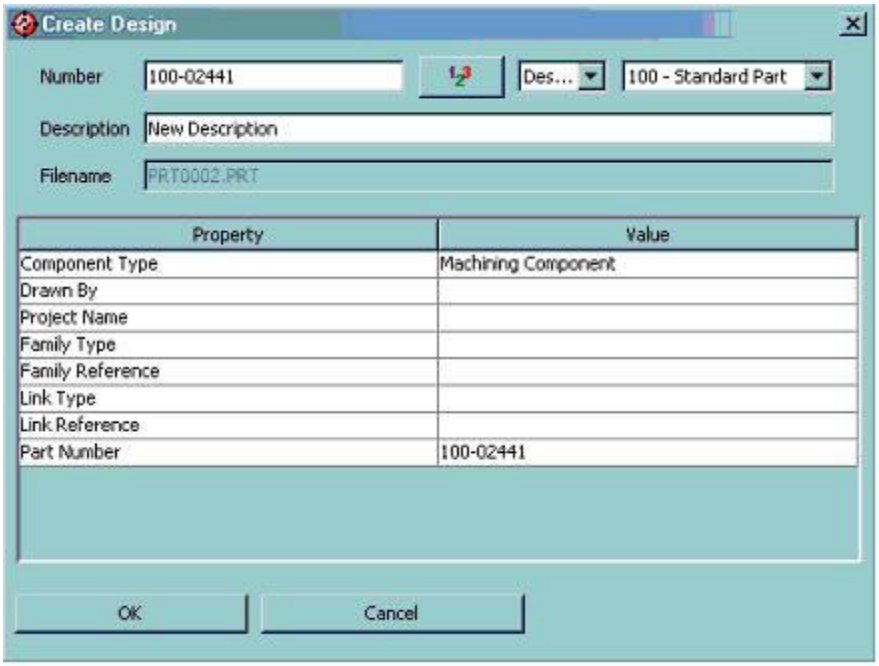
Controlling Attribute Visibility in EC Client Dialogs

Within the EC Client there are two places where attributes can be displayed for manual entry by the user. Within the Save command there is an Interactive Save dialog for Designs (or Documents), and within the Create Item/BOM command there is a similar interactive dialog for Parts. These interactive dialogs are typically used only when first creating the object in PLM, but can also be displayed during updates.

Visibility of attributes in these interactive dialogs is controlled by the XML file. Within each attribute definition line there is an option called "set". If this is set to 1, the attribute is displayed, and if it is set to 0 it is not displayed. Note that in either case, the attribute will be mapped per the settings in the INI file. This "set" option just gives an additional capability to allow the user to interactively view and modify the value prior to saving to PLM. This is a powerful capability, since these dialogs have full capability for List and Multi-List type attributes, allowing the user to pick from the list of values defined in PLM.

Here is an example showing the attribute "Component Type" as defined in the XML file, with set="1", and what the resulting interactive Save dialog looks like:

```
<attribute name="COMPONENTTYPE" id="2000008317" set="1" get="1"/>
```

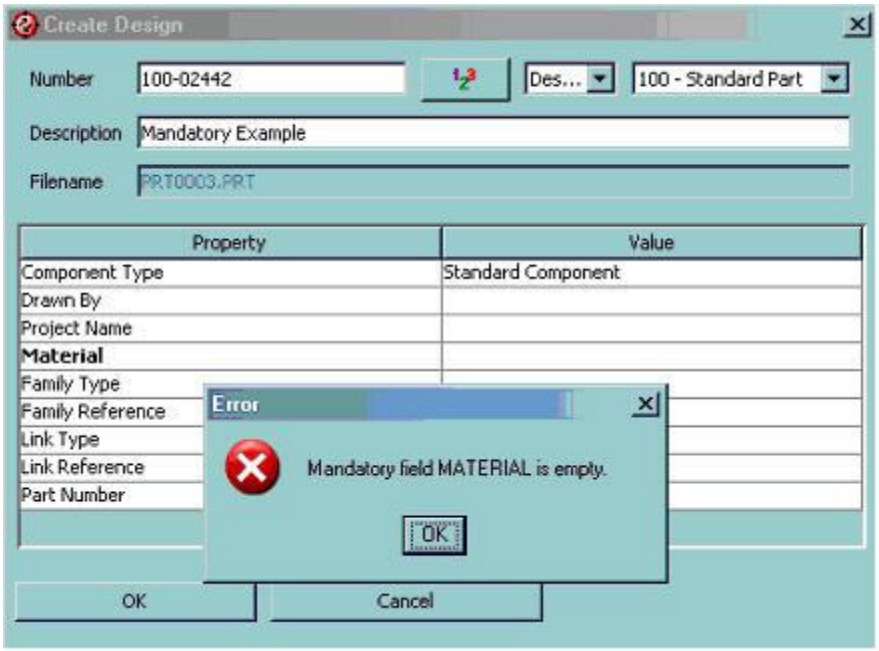


You can also enforce attributes to be entered by using a special "mandatory" flag on the attribute definition.

```
<attribute name="MATERIAL" id="2023" set="1" get="1" mandatory="1"/>
```

When the attribute is set as mandatory it shows up as bold in the dialog, and if the user does not enter a value an error message will appear when clicking on OK, as shown below.

Note The setting of "mandatory" is independent of the Agile server definition of "required" fields. For best results, mark those fields that are required by the Agile server, also as mandatory for the EC Client. Both mandatory and required fields are indicated by bold text in the EC Client user interface.



Methods for Updating Properties from PLM to CAD

As previously mentioned there are three methods for updating properties from PLM to CAD

1. On demand, using the Update Properties command
2. Automatically during the Save process
3. Automatically during the Load process

The first thing to understand is that not all CAD connectors support all three methods. The chart below summarizes the capabilities of all connectors and in which section of the INI file the settings are made:

CAD Tool	On Demand	During Save	During Load
SolidWorks	[Agile9UpdateProperties]	[Agile9SaveUpdateProperties]	[Agile9LoadUpdateProperties]
Solid Edge	[Agile9UpdateProperties]	[Agile9SaveUpdateProperties]	[Agile9LoadUpdateProperties]

Pro/E	[AgileGetProperties.PRT] [AgileGetProperties.DRW] [AgileGetProperties.ASM]	[AgileToProE.ProE] [AgileToProE.PRT] [AgileToProE.DRW] [AgileToProE.ASM]	Not directly available, however this can be accomplished by creating a mapkey in Pro/E that performs a Load and then an Update Properties
UG NX	[LoadProperties] – This defines properties updated both On Demand and during the Save process		[LoadProperties] – The same section applies to the Load operation, but only if LoadAttributes = 1
CATIA V5	[AgileGetProperties.Catia] [AgileGetProperties.CATPart] [AgileGetProperties.CATDrawing] [AgileGetProperties.CATProduct]	[AgileTo.Catia] [AgileTo.CADPart] [AgileTo.CATDrawing] [AgileTo.CATProduct]	

Updating During Save

The usefulness of updating CAD properties during the Save process is to capture attribute values entered interactively by the user and put them into the CAD file at the same time as the value is saved into PLM. The process works like this – note the highlighted step:

- User executes Save command
- During the Save command, attribute are entered interactively
- PLM objects are created (or updated) and entered attributes are saved into the objects
- Attribute values are updated into the CAD file based on the defined mapping
- The CAD file is saved into PLM

So this insures that the value in the CAD file matches the value set in PLM. Note that it is not required to keep the attribute in both the CAD file and in PLM, but many times it is advantageous to do so, so that users disconnected from PLM can see the values by looking in the CAD file.

Note With the SolidWorks Connector, the ability to update properties during Save is disabled by default, in order to improve performance. To enable this capability, please see the instructions in Chapter 5, in the section entitled “Master Switch for Update Properties on Save”.

Updating During Load

The usefulness of updating CAD properties during the Load process is to make sure that any values updated directly in PLM are updated to the CAD file. However, the problem with doing this is that by always updating every CAD file as it is loaded from PLM and CAD, it marks the CAD file as "dirty" and then upon subsequent Saves all files will appear to be modified. So it is more commonly to use the Update Properties "on demand" to update the needed files, rather than do it for all files upon Load.

Synchronizing an Attribute on a Drawing

One important use of attributes within CAD is to control textual content within drawing title blocks. By synchronizing these attributes with PLM, important information such as Part Number, Description, and Revision can be kept in synch. The following chart summarizes the steps necessary to get CAD properties to appear within drawing title blocks. For further specifics on how to create drawing text and properties please see the appropriate documentation for your CAD system.

CAD System	Steps to get CAD property to appear in drawing title block
SolidWorks	▫ Create a drawing Text property linked to a part property ▫ Declare a mapping of the property in 3DCADmapping.ini, in the sections described above
Pro/E	▫ Create a Note in Pro/E with **&Parametername** ▫ Declare a mapping of the parameter in AcpCustomer9.ini, in the sections described above
UG NX	**Option 1:** ▫ Create a named Note in NX, set the Note name to a unique value ▫ Declare a mapping for the text note within Ecu.ini in the **[FillFrame]** and/or **[FillFrameHistory]** sections, with syntax **Notename = Agile value** **Option 2:** ▫ Create a Note in NX, link the Note content to a NX Part Attribute ▫ Declare a mapping of the part attribute within Ecu.ini, in the sections described above
CATIA V5	▫ Create a text object in frame which corresponds with a CATIA property ▫ Declare a CATIA property mapping in AccCustomer9.ini, in the sections described above

Additional Text Processing

Some of the CAD connectors have capability for performance additional text processing on attribute values, for example to append prefixes or suffixes, or remove portions of the text string. See the administration section for details on what is supported for each CAD connector.

Options for Design Numbering

One of the most important decisions when implementing EC is determining the numbering scheme to be used by the Design (or Document) objects, and how this relates to the CAD filenames and Agile Part Numbers. This is a decision made during the implementation workshop, and so it is important to know what the available options are. There are 4 primary methods for numbering

using EC:

1. Using the existing filename
2. Using an autonumber from PLM
3. Using a CAD property
4. Manual entry

With any of these methods, you also have the option to automatically append the CAD file extension (e.g. ".ASM") on the end of the Design number.

The configuration option that controls the Design number is a special symbolic attribute called CAX_NEW_NUMBER. The attribute does not directly appear in the XML configuration file, it is simply used within the INI file to specify the Design number. As described in the section "Methods for Mapping from CAD to PLM", the INI file for each connector has a section where attribute are mapped. The CAX_NEW_NUMBER attribute is used within this section to define the Design number mapping. Here is an example using SolidWorks:

```
[Agile9CreateDocument]
CAX_NEW_NUMBER                = 3DCADTable.ModelTitle
```

This happens to be the way to set the Design number equal to the current filename, since "ModelTitle" is a special property in SolidWorks that contains the filename.

Basic Numbering Methods

The following table describes the process to set each of the four options for Design numbering.

Method	CAX_NEW_NUMBER setting	Initial value of Design Number field in EC Client	User action in Save dialog
1. Existing filename	Set equal to a special CAD property which equates to the filename (see admin section for each connector for details)	The filename value will be displayed	None
2. Autonumber	Leave out of INI file (or comment out)	Blank	Select desired autonumber sequence and click autonumber button to get the next value
3. CAD property	Set equal to the desired CAD property	The CAD property value will be displayed, if it exists and has an assigned value. Otherwise it will be blank	None
4. Manual entry	Any of the above	Any of the above	Type in desired Design Number

Here are some important conclusions from this chart:

- Options 1 and 3 are similar, in that you are assigning CAX_NEW_NUMBER to be equal to

some value, either the filename or a CAD property. In these cases the value will appear in the dialog.

- For option 2, you simply leave off any definition of CAX_NEW_NUMBER, and as a result the initial dialog value will be blank. The user must select the autonumber within the dialog. Note: Even if the user does not explicitly click the autonumber button to display the next number, this will be automatically done when exiting the dialog.
- Manual entry can be used in combination with any of the other methods, to override the initial default value.

Appending the CAD Filetype

It is often desired to append the CAD filetype to the Design number, for the following reasons:

- It makes the Design number look "CAD-like", and provides a way to determine the filetype at a glance
- It provides a way to separate the 3D model file from the 2D drawing file, if they have the same base number. For example, you can have D00111.PRT and D00111.DRW.

The option to automatically append the CAD file extension is located in the XML file. There are actually two settings, one to specify the CAD system being used and one to specify the delimiter used.

NumberingMode - Appends CAD-specific filetype suffix to autonumber to create Document/Design number

For CAD suffix use PROE,UG,CATIA,SOLIDWORKS,SOLIDEDGE

For no suffix use NONE

NumberingDelim - Value inserted between autonumber and CAD-specific filetype suffix

Default value is "." to match a normal CAD filename

Here is a typical example of how the settings are used:

```
<clientConfig name="NumberingMode" value="PROE"/>
<clientConfig name="NumberingDelim" value="."/>
```

With these settings, the Design numbers will have the appropriate Pro/E filetypes appended to them, using a dot as a separator. The will occur regardless of the method of determining the rest of the number field. Here are some examples:

Using an autonumber and saving a Pro/E part file

- Without using NumberingMode: D00111
- Using NumberingMode: D00111.PRT

Using manual entry and saving a Pro/E drawing file

- Without using NumberingMode: MA-4321
- Using NumberingMode: MA-4321.DRW

Note that when this NumberingMode option is used, the appended CAD filetype **does not appear in**

the EC Client dialog. You will not see the effect of it until you display the Design object and see its Number field.

Options for Part Numbering

PLM Part objects can be assigned to Design objects using the Create Item/BOM command. It is not required to perform this assignment or to publish the Part BOM structure, but when this function is used you need to be aware of the options for defining Part numbers. There are the same 4 primary methods for Part numbering as there are for Design numbering:

1. Using the existing filename
2. Using an autonumber from PLM
3. Using a CAD property
4. Manual entry

These options are only necessary when creating a new Part. If the Part already exists in PLM and you simply want to associate it to a Design, you can use the "Use Existing Item" function within the Create Item/BOM dialog.

The only differences between the options for Design numbering shown above and the options for Part numbering are the symbolic attribute used and the section of the INI file where it is defined. Instead of CAX_NEW_NUMBER the symbolic attribute is called **ITEM**. See the section "Methods for Mapping from CAD to PLM" where it describes the INI file for each connector and the appropriate section to use. Here is an example using SolidWorks:

```
[Agile9UpdateItem]
ITEM = 3DCADTable.Property.Part_Number
```

This says to set the Part number equal to the "Part_Number" property found inside the CAD file.

Note For most CAD systems there is a one-to-one mapping between each Design object and each Part object. For SolidWorks this is not the case when configurations are used. With configurations it is possible to define multiple parts or assemblies within one CAD file. Because of this, there is a special section in the INI file called [Agile9UpdateItemConfigured] which is used to define Parts related to files containing configurations. See the SolidWorks administration section for more details.

File Renaming Options

It is a common tendency with CAD users to use informal filenames when first designing a CAD model. For example a part may be called "new_housing.prt", because it is convenient to use an arbitrary name and it provides information about the file.

EC CAD connectors have the option for what is called "initial renaming", to rename the CAD filenames once they go into PLM. This provides a way to standardize filenames and to insure there are no filename conflicts caused by multiple users coincidentally using the same filename. The renamed file is typically made equal to the Design number, generally using an autonumber.

Initial renaming is set within each CAD connector's INI configuration file; see the administrator guide for details.

Here is an example of how it is used:

Before saving to PLM

CAD filename = new_housing.prt

After saving to PLM

CAD filename = D00111.prt

Design number = D00111.PRT

The original filename can be captured into an attribute in PLM as well, to enable searching on that legacy information. Symbolic attribute CAX_FIL_OLD_NAME, which is called "CAD Old Filename" in the user interface, is available for this purpose. An example of mapping this in the INI file is shown here, using SolidWorks:

```
[Agile9CreateDocument]
CAX_FIL_OLD_NAME = 3DCADTable.ModelName
```

The process by which the CAD filenames are renamed varies by CAD system. For Pro/E, UG NX, and CATIA, the renaming occurs immediately within session as the files are being saved to PLM. The new filenames are updated in the model tree in CAD. For SolidWorks and Solid Edge, the renaming does not occur immediately upon saving; the files are saved with their original names but flagged in a special way such that on the subsequent Load command they are renamed. Upon the second save everything is updated property in PLM.

Note that since this initial renaming is occuring upon the first save of the particular file in PLM, there is no concern about updating any "where used" links, because the file has not been used anywhere other than the current CAD model.

Configuring the Standard Numbering Scenario

One approach for Design and Part numbering is the most common within customer implementations and is known as the "standard numbering scenario". In this approach, an autonumber is used which equals the Part number, and with the addition of the CAD filetype equals both the Design number and the updated filename. Here is an example:

Filename before saving to PLM	Filename after saving to PLM	Design Number	Part Number
mypart123.sldprt	P12345.sldprt	P12345.SLDPRT	P12345

This numbering scheme relies on using an autonumber to form the basis of both the Design and Part number. Since the Design object is actually created first, the autonumber is pulled during the Save command, and the value is stored to be re-used when creating the Part during the Create Item/BOM command. The process works like this:

- User creates a CAD file with the arbitrary filename mypart123.sldprt.
- User saved into PLM using the CAD connector Save command

- During the save process, the user pulls an autonumber P12345, which in combination with the CAD filetype appending option, creates the Design number P12345.SLDPRT
- Since initial file renaming is in use, the filename get re-named to match the Design number
- The original base autonumber (P12345) is automatically saved by EC into a Design attribute called Part Number (symbolic name CAX_PART)
- This Part Number attribute is updated into the CAD file during the Save process using the "update during save" capability. This creates a CAD property called Part_Number.
- Later on the user creates the Part using the Create Item/BOM command. The Part number is mapped from the CAD property Part_Number, which gives the original autonumber value of P12345.

The settings required for this numbering scheme are as follows. This example uses SolidWorks:

1. In the Agile admin client, an autonumber source is created which can create the desired type of numbers, in this case Pnnnnn where nnnnn is a 5-digit number.
2. The XML configuration file must be configured to support CAD filetype appending. For the case of SolidWorks this is:

   ```
   <clientConfig name="NumberingMode" value="SOLIDWORKS"/>
   <clientConfig name="NumberingDelim" value="."/>
   ```

3. The INI configuration file must be configured to use an autonumber for the Design number, meaning that CAX_NEW_NUMBER should be omitted or commented out.

   ```
   [Agile9CreateDocument]
   ;CAX_NEW_NUMBER   = 3DCADTable.ModelTitle   (commented out!)
   ```

4. The INI configuration file must be configured to map the "Part Number" attribute from the Design object into the CAD file during the save process. Note that the value is set into the Design objects "Part Number" attribute automatically by the EC Client, no mapping is required. What we are doing here is to map it back into the CAD file.

   ```
   [Agile9SaveUpdateProperties]
   Part_Number          = CAX_PART
   ```

5. The INI configuration file must be configured to set the Part's Number attribute from the CAD property "Part_Number"

   ```
   [Agile9UpdateItem]
   ITEM                 = 3DCADTable.Property.Part_Number
   ```

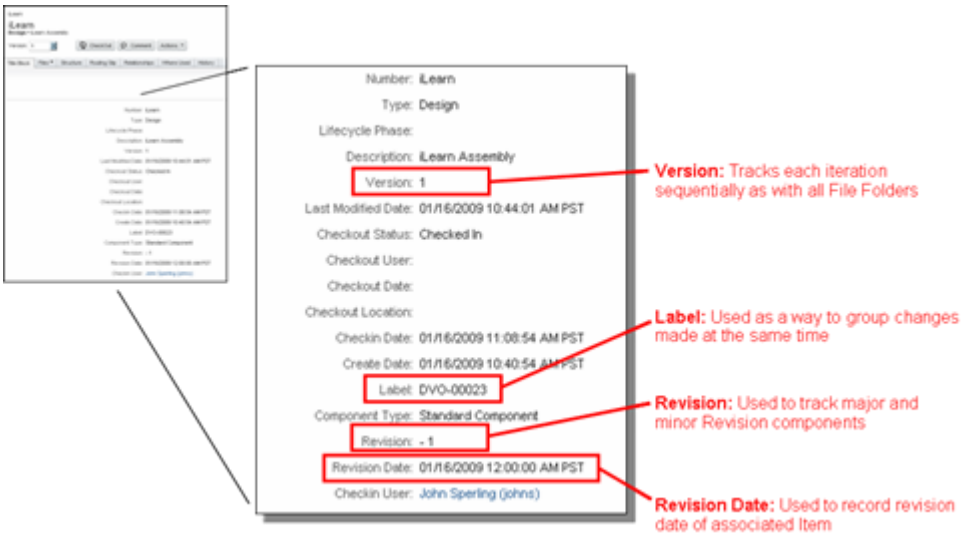
6. The INI configuration file must be configured to enable initial file renaming

   ```
   [Agile9Renaming]
   1
   ```

Overview of Design Change Process and Attributes

A number of options exist for controlling changes of CAD data using Agile EC. This section will explain the options and show how to configure them.

The picture below describes the important Design attributes related to the change process, which are Version, Label, Revision, and Revision Date.

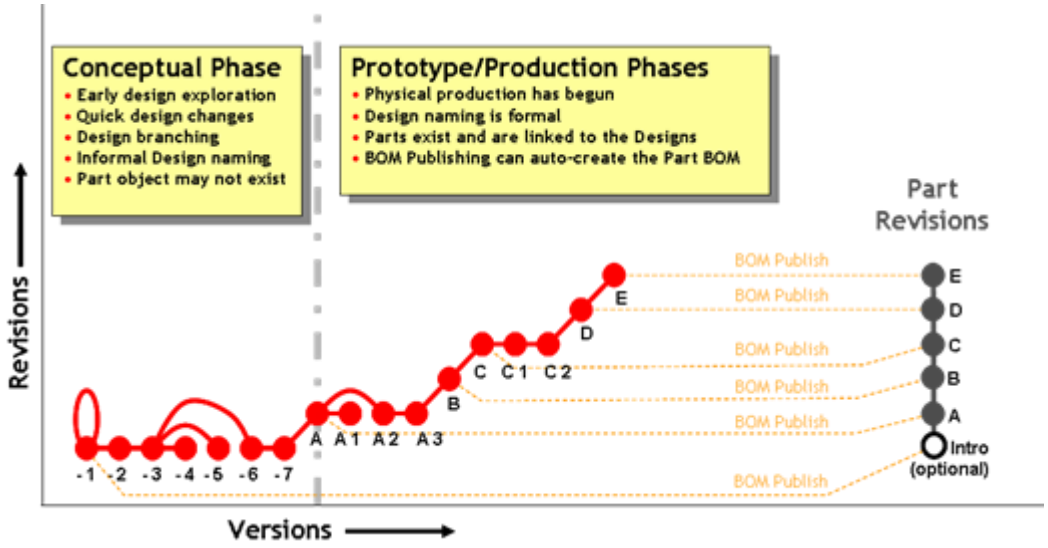


Of these four attributes, only the behavior of Version is pre-determined; all the other ones can be configured to support a variety of change processes.

The Version attribute is a numeric value starting at 1 and is incremented each time a Design goes through a check-out/check-in cycle. During this check-out/check-in cycle, the associated files, attributes, and structure of a Design object may be changed, and are associated with the specific Version. All versions are retained in PLM unless they are manually purged.

The other three attributes, Label, Revision, and Revision Date, are version-specific, meaning that each individual version of the Design has its own values for these three attributes.

