Agile Product Lifecycle Management

MCAD Connectors for Agile Engineering Collaboration User Guide

v2.4

Part No. E11888-01
January 2008

Copyright and Trademarks

CONTENTS

Preface

The Oracle|Agile 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 Oracle|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 Oracle|Agile Documentation folder available on your network from which you can access the Oracle|Agile 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 the Agile Web Client and the Agile Java Client. If you need additional assistance or information, please contact support (http://www.oracle.com/agile/support.html) (http://www.oracle.com/agile/support.html) for assistance.

Note	Before calling Agile Support about a problem with an Oracle|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.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at http://www.oracle.com/accessibility/ http://www.oracle.com/accessibility/.

Readme

Any last-minute information about Oracle|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.

Accessibility of Links to External Web Sites in Documentation

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

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
- Agile (see Introduction for supported versions) has been successfully installed on an accessible server (the prerequesites 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 consult the *Agile Installation Worksheet* and other ocumentation of Agile installation procedures.

- Agile File Management Server is usable and accessible
- A test environment is prepared
- A CAD systems is installed, and can be launched by the test user from the home directory
- Each user must locally install Java Runtime Environment (JRE). JRE 1.4.x must be used, with the exception of CATIA V4 on AIX which uses JRE 1.3.x. JRE 1.5 can still be used on the workstation for other applications; see installation instructions for further details.
- 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

Having fulfilled these prerequisites, it is time to configure your CAD Connector to Agile PLM.

Installing and Configuring Pro/ENGINEER Connector

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

The main steps are:

- Extract files from zip file
- 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 27).

Note The Pro/E 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 Agile Solution Delivery Organization if you need assistance with installation.

Extract Files

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.

Edit 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	Root directory for user data and files (Suggested value: D:\AgilePro or C:\AgilePro)
AcpLang=english	Menu and UI language

Sample values	What this command specifies
AcpJava=C:\j2sdk1.4.2_04\jre	Path to Java Runtime Environment Note: Avoid blanks or spaces in path.
CLI_JRE= 1.4	Use 1.3, 1.4 or 1.5.
AcpProEV=2002	Currently, installed version of Pro/E: ▫ "2006" designates Pro/E Wildfire 3 ▫ "2003" designates Pro/E Wildfire 2 ▫ "2002" designates Pro/E Wildfire ▫ "2001" designates Pro/E 2001
AcpStartProE=D:\proewildfire\bin\proewildfire.bat	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.
EC_VERSION=1.1	EC Client version being used

Edit 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.

Install correct 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.

Create Shortcut to new 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.

Create Toolbar in Pro/E

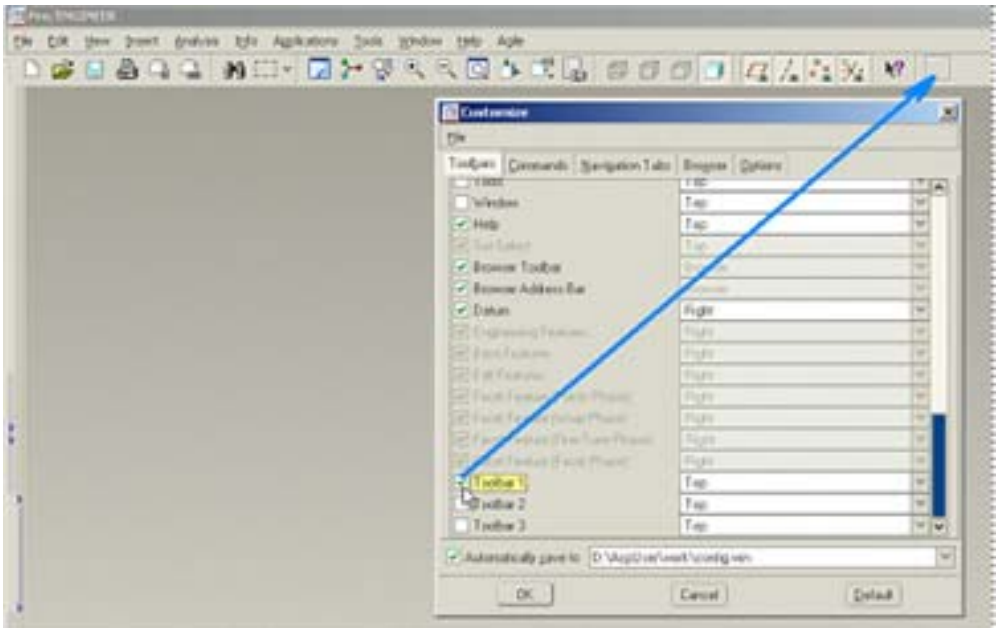
This step is optional, it will create a toolbar that you can use to run the Agile commands, in addition to the Agile menu.

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.

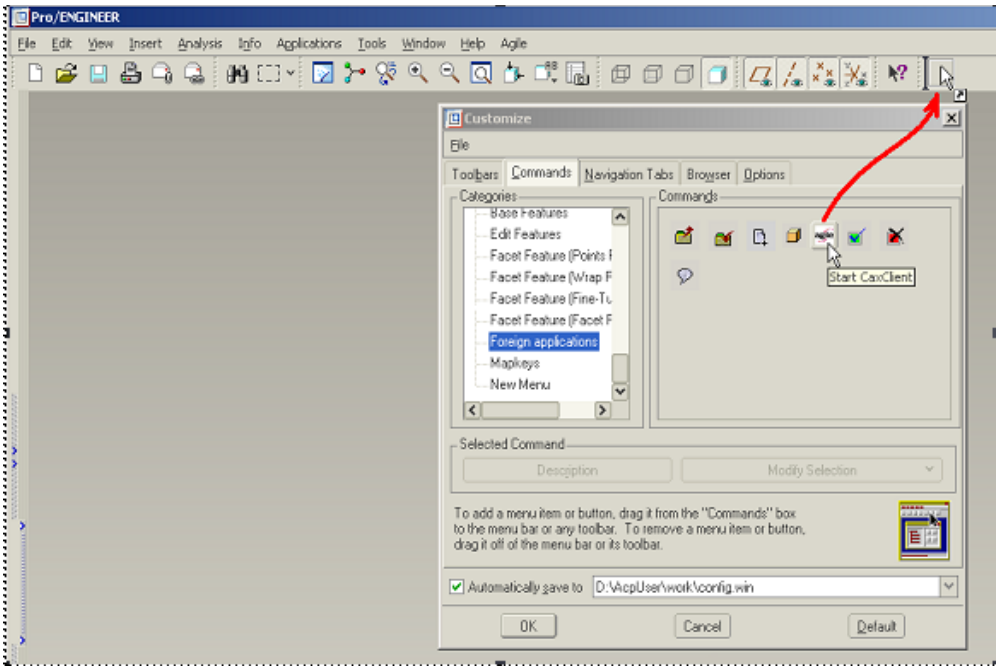
   Figure: Enabling the Toolbar

Figure 1: Toolbar in Pro/E



3. 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 other 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.

Installing and Configuring SolidWorks Connector

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

- Extract files from zip file
- Register library and executable
- Edit some parameters in the startup file
- Edit some parameters in the configuration file
- 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 34).

Extract Files

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.

Register Library and Executable

Navigate to the \AgileEC\acw directory, and double-click on registerSW2007.bat (or 2005 or 2006 as appropriate, depending on your SolidWorks version).

- You will get a popup dialog stating *DllRegisterServer in SolidWorks\agilePLMSW.dll succeeded*. Click OK to continue.
- Then it will register the CaxOleSrv.exe executable. This causes a globe icon to appear in your system tray. Just right-click on the globe and pick Exit.
- The command window will then close.

Set Java Path

You must set the path to the main system Java Runtime in the system PATH variable as follows:

- Right click on the “My Computer” icon on the desktop

- Click Properties from the menu
- Select the Advanced tab, then click Environment Variables...
- In the lower part of the dialog, scroll down until you see the variable “Path”. Select it and click on “Edit…”
- Edit the Variable Value field, and insert the path to your JRE client or hotspot directory, for example C:\Java\j2re1.4.2_04\bin\client
- Click OK three times to exit.

If you have previously installed an older version of the SolidWorks connector on this computer, note that the path to the “Not supported Dll” directory is no longer needed in the system PATH (you can remove it or just leave it).

Edit Startup File

Open the file \AgileEC\acw\Server\Scripts\AgileStartClient.bat in a text editor. Modify the value for ECP_HOME to match the full path to your \AgileEC\acw\Server directory. Modify the value of JAVA_HOME to match your JRE directory.

Edit 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=C:\Program Files\Java\jre1.5.0_07\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\Agile Connector.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.

Sample values	What this command specifies
[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. You must also create this directory on your computer.
[LogFileDir] ; D:\AgileSW\Temp\ ;	The full path of the log file directory. You must also create this directory on your computer.
[AgileURL] ; http://servername:8888/Agile ;	The URL for the EC Client.

Install correct 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\Server.

- 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\Server, overwriting the file already there.
- If you are unable to locate this file, please contact Agile Support.

Set up the Agile menu in SolidWorks

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.