The Revision field is controlled by configurable logic in EC. By default the field consists of two parts, a major and minor revision, which is very similar to how other CAD data managers work. (Note that the minor revision is also referred to as "version" but it is not the same thing as the actual Version attribute itself). An example of how the Revision field works is shown in the diagram below; the Revision field is the black text underneath each red dot.



The EC Client logic allows for a major and minor component of the Revision field, with an optional seperator (or "indicator"). The Here are some examples:

Design Revision Sequence	Design Version Sequence	Indicator	Resulting Sequence
A,B,C,D,E,etc.	NUMERIC	<null>	A1,A2,A3,etc,B1,B2,etc,C1,C2,etc
A,B,C,D,E,etc.	NUMERIC	" v"	A v1,A v2,A v3,etc,B v1,B v2,etc,C v1,C v2,etc
-,A,B,C,D,E,etc.	NUMERIC	" "	- 1,- 2,- 3,etc,A 1,A 2,A 3,etc,B 1,B 2,etc

These options are configured within the XML file, using the following lines (this is showing the last example from above):

```
<clientConfig name="DesignRevisionSequence" value=",-
,A,B,C,D,E,F,G,H,I,J,K,L,M, N,O,P,Q,R,S,T,U,V,W,X,Y,Z,"/>
<clientConfig name="DesignVersionSequence" value="NUMERIC"/>
<clientConfig name="DesignVersionIndicator" value=" "/>
```

The way the logic works in the EC Client is that the Revision starts with the first code resulting from the options described above. In the last example this is "- 1". After that, the minor revision portion (the "version") increments upon each check-out/check-in cycle. For example, after "- 1" you will get "- 2".

The question is, what causes the major revision code to increment; what causes you to go from "- 3" to "A 1". The answer is that it depends on whether you are using the BOM Publishing capability in EC to link Designs to Parts. If so, a process extension (PX) trigger on the Part workflow is used to increment the major revision. If not, the major revision can be incremented using Routing Slip approval capability in the EC Client. These two options are described next.

Change Process and Revisioning Using Part Workflow

The preferred method for controlling the change process with Designs is to let the associated Part workflow control the major revision, and the check-out/check-in of Designs control the minor revision. This is only possible if using the BOM publishing functionality of EC CAD connectors (the Create Item/BOM command).

The table below shows a typical change process when using this approach. This assumes that Part revisions are using the alpha sequence A, B, C, etc.

	Design Attributes			Part Attributes
User Action	**Version**	**Revision**	**Revision Date**	**Revision**
User saves CAD model into a Design object in PLM for the first time	1	- 1	<null>	
User checks out the Design object, makes a change in CAD, and saves back to PLM	2	- 2	<null>	
User uses "Create Item/BOM" to create a Part corresponding to the Design, and to publish the BOM	2	- 2	<null>	(A) ECO123
Change ECO123 is submitted	2	(A) ECO123	<null>	(A) ECO123
Change ECO123 is released	2	A	05/19/2009 02:37:34 PM PDT	A ECO123
User checks out the Design object, makes a change in CAD, and saves back to PLM	3	A 1	<null>	

The important points to note about this process are:

- The process extension is used in order to allow the Part workflow to set attributes on the Design objects. This is necessary because the Part workflow is outside the realm of what EC has direct control over.
- This process works only when the Design objects are attached to the Part objects when using the Create Item/BOM command. The PX determines which Design objects to modify based on the Attachments tab of the Part.
- The value of the Revision attribute for Version 2 of the Design is actually changed by the PX, going from "– 2" to "(A) ECO123" to "A".
- The Revision Date field is used to track the date and time of the Part's release.
- Version 2 of the Design is permanently locked to Revision A of the Part, after the Part is released.

- After release, the Revision field of new Design version is sequenced up from the latest major revision, for example "A" goes to "A 1", indicating the first Design version after the released rev A.

Change Process and Revisioning Using Routing Slips

An alternative method for controlling the change process with Designs is to use Routing Slips, which are associated directly to the Design objects themselves. This can be useful if the BOM publishing functionality of EC CAD connectors is not being used.

EC automates the Routing Slips capability through the use of "Label Types", which combine the following capabilities together:

- Autonumber for generating Labels, to identify all Designs changes which are being approved at the same
- User group that defines the Approvers
- User group that defines the Observers
- Whether it is a major or minor revision approval

Here is an example of some standard Label types:

Name: DV (stands for Design Version, meaning the minor revision)
Autonumber: DV00001, etc.
Approvers Group: DV Approvers
Observers Group: DV Observers
Revision Type: Minor

Name: DR (stands for Design Revision, meaning the major revision)
Autonumber: DR00001, etc.
Approvers Group: DR Approvers
Observers Group: DR Observers
Revision Type: Major

Here is a sequence of user actions showing how the Revision field of Designs is controlled by the use of different Label Types and the corresponding processes:

User Action	Version	Revision	Label	Notes
User saves CAD model into a Design object in PLM for the first time	1	- 1		
User uses "Create New Label" function, with the Label Type set to "DV"	[2]	- 2	DV00101	
User saves, using the Check In option	2	- 2	DV00101	Routing Slip notifications are sent to all approvers and observers designated by the groups assigned to the DV Label Type.

Approvers sign off on the Routing Slip				
User uses "Create New Label" function, with the Label Type set to "DR"	[3]	- 3	DR00312	
User saves, using the Check In option	3	- 3	DR00312	Routing Slip notifications are sent to all approvers and observers designated by the groups assigned to the DR Label Type.
Approvers sign off on the Routing Slip				
User checks out the Design object for work	[4]	A 1		The major revision is now incremented, because the approval of the previous version was a DR, which is a "Major revision type" Label.

The important points to note about this process are:

- The "Create New Label" function checks out the Design as well
- Routing Slip approvals are done after the Design version is checked in. If there are approvals in progress, EC will not allow checking out the next version until the approvals are complete.
- There is no configurable workflow associated with Routing Slips. It is just a simple approve or reject for all assigned approvers.
- Incrementing of the major revision happens on the first checkout AFTER the approval of a Design version that is labelled as a major revision type. This is a different type of sequencing than in the change process using Part workflow.

The configuration options to set up the Lable Types are in the XML file. This example shows the definition of the DV and DR types as described in the example:

```
<clientConfig name="LabelTypes" value="DV,DR"/>
<clientConfig name="LabelApproverDV" value="DV Approvers"/>
<clientConfig name="LabelObserverDV" value="DV Observers"/>
<clientConfig name="LabelAutonumberDV" value="DV Label"/>
<clientConfig name="LabelUseRevisionLogicDV" value="0"/>
<clientConfig name="LabelApproverDR" value="DR Approvers"/>
<clientConfig name="LabelObserverDR" value="DR Observers"/>
<clientConfig name="LabelAutonumberDR" value="DR Label"/>
<clientConfig name="LabelUseRevisionLogicDR" value="1"/>
```

The setting "LabelUseRevisionLogicXX" is what determines whether the major revision code is incremented upon the next checkout.

Additional Change Process Information

It is possible to use both change process methods in combination. The typical case of this would be to use the DV (or similar) Label Type to allow approval for minor revisions of the Designs, and then use the Part workflow process for revision approval.

When using the Label Types it is also possible to just use observers during the "approval process", which amounts to sending notifications witout actually requiring approval. In order to do this, you must assign a "LabelApproverXX" group, and define this group in PLM, but just do not include any users in the group definition.

Chapter 4

Installing and Configuring Pro/ENGINEER Connector

This chapter includes the following:

This section describes setting up the connection between your Pro/ENGINEER CAD application and Agile Engineering Collaboration.

The main steps are:

- Extract files from Pro/ENGINEER Connector zip file
- Extract files from EC Client zip file
- Configure PLM API for WAN mode (optional)
- Edit some parameters in the configuration file
- Edit some parameters in the mapping file
- Install proper **AgileAPI.jar** file
- Create shortcut to new startup file
- Create toolbar in Pro/E (optional)

The installation requires the following file:

acpNNNN.zip - Main installation package, where NNNN is the release level.

Performing the installation steps described here will enable the Agile menu to appear within Pro/E. In order to have a completely functional integration, you must also:

- Perform the core Agile configuration, as described in the Administration section of the document Agile Engineering Collaboration Client
- Configure desired Pro/E Connector parameters as described in the section Pro/E Connector Administration on page 36 on page 35.

Extracting Files for Connector

Extract the installation file to the folder location **D:\AgileEC**, or **C:\AgileEC** for a single-drive system. When you unzip, make sure that you retain the folder paths from the zip file. When the files are unzipped, you should see a folder named **acp**, which contains the Connector installation.

Extracting Files for EC Client

Extract the EC Client installation file to a temporary location. After extraction you will see 5 connector directories (acc, ace, acp, acu, acw) plus a jar directory.

- Copy the entire jar directory inside the \AgileEC\acp directory.
- From the temporary **acp\com** directory, copy the file CaxClient.bat to the **\AgileEC\acp\com** directory.
- From the temporary **acp\jar\Agile9** directory, copy the file CaxClient_Designs.xml to the **\AgileEC\acp\jar\Agile9** directory.

Configure PLM API for WAN Mode

PLM API is a high-performance web services API used to support operations across wide area network (WAN). This is an optional capability that is only necessary if you are going to be supporting MCAD connectors at remote sites (i.e. locations distant from your Agile application server). If you do not need this capability, skip to the next step.

- Log in with the Agile 9 web client to the server that will be used with the MCAD connectors. Use an account that has privileges to create File Folder objects.
- Create the following three File Folder objects, using the number listed in the first column, and then attach the file listed in the second column. Check In the File Folder following attachment. The files can be found under the jar/agile9/server folder. Note that depending on your system settings, you might need to temporarily modify the admin settings for File Folder objects to allow using an arbitrary Number field.

File Folder number	File to attach
PLMAPI_ASYNC	plm-api-server.xml
PLMAPI_CONFIG	plm-api-config.properties
PLMAPI_CONNECTOR	plm-api-sdk.xml

- Copy the file plm-api-server.jar from jar/agile9/server to the WSX extensions folder of the target Agile 9 server. Usually this folder is located in {agile_home}/integration/sdk/extensions).
- The EC Client configuration file must be edited to turn on PLM API mode. To do this, bring up either the CAXClient_Documents.xml or CAXClient_Designs.xml file (depending on which data model you are using) in a text editor. Change the following line as follows (change from 0 to 1). Save the file.

```
<clientConfig name="usePlmApi" value="1"/> <!-- Use PLM API Web
Services 1=enabled, 0=disabled -->
```

- The EC Client startup file must also be edited to enable PLM API mode. To do this, bring up acp\com\caxclient.bat in a text editor. Change the following lines by appropriately uncommenting and commenting the lines. Note that lines with "rem" in front of them are commented out and inactive:

Settings	Necessary settings for using PLM API
DFM_JAR	set DFM_JAR=%CAX_ROOT%\jar\agile9\axis.jar;.....(etc.) **rem** set DFM_JAR=
JAVA_HEAP_SIZE	rem set JAVA_HEAP_SIZE=-Xms128m -Xmx128m set JAVA_HEAP_SIZE=-Xms128m -Xmx1024m

Editing the Configuration File

Open the file **\AgileEC\acp\com\Acp.cfg** in a text editor. Edit the values as described in the table below to match your system configuration.

Sample values	What this command specifies
AcpUserRoot=D:\AcpUser	Working directory for user data and files
AcpLang=english	Menu and UI language
AcpJava=C:\j2sdk1.4.2_04\jre	Path to Java Runtime Environment. Determine the required Pro/ENGINEER connector JRE version from the chart in Appendix C, and then select the appropriate path from the available ones provided in the acp installation. If no value is set then the default %AcpRoot%\jre1.5.0 is used. Note: Avoid blanks or spaces in path.
AcpProEV=2007	Currently, installed version of Pro/E: ▫ "2007" designates Pro/E Wildfire 4 ▫ "2006" designates Pro/E Wildfire 3 ▫ "2003" designates Pro/E Wildfire 2
AcpStartProE=D:\proewildfire\bin\proewildfire.bat	▫ Set to the path and filename of your Pro/E startup script; this file is usually located in the bin directory of the Pro/ENGINEER installation. However, if your company has a customized Pro/E start script, please set this value accordingly. ▫

Editing the Mapping File

Open the file **\AgileEC\acp\ini\AcpCustomer9.ini** in a text editor. Edit the values as described in the table below to match your system configuration.

Important Avoid blank lines in this file! A comment line always starts with a # sign.

Sample values	What this command specifies
AcpAgileServerURL = http://agileserver:8888/Agile	URL for your Agile server. Change to your server name or address. This is the default server URL that will be used when you run the EC Client, it can be changed interactively.
AcpAgileUser = cax	Default username that will be used when you run the EC Client, it can be changed interactively.
AcpAgilePwd = agile	Default password that will be used when you run the EC Client, it can be changed interactively. Use "" (double quotes) for a blank entry.

Installing the AgileAPI.jar file

Important If this step is not done correctly, the connector may appear to be functioning normally but data corruption may occur!

The correct **AgileAPI.jar** file, matching the specific Agile service pack level, must be installed in the directory **\AgileEC\acp\jar\Agile9**.

- Search for the file **AgileAPI.jar** within your site's Agile server installation (such as **C:\Program Files\Agile**).
- Copy this file to **\AgileEC\acp\jar\Agile9**, overwriting the file already there.
- If you are unable to locate this file, please contact Agile Support.

Creating a Shortcut to the Startup File

In order to run Pro/E with the Connector, you must run using the startup file **\AgileEC\acp\com\acp_start.bat.** To make this more convenient, you may want to create a Windows shortcut to this file, either on your Desktop or in your Quick Launch bar.

Verify that the Pro/E Connector is working by double-clicking on your shortcut to launch Pro/E. You should see an Agile menu appear in the main menu bar.

Creating the Agile Toolbar in Pro/E

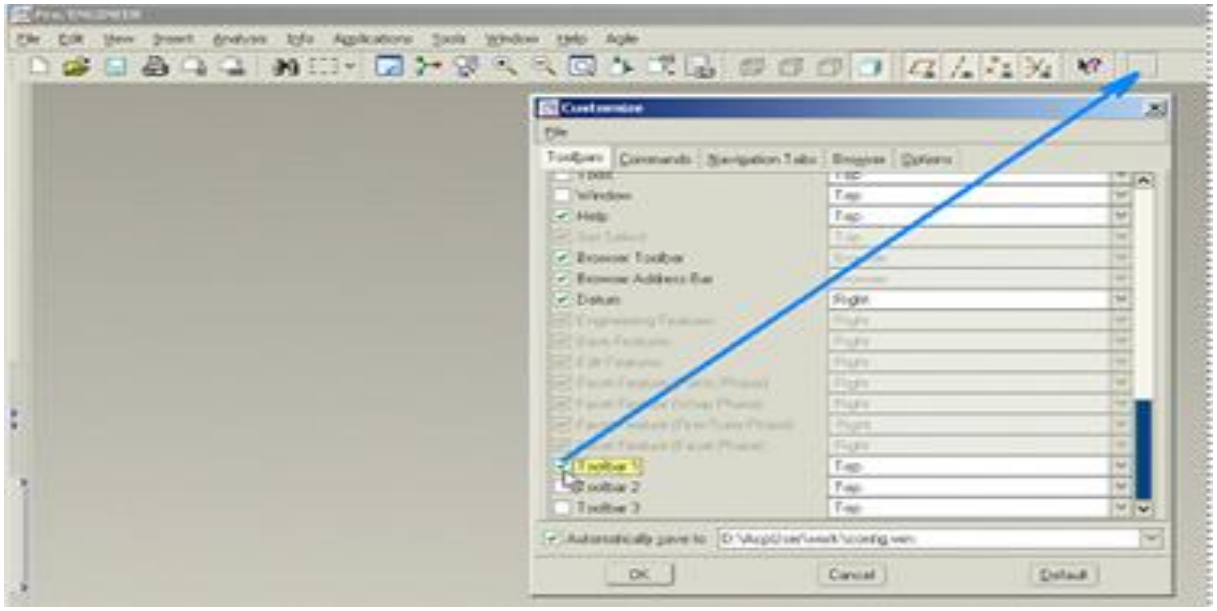
This step is optional, it will create a toolbar that you can use to run the Agile commands, in addition to the Agile menu.

To create the toolbar icon on the Pro/E toolbar:

1. Choose **Tools > Customize Screen** and select the **Toolbars** tab.
2. When you enable the Toolbar 1 field by clicking the checkbox, a new “blank icon” appears on the main toolbar.

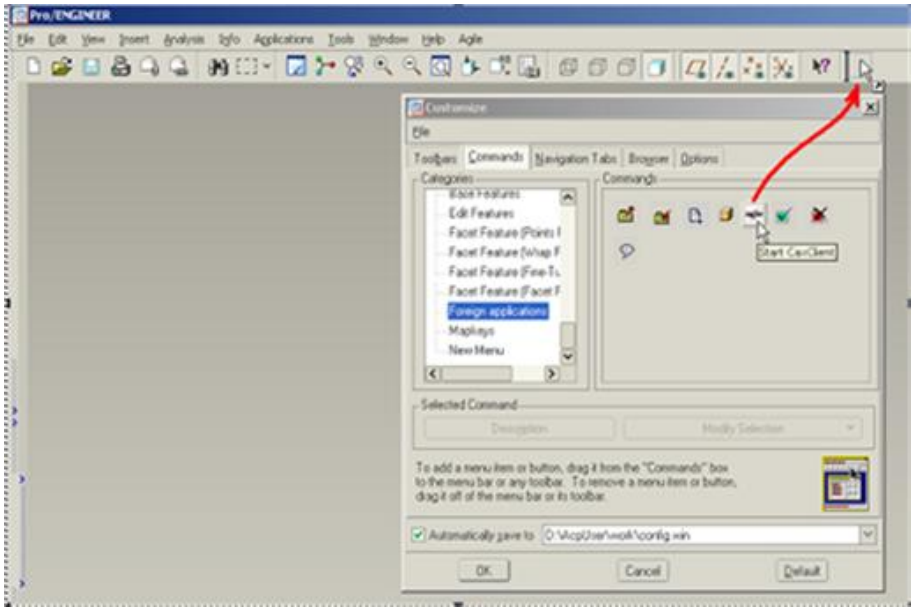
Figure: Enabling the Toolbar

Figure 1: Toolbar in Pro/E



1. Again, under **Tools > Customize Screen**, this time select the **Commands** tab. Scroll down and select **Foreign applications**. The icons of the “Agile” menu appear in the Commands area.

Figure: Moving icons to the Toolbar



Installing on Additional Computers

Once the Pro/E Connector has been installed and configured on one machine, you can install on other machines simply by copying the entire \AgileEC\acp folder structure. This works as long as the machines are configured the same in terms of their Pro/E setup, Java setup, etc.

Configuring the Pro/ENGINEER Connector

This section provides a complete summary of configuration options available for the Pro/ENGINEER connector. Once the basic installation has been done following the instructions in the previous section, you can refer here for details of all possible settings.

Note that in addition to the configuration files listed here, the EC Client must be additionally configured to provide complete operation of the Pro/ENGINEER Connector. See the EC Client Configuration Options section for details.

Table: List of all Configuration Files for the Pro/E Connector

Configuration files	Purpose	Location
Acp.cfg	System configuration	AgileEC\acp\com
AcpCustomer9.ini	Mapping and configuration	AgileEC\acp\ini

Note Configuration files typically change content between connector releases. When upgrading to a new release, please incorporate your site's configuration settings into the new version of the configuration files. Failure to do so will cause unpredictable behavior of the connector.

Configuration File Acp.cfg

The configuration file **Acp.cfg** contains basic system parameters. It is described fully in the Installing and Configuring Pro/ENGINEER Connector section on page 31 on page 30.

Mapping File AcpCustomer9.ini

This is the main file for controlling the behavior of the Pro/E Connector. This file is structured in several sections. The first line of a section starts with a left square bracket followed by a space and its name again followed by a space and the right square bracket. Each section starts with the section name. A comment line starts with the # sign #.

Note Please make sure not to leave blank lines when editing the file.

Tables below gives a description of all sections in **AcpCustomer9.ini**, and the following tables provide the details of each section.

Table: Description of all sections in AcpCustomer9.ini

Section name	Description
Initialize	Common switches to control the behavior of the Pro/E Connector
ProEToAgile.Create_DOCUMENT	This mapping section is used for initial creation of documents using the Save command.
ProEToAgile.Update_DOCUMENT	This section is used when existing documents are updated using the Save command.
ProEToAgile.Update_FILEFOLDER	This section is used when existing objects are updated via the Agile Save command.
ProEToAgile.Create_ITEM	Not used
ProEToAgile.Update_ITEM	This section is used when creating or updating parts when using the Create Item/BOM command.
AgileToProE.ProE	Defines those Agile attributes that are saved automatically into all Pro/E files, during the Save command.
AgileToProE.PRT	Defines those Agile attributes that are saved automatically into Pro/E PRT files, during the Save command.
AgileToProE.DRW	Define those Agile attributes that are saved automatically into Pro/E DRW files, during the Save command.
AgileToProE.ASM	Defines those Agile attributes that are saved automatically into Pro/E ASM files, during the Save command.
AgileGetProperties.PRT	Defines those Agile attributes that are saved into Pro/E PRT files, when using the Update Properties command.
AgileGetProperties.DRW	Defines those Agile attributes that are saved into Pro/E DRW files, when using the Update Properties command.

Section name	Description
AgileGetProperties.ASM	Defines those Agile attributes that are saved into Pro/E ASM files, when using the Update Properties command.
EcpMenu	Defines the mapping betwwen TCL procedures and menus (internal use only)

Table: [Initialize] Section Parameters

Parameter name in section [Initialize]		Parameter values	Description
AcpStartPart	=	START	Name of the default seed part
AcpStartAssembly	=	START	Name of the default seed assembly
AcpStartDrawing	=	START	Name of the default seed drawing
AcpDebug	=	**1**../..0	Turns debug mode on (1) and off (0). A log file is written to the user's working directory.
AcpAgileServerURL	=	http://agileserver:8888/Agile	Default URL that the EC Client will use to connect to the Agile Application Server
AcpAgileUser	=	cax	Default user that is used to log in to Agile when the user chooses **Connect** from the CAD system
AcpAgilePwd	=	agile	Default password to log in to Agile
AcpSaveDrwFrm	=	**1**../..0	1 = [Pro/E] drawing formats are stored in a unique document object in Agile 0 = drawing formats are stored in a local [Pro/E] path
AcpSaveLay	=	**1**../..0	1 = [Pro/E] layouts are stored in a unique document object in Agile 0 = layouts are stored in a local [Pro/E] path
AcpHelpPartIdent	=	ITEM	Name of Pro/E parameter used to identify models in the design that should not be included in the BOM, such as "skeleton parts." These objects are saved into Agile as documents, but are filtered out when using the Create Item/BOM function.
AcpHelpPartValue	=	**NO**	Value that the Pro/E parameter should be set to in order to activate the filter
AcpDefaultClass	=	DOCUMENT	System use only, do not modify
AcpCheckModify	=	**1**../..0	0 = objects will be saved to Agile whether or not they have been modified in the Pro/E session. 1 = only objects modified in the Pro/E session will be saved to Agile.