Installing and Configuring Unigraphics Connector

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

- Extract files from zip file
- Edit some parameters in the startup file
- Edit some parameters in the mapping 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. 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 Connector parameters as described in the Unigraphics Connector Administration (on page 46) section.

Note The Unigraphics 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 Agile Solution Delivery Organization if you need assistance with installation.

Extract Files

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.

Edit 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 version (v18, nx1, nx2, nx3, nx4)
set ECU_LANG=eng	User interface language (eng, ger)
set UGII_ROOT_DIR=D:\CAD\UGNX2\UGII	The root directory of the Unigraphics installation directory

Sample values	What this command specifies
set ug_start=D:\CAD\UGNX2\UGII\ugraf.exe	The complete path of the executable file to be started when using Unigraphics
set cax_usr_home=D:	Directory for saving the temporary data files (suggested value: D:\AgileUG or C:\AgileUG)
set EC_VERSION=1.1	EC Client version being used

Install correct 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.

Create Shortcut to new Startup File

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

Installing on Other Computers

Once the Unigraphics 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 setup, Java setup, etc.

Installing and Configuring CATIA V5 Connector

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

- Extract files from zip file
- 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 53).

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 Agile Solution Delivery Organization if you need assistance with installation.

Extract Files

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.

Edit 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

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

Edit 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.

Install correct 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.

Create Shortcut to new 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.

Installing on Other 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.

Installing and Configuring CATIA V4 Connector

This section describes setting up the connection between your Catia V4 CAD application and Agile PLM 9. The main steps are:

- Extract files from tar file
- Edit some parameters in the configuration file
- Edit some parameters in the declaration file
- Install proper AgileAPI.jar file

The installation requires the following file:

acc4_NNNN.tar – Main installation package, where NNNN is the release level

Performing the installation steps described here will enable the Agile menu to appear within Catia V4. 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 V4 Connector parameters as described in the section CATIA V4 Connector Administration (on page 61).

Note The Catia V4 Connector supports only Unix platforms because Catia V4 itself runs only under Unix.

Extract Files

Extract the installation file to the folder location /acc-rt inside the Agile 9 environment, for example. After you finish extracting, you should see the following directory structure below the folder named /acc-rt, which now contains the Connector installation.

Level 1	Level 2	Level 3	Files to be stored there	Description
/bin				Catia - Load (Binary extension of Catia V4)

Level 1	Level 2	Level 3	Files to be stored there	Description
	/aix_a		ACC Agl.a libepshr_eci.a libepshr.a libepclt_eci.a libtcl8.3.a libepshr.so libepshr_eci.so xw_pop	 shared library, containing java extensions for tcl standard agile library standard agile library standard agile library standard tcl library standard agile library standard agile library window handling program
		/jre		standard jre for aix
/solaris_a			ACC Agl.so libepshr_eci.so libepclt_eci.so libtcl8.3.so libepshr.so xw_pop	Catia - Load (Binary extension of Catia V4) shared library, containing java extensions for tcl standard agile library standard agile library standard tcl library standard agile library window handling program
/com			Catia.ksh CaxClient.ksh	sample script to start catia v4 sample script to start the EC 1.1
/dec			Acc.cfg AciAgile9.cfg ACC.dcls	basic configuration file of the connector connection configuration file of the connector sample catia declaration file
/ini			AccMessages.ini AccInitialize.ini AccCustomer9.ini	message resource file basic settings customer specific settings

Level 1	Level 2	Level 3	Files to be stored there	Description
/jar			CAXClientWrapper.jar	EC Client
			CaxAglDataTypes.jar	EC Client
			CaxAglProxy.jar	EC Client
			CustomConnector.jar	EC Client
			CAXConnector.jar	EC Client
			CAXClient.jar	EC Client
			CAXClient.xml	EC Client
			AgileAPI.jar	agile api
			jcert.jar	standard java
			jnet.jar	standard java
			tt.jar	standard java
			jsse.jar	standard java
			jaas.jar	standard java
			xmlParserAPIs.jar	standard java
			pxapi.jar	standard java
			xercesImpl.jar	standard java
/tcl			AccMain.tcl	entry point for all acc functions
/tcllib			AccCustomer.tcl	customer specific extension
			AccUtil.tcl	utility functions
			AglTools.tcl	tcl wrapper for EC Client access
			AccObjects.tcl	acc functions
			mkIndex.tcl	standard tcl
			word.tcl	standard tcl
			safe.tcl	standard tcl
			init.tcl	standard tcl
			history.tcl	standard tcl
			tclIndex	standard tcl
			package.tcl	standard tcl
			ldAout.tcl	standard tcl
			parray.tcl	standard tcl
			auto.tcl	standard tcl
/templates			TEMPLATE.model	template model for "create function"

Agile Menu Structure

AGILE				
	MODEL			
		NEW		
			SINGLE	Create Model (Single)
			REPL_ACT	Create Model (Replace Active)
			ADD_ACT	Create Model (Add Active)
		LOAD		
			SINGLE	Open Model (Single)
			ACTIVE	Open Model (Add Active)
			REPL_ACT	Open Model (Repl Active)
			PASSIVE	Open Model (Add Passive)
		SAVE		Save Model
		SAVE_AS		Save Model with a new number
	ABOUT			
	ITEM			Create Item
	SHOWFORM			Show version information
	UPD_TBLK			Update Titleblock
	CUSTOM			Customer specific extensions (future use)
	AGILE			
		SHOW		Maximize Agile
		START		Agile Start
		MANAGE		Manage change
		CONNECT		Agile connect
		DISCONCT		Agile disconnect

Edit Configuration Files

Open the basic configuration file of the connector /acc-rt/dec/Acc.cfg in a text editor and edit the values as described in table below to match your system configuration.

Parameter name		Sample value (default)	Description
EccCustom	=	AccInitialize.ini	Name of the mapping file

Parameter name		Sample value (default)	Description
CATIA	=	/CATIA424	Root folder of the Catia v4 installation
VERS	=	424	Version number of the Catia v4 installation
ECC_PROJECT	=	WORKSHOP	Internal catia projectname, should be equal to the real Catia v4 project name
PROJECT	=	WORKSHOP	Real Catia v4 project name
CatiaInit	=	/CATIA424/home/adm424/env/YOUR.env	Catia v4 initialization script (part of the Catia v4 standard installation)name and location are customer specific, contact catadm user for details.
JVMDir	=	/usr/java131/jre/bin:/usr/java131/jre/bin/classic	Path to libjvm.so which should be used (1.3.x on AIX and 1.4.x on Windows or Solaris).
AccJava	=	/usr/java131/jre	Path to the java installation which should be used
AccClassPath	=	$AccJava/lib/ext	Path to the jar files of the integration, due to some security restrictionsthis jar files should be placed in the lib/ext folder,otherwise it is necessary to modify the security policity of the java installation.
AccLogPath	=	/tmp	Path where the logfiles should be written to.
AccExtension	=	.so	Extension of the shared library, depends on your operating system: solaris -> .soaix -> .a

Open the connection configuration file of the connector /acc-rt/dec/AciAgile9.cfg in a text editor. This file is similar to that file used in the Catia v5 integration. Edit the values as described in the table below to match your system configuration.

Parameter name		Sample value	Description
AciJVM	=	1	
AgileHostString	=	http://ningi: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.
AgileDefaultUser	=	CAX	Default username that will be used when you run the EC Client, it can be changed interactively.
AgileDefaultPassword	=	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.

Edit Declaration File

Open the sample Catia V4 declaration file of the connector /acc-rt/dec/Acc.dcls in a text editor and edit the values as described in the table below to match your system configuration. The parameters of the first part are only necessary if this is the one and only declaration file.

Table: Parameters in the Catia v4 declaration file Acc.dcls with sample values, which can be removed in a customer environment

Sample values
catia.MODEL_KBYTES.MAX_ACTIVE_INDEX = 1000;
catia.MODEL_KBYTES.MAX_ACTIVE_DATA = 5000;
catia.MODEL_KBYTES.TOTAL_OVERLAY_INDEX = 5000;
catia.MODEL_KBYTES.GLOBAL_EXTENDED_DATA = 800;
catia.SESSION_MANAGER_KBYTES = 1000;
CATFRM.PALETTE.TYPE = 'PERMANENT';
CATUAP.*$ASM.UDMSL.LOGCHN = FALSE;

Table: Parameters in the Catia V4 declaration file Acc.dcls with sample values, which defines the name of Catia v4 load and the Catia v4 context in which this load is available

Sample values
CATCMD.FUNCTION.LOAD3D.ACC = 'ACC';
CATCMD.FUNCTION.LOAD2D.ACC = 'ACC';
CATCMD.FUNCTION.LOADDR.ACC = 'ACC';
CATCMD.FUNCTION.DETSPACE.ACC = TRUE;
CATCMD.FUNCTION.DETDRAW.ACC = TRUE;
CATCMD.FUNCTION.MODSPACE.ACC = TRUE;
CATCMD.FUNCTION.NAME.ACC = 'AGILE';
catia.START_FUNCTION = 'ACC';

Table: Parameters in the Catia v4 declaration file Acc.dcls with sample values, which defines t the Catia v4 model areas the integration is able to deal with

Sample values
alias AGILE_1 =
catia.MODEL = '"/mnts/jeltz/disk2/ecc5/db/model/accwork1", "WORK_ACC1"' ;
alias AGILE_2 =
catia.MODEL = '"/mnts/jeltz/disk2/ecc5/db/model/accwork2", "WORK_ACC2"' ;

You must adapt the physical path according to your real environment and you can modify the logical name. If it is needed to include existing model areas to the integration, you must define an alias name like "AGILE_x" for each of them.

Install correct 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 /jar directory of the installation.

- Search for the file AgileAPI.jar within your site's Agile server installation.
- Copy this file to /jar, overwriting the file already there.
- If you are unable to locate this file, please contact Agile Support.

Installing and Configuring Solid Edge Connector

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
- Register libraries and executable
- Edit some parameters in the startup file
- Edit some parameters in the configuration file
- 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 66).

Extract Files

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.

Register Libraries and Executable

Navigate to the \AgileEC\ace directory, and double-click on registerSE.bat.

- You will get a popup dialog stating *DllRegisterServer in SolidEdge\PLMSEClient.dll succeeded*. Click OK to continue.
- You will get a popup dialog stating *DllRegisterServer in SolidEdge\SEXml.dll succeeded*. Click OK to continue.
- Then it will register the CaxOleSrv.exe executable. This causes a globe icon to appear in your system tray. Just right-click on the globe and pick Exit.
- The command window will then close.

Set Java Path

You must set the path to the main system Java Runtime in the system PATH variable as follows:

- Right click on the “My Computer” icon on the desktop
- Click “Properties” from the menu
- Select the “Advanced” tab, then click “Environment Variables…”
- In the lower part of the dialog, scroll down until you see the variable “Path”. Select it and click on “Edit…”
- Edit the Variable Value field, and insert the path to your JRE client or hotspot directory, for example C:\Java\j2re1.4.2_04\bin\client
- Click OK three times to exit.

Edit Startup File

Open the file \AgileEC\ace\Server\Scripts\AgileStartClient.bat in a text editor. Modify the value for ECP_HOME to match the full path to your \AgileEC\ace\Server directory. Modify the value of JAVA_HOME to match your JRE directory.

Edit 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=C:\Program Files\Java\jre1.5.0_07\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 ;	This provides the paths to all jar files. Edit each path to match your system configuration.
[CheckOutDisk] ; D: ;	The disk drive on the client computer to be used for the working directory.
[CheckOutPath] ; \AgileSE\Work\ ;	Note The path of the working directory Note You must also create this directory on your computer.
[LogFileDir] ; D:\AgileSE\Temp\ ;	The full path of the log file directory Note 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

Install correct 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\Server.

- 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\Server, overwriting the file already there.
- If you are unable to locate this file, please contact Agile Support.

Set up the Agile menu in Solid Edge

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).

Set up the Agile toolbar in Solid Edge

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).

Chapter 2

Administration

This chapter includes the following:

Pro/E Connector Administration

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 ("Installing and Configuring Pro/ENGINEER Connector" on page 2).

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.
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

Parameter name in section [Initialize]		Parameter values	Description
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.
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

Parameter name in section [Initialize]		Parameter values	Description
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
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

SolidWorks Connector Administration

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
[LogFileDir]	Disk and path of log file directory
[AgilePartViewFile]	OBSOLETE. See [Agile9PartViewFileExtensions]
[AgileAssemblyViewFile]	OBSOLETE. See [Agile9AssemblyViewFileExtensions]
[AgileDrawingViewFile]	OBSOLETE. See [Agile9DrawingViewFileExtensions]

Option Name	Usage
[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
[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

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\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
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

Option	Description
[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 (Error! Hyperlink reference not valid.)>
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

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

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

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

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!

Configuring the PlmSWAddin.xml File

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"

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.

Unigraphics Connector Administration

Table: List of all Configuration Files for the Unigraphics 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.

Startup file acu_start.bat

The startup file acu_start.bat contains basic system parameters. It is described fully in the Installing and Configuring Unigraphics Connector (on page 10) section.

Mapping file Ecu.ini

This is the main file for controlling the behavior of the Unigraphics 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
JT	Describes the settings for generation of JT viewable files

Section name	Description
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
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

Where 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
<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]

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
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

Format in section [SaveViewable]		Scriptname
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.

CATIA V5 Connector Administration

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 Acc.cfg

The configuration file Acc.cfg contains basic system parameters. It is described fully in the Installing and Configuring CATIA V5 Connector (on page 12) section.

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
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

Parameter name in Section [Initialize]		Parameter values	Description
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
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 58) 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 58).

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 58).

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

Where 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
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

CATIA V4 Connector Administration

Table: List of Configuration Files for the CATIA V4 Connector

Configuration file	Purpose	Location
Acc.cfg	Basic system configuration	/acc-rt/dec/
AciAgile9.cfg	Connection Configuration	/acc-rt/dec/
Acc.dcls	CATIA V4 Declarations	/acc-rt/dec/
AccMessages.ini	Message resource file	/acc-rt/ini
AccInitialize.ini	Basic settings	/acc-rt/ini
AccCustomer9.ini	Customer specific settings	/acc-rt/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 Acc.cfg

The configuration file Acc.cfg contains basic system parameters. It is described fully in the Installing and Configuring CATIA V4 Connector (on page 15) section

This is the m.ain file for controlling the behavior of the CATIA V4 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 of the file AccInitialize.ini

Parameter name / Description		Sample value (default)
# AccCustomerId #	=	None
# DOCUMENT or FILEFOLDER Agile class name for catia models AccDefaultClass #	=	DOCUMENT
# name of the costumer specific mapping file AccCustomerFile #	=	AccCustomer9.ini

Parameter name / Description		Sample value (default)
# color defintions for the messagebox inside of Catia v4		
white	=	1
red	=	2
green	=	3
blue	=	4
yellow	=	5
AccInformation	=	white
AccWarning	=	yellow
AccError	=	red
AccChange	=	green
#		
# In Catia v4 filnames with some special characters may exist which are not corresponding with the characters of the filename from an operating system view.The following lines define the replace mechanism of these characters.		
AccReplaceInch	=	1
AccReplaceSpace	=	1
AccCharactersToReplace	=	° § $ % & / @
AccReplaceCharacter	=	_
#		
# folder name of template model		
AccTemplateFolder	=	/mnts/jeltz/disk2/ecc5/acc-rt/templates/
#		
# environment variable to obtain catia project name		
AccProject	=	PROJECT
#		
# name scheme to generate Catia file names		
AccSchemeOfFileName	=	%s
AccFileNameValues	=	NUMBER
#		
# customer specific procedure to generate file names		
AccFileNameProcedure	=	AccCustomProcedure
#		

Parameter name / Description		Sample value (default)
# language AccLanguage #	=	English
# name of the message definition file AccMessages #	=	AccMessages.ini
# model area DefaultArea #	=	AGILE_1
# alias -> model area relation(s) from Catia v4 dcls file, e.g. ACC.dcls AGILE_1 AGILE_2 #	 = =	 WORK_ACC1 WORK_ACC2
# modelarea -> physical path relation(s) from Catia v4 dcls file, e.g. ACC.dcls WORK_ACC1 WORK_ACC2 # [eof]	 = =	 /mnts/jeltz/disk2/ecc5/db/model/accwork1/ /mnts/jeltz/disk2/ecc5/db/model/accwork2/

Filename creation

During the first Save into Agile, a new CATIA V4 filename can be created. There are two variables in the file AccInitialize.ini 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.

Mapping file AccCustomer9.ini

This is the main file for controlling attribute mapping in the CATIA V4 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 give a description of the content of the two sections in AccCustomer9.ini.

Table: Description of the section [CatiaToAgile.DOCUMENT] in file AccCustomer9.ini

Parameter name / Description		Sample value (default)
# write the first 5 characters of the v4 file name to the DESCRIPTION field in agile DESCRIPTION #	=	Std.Name.Range-0-5
# write "Catia V4" to the CAX_CRE_SYSTEM field in agile CAX_CRE_SYSTEM #	=	Def.Catia V4.None

Table: Description of the section [FrameDefinition] in file AccCustomer9.ini

Parameter name / Description		Sample value (default)
# write the content of the agile field NUMBER to a Catia v4 text with the identifier "DocumentID" DocumentID #	=	NUMBER
# write the content of the agile field DESCRIPTION to a Catia v4 text with the identifier "DocumentName" DocumentName # [eof]	=	DESCRIPTION

How to define a text field in Catia V4

Here is an example describing the steps to define a text field in Catia V4, in order to fill with the content of a field of the Agile data set. This is the common way to fill out the titleblock of a drawing.

In the example described below the special situation using the text field “DocumentName“ is substituted by a more generic case where the name of the text field is "NewIdentifier".

Try out the following in order to create first a text field in Catia V4 and name it "NewIdentifier". Basically, it has the content "TEXT" in order to see its location at the drawing (e.g. inside the title-block).