Parameter name in section [Initialize]		Parameter values	Description
AcpParAgileNumber	=	AgileId	This parameter is the place where the name of a Pro/E parameter can be defined. It will be updated with the Agile ID number after saving to Agile.
AcpSearchAgileNumberPar	=	AgileId	"Par" refers to user-defined parameters in Pro/E. This Pro/E parameter value is used to map a CAD object to a previously existing Agile object. For example: AcpSearchAgileNumberPar=PART_NUMBER A parameter called PART_NUMBER exists in the model. Its value is set to, e.g., MODEL01234. Upon execution of the **Agile > Save** command in Pro/E, the connector tries to attach the CAD file to an existing Agile object whose ID number is MODEL01234. If such an object is not found, a new object is automatically created.
AcpUseObjectNameForId	=	1../..0	0 = files are not renamed 1 = files are renamed to match the Agile Number field or custom mapping
EcpMenuMainRes	=	[EcpGetenvAcpMainRes]	System use only, do not modify
EcpMenuCallback	=	EcpMenu	System use only, do not modify

Mapping Options for [ProEToAgile.XXXX] Sections

Each mapping consists of a pair of objects. The right side of the pair defines information that can be extracted from Pro/E. Here, Pro/E is the source of the attribute value. The left side of the pair defines the attribute value's target location in Agile.

There are several configuration options for the "right side" that define what kind of data should be extracted from Pro/E, and what kind of transformation can be applied to the data. Each right side attribute consists of three sections, for example:

```
DESCRIPTION = Std.ObjectName-Type.ToUpper
```

The first section is either Std or Par. "Std" refers to Pro/E system attributes such as file name, object type, version of Pro/E that is being used, and so forth.

Table: Standard mapping values using "Std2 prefix

Std.CreSystem	Pro/E version such as "Pro/E 2001" or "Pro/E Wildfire"
Std.VerStamp	Timestamp
Std.FileName	File name, for example "BOLT.PRT"
Std.ObjectName	Pro/E file name without the extension - "BOLT"

Std.ObjectName-Type	Object name with the type appended. This creates an easy way to differentiate an assembly from a part. Examples include: BOLT-PRT, BOLT-ASM, or BOLT-DRW.
Std.ObjectType	Pro/E object type. Possible values are PRT, ASM, DRW, or FRM.

"Par" is a reference to user-defined parameter in Pro/E, such as MATERIAL, DESCRIPTION, or ENGINEER. These types of mappings are only useful where the Pro/E file has a parameter corresponding to the name mentioned in the mapping.

Finally, the final suffix is a description of how the data should be modified. The following modifiers are possible:

Table: Suffix Options for Mapping

ToUpper	Transfer all characters to upper case
ToLower	Transfer all characters to lowercase
None	Do not modify the data
Range-<idx1>-<idx2>	Range of the string from position idx1 to idx2, example: Part.PartNumber.Range-0-2
Prefix	Prefix to be added in front of the string, example: Par.PartNumber.PrefixPRT
Suffix	Suffix to append to the string, example: Par.PartNumber.SuffixPRT

There are two special values that are used on the left side of these mappings. In the [ProEToAgile.Create_DOCUMENT] section, you use the value CAX_NEW_NUMBER to represent the Number field that will be assigned to newly created Documents. In the [ProEToAgile.Update_ITEM] section, you use the value ITEM to represent the Number field that will be assigned to newly created Parts.

The following are some example mappings for a Pro/ENGINEER part called housing.asm with a material value of Aluminum:

Table: Example Mapping Definitions

Std.ObjectName-Type.ToUpper	=	DESCRIPTION
Par.Material.None	=	MATERIAL
Std.FileName.ToLower	=	CAD_FILENAME

In this example, the Agile Description would be HOUSING-ASM, an attribute in Agile called CAD_FILENAME would have the value housing.asm and an Agile Attribute called Material would have the value Aluminum.

Mapping Options for [AgileToProE.XXXX] Sections

These section are used to define mappings from Agile to Pro/E, which occur automatically during the Save process. As this will add time to the Save process, the list of attributes should be kept to the bare minimum that absolutely need to be kept synchronized. Other attributes can be synchronized using "Update Properties", as described in the next section. The format of this section is:

```
DocNumber    =    NUMBER
```

Where the left side value is the name of the Pro/E parameter to be updated, and the right side is the Agile attribute value to be used as the source.

Mapping Options for [AgileGetProperties.XXX] Sections

These section is used to define mappings from Agile to Pro/E, which occur when the user runs the Update Properties command manually. For standard attributes the format of this section is:

CAD Parameter = <Source Table>_Field.Format

For example:

Agile_Des = Title Block_Description.ToUpper

Where the left side value is the name of the Pro/E parameter to be updated, and the right side is the Agile attribute value to be used as the source, as follows:

Section	Represents	Example
<Source Table>	Agile tab name	Title Block
Field	Agile attribute name	Description
Format	Text processing	ToUpper

For history and change history attributes, which are arranged in a table, the format of this section is:

CAD Parameter = <Filter Table>_Field,<Filter Value>,<Filter>,<Source Table>_Field.Format

For example:

Agile_CreUser = History_Action,Create,first,History_User.None

HIS_RELDATE_1 = Change History_Status,Released,last,Change History_Rel Date_int.Date01

Where the left side value is the name of the Pro/E parameter to be updated, and the right side specifies how to find the desired row and column in the table below:

Section	Represents	Example
<Filter Table>	Agile tab name to search	Title Block
Field	Desired column to search	Action
<Filter Value>	Value to detect in the column	Create
<Filter>	Which row to select, with these options: first first+n n=integer value last last-n n=integer value	first
<Source Table>	Agile tab name to retrieve value from	History
Field	Desired column to retrieve value from	User

Section	Represents	Example
Format	Text processing	None

Options for "Format"

The Format string allows you to perform additional processing on the text string being passed back into CAD. This includes predefined formats and general TCL format procedures.

Predefined formats

Format	Description
None	no processing
ToLower	convert the value to lower case
ToUpper	convert the value to upper case
Range-x-y	substring of the value from index x to index y (y may be numeric or "end")
Date01	convert int dateformat to "%d.%m.%y %H:%M:%S" example: 01.01.2007 00:00:00
Date02	convert int dateformat to "%d.%m.%Y" example: 01.01.2007
Date03	convert int dateformat to "%d.%m.%y" example: 01.01.07
Date04	convert int dateformat to "%d-%m-%y" example: 01-01-07
Date05	convert int dateformat to "%m/%d/%y" example: 01/01/07
Date06	convert int dateformat to "%d-%b-%y" example: 01-Jan-07
Prefix<str>	append a prefix <str> to the value
Suffix<str>	append a suffix <str> to the value

TCL format procedures

Any registered (tclIndex) TCL procedure that gets the current value as input and returns the formatted string. For instance:

```
proc MyFormat { value } {
  set formatedvalue $value
  return $formatedvalue
}
```

Mapping Part Attributes

In addition to mapping attributes from the CAD Document back into CAD, you can map attributes from the corresponding Part object that has been associated to the Document using the Create Item/BOM command. In order to specify a Part attribute, simply prefix the attribute value with "PART:". This example shows mapping both the Document Number and Part Number into CAD:

Agile_DocId = Title Block_Number.None

Agile_PartId = PART:Title Block_Number.None

Chapter 5

Installing and Configuring SolidWorks Connector

This chapter includes the following:

This section describes setting up the connection between your SolidWorks CAD application and Agile Engineering Collaboration. The main steps are:

- Extract files from SolidWorks Connector zip file
- Extract files from EC Client zip file
- Configure PLM API for WAN mode (optional)
- Special setup for 64-bit
- Edit some parameters in the configuration file
- Register library and executable
- Install proper **AgileAPI.jar** file
- Set up Agile menu in SolidWorks

The installation requires the following file:

acwNNNN.zip – Main installation package, where NNNN is the release level

Performing the installation steps described here will enable the Agile menu to appear within SolidWorks. In order to have a completely functional integration, you must also:

- Perform the core Agile configuration as described in the Administration section of the document *Agile Engineering Collaboration Client*.
- Configure desired SolidWorks Connector parameters as described in the section SolidWorks Connector Administration on page 49 on page 47.

Extracting Files for Connector

Extract the installation file to the folder location **D:\AgileEC** or **C:\AgileEC** for a single-drive system. When you unzip, make sure that you retain the folder paths from the zip file. When the files are unzipped, you should see a folder named **acw**, which contains the Connector installation.

Extracting Files for EC Client

Extract the EC Client installation file to a temporary location. After extraction you will see 5 connector directories (acc, ace, acp, acu, acw) plus a jar directory.

- Copy the entire jar directory inside the \AgileEC\acw directory.
- From the temporary **acw\server\scripts** directory, copy the file caxclient.bat to the **\AgileEC\acw\server\scripts** directory.
- From the temporary **acw\jar\Agile9** directory, copy the file CaxClient_Designs.xml to the **\AgileEC\acw\jar\Agile9** directory.

Configure PLM API for WAN Mode

PLM API is a high-performance web services API used to support operations across wide area network (WAN). This is an optional capability that is only necessary if you are going to be supporting MCAD connectors at remote sites (i.e. locations distant from your Agile application server). If you do not need this capability, skip to the next step.

- Log in with the Agile 9 web client to the server that will be used with the MCAD connectors. Use an account that has privileges to create File Folder objects.
- Create the following three File Folder objects, using the number listed in the first column, and then attach the file listed in the second column. Check In the File Folder following attachment. The files can be found under the jar/agile9/server folder. Note that depending on your system settings, you might need to temporarily modify the admin settings for File Folder objects to allow using an arbitrary Number field.

File Folder number	File to attach
PLMAPI_ASYNC	plm-api-server.xml
PLMAPI_CONFIG	plm-api-config.properties
PLMAPI_CONNECTOR	plm-api-sdk.xml

- Copy the file plm-api-server.jar from jar/agile9/server to the WSX extensions folder of the target Agile 9 server. Usually this folder is located in {agile_home}/integration/sdk/extensions).
- The EC Client configuration file must be edited to turn on PLM API mode. To do this, bring up either the CAXClient_Documents.xml or CAXClient_Designs.xml file (depending on which data model you are using) in a text editor. Change the following line as follows (change from 0 to 1). Save the file.

```
<clientConfig name="usePlmApi" value="1"/> <!-- Use PLM API Web
Services 1=enabled, 0=disabled -->
```

- The EC Client startup file must also be edited to enable PLM API mode. To do this, bring up \acw\server\scripts\caxclient.bat in a text editor. Change the following lines by appropriately uncommenting and commenting the lines. Note that lines with “rem” in front of them are commented out and inactive:

Settings	Necessary settings for using PLM API
DFM_JAR	set DFM_JAR=%CAX_ROOT%\jar\agile9\axis.jar;.....(etc.) **rem** set DFM_JAR=
JAVA_HEAP_SIZE	rem set JAVA_HEAP_SIZE=-Xms128m -Xmx128m set JAVA_HEAP_SIZE=-Xms128m -Xmx1024m

Configuring for a 64-bit System

With 64-bit, the macro PlmSWMacro.swp is used for the communication between SolidWorks Addin and the CaxOleServer. This macro can be found in the following directory: acw\server\Scripts and must be edited manually.

1. Start SolidWorks and select **Extras > Macro > Edit.**
2. Select the file acw\server\Scripts\PlmSWMacro.swp.

The SolidWorks VBA development environment is opened.

3. Select **Extra > References.**

All dlls registered for the System are displayed. For all dlls used by the Integration, the checkbox is marked.

4. Select the "axalantSW 1.0 Type library" and click the “Search” button to set the reference as follows:
   - SolidWorks 2009 x 64: acw\SolidWorks\agilePLMSW2009x64.dll
   - SolidWorks 2008 x 64: acw\SolidWorks\agilePLMSW2008x64.dll
   - SolidWorks 2007 x 64: acw\SolidWorks\agilePLMSW2007x64.dll
5. Click **OK** to confirm the changes.

The system will return to the SolidWorks VBA development environment.

6. Click **Debug > Compile PlmSWMacro.**
7. Select **File > Close and back to SolidWorks.**
8. You can now activate the integration in SolidWorks by selecting **Extras > Additional Applications**.

Editing the Configuration File

Open the file **\AgileEC\acw\Server\Scripts\3DCADMapping.ini** in a text editor. Edit the values as described

in the table below to match your system configuration.

Sample values	What this command specifies
[JNIOPTIONS] ; - Djava.class.path=D:\AgileEC\acw\jre1.5.0\lib\rt.jar; D:\AgileEC\acw\Server\AgileCaxConnector.jar ; D:\AgileEC\acw\Server\AgileAPI.jar; D:\AgileEC\acw\Server\xercesImpl.jar; D:\AgileEC\acw\Server\xmlParserAPIs.jar; D:\AgileEC\acw\Server\CaxAglProxy.jar; D:\AgileEC\acw\Server\CaxAglDataTypes.jar - Dagile.xml.file=D:\AgileEC\acw\Server\AgileConnector.xml -Djava.agile.gui.address=localhost -Djava.agile.gui.listener=5112 -Djava.agile.proxy.listener=5113 -Dagile.caxconnect.logfile=C:\agile.log ;-Djava.agile.proxy.logfile=C:\Proxy.log ;	This provides the paths to all jar files. Edit each path to match your system configuration. The first path listed is the JRE environment. Determine the required SolidWorks connector JRE version from the chart in Appendix C, and then select the appropriate path from the available ones provided in the acw installation. Edit all other paths to match the drive and directory in your installation. **Note** Note that the proxy.log file (the last line in this section) should be commented out for production use, because this causes an additional log file window to pop up that would be annoying to users. Uncomment only for debugging.
[CheckOutDisk] ; D: ;	The disk drive on the client computer to be used for the working directory.
[CheckOutPath] ; \AgileSW\Work\ ;	The path of the working directory. **Important** You must also create this directory on your computer.
[LogFileDir] ; D:\AgileSW\Temp\ ;	The full path of the log file directory. **Important** You must also create this directory on your computer.

Sample values	What this command specifies
[AgileURL] ; http://servername:8888/Agile ;	The URL for the EC Client.
[Agile9TemplatePath] ; D:\AgileSW\Template	The path of the template directory. This path must exist and at least one template file must exist for parts, assemblies, and drawings.

Registering the Library

Navigate to the \AgileEC\acw directory, and double-click on register.bat. A command window will appear and just follow the prompts to enter the proper SolidWorks version and Windows OS version. For example:

Please enter SolidWorks Version (SW2007 or SW2008 or SW2009 or SW2010) : SW2010

Please enter your OS architecture (x32 or x64) : x32

Registration procedure for SolidWorks2010x32 connector started

After a moment you will see a pop-up with a message similar to this:

DllRegisterServer in D:\AgileEC\acw\SolidWorks\agilePLMSW2008.dll succeeded.

Click **OK** to complete.

Installing the AgileAPI.jar file

Important If this step is not done correctly, the connector may appear to be functioning normally but data corruption may occur!

The correct **AgileAPI.jar** file, matching the specific Agile service pack level, must be installed in the directory **\AgileEC\acw\jar\Agile9**.

- Search for the file **AgileAPI.jar** within your site's Agile server installation (such as **C:\Program Files\Agile**).
- Copy this file to **\AgileEC\acw\jar\Agile9**, overwriting the file already there.
- If you are unable to locate this file, please contact Agile Support.

Setting Up the Agile Menu

To set up the Agile menu within SolidWorks, do the following steps:

- Launch SolidWorks as you normally would (Using the Start menu or desktop icon, etc.)
- In SolidWorks, go to Tools > Add-Ins and check the box next to AgilePLM. Click **OK**.
- The Agile menu should now appear in the SolidWorks menu bar. If it does not, or you receive an error at this point, something has not been set up properly.

The Agile menu in SolidWorks should now be functional.

SolidWorks Connector Administration

This section provides a complete summary of configuration options available for the SolidWorks connector. Once the basic installation has been done following the instructions in the previous section, you can refer here for details of all possible settings.

Note that in addition to the configuration files listed here, the EC Client must be additionally configured to provide complete operation of the SolidWorks Connector. See the EC Client Configuration Options section for details.

Table: List of all Configuration Files for the SolidWorks Connector

Configuration file	Purpose	Location
3DCADMapping.ini	Mapping and configuration	AgileEC\acw\Server\Scripts
PlmSWAddin.xml	Menu definition	AgileEC\acw\Server\Scripts

Note Configuration files typically change content between connector releases. When upgrading to a new release, please incorporate your site's configuration settings into the new version of the configuration files. Failure to do so will cause unpredictable behavior of the connector.

Configuring the 3DCADMapping.ini File

There is one main configuration file, which controls nearly all aspects of the SolidWorks Connector. The file is named **3DCADMapping.ini** and is located in the **..\AgileEC\acw\Server\Scripts** directory. Since this file is located within the SolidWorks Connector installation on the client machine, it is possible to customize configuration options on a per-machine basis, although typical usage is to have a common configuration file within a given site. When changes are made to the configuration file, it is necessary to exit and re-start SolidWorks in order to use the new settings.

The configuration file is made up of a series of configuration options (also called "sections"), with the option listed between square brackets, and the various settings for the option listed on the following lines. Lines beginning with a semi-colon (;) are commented out. In this file, unused options are commented out rather than deleted, which may help later if you want to enable some of the unused options.

Because this configuration file is also used for Agile 8.5 and Agile e-series installations, not all of the configuration options are valid for Agile 9. The following list summarizes the options that are valid for Agile 9.

Table: Valid configuration options in SolidWorks 3DCADMapping.ini

Option Name	Usage
[JNIOPTIONS]	Sets various Java parameters
[LogFileDir]	Drive & path of temp directory
[CheckOutDisk]	Disk drive of work directory
[CheckOutPath]	Path of work directory

Option Name	Usage
[LogFileDir]	Disk and path of log file directory
[AgilePartViewFile]	OBSOLETE. See [Agile9PartViewFileExtensions]
[AgileAssemblyViewFile]	OBSOLETE. See [Agile9AssemblyViewFileExtensions]
[AgileDrawingViewFile]	OBSOLETE. See [Agile9DrawingViewFileExtensions]
[AgileViewFileCustomScript]	Drive, path & name of a executable file which will be executed to generate a customized neutral view file to be saved
[AgilePartCheckinFile]	OBSOLETE. See [Agile9PartViewFileExtensions]
[AgileAssembly CheckinFile]	OBSOLETE. See [Agile9AssemblyViewFileExtensions]
[AgileDrawing Checkin File]	OBSOLETE. See [Agile9DrawingViewFileExtensions]
[AgileURL]	Information required to connect to the Agile server.
[Agile9CreateDocument]	This mapping section is used for setting attributes for Documents during the **Save** command, for the initial save.
[Agile9UpdateDocument]	This mapping section is used for setting attributes for Documents during the Save command, after the initial save.
[Agile9UpdateItem]	This mapping section is used for setting attributes for Parts during the **Create Item/BOM** command
[Agile9UpdateItemConfigured]	This mapping section is used for setting attributes for Parts during the Create Item/BOM command, when the CAD file is identified as configured.
[Agile9CheckinDocument]	This mapping section is used for setting file attachment attributes in Agile.
[Agile9CheckinViewableTIF]	
[AgileViewableIncludeRevision]	Appends the revision of the Document object onto the end of the viewable filenames generated in the Save comment.
[Agile9GetRevision]	Retrieves the current revision field of the Document
[Agile9UpdateProperties]	Defines the property mapping from Agile to SolidWorks, when using the Update Properties command
[Agile9SaveUpdateProperties]	Defines the property mapping from Agile to SolidWorks that occurs automatically during the **Save** command.
[Agile9LoadUpdateProperties]	Defines the property mapping from Agile to SolidWorks that occurs automatically during the **Load** command
[Agile9UpdateTitleBox]	Defines the property mapping from Agile to SolidWorks, when the properties of a drawing are updated using the Update Title Block command
[AgilePartTemplate]	Drive, path & name of a SolidWorks template part file for use with the **New** command
[AgileAssemblyTemplate]	Drive, path & name of a SolidWorks template assembly file for use with the **New** command

Option Name	Usage
[AgileDrawingTemplate]	Drive, path & name of a SolidWorks template drawing file for use with the **New** command
[Agile9Configuration]	Controls how SolidWorks configurations are handled
[EC_CLIENT_URL]	OBSOLETE
[Agile9PartViewFileExtensions]	Sets the allowable viewable file formats created during the Save command
[Agile9AssemblyViewFileExtensions]	Sets the allowable viewable file formats created during the Save command
[Agile9DrawingViewFileExtensions]	Sets the allowable viewable file formats created during the Save command
[Agile9TemplatePath]	Sets the path to where template files are stored, for use by the New command
[Agile9Units]	Sets the default units for the New command
[Agile9Renaming]	Activates the filename renaming process during the Load command
[SaveAllConfigurationsToEDrawing]	Sets whether just the active configuration or all configurations are saved to eDrawings files.
[SWTraverseMode]	Sets the type of structure traversal algorithm is used by the connector.

Table: Detailed Configuration Options

Option	Description
Syntax	**Configuration Options**
[JNIOPTIONS]	Java Parameters
-Djava.class.path=<path><file>.jar, etc.	1.For rt.jar, edit <path> to match location of Java installation. 2. For other jar files, edit <path> to match installation path (default is D:\AgileEC\acw\Server)
-Dagile.xml.file=<path>\AgileConnector.xml	Edit <path> to match installation path (default is D:\AgileEC\acw\Server)
-Djava.agile.gui.address=localhost	Do not change
-Djava.agile.gui.listener=5112	Do not change
-Djava.agile.proxy.listener=5113	Do not change
-Djava.agile.proxy.logfile=C:\Proxy.log	Uncomment this line to enable a Java debug window and log file.
[CheckOutDisk]	**Drive of work directory**
Syntax	`<drive>`
Default Value	D:
Configuration	Set to drive where work directory is located
[CheckOutPath]	**Path of work directory**

Option	Description
Syntax	<path>
Default Value	\AgileSW\Work
Configuration	Set to path where work directory is located
[LogFileDir]	**Drive & path of temp directory**
Syntax	<drive><path>
Default Value	D:\AgileSW\Temp
Configuration Options	Set to drive and path where temp directory is located
[AgileViewFileCustomScript]	**Drive, path & name of a executable file which will be executed to generate a customized view file to be saved**
Syntax	<drive><path><name><extention>
Default Value	D:\AgileEC\acw\Server\Scripts\ViewFileCustom.bat
Configuration Options	Set to drive and path where the executable file is located. In special cases this file will not be executed (see descriptions below).
[AgileURL]	**URL and Port of the Agile9 server**
Syntax	http://<server>:<port>/<Agile file name>
Default Value	http://agileserver:8888/Agile http://agileserver:8888/Agile
Configuration Options	Set to a dedicated port of a server machine where the Agile server software is located
[Agile9CreateDocument]	Defines the property mapping from SolidWorks to Agile, when the Documents are saved into Agile using the **Save** command, for the first time.

Option	Description
Configuration Options	Each mapping consists of a pair of objects. The left side of the pair defines the name of an Agile attribute that is the target of a property that is derived from the SolidWorks Model. The right side of the pair defines the SolidWorks property name. There are several configuration options for the right side of the pair that define what kind of data should be extracted from SolidWorks. Each right side attribute consists of two or three sections. All SolidWorks mappings begin with 3DCADTable. The second section can define system attributes. Possible values include: ▫ **FileStamp** – Timestamp (in seconds) ▫ **ModelPathOnly** – Directory where the CAD file is stored when saved to Agile, e.g. D:\CAD_file\Housing ▫ **ModelName** – Name of SolidWorks file with extension, e.g., BOLT.SLDPRT ▫ **ModelVersion** – SolidWorks version, e.g., SW2005 ▫ **ModelConfigurationName** – For configured parts/assemblies, the name of the configuration ▫ **ModelTitle** – Name of SolidWorks model without file extension, e.g., BOLT ▫ **ModelExtension** – SolidWorks Model type, e.g., SLDPRT, SLDASM, SLDDRW Values of the format 3DCADTable.Property.[value], where [value] is the name of a SolidWorks custom property such as Description or PartNumber. The following are some example mappings for a SolidWorks part called housing.sldpart with a custom property called Material with a value of Aluminum: ▫ CAX_FIL_NAME = 3DCADTable.ModelName ▫ DESCRIPTION = 3DCADTable.ModelTitle ▫ MATERIAL = 3DCADTable.Property.Material In this example, the Agile description is “housing”. An attribute in Agile called CAX_FIL_NAME has the value “housing.sldasm” and an Agile attribute called Material has the value “Aluminum”. The name used for the Agile attribute on the left side of the mapping is arbitrary. The actual attribute that is targeted for mapping is defined in the file CAXClient.xml, which is explained in the EC Client Configuration section. There is one special value that is used on the left side of these mappings. You use the value CAX_NEW_NUMBER to represent the Number field that will be assigned to newly created Documents.