1. View > Draw
2. Choose SP
3. Choose POINT
4. Choose COORDS
5. Type e.g. 10,10 as reference point coordinates
6. Choose TEXTD2
7. Select the reference point created before to position the text field
8. Enter the string "TEXT" as key text and confirm with YES
9. Choose IDENTIFY
10. Select the text field element
11. Type in e.g. "NewIdentifier" as KEY NEW ID and press RETURN

At least you can control the new text field using the ANALYSIS menu topic.

Solid Edge Connector Administration

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
[LogFileDir]	Disk and path of log file directory
[AgilePartViewFile]	OBSOLETE. See [Agile9PartViewFileExtensions]
[AgileAssemblyViewFile]	OBSOLETE. See [Agile9AssemblyViewFileExtensions]
[AgileDrawingViewFile]	OBSOLETE. See [Agile9DrawingViewFileExtensions]

Option Name	Usage
[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
[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

Option Name	Usage
[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>
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

Option	Description
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 ▫ ModelTitle – Name of Solid Edge model without file extension, e.g., BOLT ▫ ModelExtension – Solid Edge Model type, e.g., SLDPRT, SLDASM, SLDDRW

Option	Description
	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)
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.

Option	Description
[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"
[Agile9Configuration]	Setting that control how you identify configured parts
Syntax	ConfigProperty = Configured ConfigPropertyValue = YES

Option	Description
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 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
Syntax	0 | 1

Option	Description
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

Where 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

Section	Represents	Example
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!

Configuring the PlmSEAddin.xml File

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 3

Using EC CAD Connectors

This chapter includes the following:

Starting Engineering Collaboration Client

Engineering Collaboration is operated from within your CAD system environment. Your administrator will provide you with a startup command or icon that will start your CAD system with the Engineering Collaboration functions enabled.

Note In order to use Engineering Collaboration you must be a registered Agile user!

Menus and Toolbars

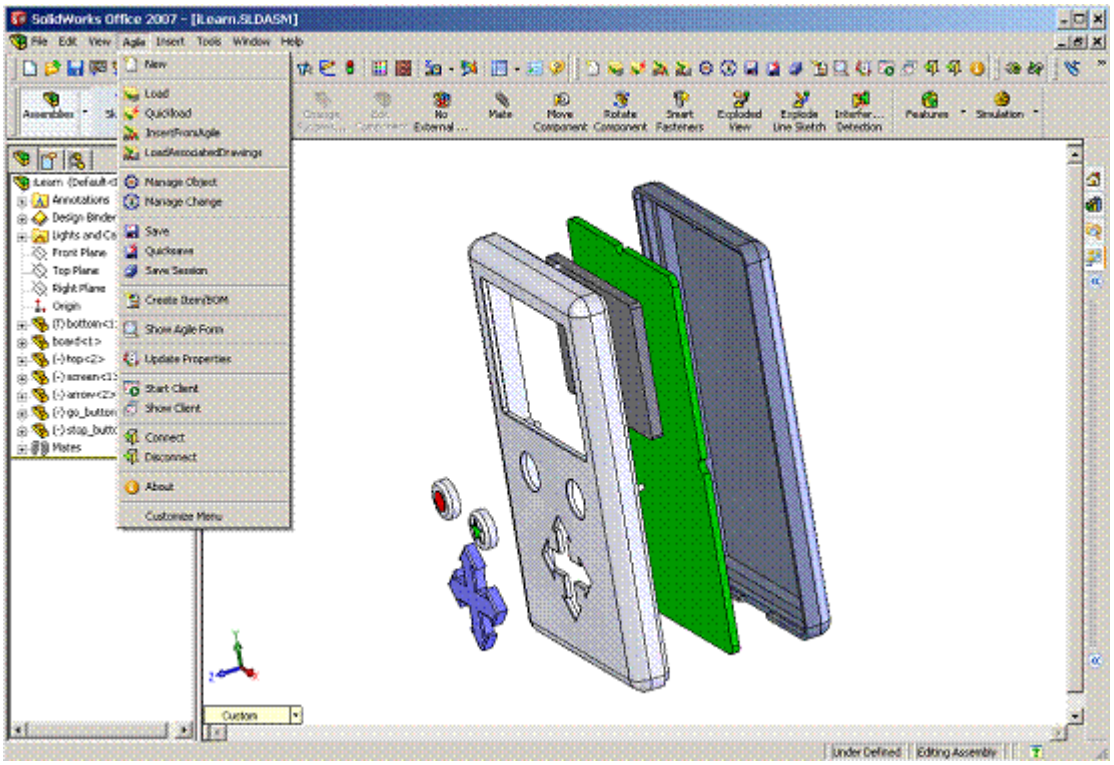
When EC is enabled, you will see either an Agile menu in your menu bar, or an Agile toolbar. Access to Engineering Collaboration functions is through this menu or toolbar.

Table: EC Access Methods according to the CAD System

CAD System	EC Access Method
Pro/ENGINEER	Menu and toolbar
SolidWorks	Menu and toolbar
Unigraphics NX	Menu and toolbar
CATIA V5	Toolbar
CATIA V4	Menu

CAD System	EC Access Method
Solid Edge	Menu and toolbar

An example of both the Agile menu and toolbar is seen in the figure below:



The contents and function of the Agile menu and toolbar, which are common to all connectors, are shown in the table below:

Table: Agile Commands

Command	System	Icon	Function
New	All		Creates a new, blank CAD model (part, assembly, or drawing) based on a selected template, and registers it in Agile.
Load	All		Loads CAD files from Agile into the current CAD session, with a dialog that allows setting load options.
Quickload	All		Loads CAD files from Agile into the current CAD session, without the load options dialog.
Insert in Assembly	All		Loads CAD files for a part or assembly from Agile into the current CAD session, inserting them into the current assembly.
Load Associated Drawings	All		Searches Agile for any drawing files related to the current CAD model, and loads the files into the current CAD session.

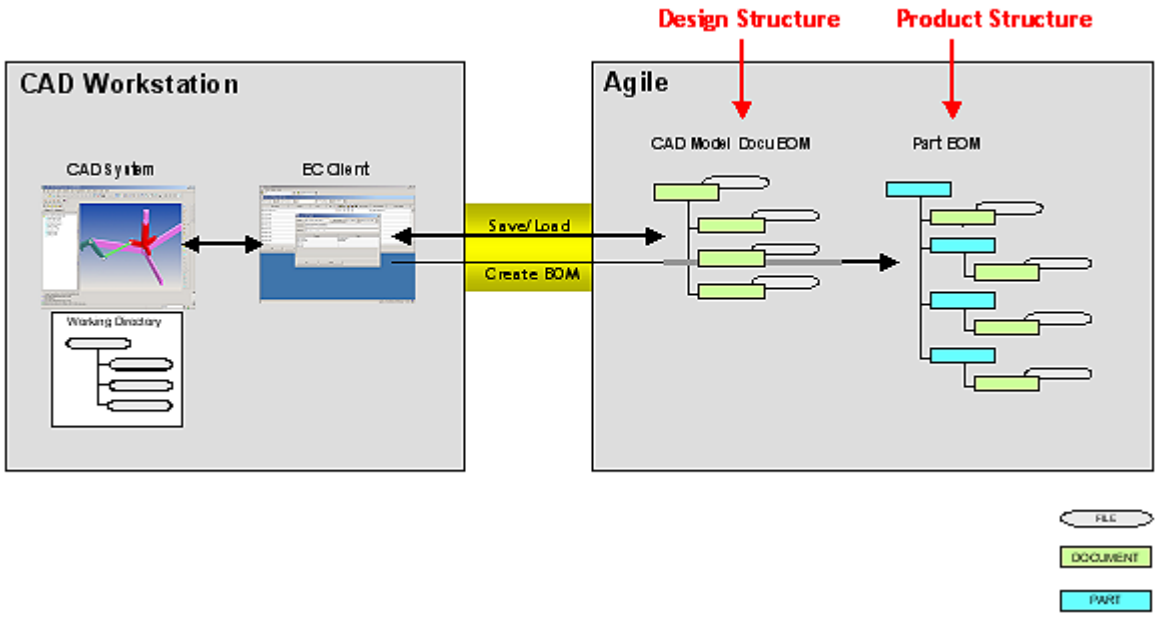
Command	System	Icon	Function
Manage Object	All		Displays checkout and revision status of the current CAD model (only). Also allows changing checkout status and creating new revisions.
Manage Change	All		Displays checkout and revision status of the current CAD model and all its components. Also allows changing checkout status and creating new revisions.
Manage Family Table	Pro/E, UG NX		Displays checkout and revision status of an entire family table, including the generic and all instances. Also allows the user to create a new revision of a generic and all related instances within Agile, at the same time.
Save	All		Saves files from the current CAD model and all its components into Agile, with a dialog that allows setting save options.
Quicksave	All		Saves files from the current CAD model and all its components into Agile, without the save options dialog.
Save Session	All		Saves files from the current CAD session into Agile, with a dialog that allows setting save options.
Save Family Table	Pro/E, UG NX		Allows the user to save an entire family table at once. With Pro/E, this process also validates the family table. The user is prompted if there are any errors with the validation process, and a log file is created for review.
Create Item/BOM	All		Creates and/or updates a Part BOM in Agile based on the structure of the current CAD model.
Show Agile Form	All		Displays the Agile form corresponding to the current CAD model.
Update Properties	All		Sets property (attribute) values in CAD based on values from Agile. Properties for the current CAD model and all its components are updated. The specific attributes to map are defined by your administrator in the configuration file.
Update Properties > First Level	Pro/E, UG NX, CATIA V5		Same as Update Properties, but only sets them for the current CAD model and the next lower level (typically used for setting drawing and model properties together).
Update Properties > Current	Pro/E, UG NX, CATIA V5		Same as Update Properties, but only sets them for the current CAD model.
Update Title Block	All		Sets text values in the CAD drawing title block, based on values from Agile. The specific attributes to map are defined by your administrator in the configuration file.
Insert CGR	CATIA V5		Adds a CGR file to the current CATProduct