Option	Description
[Agile9UpdateDocument]	Defines the property mapping from SolidWorks to Agile, when the Documents are saved into Agile using the **Save** command, after the first time. Configuration options are the same as [Agile9CreateDocument].
[Agile9UpdateItem]	Defines the property mapping from Agile to SolidWorks, when Items are created using the Save command
Configuration Options	See at [Agile9UpdateDocument] section There is one special value that is used on the left side of these mappings. You use the value ITEM to represent the Number field that will be assigned to newly created Parts.
[Agile9UpdatedItemConfigured]	Defines the property mapping from Agile to SolidWorks, when Items are created or updated using the Create Item/BOM command, and the Items are marked as Configured (see Agile9Configuration section)
Configuration Options	see the [Agile9UpdateDocument] section There is one special value that is used on the left side of these mappings. You use the value ITEM to represent the Number field that will be assigned to newly created Parts.
[Agile9CheckinDocument]	Defines the property mapping for file attachements, when the files are checked in during the **Save** command.
Configuration Options	System parameters - do not change
[Agile9UpdateProperties]	Defines the property mapping from Agile to SolidWorks, when using the Update Properties command manually.
Configuration Options	Each mapping consists of a pair of objects. The left side of the pair defines the name of a SolidWorks property that is being set. The right side of the pair defines the Agile attribute. For details of the mapping see the section " Mapping Options for Update Properties Sections"
[Agile9SaveUpdateProperties]	Defines the property mapping from Agile to SolidWorks which occurs automatically during the **Save** command
Configuration Options	Each mapping consists of a pair of objects. The left side of the pair defines the name of a SolidWorks property that is being set. The right side of the pair defines the Agile attribute. For details of the mapping see the section " Mapping Options for Update Properties Sections"

Option	Description
[Agile9LoadUpdateProperties]	Defines the property mapping from Agile to SolidWorks, which occurs automatically during the Load command
Configuration Options	Each mapping consists of a pair of objects. The left side of the pair defines the name of a SolidWorks property that is being set. The right side of the pair defines the Agile attribute. For details of the mapping see the section " Mapping Options for Update Properties Sections"
[Agile9UpdateTitleBox]	Defines the property mapping from Agile to SolidWorks, when the properties of a drawing are updated using the **Update Title Block** command
Configuration Options	Each mapping consists of a pair of objects. The left side of the pair defines the name of a SolidWorks property that is being set. The right side of the pair defines the Agile attribute. For details of the mapping see the section " Mapping Options for Update Properties Sections"
[Agile9Configuration]	Setting that control how you identify configured parts
Syntax	`ConfigProperty = Configured` `ConfigPropertyValue = YES`
Configuration Options	ConfigProperty is the name of the SolidWorks Custom Property that is used to identify configured parts. ConfigPropertyValue is the value that must be set for this property, to indicate that this part is to treated as containing multiple part configurations. When this value is set for a part or assembly, each different configuration is treated as a unique part, when generating the Part BOM with the Create Item /BOM command.
[Agile9PartViewFileExtensions]	Sets the allowable viewable file formats for parts, as created during the Save command
Syntax	`(x_t$)\|(x_b$)|(igs$)\|(step$)|(sat$)\|(stl$)|(wrl$)\|(eprt$)|(pdf$)\|(u3d$)|(3dxml$)\|(xaml$)|(cgr$)\|(jpg$)|(hcg$)\|(hsf$)|(tif$)`
[Agile9AssemblyViewFileExtensions]	Sets the allowable viewable file formats for assemblies, as created during the Save command
Syntax	`(x_t$)\|(x_b$)|(igs$)\|(step$)|(sat$)\|(stl$)|(wrl$)\|(eprt$)|(pdf$)\|(u3d$)|(3dxml$)\|(xaml$)|(cgr$)\|(jpg$)|(hcg$)\|(hsf$)|(tif$)`
[Agile9DrawingViewFileExtensions]	Sets the allowable viewable file formats for drawings, as created during the Save command

Option	Description
Syntax	(dxf$)\|(dwg$)|(edrw$)\|(jpg$)|(pdf$)\|(tif$)
[Agile9TemplatePath]	Sets the path to where template files are stored, for use by the New command
Syntax	<drive><path>
Configuration Options	The designated path will be scanned for *.prtdot, *.asmdot, and *.drwdot files, and these files will be made available as templates within the New command.
[Agile9Units]	Sets the default units for the New command
Syntax	Millimeters | Inches
[Agile9Renaming]	Activates the filename renaming process during the Load command
Syntax	0 | 1
Configuration Options	0 = Do not rename files during the Load process 1 = Rename files during the Load process, so that the filename matches the Agile Number field or a customized text string. This is used to support both the initial rename use case and the “save as” use case. For details of how to customize the filename see the file acwCustomer.tcl
[SaveAllConfigurationsToEDrawing]	Sets whether just the active configuration or all configurations are saved to eDrawing files.
Syntax	0 | 1
Configuration Options	0 = Save only the active configuration to the eDrawing file 1 = Save all configurations to the eDrawing file
[SWTraverseMode]	Sets the type of structure traversal algorithm is used by the connector.
Syntax	0 | 1
Configuration Options	0 = Use the old and slow traverse mode from the SolidWorks standard API 1 = Use the new and fast traverse mode from the SoldWorks DocumentManager API

Mapping Options for Update Properties Sections - SolidWorks

Multiple sections of the 3DCADMapping.ini file, as listed above, are used to define mappings from

Agile to SolidWorks. For standard attributes the format of this section is:

CAD Parameter = <Source Table>_Field.Format

For example:

Agile_Des = Title Block_Description.ToUpper

Where the left side value is the name of the SolidWorks parameter to be updated, and the right side is the Agile attribute value to be used as the source, as follows:

Section	Represents	Example
<Source Table>	Agile tab name	Title Block
Field	Agile attribute name	Description
Format	Text processing	ToUpper

For history and change history attributes, which are arranged in a table, the format of this section is:

CAD Parameter = <Filter Table>_Field,<Filter Value>,<Filter>,<Source Table>_Field.Format

For example:

Agile_CreUser = History_Action,Create,first,History_User.None

HIS_RELDATE_1 = Change History_Status,Released,last,Change History_Rel Date_int.Date01

Where the left side value is the name of the SolildWorks parameter to be updated, and the right side specifies how to find the desired row and column in the table below:

Section	Represents	Example
<Filter Table>	Agile tab name to search	Title Block
Field	Desired column to search	Action
<Filter Value>	Value to detect in the column	Create
<Filter>	Which row to select, with these options: first first+n n=integer value last last-n n=integer value	first
<Source Table>	Agile tab name to retrieve value from	History
Field	Desired column to retrieve value from	User
Format	Text processing	None

Options for "Format"

The Format string allows you to perform additional processing on the text string being passed back into CAD. This includes predefined formats and general TCL format procedures.

Predefined formats

Format	Description
None	no processing
ToLower	convert the value to lower case
ToUpper	convert the value to upper case
Range-x-y	substring of the value from index x to index y (y may be numeric or "end")
Date01	convert int dateformat to "%d.%m.%y %H:%M:%S" example: 01.01.2007 00:00:00
Date02	convert int dateformat to "%d.%m.%Y" example: 01.01.2007
Date03	convert int dateformat to "%d.%m.%y" example: 01.01.07
Date04	convert int dateformat to "%d-%m-%y" example: 01-01-07
Date05	convert int dateformat to "%m/%d/%y" example: 01/01/07
Date06	convert int dateformat to "%d-%b-%y" example: 01-Jan-07
Prefix<str>	append a prefix <str> to the value
Suffix<str>	append a suffix <str> to the value

TCL format procedures

Any registered (tclIndex) TCL procedure that gets the current value as input and returns the formatted string. For instance:

```
proc MyFormat { value } {
```

```
    set formatedvalue $value
    return $formatedvalue
}
```

Mapping Part Attributes

In addition to mapping attributes from the CAD Document back into CAD, you can map attributes from the corresponding Part object that has been associated to the Document using the Create Item/BOM command. In order to specify a Part attribute, simply prefix the attribute value with "PART:". This example shows mapping both the Document Number and Part Number into CAD:

Agile_DocId = Title Block_Number.None

Agile_PartId = PART:Title Block_Number.None

Note If you do NOT have any Part attributes mapped back into CAD, you can improve performance of the connector somewhat by using the following setting in the CAXClient_Designs.xml file:

Note <clientConfig name="skipGetItemProperties" value="1"/>

Note By setting this to 1, the logic to check associated Parts is skipped, thereby improving performance. Design attributes are still exchanged.

Master Switch for Update Properties on Save

The Update Properties on Save functionality has a "master switch" which can be used to turn this functionality on or off. By turning it off, performance of the Save process can be improved, which is most noticable for CAD models with many components. Of course when it is turned off, the trade-off is that no properties can be exchanged during the Save process and this must be done manually by the user, using the Update Properties command. The master switch is contained in the menu definition file PlmSWAddin.xml as described in the next section. The switch is accomplished by having two sets of menu callbacks, one that includes the update properties on save, and the other which doesn't:

Includes Update Properties on Save

Save: AgileSave202

QuickSave: AgileSave203

Save Session: AgileSave204

Does NOT include Update Properties on Save

Save: AgileSave205

QuickSave: AgileSave206

Save Session: AgileSave207

Simply edit the menu definition file with a text editor, and change all instances of the callback names to the proper number (e.g. change AgileSave202 to AgileSave205, or vice versa).

Controlling Custom vs. Configuration-specific Properties

In the following sections:

- [Agile9UpdateDocument]
- [Agile9UpdateItem]
- [Agile9UpdateItemConfigured]

You can use the "Custom_" and "ActiveConfiguration_" modifiers to control whether the properties are coming from Custom or Configuration-specific Properties.

For example:

```
ITEM  = 3DCADTable.Property.Custom_PartNumber
```

Sets the Part number attribute using a Custom property called "PartNumber"

```
DESCRIPTION = 3DCADTable.Property.ActiveConfiguration_Description
```

Sets the Description attribute from a configuration-specific property called "Description".

If you omit the "Custom_" or "ActiveConfiguration_" modifier, it defaults to configuration-specific. Note also that SolidWorks properties are case-sensitive!

Modifying the Agile Menu Definition

There is another configuration file, which controls the layout of the Agile menu. The file is named **PlmSWAddin.xml** and is located in the **AgileEC\acw\Server\Scripts** directory. Since this file is located within the SolidWorks Connector installation on the client machine, it is possible to customize menu options on a per-machine basis, although typical usage is to have a common configuration file within a given site. When changes are made to the configuration file, it is necessary to exit and re-start SolidWorks in order to use the new menus.

Configuration of the menus is limited to:

- Removal of unneeded commands and menus
- Renaming of commands and menus
- Restructuring of commands and menus
- Addition or removal of menu separators

The portion of the file which can be configured is within the <CaxMenu_EN> tags (for English language). Within this section you will see four sets of tags, which contain the menu entries for the following situations in SolidWorks:

<Base> - Menus when no SolidWorks component is active

<Part> - Menus when a single Part is active

<Assembly> - Menus when an Assembly is active

<Drawing> - Menus when a Drawing is active

For example, the <Base> section looks like this when you call up the file in an editor.

However, note that for each of these lines, there is additional text if you scroll over to the right side. Make sure when cutting and pasting lines, that you get the entire line. The portion of the lines that you would need to edit is limited to what is shown above (i.e. do not edit any part of the text further to the right).

The editable sections of the file are described as follows:

<menu…> tags	Defines the type of menu entry
Syntax	menu – Indicates a menu or sub-menu entrymenu item – Indicates a menu command
type	Distinguishes the entries for each section.
Syntax	type = “<number>” where <number> equals: 0 = Base menu1 = Part menu 2 = Assembly menu 3 = Drawing menu
text	Defines the menu text and hierarchy level
Syntax	For **menu**: text = “<menu>@<next-higher-menu>” For **menuitem**: text = “<command>@<menu>@<next-higher-menu>”
Examples	Example of first-level menu and a command within it: <menu type = "0" text = "Agile" <menuitem type = "0 text = "Connect@Agile" Example of second-level menu and a command within it: <menu type = "0" text = "New@Agile" <menuitem type = "0" text = "Part@New@Agile"

For turning on or off the update properties on save please modify the following setting:

Removing Commands and Menus

It can be useful to remove commands from the menus, for example to eliminate commands that do not fit with your specific business processes. To remove a command from the menus, simply delete the entire line containing the command that you want to remove. Remember to delete it from all menus that it appears in (<Base>, <Part>, etc.). You can also remove entire sub-menus, but make sure to also remove or restructure all commands within the sub-menu.

Renaming Commands and Menus

Commands and menus can be renamed simply by changing the text values. Remember to rename them in all menus that they appear in (<Base>, <Part>, etc.). If you rename a menu, make sure to also change the menu portion of the text field in each command in the menu.

Restructuring Commands and Menus

Commands can be restructured, for example to move them in or out of a sub-menu. To move a command from a sub-menu to the next higher menu, change the text field of the command to remove the reference to the sub-menu. To move a command into a sub-menu, do the reverse. You can add your own sub-menus, if necessary, by adding additional menu lines.

Adding or Removing Menu Separators

Menu separators can easily be added or removed. Separators are defined by lines in the file such as this:

```
<menuitem type = "1" text = "@Agile" position = "-1" callback =
"Separator" enablemethod = "" hint = "" />
```

The difference between this menuitem, and one for a real command, is that the text entry has no command text before the first @ sign, and the callback entry is "Separator". You can add or remove any of these separator lines to control the positioning of separators in the menus.

Chapter 6

Installing and Configuring Unigraphics NX Connector

This chapter includes the following:

This section describes setting up the connection between your Unigraphics NX CAD application and Agile Engineering Collaboration. The main steps are:

- Extract files from Unigraphics NX Connector zip file
- Extract files from EC Client zip file
- Configure PLM API for WAN mode (optional)
- Edit some parameters in the startup file
- Install proper **AgileAPI.jar** file
- Create shortcut to new startup file

The installation requires the following file:

acuNNNN.zip – Main installation package, where NNNN is the release level.

Performing the installation steps described here will enable the Agile menu to appear within Unigraphics NX. In order to have a completely functional integration, you must also:

- Perform the core Agile configuration as described in the Administration section of the document Agile Engineering Collaboration Client.
- Configure desired Unigraphics NX Connector parameters as described in the Unigraphics NX Connector Administration on page 67 section on page 63.

Note	The Unigraphics NX Connector supports Unix platforms in addition to Windows. Only instructions for Windows are provided below, but the process for Unix is very similar, using equivalent Unix commands and directory paths. Please consult with Oracle Consulting Services if you need assistance with installation.

Extracting Files for Connector

Extract the installation file to the folder location **D:\AgileEC** or **C:\AgileEC** for a single-drive system. When you unzip, make sure that you retain the folder paths from the zip file. When the files are unzipped, you should see a folder named **acu**, which contains the Connector installation.

Extracting Files for EC Client

Extract the EC Client installation file to a temporary location. After extraction you will see 5 connector directories (acc, ace, acp, acu, acw) plus a jar directory.

- Copy the entire jar directory inside the **\AgileEC\acw** directory.
- From the temporary **acu\com** directory, copy the file CaxClient.bat to the **\AgileEC\acu\com** directory.
- From the temporary **acu\jar\Agile9** directory, copy the file CaxClient_Designs.xml to the **\AgileEC\acu\jar\Agile9** directory.

Configure PLM API for WAN Mode

PLM API is a high-performance web services API used to support operations across wide area network (WAN). This is an optional capability that is only necessary if you are going to be supporting MCAD connectors at remote sites (i.e. locations distant from your Agile application server). If you do not need this capability, skip to the next step.

- Log in with the Agile 9 web client to the server that will be used with the MCAD connectors. Use an account that has privileges to create File Folder objects.
- Create the following three File Folder objects, using the number listed in the first column, and then attach the file listed in the second column. Check In the File Folder following attachment. The files can be found under the jar/agile9/server folder. Note that depending on your system settings, you might need to temporarily modify the admin settings for File Folder objects to allow using an arbitrary Number field.

File Folder number	File to attach
PLMAPI_ASYNC	plm-api-server.xml
PLMAPI_CONFIG	plm-api-config.properties
PLMAPI_CONNECTOR	plm-api-sdk.xml

- Copy the file plm-api-server.jar from jar/agile9/server to the WSX extensions folder of the target Agile 9 server. Usually this folder is located in {agile_home}/integration/sdk/extensions).
- The EC Client configuration file must be edited to turn on PLM API mode. To do this, bring up either the CAXClient_Documents.xml or CAXClient_Designs.xml file (depending on which data model you are using) in a text editor. Change the following line as follows (change from 0 to 1). Save the file.

```
<clientConfig name="usePlmApi" value="1"/> <!-- Use PLM API Web
Services 1=enabled, 0=disabled -->
```

Settings	Necessary settings for using PLM API
DFM_JAR	set DFM_JAR=%CAX_ROOT%\jar\agile9\axis.jar;.....(etc.) **rem** set DFM_JAR=
JAVA_HEAP_SIZE	rem set JAVA_HEAP_SIZE=-Xms128m -Xmx128m set JAVA_HEAP_SIZE=-Xms128m -Xmx1024m

Editing the Startup File

Open the file **\AgileEC\acu\com\acu_start.bat** in a text editor. Edit the values as described in table below to match your system configuration.

Sample values	What this command specifies
set ECU_UGV=nx4	Unigraphics NX version (nx3, nx4, nx5)
set ECU_LANG=eng	User interface language (eng, ger)
set UGII_ROOT_DIR=D:\CAD\UGNX2\UGII	The root directory of the Unigraphics NX installation directory
set ug_start=D:\CAD\UGNX2\UGII\ugraf.exe	The complete path of the executable file to be started when using Unigraphics NX
set cax_usr_home=D:	Directory for saving the temporary data files (suggested value: **D:\AgileUG or C:\AgileUG**)

Creating a Shortcut to the Startup File

In order to run Unigraphics NX with the Connector, you must run using the startup file **\AgileEC\acu\com\acu_start.bat**. To make this more convenient, you may want to create a Windows shortcut to this file, either on your Desktop or in your Quick Launch bar.

Verify that the Unigraphics NX Connector is working by double-clicking on your shortcut to launch Unigraphics NX. You should see an Agile menu appear in the main menu bar.

Installing the AgileAPI.jar file

Important If this step is not done correctly, the connector may appear to be functioning normally but data corruption may occur!

The correct **AgileAPI.jar** file, matching the specific Agile service pack level, must be installed in the directory **\AgileEC\acu\jar\agile9.**

- Search for the file **AgileAPI.jar** within your site's Agile server installation (such as **C:\Program Files\Agile**).
- Copy this file to **\AgileEC\acu\jar\agile9**, overwriting the file already there.
- If you are unable to locate this file, please contact Agile Support.

Installing on Additional Computers

Once the Unigraphics NX Connector has been installed and configured on one machine, you can install on other machines simply by copying the entire **\AgileEC\acu** folder structure. This works as long as the machines are configured the same in terms of their Unigraphics NX setup, Java setup, etc.

Unigraphics NX Connector Administration

This section provides a complete summary of configuration options available for the Unigraphics NX connector. Once the basic installation has been done following the instructions in the previous section, you can refer here for details of all possible settings.

Note that in addition to the configuration files listed here, the EC Client must be additionally configured to provide complete operation of the Unigraphics NX Connector. See the EC Client Configuration Options section for details.

Table: List of all Configuration Files for the Unigraphics NX Connector

Configuration File	Purpose	Location
acu_start.bat	Startup file with system parameters	AgileEC\acu\com
Ecu.ini	Mapping and configuration	AgileEC\acu\ini\Agile9
ecu.men	Menu definition	AgileEC\acu\ini\agile9\<lang>\startup

Note Configuration files typically change content between connector releases. When upgrading to a new release, please incorporate your site's configuration settings into the new version of the configuration files. Failure to do so will cause unpredictable behavior of the connector.

Mapping File Ecu.ini

This is the main file for controlling the behavior of the Unigraphics NX Connector. This file is structured in several sections. The first line of a section starts with a left square bracket followed by a space and its name again followed by a space and the right square bracket. Each section starts with the section name. A comment line starts with the # sign.

Note Please make sure not to leave blank lines when editing the file.

The following tables provide the details of each section of Ecu.ini:

Table: Description of all sections in the Ecu.ini file

Section name	Description
Initialize	Common necessary switches to enable a reasonable behavior of the connector
LoadProperties	Describes the assignment of Agile attributes to UG-part properties, when using the **Update Properties** command, and also automatically during the **Save** command
SaveProperties	Dscribes the assignment of UG properties to Agile attributes, during the **Save** command
FillFrame	Describes the attributes which will be transferred from Agile into the drawing title block, when using the **Update Title Block** command
FillFrameHistory	Describes the History and Change History attributes which will be transferred from Agile into the drawing title block, when using the Update Title Block command

Section name	Description
JT	Describes the settings for generation of JT viewable files
SaveViewable	Describes the available options for save with a viewable file and which custom script is called or execution
CGM	Describes the assignment of a pen-definition to a specific pen when generating a Computer-Graphic Metafile, obsolete with NX3

Table: Details of the Initialize section

Parameter name in section [Initialize]		Parameter values	Description
JNI_DEBUG	=	**1**	Enables proxy.log for debug of CAD input and output, disabled by commenting out using a # sign at the beginning of the line
LoadAttributes	=	**1..**/..0	Set part attributes defined in this section after loading a part in UG
LoadFrame	=	**1..**/..0	Fill the frame title box text notes after loading a drawing in UG
EcuChangeFrame	=	**1..**/..0	Automatic replacement of the old frame with a frame template on local disc
EcuEmptyText	=	-	Placeholder for empty text notes in the drawing title block
DefaultUnits	=	**MILLIMETERS**/INCHES	The default UG part unit used in **New** command dialog
CheckOutOnModify = 0	=	1../..0	Automatic attachment checkout on UG
SaveOnDiscThenToAgile = 0	=	1../..0	Save all to disc first then to agile PLM
RenameOnInitialSave = 0	=	**1..**/..0	Rename on first save to agile
RenameOnSaveAs = 1	=	**1..**/..0	Rename on save as
AcuFileNameProcedure = AcuCustomProcedure	=	**1..**/..0	Customer file name generation

The section [LoadProperties] describes the assignment of Agile attributes to UG properties. There are three situations where these attributes will be assigned:

- When the user picks the **Update Properties** command
- Automatically during the **Save** command
- Automatically during the **Load** command, if LoadAttributes = 1

Table: Load Attributes Examples

UG-Part Attribute in section [LoadProperties]		Object.Field
PLM_DOC_NUMBER	=	Doc.NUMBER

UG-Part Attribute in section [LoadProperties]		Object.Field
CAX_MULTI	=	Doc.CAX_MULTI

Mapping Options for Load Properties Sections

For standard attributes the format of the [LoadProperties] section is:

CAD Parameter = <Source Table>_Field.Format

For example:

Agile_Des = Title Block_Description.ToUpper

The left side value is the name of the UG property to be updated.

The right side can be either the symbolic attribute name from the **CaxClient.xml** file (such as NUMBER, DESCRIPTION, etc.) or any Agile attribute represented as follows:

Section	Represents	Example
<Source Table>	Agile tab name	Title Block
Field	Agile attribute name	Description
Format	Text processing	ToUpper

For history and change history attributes, which are arranged in a table, the format of this section is:

CAD Parameter = <Filter Table>_Field,<Filter Value>,<Filter>,<Source Table>_Field.Format

For example:

Agile_CreUser = History_Action,Create,first,History_User.None

HIS_RELDATE_1 = Change History_Status,Released,last,Change History_Rel Date_int.Date01

Where the left side value is the name of the UG property to be updated, and the right side specifies how to find the desired row and column in the table below:

Section	Represents	Example
<Filter Table>	Agile tab name to search	Title Block
Field	Desired column to search	Action
<Filter Value>	Value to detect in the column	Create
<Filter>	Which row to select, with these options: first first+n n=integer value last last-n n=integer value	first

Section	Represents	Example
<Source Table>	Agile tab name to retrieve value from	History
Field	Desired column to retrieve value from	User
Format	Text processing	None

Options for "Format"

The Format string allows you to perform additional processing on the text string being passed back into CAD. This includes predefined formats and general TCL format procedures.

Predefined formats

Format	Description
None	no processing
ToLower	convert the value to lower case
ToUpper	convert the value to upper case
Range-x-y	substring of the value from index x to index y (y may be numeric or "end")
Date01	convert int dateformat to "%d.%m.%y %H:%M:%S" example: 01.01.2007 00:00:00
Date02	convert int dateformat to "%d.%m.%Y" example: 01.01.2007
Date03	convert int dateformat to "%d.%m.%y" example: 01.01.07
Date04	convert int dateformat to "%d-%m-%y" example: 01-01-07
Date05	convert int dateformat to "%m/%d/%y" example: 01/01/07
Date06	convert int dateformat to "%d-%b-%y" example: 01-Jan-07
Prefix<str>	append a prefix <str> to the value
Suffix<str>	append a suffix <str> to the value

TCL format procedures

Any registered (tclIndex) TCL procedure that gets the current value as input and returns the formatted string. For instance:

```
proc MyFormat { value } {
  set formatedvalue $value
  return $formatedvalue
}
```

Mapping Part Attributes

In addition to mapping attributes from the CAD Document back into CAD, you can map attributes from the corresponding Part object that has been associated to the Document using the Create Item/BOM command. In order to specify a Part attribute, simply prefix the attribute value with "PART:". This example shows mapping both the Document Number and Part Number into CAD:

Agile_DocId = Title Block_Number.None

Agile_PartId = PART:Title Block_Number.None

[SaveProperties]

Icon Lock Localization Description BasedOn Modified ModifiedBy Code Version State

2 [SaveProperties] Normal Template 16.01.2008 06:22:43 trana 27786 1

Icon Lock Localization Description BasedOn Modified ModifiedBy Code Version State

2 [SaveProperties] Normal Template 16.01.2008 06:22:43 trana 27786 1

The section [SaveProperties] describes the assignment of UG properties to Agile attributes, in a variety of situations. These situations include:

- Creating Documents with the Save command (Create.Doc)
- Updating Documents with the Save command (Update.Doc)
- Creating Parts with the Create Item/BOM command (Create.Item)
- Updating Parts with the Create Item/BOM command (Update.Item)

Table: Save Properties Examples

Mode.Object.Field in section [SaveProperties]		Type.Attribute.Format
Create.Doc.CAX_NEW_NUMBER	=	System.FileName.ToUpper
Create.Doc.CAX_CRE_SYSTEM	=	String.Unigraphics.None
Create.Doc.CAX_FIL_NAME	=	System.ObjectName.None
Update.Doc.CAX_FIL_NAME	=	System.ObjectName.None
Update.Doc.CAX_CRE_SYSTEM	=	String.Unigraphics.None
Create.Item.ITEM	=	System.ObjectName.None
Update.Item.ITEM	=	System.ObjectName.None

Table: Type Attribute Values in SaveProperties Section

Type Attribute Values in section [SaveProperties]	Description
Attribute.<Attributename>	Returns the value of the UG part attribute that matches the name in <Attributename>, example: Attribute.PART_ID.None -> delivers the value in PART_ID without additional string conversion
String.<fixed String>	Sets the string given in <fixed String> as default setting for the mapped field
System.Timestamp	Returns the current Timestamp of the part file on disk.
System.Object.Name	Returns the current Name of the part file without path or extension.
System.FileName	Returns the current Name of the part file without path but with extension.
System.FullName	Returns the current Name of the part file including path and extension.
System.Version	Returns the current UG-Version

Table: String Formatting Options in SaveProperties Section

Format string in section [SaveProperties]	Description
ToUpper	Converts to upper case
ToLower	Converts to lower case
Range-<idx1>-<idx2>	Range of the string from position idx1 to idx2, example: System.ObjectName.Range-2-3
Prefix	Prefix to be added in front of the string, example: System.ObjectName.PrefixPRT
Suffix	Suffix to append to the string, example: System.ObjectName.SuffixPRT
None	No string conversion

The sections [FillFrame] and [FillFrameHistory] describe the source and the format for the content which will be transferred from Agile and displayed in the drawing title block. The structure of such a line is TextNoteName = Object.Field.

Table: Fill Frame Examples

Example			Description
ZVS1:20:1	=	Doc.Number	Line length is 20 and it is a multi line attribute

Example			Description
ZVS1:20:0	=	Doc.Number	Line length is 20 and will be cut after 20th char and it is no multi line attribute
ZVS1	=	Doc.Number	Standard line length and no multi line attribute

The formatting of the sections [FillFrame] and [FillFrameHistory] follow the syntax described above in the section Mapping Options for Load Properties Sections.

The section [JT] describes how JT format viewable files are generated. The options are:

Monolithic = 1 creates one JT container file with all components inside

Monolithic = 0 creates one JT file for each component

CheckInComponents = 1 works for Monolithic=0 only and copies all JT file of any component into the PLM vault

The section [SaveViewable] describes the available options for saving viewable files with the Save command. The structure of such a line is:

Format = Scriptname

Table: SaveViewable Examples

Format in section [SaveViewable]		Scriptname
CGM	=	AcuSaveCGM.tcl
PDF	=	AcuSavePDF.tcl

The section [CGM] describes the assignment of a pen-definition to a specific pen when generating a Computer-Graphic Metafile. The structure of such a line is:

```
open   =   number of the format description.
```

Table: CGM Examples

Pen1	=	1
Pen1	=	2
Pen1	=	3
Pen1	=	4
Pen1	=	5
Pen1	=	6
Pen1	=	7
Pen1	=	8
Pen1	=	9
Pen1	=	10
Pen1	=	11

Pen1	=	12
Pen1	=	13
Pen1	=	14
Pen1	=	15
PenSelection	=	1
TextRepresentation	=	2

Menu Definition File ecu.men

The Menufiles of the Connector are implemented using UG-Menuscript language. The definition file is named ecu.men and is located in the language specific subdirectory of your Connector, for instance the English Menufile is located in AgileEC/acu/ini/agile9/eng/startup. See *UG-NX* documentation for details about UG-Menuscript.

Chapter 7

Installing and Configuring CATIA V5 Connector

This chapter includes the following:

This section describes setting up the connection between your CATIA V5 CAD application and Agile Engineering Collaboration. The main steps are:

- Extract files from CATIA V5 Connector zip file
- Extract files from EC Client zip file
- Configure PLM API for WAN mode (optional)
- Edit some parameters in the configuration file
- Edit some parameters in the environment file
- Install proper **AgileAPI.jar** file
- Create shortcut to new startup file

The installation requires the following file:

accNNNN.zip – Main installation package, where NNNN is the release level

Performing the installation steps described here will enable the Agile toolbars to appear within CATIA V5. In order to have a completely functional integration, you must also:

- Perform the core Agile configuration as described in the Administration section of the document *Agile Engineering Collaboration Client*.
- Configure desired CATIA V5 Connector parameters as described in the section CATIA V5 Connector Administration on page 79 on page 75.

Note	The CATIA V5 Connector supports Unix platforms in addition to Windows. Only instructions for Windows are provided below, but the process for Unix is very similar, using equivalent Unix commands and directory paths. Please consult with the Oracle Consulting Services if you need assistance with installation.

Extracting Files for Connector

Extract the installation file to the folder location **D:\AgileEC\acc** or **C:\AgileEC\acc** for a single-drive system. When you unzip, make sure that you retain the folder paths from the zip file. When the files are unzipped, you should have a folder named **acc**, which contains the Connector installation.

Extracting Files for EC Client

Extract the EC Client installation file to a temporary location. After extraction you will see 5 connector directories (acc, ace, acp, acu, acw) plus a jar directory.

- Copy the entire jar directory inside the **\AgileEC\acc** directory.
- From the temporary **acc\com** directory, copy the file CaxClient.bat to the **\AgileEC\acc\com** directory.
- From the temporary **acc\jar\Agile9** directory, copy the file CaxClient_Designs.xml to the **\AgileEC\acc\jar\Agile9 d**irectory.

Configure PLM API for WAN Mode

PLM API is a high-performance web services API used to support operations across wide area network (WAN). This is an optional capability that is only necessary if you are going to be supporting MCAD connectors at remote sites (i.e. locations distant from your Agile application server). If you do not need this capability, skip to the next step.

- Log in with the Agile 9 web client to the server that will be used with the MCAD connectors. Use an account that has privileges to create File Folder objects.
- Create the following three File Folder objects, using the number listed in the first column, and then attach the file listed in the second column. Check In the File Folder following attachment. The files can be found under the jar/agile9/server folder. Note that depending on your system settings, you might need to temporarily modify the admin settings for File Folder objects to allow using an arbitrary Number field.

File Folder number	File to attach
PLMAPI_ASYNC	plm-api-server.xml
PLMAPI_CONFIG	plm-api-config.properties
PLMAPI_CONNECTOR	plm-api-sdk.xml

- Copy the file plm-api-server.jar from jar/agile9/server to the WSX extensions folder of the target Agile 9 server. Usually this folder is located in {agile_home}/integration/sdk/extensions).
- The EC Client configuration file must be edited to turn on PLM API mode. To do this, bring up either the CAXClient_Documents.xml or CAXClient_Designs.xml file (depending on which data model you are using) in a text editor. Change the following line as follows (change from 0 to 1).

Save the file.

```
<clientConfig name="usePlmApi" value="1"/> <!-- Use PLM API Web
Services 1=enabled, 0=disabled -->
```

- The EC Client startup file must also be edited to enable PLM API mode. To do this, bring up acc\com\caxclient.bat in a text editor. Change the following lines by appropriately uncommenting and commenting the lines. Note that lines with "rem" in front of them are commented out and inactive:

Settings	Necessary settings for using PLM API
DFM_JAR	set DFM_JAR=%CAX_ROOT%\jar\agile9\axis.jar;…….(etc.) **rem** set DFM_JAR=
JAVA_HEAP_SIZE	rem set JAVA_HEAP_SIZE=-Xms128m -Xmx128m set JAVA_HEAP_SIZE=-Xms128m -Xmx1024m

Creating a Shortcut to the Startup File

In order to run CATIA V5 with the Connector, you must run using the startup file **\AgileEC\acc\com\cv5.cmd**. To make this more convenient, you may want to create a Windows shortcut to this file, either on your Desktop or in your Quick Launch bar.

Verify that the CATIA V5 Connector is working by double-clicking on your shortcut to launch CATIA V5. You should see the Agile toolbars appear in the CATIA user interface.

Editing the Configuration File

Open the file **\AgileEC\acc\com\Acc.cfg** in a text editor. Edit the values to match your system configuration.

Sample values	What this command specifies
`AccUserRoot=f:\AccUser`	Root directory for user data and files (suggested value:**D:\AgileCAT** or **C:\AgileCAT**)
`AccTemplateFolder=f:\acc2-work\templates\`	Template folder for use with the "New" command. A default template folder is provided at **D:\AgileEC\acc\templates**
`CatiaEnv=CATIA_ECC`	Your CATIA environment. This selects the CATIA environment file to use
`AccJava=D:\j2sdk1.4.1_02\jre`	Path to Java Runtime Environment (Note: Avoid blanks or spaces in path.) Determine the required CATIA connector JRE version from the chart in Appendix C, and then select the appropriate path from the available ones provided in the acc installation.

Sample values	What this command specifies
`Csp=r13spx`	CATIA version (r14spx, r15spx, r16spx)
`CatiaBin=z:\Programme\DassaultSystemes\B13\intel_a\code\bin\CNEXT.exe`	Path to CATIA executable

Editing the Environment File

You must edit your CATIA environment file to put in the appropriate folder paths. The file to edit depends on your CATIA version and environment name. By default, the filename is **CATIA_ECC.txt**, and is located in the **\AgileEC\acc\bin\<os>\<agile_version>\<catia_version>** folder.

For example, **\AgileEC\acc\bin\intel_a\agile9\r13spx\CATIA_ECC.txt.**

Edit all folder paths within this file to match your system's CATIA installation.

Installing the AgileAPI.jar file

Important If this step is not done correctly, the connector may appear to be functioning normally but data corruption may occur!

The correct **AgileAPI.jar** file, matching the specific Agile service pack level, must be installed in the directory **\AgileEC\acc\jar\agile9.**

- Search for the file **AgileAPI.jar** within your site's Agile server installation (such as **C:\Program Files\Agile**).
- Copy this file to **\AgileEC\acc\jar\agile9**, overwriting the file already there.
- If you are unable to locate this file, please contact Agile Support.

Installing on Additional Computers

Once the CATIA V5 Connector has been installed and configured on one machine, you can install on other machines simply by copying the entire **\AgileEC\acc** folder structure. This works as long as the machines are configured the same in terms of their CATIA V5 setup, Java setup, etc.

CATIA V5 Connector Administration

This section provides a complete summary of configuration options available for the CATIA V5 connector. Once the basic installation has been done following the instructions in the previous section, you can refer here for details of all possible settings.

Note that in addition to the configuration files listed here, the EC Client must be additionally configured to provide complete operation of the CATIA V5 Connector. See the EC Client Configuration Options section for details.

Table: List of all Configuration Files for the CATIA V5 Connector

Configuration file	Purpose	Location
Acc.cfg	System configuration	AgileEC\acc\com
AccInitialize.ini	Configuration	AgileEC\acc\ini
AccCustomer9.ini	Mapping	AgileEC\acc\ini

Note Configuration files typically change content between connector releases. When upgrading to a new release, please incorporate your site's configuration settings into the new version of the configuration files. Failure to do so will cause unpredictable behavior of the connector.

Configuration File AccInitialize.ini

This is the main file for controlling the behavior of the CATIA V5 Connector. This file has a single [Initialize] section. A comment line starts with the # sign.

Note Please make sure not to leave blank lines when editing the file.

Table: [Initialize] Section Parameters

Parameter name in Section [Initialize]		Parameter values	Description
AccCustomerId	=	None	System setting (do not change)
AccLanguage	=	**English**	Language setting
AccMappingFile	=	Acc.ini	Mapping file name
AccCustomerFile	=	AccCustomer9.ini	Customer file name
AccMessages	=	AccMessages.ini	Messages file name
AccDebug	=	**1**../..0	Turns debug mode on (1) and off (0). A log file is written to the user's working directory.
AccAgileServerURL	=	http://agileserver:8888/Agile	Default URL that the EC Client will use to connect to the Agile Application Server

Parameter name in Section [Initialize]		Parameter values	Description
AccAgileUser	=	cax	Default user that is used to log in to Agile when the user chooses Connect from the CAD system
AccAgilePwd	=	agile	Default password to log in to Agile
AccDefaultClass	=	DOCUMENT	The value used here must agree with the value for defaultClass in CAXclient.xml. System setting (do not change)!
AccHelpPartIdent	=	ITEM	Name of CATIA V5 property used to identify models in the design that should not be included in the BOM. These objects are saved into Agile as Documents, but are filtered out when using the Create Item/BOM function.
AccHelpPartValue	=	NO	Value that the CATIA V5 property should be set to in order to activate the filter
AccAgileBackupId	=	AgileID	Indicates the field to use for re-associating a file to the correct Agile Document. This assignment tracks the Agile Document number.
AccAgileBackupName	=	AgileName	Indicates the field to use for re-associating a file to the correct Agile Document. This assignment tracks the Agile filename.
AccEnableRename	=	1	0 = files are not renamed 1 = files are renamed to match the Agile Number field or custom mapping
AccSchemeOfFileName	=	%	Format definition (in “C” style) used to define the CATIA filename
AccFileNameValues	=	NUMBER / CATIAFILE	Basis of the filename. Standard values are either NUMBER (Agile Document Number) or CATIAFILE (original filename)

Filename Creation

During the first Save into Agile, a new CATIA V5 filename can be created. In the file AccInitialize.ini are two variables that control this process:

- AccFilenameValues
- AccSchemeOfFileName

AccFilenameValues can contain a list of attributes from Agile either defined in the EC Client definition file or simply “CATIAFILE”. “CATIAFILE” means the usage of the original Catia file name. AccSchemeOfFileName is a format definition based on the “C” style.

#

```
AccSchemeOfFileName  = %s
```

AccFileNameValues = NUMBER

After checkin of a part to Agile, the object will be renamed to D00444.CATPart because D00444 is the number of the Agile document.

```
#
AccSchemeOfFileName  = %s
AccFileNameValues  = CATIAFILE
```

After checkin of a part to Agile the object will not be renamed.

```
#
AccSchemeOfFileName  = CAT-%s
AccFileNameValues  = NUMBER
```

After checkin of a part to Agile the object will be renamed to CAT-D00444.CATPart.

[Customer Functions] Section

To better support the ability for project-based customization of TCL scripting, entry points are now provided for TCL add-ins through the [CustomerFunctions] section in AccInitialize.ini.

[CustomerFunctions]

...

<EntryPoint> = <Customer specific procedure>

....

There are 7 predefined entrypoints:

1. CatiaScanTree-01
2. CatiaScanTree-02
3. CatiaScanTree-03
4. CatiaAccSaveToAgile-01
5. CatiaAccLoad-01
6. CatiaAccSave-01
7. CatiaAccUpdateFrame-01

Mapping File AccCustomer9.ini

This is the main file for controlling attribute mapping in the CATIA V5 Connector. This file is structured in several sections. The first line of a section starts with a left square bracket followed by a space and its name again followed by a space and the right square bracket. Each section starts with the section name. A comment line starts with the # sign.

Note Please make sure not to leave blank lines when editing the file.

The following table gives a description of all sections in **AccCustomer9.ini**, and the following tables provide the details of each section.

Table: Description of all sections in AccCustomer9.ini

Section name	Description
CatiaToAgile.DOCUMENT	This mapping section is used for assigning attributes when Documents using the **Save** command.
CatiaToAgileUpdate.DOCUMENT	This mapping section is used for assigning attributes when updating Documents using the Save command.
CatiaToAgile.FILEFOLDER	OBSOLETE
CatiaToAgile.ITEM	This mapping section is used for creating and updating Parts using the **Create Item/BOM** command.
AgileTo.Catia	Defines those Agile attributes that are saved automatically into all CATIA V5 files, during the **Save** command.
AgileTo.CATPart	Defines those Agile attributes that are saved automatically into CATIA V5 CATPart files, during the **Save** command.
AgileTo.CATDrawing	Defines those Agile attributes that are saved automatically into CATIA V5 CATDrawing files, during the **Save** command.
AgileTo.CATProduct	Defines those Agile attributes that are saved automatically into CATIA V5 CATProduct files, during the **Save** command.
AgileGetProperties.Catia	Defines those Agile attributes that are saved into all CATIA V5 files, when using the **Update Properties** command.
AgileGetProperties.CATPart	Defines those Agile attributes that are saved into CATIA V5 CATPart files, when using the **Update Properties** command.
AgileGetProperties.CATDrawing	Defines those Agile attributes that are saved into CATIA V5 CATDrawing files, when using the **Update Properties** command.
AgileGetProperties.CATProduct	Defines those Agile attributes that are saved into CATIA V5 CATProduct files, when using the **Update Properties** command.
FrameDefinition	Defines those Agile attributes that are mapped onto drawing title blocks, when using the **Update Title Block** command.
AccCreateObjectTypes	Not used
CatiaToAgileNew.DOCUMENT	This mapping section is used for assigning attributes when creating Documents using the New command.
AccSaveViewable.CATPart	Defines types of viewable files that can be saved for CATParts in the **Save With…** command
AccSaveViewable.CATProduct	Defines types of viewable files that can be saved for CATProducts in the **Save With…** command

Section name	Description
AccSaveViewable.CATDrawing	Defines types of viewable files that can be saved for CATDrawings in the **Save With…** command

Mapping Options for [CatiaToAgile.XXXX] Sections

Each mapping consists of a pair of objects. The right side of the pair defines information that can be extracted from CATIA V5. Here, CATIA V5 is the source of the attribute value. The left side of the pair defines the attribute value's target location in Agile.

There are several configuration options for the "right side" that define what kind of data should be extracted from CATIA V5, and what kind of transformation can be applied to the data. Each right side attribute consists of three sections, for example:

```
DESCRIPTION   =  Std.DescriptionReference.ToUpper
```

The first section is either Std, Par, or Def. "Std" refers to CATIA V5 system attributes, as listed here:

Table: Standard mapping values using "Std" prefix

Std.DescriptionReference
Std.Extension
Std.PartNumber
Std.Definition
Std.Nomenclature
Std.Revision

"Par" is a reference to user-defined property in CATIA V5, such as MATERIAL, DESCRIPTION, or ENGINEER. These types of mappings are only useful where the CATIA V5 file has a property corresponding to the name mentioned in the mapping.

"Def" is a default fixed string value.

Finally, the final suffix is a description of how the data should be modified. The following modifiers are possible:

Table: Suffix Options for Mapping

ToUpper	Transfer all characters to upper case
ToLower	Transfer all characters to lowercase
None	Do not modify the data
Range-<idx1>-<idx2>	Range of the string from position idx1 to idx2, example: Part.PartNumber.Range-0-2
Prefix	Prefix to be added in front of the string, example: Par.PartNumber.PrefixPRT
Suffix	Suffix to append to the string, example: Par.PartNumber.SuffixPRT

There are two special values that are used on the left side of these mappings. In the [CatiaToAgile.DOCUMENT] section, you use the value CAX_NEW_NUMBER to represent the

Number field that will be assigned to newly created Documents. In the [CatiaToAgile.ITEM] section, you use the value ITEM to represent the Number field that will be assigned to newly created Parts.

Mapping Options for [AgileTo.XXXX] Sections

These section are used to define mappings from Agile to CATIA, which occur automatically during the Save process. As this will add time to the Save process, the list of attributes should be kept to the bare minimum that absolutely need to be kept synchronized. Other attributes can be synchronized using “Update Properties”, as described in the next section. For formatting details see Mapping Options for Update Properties Sections - CATIA on page 84 on page 80 below.