Command	System	Icon	Function
Save with CGR	CATIA V5		Saves the CGR file along with the native CATPart or CATProduct file, and establishes the relationship between them
Open Native File	CATIA V5		Opens the native CATPart or CATProduct files for the selected CGR file(s)
Reload CGR	CATIA V5		Updates the selected CGR file(s) with the latest version from PLM
Start Client	All		Used to start (or re-start) the EC Client window.
Show Client	All		Brings up the EC Client window. This is mostly useful on Unix systems which do not have sophisticated window management.
Connect	All		Used to connect to Agile. The user will be prompted to log in. Note that the connector will automatically perform a Connect operation when the user attempts to use any Agile menu command.
Disconnect	All		Disconnects from Agile, without exiting from the CAD session. This frees up the Agile PC license.
About	All		Displays information about the current version of the CAD Connector.

CAD Connector Functionality

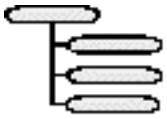
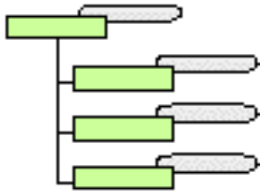
In order to understand the details of the CAD Connector functionality, it is important to understand the overall process and how the data is stored in Agile. The figure below gives a high-level view of the process.

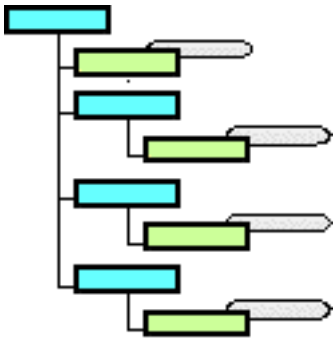
Figure: Agile Engineering Collaboration Process



One important concept is that EC creates two distinct structures inside Agile, one called the Design Structure, and the other called the Product Structure. Each of these is based on the CAD file structure that is created by the designer in the CAD system. The following table explains the purpose of each of these structures:

Table: EC Structure types

Structure Type	Data Type	How Created	Purpose
CAD File Structure 	Files	Created by building CAD models. The structure is known within the CAD files.	Defines the assembly structure of CAD models. Also used to structure the relationships between CAD drawings and the components on the drawing.
Design Structure 	Agile Document object, with attached files (typically a sub-class called CAD Model)	Created using the EC Agile > Save command	Manages the CAD files within Agile, for saving and loading designs. The Design Structure matches the CAD File Structure on a one-for-one basis.

Structure Type	Data Type	How Created	Purpose
Product Structure 	Agile Part object	Created using the EC Agile > Create Item/BOM command	The Product Structure represents the physical product that you are going to build. When the Product Structure is created by EC, the Design Structure Documents are either linked inside the structure, or linked as Relationships to provide access to the Documents and their files.

To enable you to create and modify these structures in Agile, the CAD Connectors have four main functions, each of which has its own dialog window in the EC Client. These functions are Save, Load, Manage Change, and Create Item/BOM. These functions are described in the following sections, followed by other miscellaneous functions.

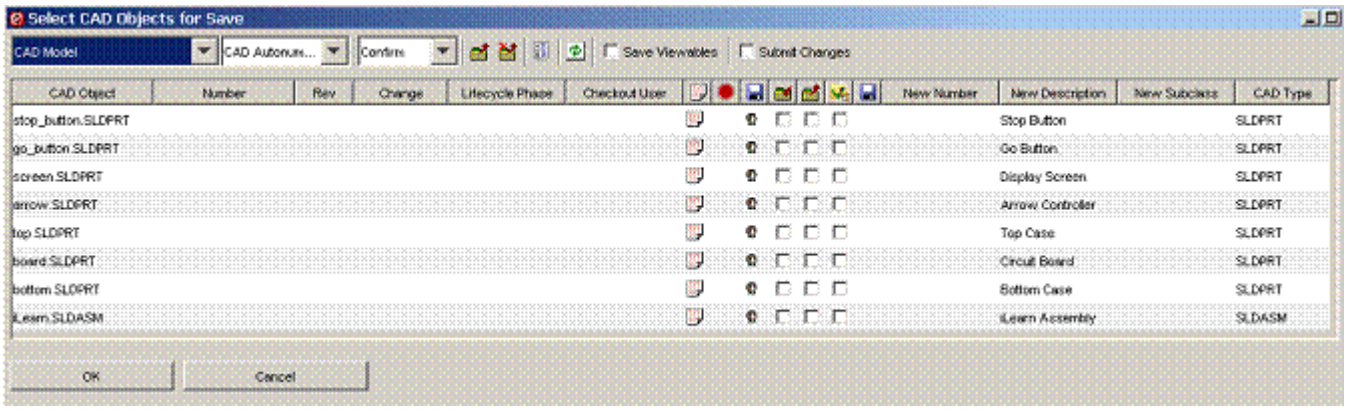
Saving to Agile

Introduction

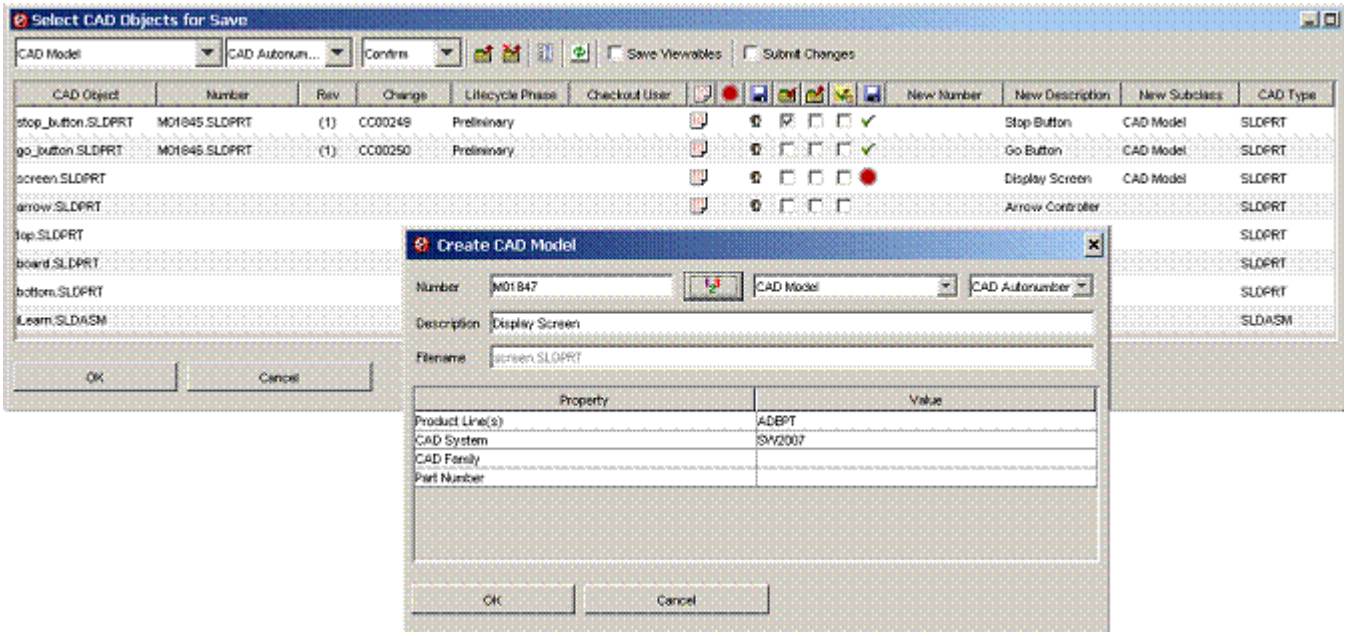
Saving into Agile, using the Agile > Save command, is used to create a Design Structure in Agile. This structure stores all CAD design files (parts, assemblies, drawings, etc.) in a way that supports CAD work-in-progress design, and makes the data available to the rest of the organization, as privileges permit. It will save the current CAD model (whatever is in the active CAD window), including all lower-level components.

Using the Save Command

When you execute the Agile > Save command, the EC Client will pop forward, and display a Save dialog similar to that is shown in the figure below.