Mapping Options for [AgileGetProperties.XXX] Sections

These section is used to define mappings from Agile to CATIA V5, which occur when the user runs the Update Properties command manually. For formatting details see Mapping Options for Update Properties Sections - CATIA on page 84 on page 80.

Mapping Options for [FrameDefinition] Section

These section is used to define mappings from Agile attributes to the CATIA V5 drawing title block, which occurs when the user runs the Update Title Block command. For formatting details see Mapping Options for Update Properties Sections - CATIA on page 84 on page 80.

Mapping Options for Update Properties Sections - CATIA

Multiple sections of the AccCustomer9.ini file, as listed above, are used to define mappings from Agile to CATIA. For standard attributes the format of this section is:

CAD Parameter = <Source Table>_Field.Format

For example:

Agile_Des = Title Block_Description.ToUpper

The left side value is the name of the CATIA parameter to be updated. For the [AgileTo.XXXX] and [AgileGetProperties.XXX] sections, the formatting of the left side matches the description shown for the RIGHT side of the [CatiaToAgile.XXXX] section (see above for details). For the [FrameDefinition] section, the left side represents a CATIA text property in the format Text.n, where n is an integer.

The right side can be either the symbolic attribute name from the CaxClient.xml file (such as NUMBER, DESCRIPTION, etc.) or any Agile attribute represented as follows:

Section	Represents	Example
<Source Table>	Agile tab name	Title Block
Field	Agile attribute name	Description
Format	Text processing	ToUpper

For history and change history attributes, which are arranged in a table, the format of this section is:

CAD Parameter = <Filter Table>_Field,<Filter Value>,<Filter>,<Source Table>_Field.Format

For example:

Agile_CreUser = History_Action,Create,first,History_User.None

HIS_RELDATE_1 = Change History_Status,Released,last,Change History_Rel Date_int.Date01

Where the left side value is the name of the CATIA parameter to be updated, and the right side specifies how to find the desired row and column in the table below:

Section	Represents	Example
<Filter Table>	Agile tab name to search	Title Block
Field	Desired column to search	Action
<Filter Value>	Value to detect in the column	Create
<Filter>	Which row to select, with these options: first first+n n=integer value last last-n n=integer value	first
<Source Table>	Agile tab name to retrieve value from	History
Field	Desired column to retrieve value from	User
Format	Text processing	None

Options for "Format"

The Format string allows you to perform additional processing on the text string being passed back into CAD. This includes predefined formats and general TCL format procedures.

Predefined formats

Format	Description
None	no processing
ToLower	convert the value to lower case
ToUpper	convert the value to upper case
Range-x-y	substring of the value from index x to index y (y may be numeric or "end")
Date01	convert int dateformat to "%d.%m.%y %H:%M:%S" example: 01.01.2007 00:00:00
Date02	convert int dateformat to "%d.%m.%Y" example: 01.01.2007
Date03	convert int dateformat to "%d.%m.%y" example: 01.01.07
Date04	convert int dateformat to "%d-%m-%y" example: 01-01-07
Date05	convert int dateformat to "%m/%d/%y" example: 01/01/07

Format	Description
Date06	convert int dateformat to "%d-%b-%y" example: 01-Jan-07
Prefix<str>	append a prefix <str> to the value
Suffix<str>	append a suffix <str> to the value

TCL format procedures

Any registered (tclIndex) TCL procedure that gets the current value as input and returns the formatted string. For instance:

```
proc MyFormat { value } {
  set formatedvalue $value
  return $formatedvalue
}
```

Mapping Part Attributes

In addition to mapping attributes from the CAD Document back into CAD, you can map attributes from the corresponding Part object that has been associated to the Document using the Create Item/BOM command. In order to specify a Part attribute, simply prefix the attribute value with “PART:”. This example shows mapping both the Document Number and Part Number into CAD:

Agile_DocId = Title Block_Number.None

Agile_PartId = PART:Title Block_Number.None

Chapter 8

Installing and Configuring Solid Edge Connector

This chapter includes the following:

This section describes setting up the connection between your Solid Edge CAD application and Agile Engineering Collaboration.

The main steps are:

- Extract files from zip file
- Extract files from EC Client zip file
- Configure PLM API for WAN mode (optional)
- Edit some parameters in the configuration file
- Register libraries and executable
- Install proper **AgileAPI.jar** file
- Set up Agile menu in Solid Edge
- Set up Agile toolbar in Solid Edge

The installation requires the following file:

- **aceNNNN.zip** – Main installation package, where NNNN is the release level

Performing the installation steps described here will enable the Agile commands to appear within Solid Edge. In order to have a completely functional integration, you must also:

- Perform the core Agile configuration as described in the Administration section of the document *Agile Engineering Collaboration Client*.
- Configure desired Solid Edge Connector parameters as described in the section Solid Edge Connector Administration on page 92 on page 101.

Extracting Files for Connector

Extract the installation file to the folder location **D:\AgileEC** or **C:\AgileEC** for a single-drive system. When you unzip, make sure that you retain the folder paths from the zip file. When the files are unzipped, you should see a folder named **ace**, which contains the Connector installation.

Extracting Files for EC Client

Extract the EC Client installation file to a temporary location. After extraction you will see 5 connector directories (acc, ace, acp, acu, acw) plus a jar directory.

- Copy the entire jar directory inside the **\AgileEC\ace** directory.
- From the temporary **ace\server\scripts** directory, copy the file caxclient.bat to the **\AgileEC\ace\server\scripts** directory.
- From the temporary **ace\jar\Agile9** directory, copy the file CaxClient_Designs.xml to the **\AgileEC\ace\jar\Agile9** directory.

Configure PLM API for WAN Mode

PLM API is a high-performance web services API used to support operations across wide area network (WAN). This is an optional capability that is only necessary if you are going to be supporting MCAD connectors at remote sites (i.e. locations distant from your Agile application server). If you do not need this capability, skip to the next step.

- Log in with the Agile 9 web client to the server that will be used with the MCAD connectors. Use an account that has privileges to create File Folder objects.
- Create the following three File Folder objects, using the number listed in the first column, and then attach the file listed in the second column. Check In the File Folder following attachment. The files can be found under the jar\agile9\server folder. Note that depending on your system settings, you might need to temporarily modify the admin settings for File Folder objects to allow using an arbitrary Number field.

File Folder number	File to attach
PLMAPI_ASYNC	plm-api-server.xml
PLMAPI_CONFIG	plm-api-config.properties
PLMAPI_CONNECTOR	plm-api-sdk.xml

- Copy the file plm-api-server.jar from jar\agile9\server to the WSX extensions folder of the target Agile 9 server. Usually this folder is located in {agile_home}\integration\sdk\extensions).
- The EC Client configuration file must be edited to turn on PLM API mode. To do this, bring up the \jar\agile9\CAXClient_Designs.xml file in a text editor. Change the following two lines as follows (change from 0 to 1). Save the file.

```
<clientConfig name="DFM" value="1"/> <!-- Use direct DFM file transfer
1=enabled, 0=disabled -->
```

```
<clientConfig name="usePlmApi" value="1"/> <!-- Use PLM API Web
Services 1=enabled, 0=disabled -->
```

- The EC Client startup file must also be edited to enable PLM API mode. To do this, bring up \ace\server\scripts\caxclient.bat in a text editor. Change the following lines by appropriately uncommenting and commenting the lines. Note that lines with “rem” in front of them are commented out and inactive:

Settings	Necessary settings for using PLM API
DFM_JAR	set DFM_JAR=%CAX_ROOT%\jar\agile9\axis.jar;.....(etc.) **rem** set DFM_JAR=
JAVA_HEAP_SIZE	rem set JAVA_HEAP_SIZE=-Xms128m -Xmx128m set JAVA_HEAP_SIZE=-Xms128m -Xmx1024m

Editing the Configuration File

Open the file \AgileEC\ace\Server\Scripts\3DCADMapping.ini in a text editor. Edit the values as described in table below to match your system configuration.

Sample values	What this command specifies
[JNIOPTIONS] ; -Djava.class.path=D:\AgileEC\ace\jre1.5.0\lib\rt.jar; D:\AgileEC\ace\Server\AgileCaxConnector.jar; D:\AgileEC\ace\Server\AgileAPI.jar; D:\AgileEC\ace\Server\xercesImpl.jar; D:\AgileEC\ace\Server\xmlParserAPIs.jar; D:\AgileEC\ace\Server\CaxAglProxy.jar; D:\AgileEC\ace\Server\CaxAglDataTypes.jar -Dagile.xml.file=D:\AgileEC\ace\Server\AgileConnector.xml -Djava.agile.gui.address=localhost -Djava.agile.gui.listener=5112 -Djava.agile.proxy.listener=5113 -Dagile.caxconnect.logfile=C:\agile.log ;-Djava.agile.proxy.logfile=C:\Proxy.log ;	To all jar files. Edit each path to match your system configuration. The first path listed is the JRE environment. Determine the required SolidWorks connector JRE version from the chart in Appendix C, and then select the appropriate path from the available ones provided in the acw installation. Edit all other paths to match the drive and directory in your installation. **Note** The proxy.log file (the last line in this section) should be commented out for production use, because this causes an additional log file window to pop up that would be annoying to users. Uncomment only for debugging.

Sample values	What this command specifies
[CheckOutDisk] ; D: ;	The disk drive on the client computer to be used for the working directory.
[CheckOutPath] ; \AgileSE\Work\ ;	The path of the working directory **Important** You must also create this directory on your computer.
[LogFileDir] ; D:\AgileSE\Temp\ ;	The full path of the log file directory **Important** You must also create this directory on your computer.
[AgileURL] ; http://servername:8888/Agile ;	The URL for the EC Client

Registering Libraries

- Navigate to the \AgileEC\ace directory, and double-click on registerSE.bat. A command window will appear; eventually this will show a “SUCCESS” message and then terminate.

Installing the AgileAPI.jar File

Note If this step is not done correctly, the connector may appear to be functioning normally but data corruption may occur!

The correct **AgileAPI.jar** file, matching the specific Agile service pack level, must be installed in the directory **\AgileEC\ace\jar\Agile9.**

- Search for the file **AgileAPI.jar** within your site’s Agile server installation (such as **C:\Program Files\Agile**).
- Copy this file to **\AgileEC\ace\jar\Agile9**, overwriting the file already there.
- If you are unable to locate this file, please contact Agile Support.

Setting Up the Agile Menu

To set up the Agile menu within Solid Edge, do the following steps:

- Launch Solid Edge as you normally would (Using the Start menu or desktop icon, etc.).
- Create or open a file (so that you are not at the startup screen).
- Go to Tools > Add-Ins > Add-In Manager… and check the box next to Agile. Click **OK**.
- Now if you navigate to Tools > Add-Ins you will see an Agile menu. Since it is not possible within Solid Edge to have this menu appear on the top menu bar, it is advisable to add the Agile toolbar (see next step).

Setting Up the Agile Toolbar

To set up the Agile command icons on the Solid Edge toolbar, do the following steps:

- Launch Solid Edge, and create or open a file (so that you are not at the startup screen)
- Go to Tools > Customize…
- On the Toolbars tab, scroll down and highlight Agile
- Drag any icon from the Buttons area on the right, up to a blank area of the main toolbar (the gray area). When you let go, a new toolbar will be created and the icon will be inserted in it.
- Drag the remaining icons into the toolbar you just created. When completed you can dock the toolbar with the other standard toolbars.
- You must repeat the above steps for each separate mode of Solid Edge (Part, Assembly, Drawing).

Solid Edge Connector Administration

This section provides a complete summary of configuration options available for the Solid Edge connector. Once the basic installation has been done following the instructions in the previous section, you can refer here for details of all possible settings.

Note that in addition to the configuration files listed here, the EC Client must be additionally configured to provide complete operation of the Solid Edge Connector. See the EC Client Configuration Options section for details.

Table: List of all Configuration Files for the Solid Edge Connector

Configuration file	Purpose	Location
3DCADMapping.ini	Mapping and configuration	AgileEC\ace\Server\Scripts
PlmSEAddin.xml	Menu definition	AgileEC\ace\Server\Scripts

Note Configuration files typically change content between connector releases. When upgrading to a new release, please incorporate your site's configuration settings into the new version of the configuration files. Failure to do so will cause unpredictable behavior of the connector.

Configuring the 3DCADMapping.ini File

There is one main configuration file, which controls nearly all aspects of the Solid Edge Connector. The file is named **3DCADMapping.ini** and is located in the **..\AgileEC\ace\Server\Scripts** directory. Since this file is located within the Solid Edge Connector installation on the client machine, it is possible to customize configuration options on a per-machine basis, although typical usage is to have a common configuration file within a given site. When changes are made to the configuration file, it is necessary to exit and re-start Solid Edge in order to use the new settings.

The configuration file is made up of a series of configuration options (also called "sections"), with the option listed between square brackets, and the various settings for the option listed on the following lines. Lines beginning with a semi-colon (;) are commented out. In this file, unused options are commented out rather than deleted, which may help later if you want to enable some of the unused options.

Because this configuration file is also used for Agile 8.5 and Agile e-series installations, not all of the configuration options are valid for Agile 9. The following list summarizes the options that are valid for Agile 9.

Table: Valid configuration options in Solid Edge 3DCADMapping.ini

Option Name	Usage
[JNIOPTIONS]	Sets various Java parameters.
[LogFileDir]	Drive & path of temp directory
[CheckOutDisk]	Disk drive of work directory
[CheckOutPath]	Path of work directory

Option Name	Usage
[LogFileDir]	Disk and path of log file directory
[AgilePartViewFile]	OBSOLETE. See [Agile9PartViewFileExtensions]
[AgileAssemblyViewFile]	OBSOLETE. See [Agile9AssemblyViewFileExtensions]
[AgileDrawingViewFile]	OBSOLETE. See [Agile9DrawingViewFileExtensions]
[AgileViewFileCustomScript]	Drive, path & name of a executable file which will be executed to generate a customized neutral view file to be saved.
[AgilePartCheckinFile]	OBSOLETE. See [Agile9PartViewFileExtensions]
[AgileAssembly CheckinFile]	OBSOLETE. See [Agile9AssemblyViewFileExtensions]
[AgileDrawing Checkin File]	OBSOLETE. See [Agile9DrawingViewFileExtensions]
[AgileURL]	Information required to connect to the Agile server.
[Agile9CreateDocument]	This mapping section is used for setting attributes for Documents during the **Save** command, for the initial save.
[Agile9UpdateDocument]	This mapping section is used for setting attributes for Documents during the Save command, after the initial save.
[Agile9UpdateItem]	This mapping section is used for setting attributes for Parts during the **Create Item/BOM** command.
[Agile9UpdateItemConfigured]	This mapping section is used for setting attributes for Parts during the Create Item/BOM command, when the CAD file is identified as configured.
[Agile9CheckinDocument]	This mapping section is used for setting file attachment attributes in Agile.
[AgileViewableIncludeRevision]	Appends the revision of the Document object onto the end of the viewable filenames generated in the Save comment.
[Agile9GetRevision]	Retrieves the current revision field of the Document
[Agile9UpdateProperties]	Defines the property mapping from Agile to Solid Edge, when using the Update Properties command.
[Agile9SaveUpdateProperties]	Defines the property mapping from Agile to Solid Edge that occurs automatically during the **Save** command.
[Agile9LoadUpdateProperties]	Defines the property mapping from Agile to Solid Edge that occurs automatically during the **Load** command
[Agile9UpdateTitleBox]	Defines the property mapping from Agile to Solid Edge, when the properties of a drawing are updated using the Update Title Block command
[AgilePartTemplate]	OBSOLETE. See [Agile9TemplatePath]
[AgileAssemblyTemplate]	OBSOLETE. See [Agile9TemplatePath]
[AgileDrawingTemplate]	OBSOLETE. See [Agile9TemplatePath]
[Agile9Configuration]	Controls how SolidWorks configurations are handled
[Agile9PartViewFileExtensions]	Sets the allowable viewable file formats created during the Save command

Option Name	Usage
[Agile9AssemblyViewFileExtensions]	Sets the allowable viewable file formats created during the Save command
[Agile9DrawingViewFileExtensions]	Sets the allowable viewable file formats created during the Save command
[Agile9TemplatePath]	Sets the path to where template files are stored, for use by the New command
[Agile9Units]	Sets the default units for the New command
[Agile9Renaming]	Activates the filename renaming process during the Load command

Table: Detailed Configuration Options

Option	Description
Syntax	**Configuration Options**
[JNIOPTIONS]	Java Parameters
-Djava.class.path=<path><file>.jar, etc.	1.For rt.jar, edit <path> to match location of Java installation. 2. For other jar files, edit <path> to match installation path (default is D:\AgileEC\ace\Server)
-Dagile.xml.file=<path>\AgileConnector.xml	Edit <path> to match installation path (default is D:\AgileEC\ace\Server)
-Djava.agile.gui.address=localhost	Do not change
-Djava.agile.gui.listener=5112	Do not change
-Djava.agile.proxy.listener=5113	Do not change
-Djava.agile.proxy.logfile=C:\Proxy.log	Uncomment this line to enable a Java debug window and log file.
[CheckOutDisk]	**Drive of work directory**
Syntax	<drive>
Default Value	D:
Configuration	Set to drive where work directory is located
[CheckOutPath]	**Path of work directory**
Syntax	<path>
Default Value	\AgileSE\Work
Configuration	Set to path where work directory is located
[LogFileDir]	**Drive & path of temp directory**
Syntax	<drive><path>

Option	Description
Default Value	D:\AgileSE\Temp
Configuration Options	Set to drive and path where temp directory is located
[AgileViewFileCustomScript]	Drive, path & name of a executable file which will be executed to generate a customized view file to be saved
Syntax	<drive><path><name>
Default Value	D:\AgileEC\ace\Server\Scripts\ViewFileCustom.bat
Configuration Options	Set to drive and path where the executable file is located. In special cases this file will not be executed (see descriptions below).
[AgileURL]	URL and Port of the Agile9 server
Syntax	http://<server>:<port>/<Agile file name> **Error! Hyperlink reference not valid.**
Default Value	http://agileserver:8888/Agile
Configuration Options	Set to a dedicated port of a server machine where the Agile server software is located
[Agile9CreateDocument]	Defines the property mapping from Solid Edge to Agile, when the Documents are saved into Agile using the **Save** command, for the first time.
Configuration Options	Each mapping consists of a pair of objects. The left side of the pair defines the name of an Agile attribute that is the target of a property that is derived from the Solid Edge Model. The right side of the pair defines the Solid Edge property name. There are several configuration options for the right side of the pair that define what kind of data should be extracted from Solid Edge. Each right side attribute consists of two or three sections. All Solid Edge mappings begin with 3DCADTable. The second section can define system attributes. Possible values include: ▫ FileStamp – Timestamp (in seconds) ▫ ModelPathOnly – Directory where the CAD file is stored when saved to Agile, e.g. D:\CAD_file\Housing ▫ ModelName – Name of Solid Edge file with extension, e.g., BOLT.SLDPRT ▫ ModelVersion – Solid Edge version ▫ ModelConfigurationName – For configured parts/assemblies, the name of the configuration

Option	Description
	▫ ModelTitle – Name of Solid Edge model without file extension, e.g., BOLT ▫ ModelExtension – Solid Edge Model type, e.g., SLDPRT, SLDASM, SLDDRW Values of the format 3DCADTable.Property.[value], where [value] is the name of a Solid Edge custom property such as Description or PartNumber. The following are some example mappings for a Solid Edge part called housing.sldpart with a custom property called Material with a value of Aluminum: ▫ CAX_FIL_NAME = 3DCADTable.ModelName ▫ DESCRIPTION = 3DCADTable.ModelTitle ▫ MATERIAL = 3DCADTable.Property.Material In this example, the Agile description is "housing". An attribute in Agile called CAX_FIL_NAME has the value "housing.sldasm" and an Agile attribute called Material has the value "Aluminum". The name used for the Agile attribute on the left side of the mapping is arbitrary. The actual attribute that is targeted for mapping is defined in the EC Client configuration. There is one special value that is used on the left side of these mappings. You use the value CAX_NEW_NUMBER to represent the Number field that will be assigned to newly created Documents.
[Agile9UpdateDocument]	Defines the property mapping from Solid Edge to Agile, when the Documents are saved into Agile using the Save command, after the first time. Configuration options are the same as [Agile9CreateDocument].
[Agile9UpdateItem]	Defines the property mapping from Agile to Solid Edge, when Items are created using the Save command
Configuration Options	See at [Agile9UpdateDocument] section There is one special value that is used on the left side of these mappings. You use the value ITEM to represent the Number field that will be assigned to newly created Parts.
[Agile9UpdatedItemConfigured]	Defines the property mapping from Agile to Solid Edge, when Items are created or updated using the Create Item/BOM command, and the Items are marked as Configured (see Agile9Configuration section)

Option	Description
Configuration Options	See the [Agile9UpdateDocument] section There is one special value that is used on the left side of these mappings. You use the value ITEM to represent the Number field that will be assigned to newly created Parts.
[Agile9CheckinDocument]	Defines the property mapping for file attachements, when the files are checked in during the **Save** command.
Configuration Options	System parameters - do not change
[Agile9UpdateProperties]	Defines the property mapping from Agile to Solid Edge, when using the Update Properties command manually.
Configuration Options	Each mapping consists of a pair of objects. The left side of the pair defines the name of a Solid Edge property that is being set. The right side of the pair defines the Agile attribute. For details of the mapping see the section " Mapping Options for Update Properties Sections"
Examples	AgileNumber = Title Block.Number Description = Title Block.Description
[Agile9SaveUpdateProperties]	Defines the property mapping from Agile to Solid Edge which occurs automatically during the **Save** command
Configuration Options	Each mapping consists of a pair of objects. The left side of the pair defines the name of a Solid Edge property that is being set. The right side of the pair defines the Agile attribute. For details of the mapping see the section " Mapping Options for Update Properties Sections"
[Agile9LoadUpdateProperties]	Defines the property mapping from Agile to Solid Edge, which occurs automatically during the Load command
Configuration Options	Each mapping consists of a pair of objects. The left side of the pair defines the name of a Solid Edge property that is being set. The right side of the pair defines the Agile attribute. For details of the mapping see the section " Mapping Options for Update Properties Sections"
[Agile9UpdateTitleBox]	Defines the property mapping from Agile to Solid Edge, when the properties of a drawing are updated using the **Update Title Block** command
Configuration Options	Each mapping consists of a pair of objects. The left side of the pair defines the name of a Solid Edge property that is being set. The right side of the pair defines the Agile attribute. For details of the mapping see the section " Mapping Options for Update Properties Sections"

Option	Description
[Agile9Configuration]	Setting that control how you identify configured parts
Syntax	ConfigProperty = Configured ConfigPropertyValue = YES
Configuration Options	ConfigProperty is the name of the Solid Edge Custom Property that is used to identify configured parts. ConfigPropertyValue is the value that must be set for this property, to indicate that this part is to be treated as containing multiple part configurations. When this value is set for a part or assembly, each different configuration is treated as a unique part, when generating the Part BOM with the Create Item /BOM command.
[Agile9PartViewFileExtensions]	Sets the allowable viewable file formats for parts, as created during the Save command
Syntax	(igs$)\|(jt$)|(sat$)\|(step$)|(xgl$)\|(x_b$)|(x_t$)\|(ems$)|(stl$)\|(plmxml$)|(model$)\|(catpart$)|(pdf$)
[Agile9AssemblyViewFileExtensions]	Sets the allowable viewable file formats for assemblies, as created during the Save command
Syntax	(bmk$)\|(igs$)|(sat$)\|(step$)|(sgl$)\|(x_b$)|(x_t$)\|(plmxml$)|(model$)\|(jt$)|(catpart$)\|(pdf$)
[Agile9DrawingViewFileExtensions]	Sets the allowable viewable file formats for drawings, as created during the Save command
Syntax	(igs$)\|(dgn$)|(dwg$)\|(dxf$)|(pdf$)
[Agile9TemplatePath]	Sets the path to where template files are stored, for use by the New command
Syntax	<drive><path>
Configuration Options	The designated path will be scanned for *.prtdot, *.asmdot, and *.drwdot files, and these files will be made available as templates within the New command.
[Agile9Units]	Sets the default units for the New command
Syntax	Millimeters | Inches
[Agile9Renaming]	Activates the filename renaming process during the Load command

Option	Description
Syntax	0 | 1
Configuration Options	0 = Do not rename files during the Load process 1 = Rename files during the Load process, so that the filename matches the Agile Number field or a customized text string. This is used to support both the initial rename use case and the "save as" use case. For details of how to customize the filename see the file acwCustomer.tcl

Mapping Options for Update Properties Sections - Solid Edge

Multiple sections of the AccCustomer9.ini file, as listed above, are used to define mappings from Agile to Solid Edge. For standard attributes the format of this section is:

CAD Parameter = <Source Table>_Field.Format

For example:

Agile_Des = Title Block_Description.ToUpper

The left side value is the name of the Solid Edge parameter to be updated, For the [AgileTo.XXXX] and [AgileGetProperties.XXX] sections, the formatting of the left side matches the description shown for the RIGHT side of the [CatiaToAgile.XXXX] section (see above for details). For the [FrameDefinition] section, the left side represents a CATIA text property in the format Text.n, where n is an integer.