After changing any options within the dialog (see below for details), you click the OK button to start the save process. When saving the first time, interactive mode is used which brings up a separate dialog window for each file, in order to capture property information needed for the initial save, as shown in image below.



Details of the Save dialog are shown in the Figures "Save Dialog Toolbars" & "Details of Save Dialog" below, and described in the Tables "Toolbar Options" and "List Fields and Control".

Figure: Save Dialog Toolbars

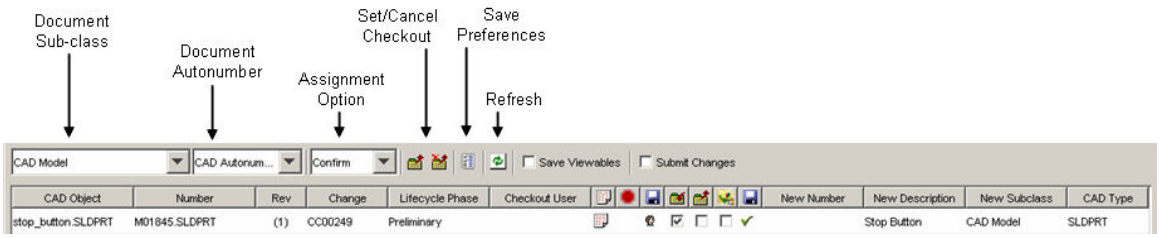


Table: Toolbar Options

Document sub-class	Selector which allows you to pick the default Document sub-class to use for saving CAD files in Agile. Typically set to "CAD Model". Can be overridden in the interactive save dialog.
Document Autonumber	Selector for the default autonumber to use for saving into Agile. Can be overridden in the interactive save dialog.
Assignment Option	Controls how the EC Client handles the situation when you try to create a document that already exists in Agile. Options are: ▫ Confirm (prompts the user each time ▫ Overwrite (assign to existing document, and overwrite the file) ▫ Don't Assign (do not assign) ▫ Assign Only (assign to existing document, but don't overwrite the file).
Set Checkout	Set checkout reservation for the selected components
Cancel Checkout	Cancel checkout reservation for the selected components
Save Preferences	Brings up a dialog which allows you to set the default Document sub-class and autonumber, desired viewable file formats, and default property mappings.
Refresh	Updates the contents of the dialog from PLM.
Save Viewables	If checked, viewable files will be generated for the components in the dialog, based on the Save Preferences settings for viewables.
Submit Changes	If checked, an additional dialog will appear at the end of the Save process, allowing the user to submit change objects through their workflow

Figure: Details of Save Dialog

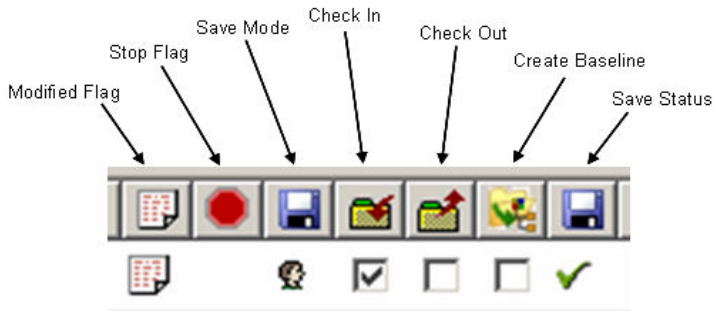


Table: List Fields and Controls

CAD Object	The CAD filename that is being saved

Number	Number of the CAD Model (Document) object in Agile
Rev	Current revision in Agile. Parentheses indicate a pending revision.
Change	The Change object associated with the current revision
Lifecycle Phase	The lifecycle phase associated with the current revision
Checkout User	The name of the current checkout user, if any
[Modified Flag] 	Indicates whether the file has been modified in CAD. If so, the Modified Flag icon will be displayed in the column, and the Save Mode will be set to save the file.
[Stop Flag] 	Indicates whether or not you have the ability to save this component into Agile, based on your privileges and the state of the object in Agile. If not, a Stop Sign icon will be displayed in the column.
[Save Mode]  	Controls whether the file is saved or not, and in what mode. There are three possible options, that can be selected directly in the user interface: - Do not save the file - Save the file in batch mode - Save the file in interactive mode
[Check In]	If set, the file is checked in after being saved (reservation released and available to other users)
[Check Out]	If set, the file is checked out (reserved) after being saved.
[Save Baseline]	Saves a baseline, which is a "snapshot" of all files being saved, and which is primarily used for viewing within Agile (see Saving with Baseline (on page 93) for more details).
[Save Status]  	Indicates the progress of the save operation as follows: - File has been successfully saved - Save process is stopped at this point
New Number	This serves two purposes: 1. On the initial save, this will display the pre-defined mapping of the Number field, for this item. 2. On subsequent saves, can be used to enter a new Document Number to re-assign the file to.
New Description	This serves two purposes: 1. On the initial save, this will display the pre-defined mapping of the Description field, for this item. 2. On subsequent saves, can be used to enter a new Description value.
New Subclass	This serves two purposes: 1. On the initial save, this will display the selected sub-class for this item. 2. On subsequent saves, can be used to enter a new sub-class for the item.
CAD Type	Shows the CAD file extension, which can be used for sorting

Details of the Interactive Save dialog are shown in the figure below and described in the table.

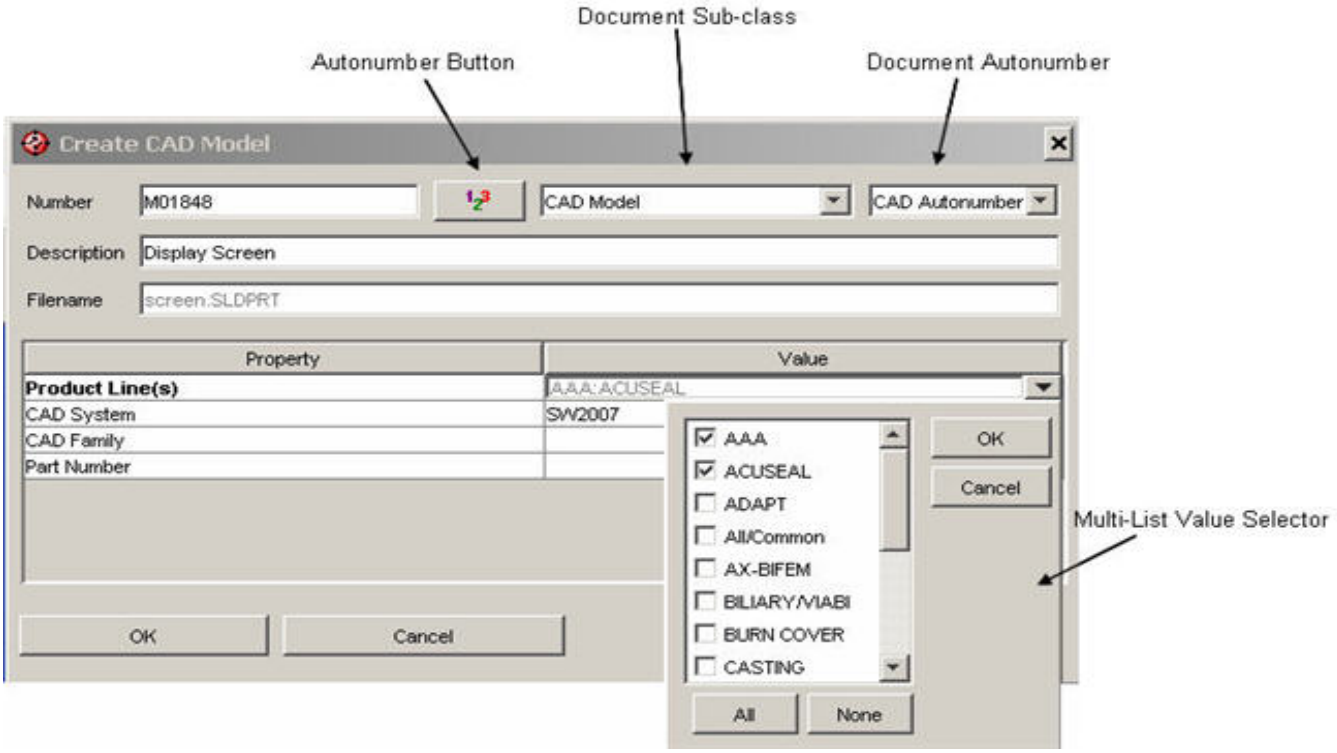


Table: Details of the Interactive Save Dialog

Number	The value that will become the Number assigned to the Agile Document that is being created.
Description	The value that will become the Description assigned to the Agile Document that is being created.
Filename	The CAD filename that is being saved into Agile.
Autonumber Button	If you click the button, it will put the next available autonumber from the selected sub-class and autonumber, into the Number field.
Document sub-class	Selector which allows you to pick the Document sub-class to use for saving this particular CAD file into Agile. This is for overriding the default value set in the main Save dialog.
Document Autonumber	Selector for the autonumber to use for saving this particular CAD file into Agile. This is for overriding the default value set in the main Save dialog.
Property / Value area	This area displays, and allows editing, for other properties that are being set from CAD into Agile. Values may be pre-populated based on the settings in the Save Preferences dialog.