The right side can be either the symbolic attribute name from the CaxClient.xml file (such as NUMBER, DESCRIPTION, etc.) or any Agile attribute represented as follows:

Section	Represents	Example
<Source Table>	Agile tab name	Title Block
Field	Agile attribute name	Description
Format	Text processing	ToUpper

For history and change history attributes, which are arranged in a table, the format of this section is:

CAD Parameter = <Filter Table>_Field,<Filter Value>,<Filter>,<Source Table>_Field.Format

For example:

Agile_CreUser = History_Action,Create,first,History_User.None

HIS_RELDATE_1 = Change History_Status,Released,last,Change History_Rel Date_int.Date01

Where the left side value is the name of the Solid Edge parameter to be updated, and the right side specifies how to find the desired row and column in the table below:

Section	Represents	Example
<Filter Table>	Agile tab name to search	Title Block
Field	Desired column to search	Action
<Filter Value>	Value to detect in the column	Create
<Filter>	Which row to select, with these options: first first+n n=integer value last last-n n=integer value	first
<Source Table>	Agile tab name to retrieve value from	History
Field	Desired column to retrieve value from	User
Format	Text processing	None

Options for "Format"

The Format string allows you to perform additional processing on the text string being passed back into CAD. This includes predefined formats and general TCL format procedures.

Predefined formats

Format	Description
None	no processing
ToLower	convert the value to lower case
ToUpper	convert the value to upper case
Range-x-y	substring of the value from index x to index y (y may be numeric or "end")
Date01	convert int dateformat to "%d.%m.%y %H:%M:%S" example: 01.01.2007 00:00:00
Date02	convert int dateformat to "%d.%m.%Y" example: 01.01.2007
Date03	convert int dateformat to "%d.%m.%y" example: 01.01.07
Date04	convert int dateformat to "%d-%m-%y" example: 01-01-07
Date05	convert int dateformat to "%m/%d/%y" example: 01/01/07
Date06	convert int dateformat to "%d-%b-%y" example: 01-Jan-07
Prefix<str>	append a prefix <str> to the value
Suffix<str>	append a suffix <str> to the value

TCL format procedures

Any registered (tclIndex) TCL procedure that gets the current value as input and returns the formatted string. For instance:

```
proc MyFormat { value } {
  set formatedvalue $value
  return $formatedvalue
}
```

Mapping Part Attributes

In addition to mapping attributes from the CAD Document back into CAD, you can map attributes from the corresponding Part object that has been associated to the Document using the Create Item/BOM command. In order to specify a Part attribute, simply prefix the attribute value with "PART:". This example shows mapping both the Document Number and Part Number into CAD:

Agile_DocId = Title Block_Number.None

Agile_PartId = PART:Title Block_Number.None

Controlling Custom vs. Configuration-specific Properties

In the following sections:

- [Agile9UpdateDocument]
- [Agile9UpdateItem]
- [Agile9UpdateItemConfigured]

You can use the "Custom_" and "ActiveConfiguration_" modifiers to control whether the properties are coming from Custom or Configuration-specific Properties. For example:

```
ITEM  = 3DCADTable.Property.Custom_PartNumber
```

Sets the Part number attribute using a Custom property called "PartNumber"

```
DESCRIPTION = 3DCADTable.Property.ActiveConfiguration_Description
```

Sets the Description attribute from a configuration-specific property called "Description".

If you omit the "Custom_" or "ActiveConfiguration_" modifier, it defaults to configuration-specific. Note also that Solid Edge properties are case-sensitive!

Modifying the Agile Menu Definition

There is another configuration file, which controls the layout of the Agile menu. The file is named **PlmSEAddin.xml** and is located in the **AgileEC**\ace**Server****Scripts** directory. Since this file is located within the Solid Edge Connector installation on the client machine, it is possible to customize menu options on a per-machine basis, although typical usage is to have a common configuration file within a given site. When changes are made to the configuration file, it is necessary to exit and re-start Solid Edge in order to use the new menus.

Configuration of the menus is limited to:

- Removal of unneeded commands and menus
- Renaming of commands and menus
- Restructuring of commands and menus

- Addition or removal of menu separators

The portion of the file which can be configured is within the <CaxMenu_EN> tags (for English language). Within this section you will see four sets of tags, which contain the menu entries for the following situations in Solid Edge:

<Base> - Menus when no Solid Edge component is active

<Part> - Menus when a single Part is active

<Assembly> - Menus when an Assembly is active

<Drawing> - Menus when a Drawing is active

For example, the <Base> section looks like this when you call up the file in an editor:

However, note that for each of these lines, there is additional text if you scroll over to the right side. Make sure when cutting and pasting lines, that you get the entire line. The portion of the lines that you would need to edit is limited to what is shown above (i.e. do not edit any part of the text further to the right).

The editable sections of the file are described as follows:

Editable Sections	Description
<menu…> tags	Defines the type of menu entry
Syntax	menu – Indicates a menu or sub-menu entrymenu item – Indicates a menu command
type	Distinguishes the entries for each section.
Syntax	type = “<number>” where <number> equals: 0 = Base menu1 = Part menu 2 = Assembly menu 3 = Drawing menu
text	Defines the menu text and hierarchy level
Syntax	For **menu**: text = “<menu>@<next-higher-menu>” For **menuitem**: text = “<command>@<menu>@<next-higher-menu>”

Examples	Example of first-level menu and a command within it: <menu type = "0" text = "Agile" <menuitem type = "0 text = "Connect@Agile" Example of second-level menu and a command within it: <menu type = "0" text = "New@Agile" <menuitem type = "0" text = "Part@New@Agile"

Removing Commands and Menus

It can be useful to remove commands from the menus, for example to eliminate commands that do not fit with your specific business processes. To remove a command from the menus, simply delete the entire line containing the command that you want to remove. Remember to delete it from all menus that it appears in (<Base>, <Part>, etc.). You can also remove entire sub-menus, but make sure to also remove or restructure all commands within the sub-menu.

Renaming Commands and Menus

Commands and menus can be renamed simply by changing the text values. Remember to rename them in all menus that they appear in (<Base>, <Part>, etc.). If you rename a menu, make sure to also change the menu portion of the text field in each command in the menu.

Restructuring Commands and Menus

Commands can be restructured, for example to move them in or out of a sub-menu. To move a command from a sub-menu to the next higher menu, change the text field of the command to remove the reference to the sub-menu. To move a command into a sub-menu, do the reverse. You can add your own sub-menus, if necessary, by adding additional menu lines.

Adding or Removing Menu Separators

Menu separators can easily be added or removed. Separators are defined by lines in the file such as this:

```
<menuitem type = "1" text = "@Agile" position = "-1" callback =
"Separator" enablemethod = "" hint = "" />
```

The difference between this menuitem, and one for a real command, is that the text entry has no command text before the first @ sign, and the callback entry is "Separator". You can add or remove any of these separator lines to control the positioning of separators in the menus.

Chapter 9

EC Client Configuration Options

This chapter includes the following:

Regardless of which CAD Connector you are using, the main user interface is the EC Client. The configuration of the EC Client is done the same way for all CAD connectors. There are two main files to be aware of, the startup file (CaxClient.bat) and the configuration file (CaxClient_{type}.xml). The options available in these files are described in this chapter, as well as the necessary configuration that needs to be done using the Agile Admin client, to allow the EC Client and the CAD Connectors to work properly with Agile PLM.

Startup File - CaxClient.bat

The EC Client is started by the startup file CaxClient.bat. Each CAD connector has this file, which is triggered by the Agile > Start Client command.

CAD Connector	Location
Pro/ENGINEER	AgileEC\acp\com
SolidWorks	AgileEC\acw\Server\Scripts
Unigraphics NX	AgileEC\acu\com
CATIA V5	AgileEC\acc\com
CATIA V4	/acc-rt/jar
SolidEdge	AgileEC\ace\Server\Scripts

There are three settings that can be modified in this file, as listed below. Note that CAXCLIENTXML is the primary switch that controls whether Designs or Documents are used for the EC data model.

Settings	What this settings specifies
JAVA_HOME	JRE directory path. Select the appropriate path from the available ones provided in the acw installation. See "Set Java Version" above.
DFM_JAR	Sets the jar files necessary for direct DFM file access. Do not set if using JRE 1.4.
CAXCLIENTXML	Sets the appropriate XML file for running in Document mode or

	Design mode.

Configuration File - CAXClient_{type}.xml

The EC Client is configured by editing an XML file called CAXClient_{type}.xml, where {type} is either Documents or Designs. Each file, depending on the connector you choose to work with, is located in one of the following locations.

CAD Connector	Location
Pro/ENGINEER	AgileEC\acp\jar\Agile9
SolidWorks	AgileEC\acw\jar\Agile9
Unigraphics NX	AgileEC\acu\jar\Agile9
CATIA V5	AgileEC\acc\jar\Agile9
CATIA V4	/acc-rt/jar
SolidEdge	AgileEC\ace\jar\Agile9

In each case, this file is configured in exactly the same way. This configuration file is used to:

- control the behavior of the EC Client
- define the data model used for storing CAD data in Agile

Since you can run the EC client with one of the two data models (Documents or Designs), there are now two independent XML files. The "CAXCLIENT" parameter in the CAXClient bat script determines which file is used, and therefore which data model is used.

clientConfig Parameters

The tag <clientConfig> indicates the parameters in this section. Configure these parameters as desired to match system and process requirements at your site.

clientConfig section

Name	Value	Remarks
userName	cax	default user (filled in login form)
serverURL	http://localhost:8888/Agile	used server connect string
dateFormat	yyyy-MM-dd HH:mm:ss z	default date format
enableFields	ON | **OFF**	If ON, will automatically turn on the necessary attributes configured for the sub-classes defined in CAXClient.xml (does not name the attributes however).

Name	Value	Remarks
editMode	**ON** | OFF	Allows editing within forms
ping	**ON** | OFF	Helps to keep the client session alive under certain conditions
ObjectCache	ON | **OFF**	ADVANCED OPTION - Contact Agile Solution Delivery for assistance
SessionXMLCache	**ON** | OFF	ADVANCED OPTION - Contact Agile Solution Delivery for assistance
CopySourceAttachmentsForward	ON | **OFF**	If ON - removes all non-SOURCE attachments automatically when creating a new pending revision using EC Client. Useful for removing viewable and baseline attachments.
SingleFieldUpdate	ON | **OFF**	If ON - updates objects one attribute at a time, rather than all at once. Required only to bypass an update problem on AIX JRE platform.
drsListenPort	5112	Dresden (drs) protocol listen port (do not change)
creationMode	**INT** | BAT	Default mode (possible is INT or BAT) for creation of new objects in Save Command (interactive or batch-mode)
creationModePart	**INT** | BAT	Default mode (possible is INT or BAT) for creation of new parts in create Item BOM command (interactive or batch-mode)
revisionSequence (also revisionSequenceECO, revisionSequenceDVO etc.)	Examples: ,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z, ,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,	Default Sequence of Revision letters. Keep a comma at the beginning and the end of the sequence. You can create multiple entries of this line, by appending the name of an Agile Change sub-class on the end. That will assign the given revision sequence to that sub-class only.
revisionSequenceEditor	**ON** | OFF	If ON - allows selecting a revision from the pre-defined revisionSequence list in Manage Change. If OFF - you must enter the revision.
EcoDefaultWorkflow	**Examples**: "Default Change Orders" "{class} Default Workflow"	The workflow assigned automatically when creating a Change object, ECO in Manage Change. If the name contains the string "{class}", the name of the Change sub-class is inserted. This allows assigning a unique workflow per Change sub-class.

Name	Value	Remarks
EcoIgnoreRuleForLatest	**ON** | OFF	If ON - allows you to select a non-latest revision for creation of a pending revision, in the Manage Change command (although it displays a warning icon). This supports design branching. If OFF - attempting to create a pending revision from a non-latest revision will not work (Displays a stop sign).
NumberingMode	**Allowable values**: NONE Pro/E UG CATIA SOLIDWORKS SOLIDEDGE	If set to a value other than NONE, when creating a new CAD model using the Agile > New command, the appropriate CAD extension for the given CAD system will be appended to the autonumber from Agile, to create the Document number. The value of NumberingDelim is also inserted (see next entry). For example, if this is set to SOLIDWORKS, and the next available autonumber in the New dialog is P12345, the Document number will be P12345.SLDPRT (instead of P12345).
NumberingDelim	**Examples**: "." (period) "-" (dash)	You insert the value between the autonumber value and the value of the NumberingMode parameter (see above).
NumberingCreateParts	ON | **OFF**	If ON - creates a Part object automatically during the New command, and links it to the created Document. The Part number will match the selected autonumber. In the example above, the Part will be P12345 and the Document will be P12345.SLDPRT.
InitialECOClass	**Allowable values**: NONE A name of a Change sub-class	If set to something other than NONE, when Documents are first created in the EC Client (using New or Save), they will be automatically placed on a Change object, using this sub-class.
InitialRev	**Examples**: "1" "A" "-" (dash)	If InitialECOClass is not NONE, this will be the value of the initial pending revision created for Documents during New or Save.
InitialPartECOClass	**Allowable values**: 32105 A name of a Change sub-class	If set to something other than NONE, when Parts are first created in the EC Client (using New, Save, or Create Item/BOM), they will be automatically placed on a Change object, using this sub-class.

Name	Value	Remarks
InitialPartRev	**Examples**: "1" "A" "-" (dash)	If InitialPartECOClass is not NONE, this will be the value of the initial pending revision created for Parts during New, Save, or Create Item/BOM.
AutoCreateECO	ON | **OFF**	If ON, automatically creates Change objects and assigns revision values based on values coming from CAD. Used for dataload operations.
AddPartToECO	ON | **OFF**	If ON, and if Parts are automatically generated using the NumberingCreateParts option, then the Parts will automatically be assigned as affected items on the same Change as the Documents, in the Manage Change command.
assignDrawingToPart	ON | **OFF**	If ON, will link the 2D drawing Document as well as the 3D model Document into the Part BOM, during the Create Item/BOM command. Is only valid if disableDocumentPartLink is OFF.
disableDocumentPartLink	ON | **OFF**	If ON, the Create Item/BOM command will not link the Documents into the Part BOM, but will instead set attributes on each object, referencing each other (for 9.2.1) or Relationship links (for 9.2.2).
checkLocalStamp	**0** | 1	ADVANCED OPTION - Contact Agile Solution Delivery for assistance
overWrite	**ASK** | ON | OFF	Sets the default value of the Assignment selector found at the top of the Save dialog ASK = Confirm ON = Assign and Overwrite OFF = Don't Assign or Overwrite
autoSaveBatch	**0** | 1	If set to 1- saves in batch mode with no client interaction. Used for dataload operations.
mapAttributesToFileFolder	ON | **OFF**	If ON, will set attributes on the File Folder object attached to the Document, that are equal to the attributes set on the Document. This requires configuration of the File Folder section of the CAXClient.xml file. This is used to better control access at the File Folder level.
getFilesSelective	**0** | 1	If set to1- Enables an additional check box column in the Load dialog that allows users to select which files to load. **CAUTION**: Only some CAD systems support

Name	Value	Remarks
		this capability.
baselineColumn	0 | **1**	If set to 0 - Removes the Create Baseline check box column in the Save dialog, preventing users from creating Baselines. If set to 1- Displays a save Baseline column in Save dialog.
DefaultUnit	**Metric** | Inches	Sets the default units for the New dialog
LoadOptionsLatest	**Allowable values (one or more):** LATEST_PENDING RELEASED ASSAVED	Sets valid options for structure resolution during Load command, for a selected pending revision. The user will receive a pop-up dialog prompting them to select an option.
LoadOptionsNonLatest	**Allowable values (one or more):** LATEST_PENDING RELEASED ASSAVED	Sets valid options for structure resolution during Load command, for a selected released revision. The user will receive a pop-up dialog prompting them to select an option.
LoadSimplified	0 | 1 | NONE	Enables the "Simplified Reps" check box in the Load dialog, which is only valid for Pro/ENGINEER. If set to 0 - the check box is enabled and not checked. If set to 1 - the check box is enabled and checked. If set to NONE - the check-box is disabled.
ViewablesDrawing	**Example:** TIF,PDF,CGM,PDF,JPG,HPGL,PS,DXF,DWG,EDRW,CALS	The list of allowable viewable filetypes for Drawings that will be on display in the Save Preferences dialog. **Note**: Further configuration may be necessary to generate the desired filetype.
ViewablesModel	**Example:** JT,CGR,WRL,STL,X_T,X_B,STEP,IGES,EPRT,EASM,ED, EDP,EDZ,3DXML,SAT,VDA	The list of allowable viewable filetypes for Models (parts and assemblies), that will display in the Save Preferences dialog. **Note**: Further configuration may be necessary to generate the desired filetype.

Name	Value	Remarks
AutoSubmitXYZ (Where XYZ is the name of a Change sub-class. Example: DVO)	**Examples:** Released Submitted	Gives the target workflow status for given Change sub-class, for use with "Submit Changes" option in Save dialog. The workflow must be configured with a valid path using "default next status" to target workflow status.
Viewer	**Allowable values:** ON OFF LOCAL	Controls the viewing function from the EC Client. If set to OFF - view button is disabled. If set to LOCAL - enables the view button and supports launching file attachments using client-side MIME type applications. If set to ON - view button is enabled and uses the Agile Viewer.
PublishModelFiles	**Allowable Values** PRT, JT, CGM, EPRT, EASM, SLDASM, SLDPRT	Sets the filetypes from 3D Design objects (parts and assemblies) which attach to Part objects during the Create Item/BOM process.
PublishDrawingFiles	**Allowable Values** EDRW, CGM, TIF, PDF, PRT	Sets the filetypes from CAD drawings (parents of 3D models) which attach to Part objects during the Create Item/BOM process.
Label Types	**Examples** DV DC	
LabelApproversXYZ (where XYZ is the name of the Label Type. Example: DV)	**Examples:** DV Approvers	Name of the approver group for a specific label type. When you choose this label type, Users within this group are assigned to the Routing Slip
LabelObserversXYZ (where XYZ is the name of the Label Type. Example: DV)	**Examples:** DV Observers	Name of the observer group for a specific label type. When you choose this label type, Users within this group are assigned to the Routing Slip.
LabelAutonumber XYZ (where XYZ is the name of the Label Type. Example:DV)	**Examples:** DV Label	Name of the Autonumber to use for the specific label type. **Note**: The prefix of the autonumber must begin with label value. For example, the autonumber prefix for DV must begin with "DV".
LabelUseRevisionLogicXYZ (where XYZ is the name of the Label Type. Example:DR	0 | 1	For this label type, will the major revision increment upon the next checkout. 0=No, 1=Yes.

Name	Value	Remarks
DesignSequenceRevision	Examples; A,B,C,D,E,F,G,H,I,J,K,L,M, N,O,P,Q,R,S,T,U,V,W,X,Y, Z	Revision sequence for the major revision component of the Design Revision field.
DesignVersionSequence	Either the value "NUMERIC" or an actual sequence. (similar to above)	Revision sequence for the minor revision component of the Design Revision field.
DesignVersionIndicator	Examples: "v" A v1 "" A1	Separator text between major and minor revision components of the Design Revision field.
AllApprovalsRequired	0 | **1**	Controls ability to check out Design based on existing Routing Slip approval. If set to 1, all approvers have to approve. If set to 0, only one approval (with no rejections) is required.
CheckOutWhileInApproval	**0** | 1	If set to 1 - Allows Users to check out the next Design version even if the current one is still in approval. If set to 0 - Users cannot check out until approval occurs. **Note**: If check out occurs, prior to approval, major revisioning will not occur.
DFM	**0** | 1	If set to 1 - enables DFM capability If set to 0 - disables DFM capability.
usePlmApi	**0** | 1	If set to 1 – enables PLM API for WAN usage If set to 0 – disables PLM API for WAN usage
useStreamCompression	**0** | 1	If set to 1 – enables compression for PLM API If set to 0 – disables compression for PLM API
PublishBOMStructure	**ON** | OFF	If set to OFF, the part BOM will not be published only the relations between models and parts
generateAssemblyThumbnails	**0** | 1	If set to 1 – trigger generating of thumbnails for assemblies If set to 0 – do not trigger generating of thumbnails for assemblies
skipGetItemProperties	**0** | 1	If set to 1 – does not return attributes from associated Parts, thereby improving performance in cases where Part attributes are

Name	Value	Remarks
		not used. If set to 0 – return attributes from Parts

fileOperation Parameters

The tag <fileOperation> indicates the parameters in the section. They control the behavior of the Save and Load operations. Configure them to match the process requirements at your site.

fileOperation section

Name Parameter	Value	Remarks
get		Parameters relating to the "Load" command
autoCheckout		"0" no automatic checkout (reserve) "1" automatic reserve
URLcopy	0	Obsolete
put		Parameters relating to the "Save" command
saveOption		"1" enable save if object isn't checked out by another user "2" enable save if object isn't checked out by another user and the object has not been changed in PLM (not out-of-date) "3" obsolete (out-of-date) "4" enable save only if object is checked out by current user
timeStampField	6151	Obsolete
viewable	VIEWABLE	The value used to identify source (native) CAD files, in the Attachment Type attribute of the Attachments tab on the Document.
source	SOURCE	The value used to identify viewable files, in the Attachment Type attribute of the Attachments tab on the Document.
baseline	BASELINE	The value used to identify baselines, in the Attachment Type attribute of the Attachments tab on the Document.
viewables	,CGM,TIF,TIFF,JPG,JPEG,GIF,PDF,JT, X_T,CGR,EPRT, EASM,EDRW	A comma-separated list of file extensions flagged as viewable file types. Make sure there is a comma at the beginning and end of the list.
keepCheckout	**0** | 1	Default pre-selection of checkboxes for objects in save dialog. "1" - indicates the file is still checked out after the Save command is completed.

Name Parameter	Value	Remarks
checkInDefault	0 | **1**	Default pre-selection of checkboxes for objects in save dialog, "1" - indicates the filefolder is checked in after file transport to make changes visible to other users.

Setting EC Client Data Model

The EC Client also allows you to switch between the Document and Design object based on the properties set in your configurations file. Since these settings pre-exist in the CAXClient_Designs.xml and CAXClient_Documents.xml, respectively, in order to switch between data models, it is necessary only to change the setting in the CaxClient.bat file as described in the beginning of this chapter.

Setting for Design objects:

<createObject defaultClass="FILEFOLDER"/>

<setProperties defaultClass='FILEFOLDER'/>

Setting for Document objects:

<createObject defaultClass="DOCUMENT"/>

<setProperties defaultClass="DOCUMENT"/>

Agile Data Model Parameters

The CAXClient_{type}.xml file defines the types of objects that are used by Agile to store your CAD data, and also which attributes from those objects are accessed. The object definitions are within the <objectProperties> section, and consist of <subclass> definitions which define tabs and attributes.

There are three main <subclass> sections: File Folder (Design), Document, and Item.

- The File Folder subclass section is used when using Design objects to manage CAD files.
- The Document subclass section is used when using Document objects to manage CAD files.
- The Item subclass defines the object class for the Part BOM (the "Product Structure").

Subclass definitions appear as follows:

```
<subclass name="DOCUMENT" type="2" id="24147">
```

Where type is the Agile object type, and id is the Agile sub-class ID. By setting the id equal to a specific Agile sub-class (such as CAD Model), you can control which Document sub-class appears in the EC Client by default.

Note The drawback of this setting is that the sub-class ID cannot be determined directly through the Agile administrator client. However, when you run EC Client it will output a list of all object sub-class IDs in the CaxClient.log file.

Tab definitions appear as follows:

<table name="TITLE_BLOCK" id="0">

Within each tab, attribute definitions appear as follows:

<attribute name="DESCRIPTION" id="1002" set="1" get="1" mandatory="1"/>

This has the following parameters:

id = the Agile base ID of the attribute

set = 0 or 1 where 1 means to include this attribute in UI for creating or setting object properties

get = 0 or 1 where 1 means to return this attribute to the CAD Connector

mandatory = 0 of 1 where 1 means that the user cannot continue until a value is entered.

Note The setting of "mandatory" is independent of the Agile server definition of "required" fields. For best results, mark those fields that are required by the Agile server, also as mandatory for the EC Client. Bold text in the EC Client user interface indicates both mandatory and required fields.

You define the Agile data model for EC by determining which attributes to set within the CAXClient.xml file, and then configuring these attributes in Agile. These attributes are set in the Agile Java Client.

Data Model Configuration for Design Objects

If you are using Design objects as the basis of storing CAD files, most data model configurations are already in place. The main task is to re-configure the "Model Name" attribute from a text attribute to a multitext attribute.

- Rename the existing “Model Name” attribute (ID 2000008376) to a name of your choice.
- Rename the “Structure MultiText01” attribute (ID 2000008365) to “Model Name”

In the FILEFOLDER subclass section of the CAXClient.xml file, set the CAX_FILENAME entry to the new ID as follows:

```
<attribute name="CAX_FILENAME" id="2000008365" set="0" get="0"/>
```

The CAXClient_{type}.xml file defines the types of objects that are used by Agile to store your CAD data, and also which attributes from those objects are accessed. The object definitions are within the <objectProperties> section, and consist of <subclass> definitions which define tabs and attributes.

There are three main <subclass> sections: File Folder (Design), Document, and Item.

- The File Folder subclass section is used when using Design objects to manage CAD files.

- The Document subclass section is used when using Document objects to manage CAD files.
- The Item subclass defines the object class for the Part BOM (the "Product Structure")

Subclass definitions appear as follows:

<subclass name="DOCUMENT" type="2" id="24147">

Where type is the Agile object type, and id is the Agile sub-class ID. By setting the id equal to a specific Agile sub-class (such as CAD Model), you can control which Document sub-class appears in the EC Client by default.

Note The drawback of this setting is that the sub-class ID cannot be determined directly through the Agile administrator client. However, when you run EC Client it will output a list of all object sub-class IDs in the CaxClient.log file.

Tab definitions appear as follows:

<table name="TITLE_BLOCK" id="0">

Within each tab, attribute definitions appear as follows:

<attribute name="DESCRIPTION" id="1002" set="1" get="1" mandatory="1"/>

This has the following parameters:

id = the Agile base ID of the attribute

set = 0 or 1 where 1 means to include this attribute in UI for creating or setting object properties

get = 0 or 1 where 1 means to return this attribute to the CAD Connector

mandatory = 0 of 1 where 1 means that the user cannot continue until a value is entered.

Note The setting of "mandatory" is independent of the Agile server definition of "required" fields. For best results, mark those fields that are required by the Agile server, also as mandatory for the EC Client. Bold text in the EC Client user interface indicates both mandatory and required fields.

You define the Agile data model for EC by determining which attributes to set within the CAXClient_{type}.xml file, and then configuring these attributes in Agile. These attributes are set in the Agile Admin Client. The following two sections show the steps for the two different EC data models.

Data Model Configuration for Design Objects

If you are using the Design object with your EC CAD connectors to manage CAD data, configure Agile PLM using the following steps:

- Check the configuration of the Design object attributes, following the "Design Object" section in

Appendix B.

- If using Agile 9.2.2.x, make sure to follow the instructions listed for Note 1, to fix a configuration problem with the Model Name attribute on the Design Structure tab.

- Configure the required Part object attributes, following the "Part Object" section in Appendix B.
- For both the Design and Part objects, make sure that any privileges which reference these objects are properly set such that the "Applied To" field of the privileges includes all the required attributes.

Data Model Configuration for Document Objects

If you are using the Document object with your EC CAD connectors to manage CAD data, configure Agile PLM using the following steps:

- Create a new Document sub-class to be used by the MCAD data. For example this could be called "Model" or "CAD Model".
- Edit the CAXClient_Documents.xml file, and modify the ID of the Document sub-class line to match the actual ID of the new sub-class. Do NOT change the "Document" name in the XML file to match the sub-class name. It should remain as "Document".
  - The ID can be determined by running the EC Client and checking the CaxClient.log file.
- Configure the required attributes for this new Document sub-class, following the "Document Object" section in Appendix B.
- Configure the required Part object attributes, following the "Part Object" section in Appendix B.
- For both the Document and Part objects, make sure that any privileges which reference these objects are properly set such that the "Applied To" field of the privileges includes all the required attributes.

EC Client Log File

Each time an EC Client session starts, a log file called CaxClient.log is created in the user's working directory. This log file can be viewed or printed to help debug any configuration problems that might occur. Typically Oracle Support will request this file when helping to resolve any issue with Agile EC.

Agile Roles and Privileges

For detailed information on Agile Roles and Privileges, see Agile PLM Administrator Guide. However, given below are some aspects critical to proper operation of the EC Client.

It is typical to create a role or roles for those CAD users who will create objects in Agile through EC CAD Connectors. They should have privileges to modify objects from any Document or Design subclass that you have defined to be used to store CAD data. Other users should not have Modify privileges to these subclasses. Typically non-CAD users would not have Modify privileges to these subclasses.

When using the Design object for storing CAD data, there is a standard role called "Design Engineer" that has a good basic set of privileges for managing Designs. This can be modified as

desired.

The necessary privileges for performing functions supported by the EC CAD connectors when using the Design object are as follows:

EC Function	Object Class	Privileges
Save, Load, Manage Change	Designs	Discover, Create, Read, Modify, Delete, Checkout, Checkin, Cancel Checkout, Print, Get, View
Create Item/BOM	Parts	Discover, Create, Read, Modify
	Eng Changes	Discover, Create, Read, Modify, Change Status

Pay particular attention to the “Applied To” field of these privileges, since as you enable new attributes aspart of the data model configuration
they will not be part of the selected set of attributes by default.

EC Client Customizing

The MCAD systems call CAD Connectors. The MCAD system delivers and receives data to and from Agile using I Ag I Parameter objects. There are three different ways to modify this dataflow:

- Add your own ActionHandler to CAD Connector
- Extend your own Connector from CAD Connector
- Extend your own Connector from CAD Connector and add your own ActionHandler to CAD Connector (combination of both steps above).

1. **To add your own ActionHandler**:
    1. Create your own JAVA class, which extends CAXAction and implements ICAXAction. Please see example of com.Agile.cax.custom.CustomAction in CustomConnector.jar. ICAXAction implements methods, called within the workflow of, save and load processes. Here you can modify the behavior using pre-defined entries.
    2. Create your own JAVA-jar-File, which contains your class. Compile and link against the CAXConnector.jar, CAXClient.jar, CaxAglDataTypes.jar and AgileAPI.jar.
    3. Ensure your jar-file and all needed sub-jars are contained in your java class path when the CAXClient is started. You can check cax_client.bat for that.
    4. Register your ActionListener for CAXConnector in CAXClient.xml. If your class is com.Agile.cax.custom.CustomAction, the line looks like this:

```
<drsExtension logicalName="com.Agile.cax.CAXConnector"
className="com.Agile.cax.CAXConnector"
actionHandler="com.Agile.cax.custom.CustomAction" >
```

Note You can only add this ActionListener to the mentioned line. Ignore all other drsExtension tags.

    After restart of CAXClient you see your registered handler in stdout.

2. **To extend CAXConnector:**

1. Create your own JAVA class, which extends CAXConnector. Please see the example of com.Agile.cax.custom.CustomConnector in CustomConnector.jar. This CAXConnector implements all methods which are called directly by MCAD. Here you can modify the behavior by changing the dataflow or making a complete replacement of single methods (overloading).
2. Please follow 1.1 and 1.2 for creating and using your jar file in CAXClient.
3. Register your connector in CAXClient.xml by creating your own drsExtension tag (see CustomConnector for example). Assume you implemented your own Connector in class com.Agile.cax.custom.CustomConnector, then the lines look like this:

```
<drsExtension logicalName="com.Agile.cax.custom.CustomConnector"
```

```
className="com.Agile.cax.custom.CustomConnector" ></drsExtension>
```

4. Register your connector and the changed methods in CAXClient.xml. Each method called by MCAD has a name, which is mapped to a JAVA class in XML: <classMapping methodName="symbolicName" className="containingClass">

If you implemented your own method "searchObject" in your connector com.Agile.cax.custom.CustomConnector, the XML entry has to be modified like this:

```
<classMapping methodName="searchObject" className="
com.Agile.cax.custom.CustomConnector">
```

The available methodName's are predefined and called by CAD.

3. **To extend CAXConnector and ActionHandler combined:**
    1. In order to extend your own Connector and use an ActionHandler just combine the steps in 1 and 2 above.

ActionHandlers only work for a CAXConnector and can only be registered there. A "CustomConnector" may not have its own ActionHandler, but you can register this ActionHandler at the CAXConnector. Always first try to use own ActionHandlers instead of own Connectors, because making your own Connector is much more complex than making your own ActionHandler

Appendix A

Tips and Tricks

Tip #1 **Reloading INI file**	If you have a CAD connector running and you need to make a change to the INI configuration file (e.g. 3DCADMapping.ini for SolidWorks, AcpCustomer9.ini for Pro/E), you can have the changes take effect without restarting the CAD system. Once the INI file updates are saved, simply pick the "Agile > About" command within the CAD system. This reloads the configuration information from the INI file.
Tip #2 **Removing file tracking**	EC uses a few mechanisms to track the fact that CAD files have already been stored in PLM. There are times when you may purposely want to defeat this tracking, for example to repeat a demo. There are two basic mechanisms: Master Tracking File – Same for all CAD systems .caxsession.xml file, located in C:\Documents and Settings\{username} File-based Tracking – Different depending on the CAD system For SolidWorks and Solid Edge: .CID file created for each tracked CAD file For Pro/E, UG NX, and CATIA: Attribute saved inside CAD file (usually AGILEID) Removing tracking: Stop and CAD system and EC Client Delete the .caxsession.xml file For SolidWorks or Solid Edge, delete the .CID files of the CAD files for which you wish to remove tracking. For all other systems, run the CAD system and delete the AGILEID attribute from each file. However, if the CAD files were originally saved from outside the EC working directory, the AGILEID attribute will not be present so no further work is required.
Tip #3 **Resizing EC Client Dialogs**	If an EC Client dialog (such as Save or Load) is extending outside the frame of the EC Client window itself, just double-click on the title bar of the dialog and it will resize to fit within the client window.
Tip #4 **Save Preferences Defaults**	In Save Preferences, when entering a Default value, make sure to click out of the data entry field or the value will not be saved.
Tip #5 **Starting the EC Client**	The SolidWorks and Solid Edge connectors do not by default start the EC Client when the CAD system is started. If it is desired to do this, you can create a Windows batch file to start both the CAD system and the EC Client concurrently.

Appendix B

PLM Data Model Configuration

This Appendix includes the following:

This appendix lists the necessary settings to make in the PLM data model for using EC MCAD Connectors.

Design Object

The following table shows the necessary attributes to configure when using Design objects as the basis of MCAD data management. Using the Agile Admin client, check to make sure that all attribute marked as Required in this table are configured and set to visible. If a specific attribute needs to be changed to a different base ID due to customer requirements, see the section entitled "Changing an Attribute to a Different Base ID" in the Getting Started chapter of this document.

For the Design object, all of these attributes are pre-configured in the standard database dump, other than what is described in the notes below.

		Attribute	Base ID	Required	Agile Name	CAXClient.xml ID
Designs	Page Two	PAGE_TWO.TEXT01	2007	X	Design System	CAX_CRE_SYSTEM
		PAGE_TWO.MULTITEXT10	2017	X	CAD Filename	CAX_FIL_NAME
		PAGE_TWO.MULTITEXT20	2018		Old CAD Filename	CAX_FIL_OLD_NAME*
		PAGE_TWO.TEXT03	2009	X	Filetype	CAX_TYPE
		PAGE_TWO.TEXT04	2010	X	Subtype	CAX_SUBTYPE
		PAGE_TWO.TEXT05	2011	X	Family	CAX_FAM
		PAGE_TWO.TEXT06	2012	X	Variant	CAX_VAR
		PAGE_TWO.TEXT07	2013		Drawing Name	CAX_DWG_NAME*
		PAGE_TWO.TEXT08	2014		Frame ID	CAX_FRAME_ID*
		PAGE_TWO.TEXT09	2015		Name Format	CAX_NAM_FMT*
		PAGE_TWO.TEXT10	2016		Project Name	CAX_PROJECT*
		PAGE_TWO.TEXT12	1302	X	Part Number	CAX_PART
		PAGE_TWO.MULTITEXT32	1332	X	Model Type	CAX_MODEL_TYPE
		PAGE_TWO.MULTITEXT33	1333	X	Model Reference	CAX_MODEL_REF
		PAGE_TWO.MULTITEXT34	1334	X	Link Type	CAX_LINK_TYPE
		PAGE_TWO.MULTITEXT35	1335	X	Link Reference	CAX_LINK_REF
		PAGE_TWO.TEXT17	1307		TDM Version	TDM_VERSION*
		PAGE_TWO.TEXT18	1308		TDM Revision	TDM_REVISION*
	Structure	FOLDER_STRUCTURE.Filename (note 1)	2000008376	X	Model Name	CAX_FILENAME
		FOLDER_STRUCTURE.Identifier	2000008377	X	Identifier	CAX_IDENT
		FOLDER_STRUCTURE.Component	2000008378	X	Component	CAX_COMPONENT
		FOLDER_STRUCTURE.Reference	2000008379	X	Reference	CAX_REFERENCE
		FOLDER_STRUCTURE.Configuration	2000008380	X	Configuration	CAX_CONFIG
	WhereUsed - Design	There is no need to configure Where Used attributes. Ignore the attribute settings in the CaxClient_Designs.xml file				
	Relationships	RELATIONSHIP.TEXT01	5846	X	Link Type	N/A
	Files	FILES.CATEGORY	2000008509	X	File Category	N/A
				List Values:		
				Source		
				Viewable		
						*In file but commented out

To insure that the Design class can be access by EC correctly, it is necessary to check all privileges associated with the Design object to make sure that all the above required attributes are selected in the "Applied To" list of each privilege.

Document Object

The following table shows the necessary attributes to configure when using Document objects (DocuBOMs) as the basis of MCAD data management. Using the Agile Admin client, create a Document sub-class to be used for MCAD (e.g. "CAD Model"), and define all attributes marked as Required in this table. This needs to be done manually, since this is not pre-configured in the standard PLM database dump. If a specific attribute needs to be changed to a different base ID due to customer requirements, see the section entitled "Changing an Attribute to a Different Base ID" in the Getting Started chapter of this document.

		Attribute	Base ID	Required	Agile Name	CAXClient.xml ID
Documents	Page Three	PAGE_THREE.TEXT11	1585	X	CAD System	CAX_CRE_SYSTEM
		PAGE_THREE.TEXT12	1586	X	CAD Filename	CAX_FIL_NAME
		PAGE_THREE.TEXT13	1587		CAD Old Filename	CAX_FIL_OLD_NAME*
		PAGE_THREE.TEXT14	1588	X	CAD Timestamp	CAX_TIMESTAMP
		PAGE_THREE.TEXT15	1589	X	CAD Type	CAX_TYPE
		PAGE_THREE.TEXT16	1590	X	CAD Subtype	CAX_SUBTYPE
		PAGE_THREE.TEXT17	1591	X	CAD Family	CAX_FAM
		PAGE_THREE.TEXT18	1592	X	CAD Variant	CAX_VAR
		PAGE_THREE.TEXT19	1593		CAD Drawing Name	CAX_DWG_NAME*
		PAGE_THREE.TEXT20	1594		CAD Frame ID	CAX_FRAME_ID*
		PAGE_THREE.TEXT21	1595		CAD Name Format	CAX_NAM_FMT*
		PAGE_THREE.TEXT22	1596		CAD Project Name	CAX_PROJECT*
		PAGE_THREE.TEXT23	1597	X	Part Number	CAX_PART
		PAGE_TWO.MULTITEXT31	1570		CAD File Path	CAX_FIL_PATH*
		PAGE_TWO.MULTITEXT32	1571		CAD Old File Path	N/A
		PAGE_THREE.TEXT01	1575	X	CAD Model Type	CAX_MODEL_TYPE
		PAGE_THREE.TEXT02	1576	X	CAD Model Reference	CAX_MODEL_REF
		PAGE_THREE.TEXT03	1577	X	CAD Link Type	CAX_LINK_TYPE
		PAGE_THREE.TEXT04	1578	X	CAD Link Reference	CAX_LINK_REF
	BOM	BOM.MULTITEXT30	1341	X	CAD Filename	CAX_FILENAME
		BOM.MULTITEXT31	1342	X	CAD Revision	CAX_TRANSFORM
		BOM.TEXT01	2175	X	CAD Ident	CAX_IDENT
		BOM.TEXT02	2176	X	CAD Component	CAX_COMPONENT
		BOM.TEXT03	2177	X	CAD Reference	CAX_REFERENCE
		BOM.TEXT04	2178	X	CAD Config	CAX_CONFIG
	WhereUsed -and- Pending Change Where Used	BOM.MULTITEXT30	1413	X	CAD Filename	CAX_FILENAME
		BOM.MULTITEXT31	1414	X	CAD Revision	CAX_TRANSFORM
		BOM.TEXT01	2216	X	CAD Ident	CAX_IDENT
		BOM.TEXT02	2217	X	CAD Component	CAX_COMPONENT
		BOM.TEXT03	2218	X	CAD Reference	CAX_REFERENCE
		BOM.TEXT04	2219	X	CAD Config	CAX_CONFIG
	Relationships	RELATIONSHIP.TEXT01	5846	X	Link Type	N/A
	Attachments	ATTACHMENT_MAP.ATTACHMENTTYPE	4681	X	Attachment Type	N/A
				Add these list values:		
				SOURCE		
				VIEWABLE		

*In file but commented out

To insure that the Document sub-class can be access by EC correctly, it is necessary to check all privileges
associated with the sub-class to make sure that all the above required attributes are selected in the

"Applied To" list of each privilege.

Part Object

The following table shows the necessary attributes to configure when using Parts objects with EC MCAD Connectors, specifically with the "Create Item/BOM" command which publishes Part BOMs. This configuration is necessary when using either Designs or Documents as the basis of the MCAD data management. Using the Agile Admin client, define all attributes marked as Required in this table. This needs to be done manually, since this is not pre-configured in the standard PLM database dump. If a specific attribute needs to be changed to a different base ID due to customer requirements, see the section entitled "Changing an Attribute to a Different Base ID" in the Getting Started chapter of this document.

		Attribute	Base ID	Required	Agile Name	CAXClient.xml ID
Parts	Page Two	PAGE_TWO.TEXT11	1301	X	CAD System	CAX_CRE_SYSTEM
		PAGE_TWO.TEXT12	1302	X	CAD Filename	CAX_FIL_NAME
		PAGE_TWO.TEXT13	1303		CAD Old Filename	CAX_FIL_OLD_NAME*
		PAGE_TWO.TEXT14	1304		CAD Timestamp	CAX_TIMESTAMP*
		PAGE_TWO.TEXT15	1305	X	CAD Type	CAX_TYPE
		PAGE_TWO.TEXT16	1306	X	CAD Subtype	CAX_SUBTYPE
		PAGE_TWO.TEXT17	1307		CAD Family	CAX_FAM*
		PAGE_TWO.TEXT18	1308	X	CAD Variant	CAX_VAR
		PAGE_TWO.TEXT19	1309		CAD Drawing Name	CAX_DWG_NAME*
		PAGE_TWO.TEXT20	1310		CAD Frame ID	CAX_FRAME_ID*
		PAGE_TWO.TEXT21	1311		CAD Name Format	CAX_NAM_FMT*
		PAGE_TWO.TEXT22	1312		CAD Project Name	CAX_PROJECT*
		PAGE_TWO.TEXT23	1313	X	Published From	CAX_PUBLISHED
		PAGE_TWO.MULTITEXT31	1331		CAD File Path	CAX_FIL_PATH*
		PAGE_TWO.MULTITEXT32	1332		CAD Old File Path	CAX_FIL_OLD_PATH*
		PAGE_TWO.TEXT01	2007	X	CAD Model	CAX_DOC
		PAGE_TWO.TEXT02	2008	X	CAD Model	CAX_DOC_CONFIG
	BOM	BOM.MULTITEXT30	1341	See note 1	CAD Filename	CAX_FILENAME
		BOM.MULTITEXT31	1342	X	CAD Revision	CAX_TRANSFORM
		BOM.TEXT01	2175	X	CAD Ident	CAX_IDENT
		BOM.TEXT02	2176	See note 1	CAD Component	CAX_COMPONENT
		BOM.TEXT03	2177	See note 1	CAD Reference	CAX_REFERENCE
		BOM.TEXT04	2178	X	CAD Config	CAX_CONFIG
	WhereUsed -and- Pending Change Where Used	BOM.MULTITEXT30	1413	See note 1	CAD Filename	CAX_FILENAME
		BOM.MULTITEXT31	1414	X	CAD Revision	CAX_TRANSFORM
		BOM.TEXT01	2216	X	CAD Ident	CAX_IDENT
		BOM.TEXT02	2217	See note 1	CAD Component	CAX_COMPONENT
		BOM.TEXT03	2218	See note 1	CAD Reference	CAX_REFERENCE
		BOM.TEXT04	2219	X	CAD Config	CAX_CONFIG
						*In file but commented out

To insure that the Part class can be access by EC correctly, it is necessary to check all privileges associated with the class to make sure that all the above required attributes are selected in the "Applied To" list of each privilege.

This page is blank.