Note Your site most likely has pre-defined mappings for Number, Description, and other properties. You should check with your administrator to understand the allowable values to use. Also, these properties can be set as “Required”, meaning that you must enter a value before exiting the dialog. Required attributes are displayed in bold text.

Note You can cancel out of any individual interactive save dialog (for example to check some other data in Agile), and the whole save process will be paused at that point. If you click the OK button in the main Save dialog, it will re-start the save process from where you left off.

Multi-Select and Context Menus

Since you will often have many items listed in the Save dialog, it is convenient to be able to set options for multiple items at a time. This is made possible by multi-select and context menus. To multi-select, simply click within any item in the window, and either hold down on the left mouse button and drag the cursor, or use Shift-click or Control-click. Once you have selected the desired items, you can use the context menu (right mouse button) to execute any of the commands listed in the following table.

Table: Save Dialog - Context Menu

Open Form	Open the form of the selected item
Save Mode	
Save Interactive	Sets all selected items to be saved, in interactive mode.
Save in Batch	Sets all selected items to be saved, in batch mode.
Deselect from Save	Sets all selected items to NOT be saved.
Rename/SaveAs	
Assign new Autonumber	Sets the New Number field of all selected items to the next available autonumber.
Assign Current Subclass	Sets the New Subclass field of all selected items to the current default sub-class.
Clear new Number/Subclass	Removes any settings within the New Number and New Subclass fields, for selected items.
CheckOut	
CheckOut	Sets checkout reservation for currently selected items.
Cancel CheckOut	Cancels checkout reservation for currently selected items.
keep Checked Out	Turns on the Check Out checkbox, for all selected items.
don’t keep Checked Out	Turns off the Check Out checkbox, for all selected items.
CheckIn after Save	Turns on the Check In checkbox, for all selected items.
don’t CheckIn after Save	Turns off the Check In checkbox, for all selected items.
Change	
Open Form	Opens the form of the Change object associated with the highlighted item
Assign to Current Change	Assigns selected items to the Change object shown in the Current Change field at the top of the dialog.

Create new Change (Autonumber)	Creates a new Change object, using the defined Change sub-class, and assigns selected items to it.
Make Current Change	Sets the number of the selected item's Change object into the Current Change field at the top of the dialog. This is so that you can use it for other items.
Load Items Current Change	If an item has been assigned to a Change outside of the EC Client, you can use this option to load that Change assignment into the EC Client.
Remove from Change	Removes selected items from their current Change assignment
Set Lifecycle Phase	Sets the selected items to the current default lifecycle phase

Save Preferences

The Save Preferences dialog is accessed using the button in the Save dialog. It allows you to set the following preferences:

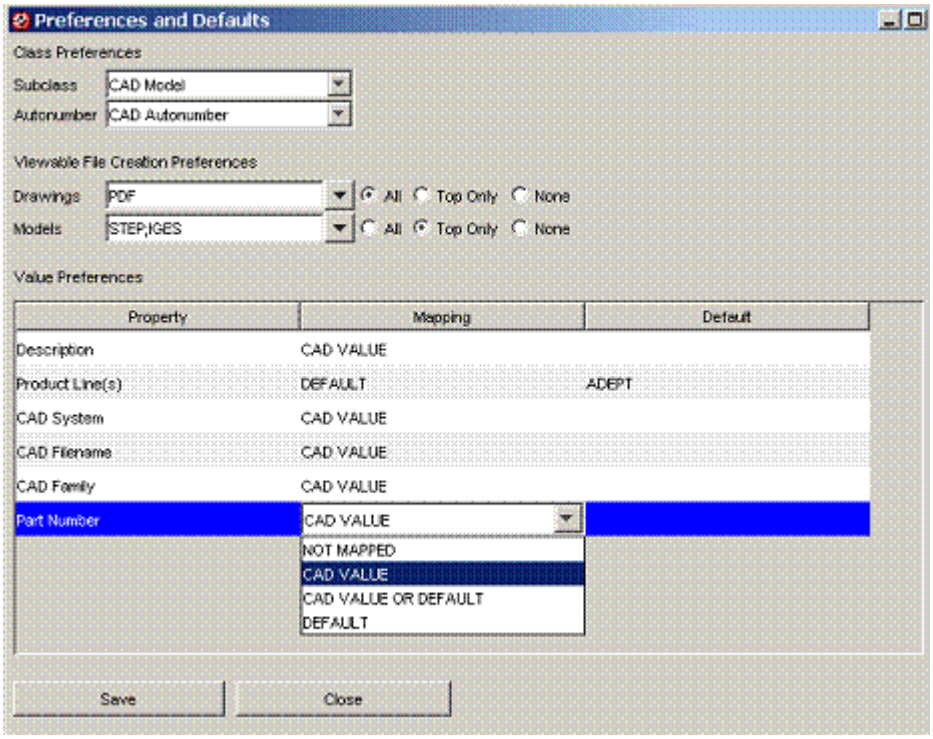
Class Preferences – The default subclass and autonumber used for creating new Documents.

Viewable File Creation Preferences – The types of viewable files that will be automatically created and attached in PLM along with the native file. This can be set independently for Drawings and Models (parts and assemblies), and can be set to generate the viewable files for all CAD files, only the top CAD file, or no CAD files. Note that there is a "master switch" in the Save dialog which turns on and off the entire viewable file creation process for each particular Save command. Also note that depending on the CAD system, additional configuration work may be necessary to automatically create the viewable files (please contact your administrator).

Property Value Preferences – This section allows you to pre-define the properties that are mapped between CAD and PLM, as part of the Save process. By setting these preferences appropriately, you can reduce the use of the interactive save dialog and speed up the save process. The four mapping options are:

- Not Mapped – No value is to be set for this property
- CAD Value – Use the value defined in the CAD properties, based upon the mapping defined by your administrator
- CAD Value or Default – Use the value defined in the CAD properties, but if no value exists then use the default value in the "Default" column.

- Default – Use the value in the “Default” column.



Other Comments about Save

For the initial save of a CAD design into Agile, each file within an assembly needs to be saved. The save mode is set to interactive by default, since generally the user needs to input some property values. For subsequent saves, only those files which have been modified are flagged to be saved, and the save mode is set to batch. If you want to save more or fewer items than what has been identified automatically, simple change the setting for Save Mode for those items.

Design Structures

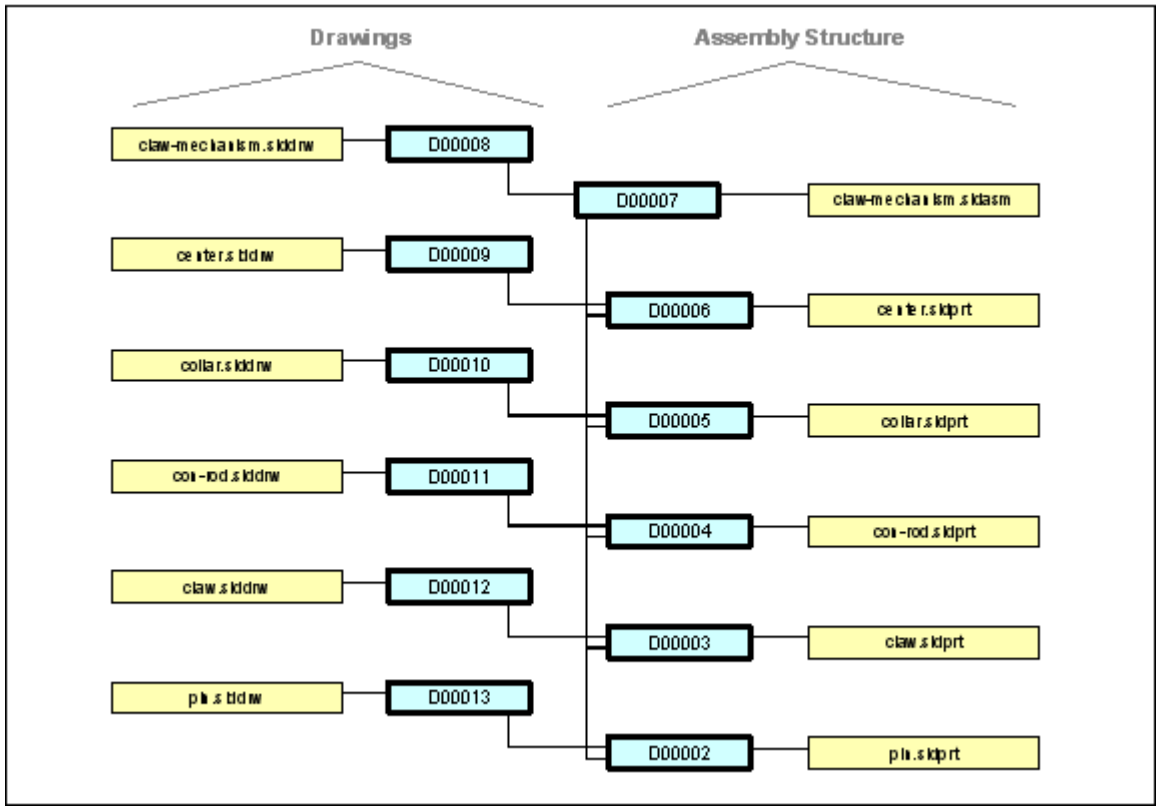
The Save command creates design structures in Agile, to hold the CAD files. It is important to understand how these structures are set up in Agile. CAD designs consist mainly of three types of files – Parts, Assemblies, and Drawings. In terms of the design structure hierarchy, Parts are the lowest level. They are combined into parent Assemblies, which in turn are combined into higher-level Assemblies. Drawings are represented at a higher level than either Parts or Assemblies; essentially a Drawing is a “parent” of whatever assembly or part that is on the drawing.

The figure below illustrates a case of a SolidWorks five-part assembly, where each of the parts, and also the assembly, have corresponding drawings. The assembly (D00007) is the parent of all the parts, and each drawing is the parent of its respective assembly or part. It is important to keep this structure in mind when using EC operations like Save and Load, and also when browsing through design structure data in Agile.

For example, since the Save and Load commands work on a single structure at a time, you can save drawing D00008 and it will save assembly D00007 and the five parts, but not the five part drawings (because they are not in the same tree structure – you would not see them within the

BOM tab of D00008 in Agile). For this situation you can use the Save Session command, which will include all files active in the CAD session within the Save dialog.

Figure: Design Structure Hierarchy



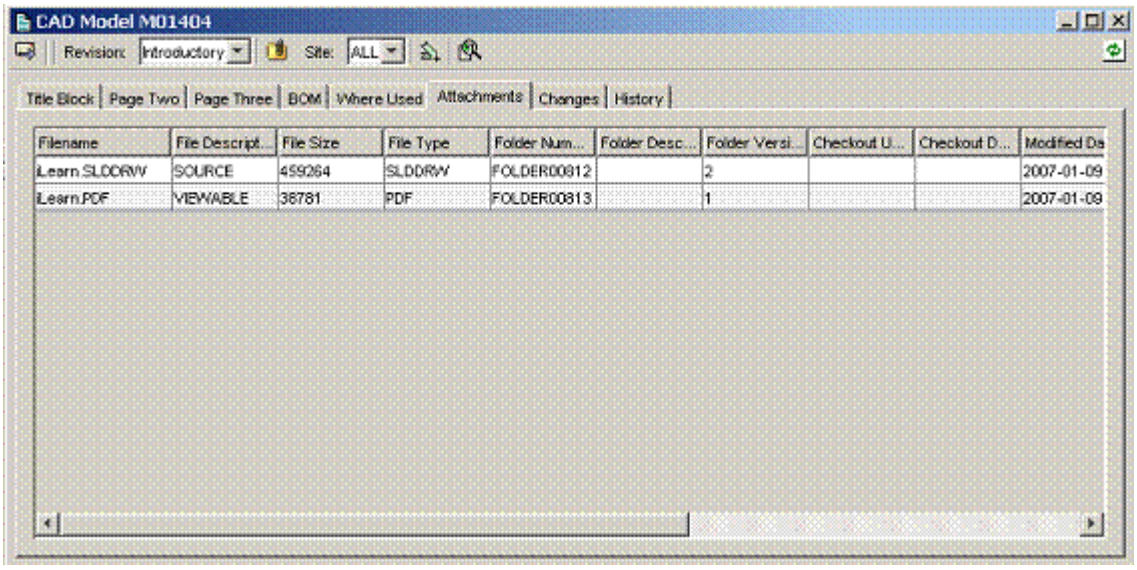
Similar types of structures are used to store other specialty types of CAD files, such as Pro/E family table generics, and drawing format files. In each case, the Connector is set up to recognize the files as being stored in these particular type of structures, so they must not be manually modified in Agile or errors may result.

Saving with Derived Files

By configuring the Save Preferences dialog (see above) you can set the system to save additional derived or viewable files, such as PDF or IGES, in combination with the regular save operation. The reason these operations are combined is to make sure that both the additional file and the native file(s) are in synch with each other. The additional files are attached to the same Agile Document object as the native CAD file. The available file format options within the Save Preferences menu are customizable for your site; your administrator will configure this as appropriate.

The figure "Derived File Saved into Agile" shows an example of the results of using this function to generate a CGM PDF file in addition to the native CAD file.

Figure: Derived File Saved into Agile



Note In addition to this function with the Agile menu, it is also possible to manually attach files to Agile Document using the “Add File” button on the Document’s form in the EC Client.

Note First generate the file using your CAD system, then use the Agile > Show Agile Form command to bring up the form for the current CAD model, then use the Add File button to add the file.

Saving with Baseline

Baselines are additional data structures in Agile, used to store a “snapshot” of an entire CAD assembly or CAD drawing. The primary use of a Baseline is to support native CAD 3D model or 2D drawing viewing with the Agile Advanced Viewer. To save a Baseline, you check the Create Baseline checkbox in the Save dialog. The guidelines for which items you should check are as follows:

- Always check the checkbox for the highest level item in the tree (the current assembly or drawing object in CAD that you are saving).
- Check any other assemblies that you would like to be able to view in the Advanced Viewer.
- Don’t check any parts. This does not add any value.

Note It is important to note that saving a Baseline doubles the time to perform the Save command, and also doubles your file storage in Agile. This is because the files have to be saved into Agile twice, once for the normal design structure (to support the save/load process), and once for the Baseline.

Note Starting with Agile PLM version 9.2.2 service pack 2, there is enhanced capability with the Agile Viewer that allows CAD assembly structures to be viewed directly. This may reduce or eliminate your need to save Baselines (see the EC Client Administration chapter for information on how to disable the Baseline column from the Save dialog).

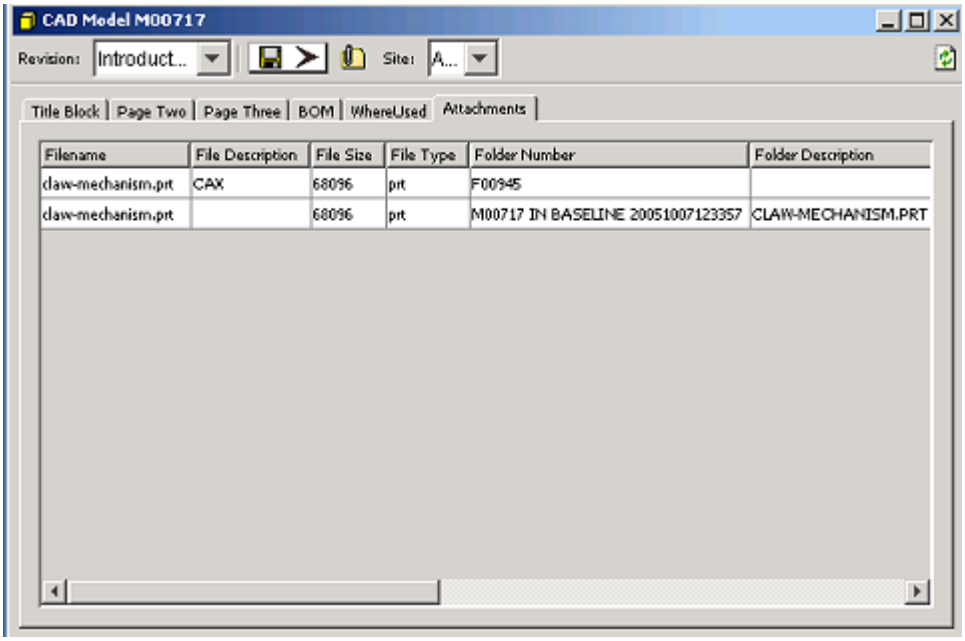
Agile CAD Model Document saved with Baseline shows an example of a Document where a

Baseline has been stored. You will see the CAD filename listed a second time, and the Folder Number value is set to, for example:

M00717 IN BASELINE 20051001123357

Where M00717 is the number of the Document, and 20051001123357 is the date and time that the Baseline was generated. This folder is the connection to the baseline. In the standard Agile client, if you view this folder, you will see the 3D Model or 3D CAD drawing in the Advanced Viewer.

Figure: Agile CAD Model Document saved with Baseline



In addition to the folder that is shown here, there is another folder that is generated for the baseline, called the Combined Folder. This is the folder that has all the files for the baseline in it. This folder has a name like:

BASELINE 20051001123357 F00948

Where F00948 is an autonumber.

Normally you will not need to directly access this folder, but it is worthwhile to mention that since this folder contains all files that were saved for the baseline, it is an easy way to extract a complete CAD model from Agile, for example for a supplier to access.

Note	The term “Folders”, as described here, refers to the Agile File Folder object class. You can search for File Folders directly within the EC Client or standard Agile client.

Other Save Commands

Two additional Save commands are available:

Quicksave – This command saves the same files as the Save command, but does not open up the Save dialog. This is typically used for interim saves, when no user input is required. If the

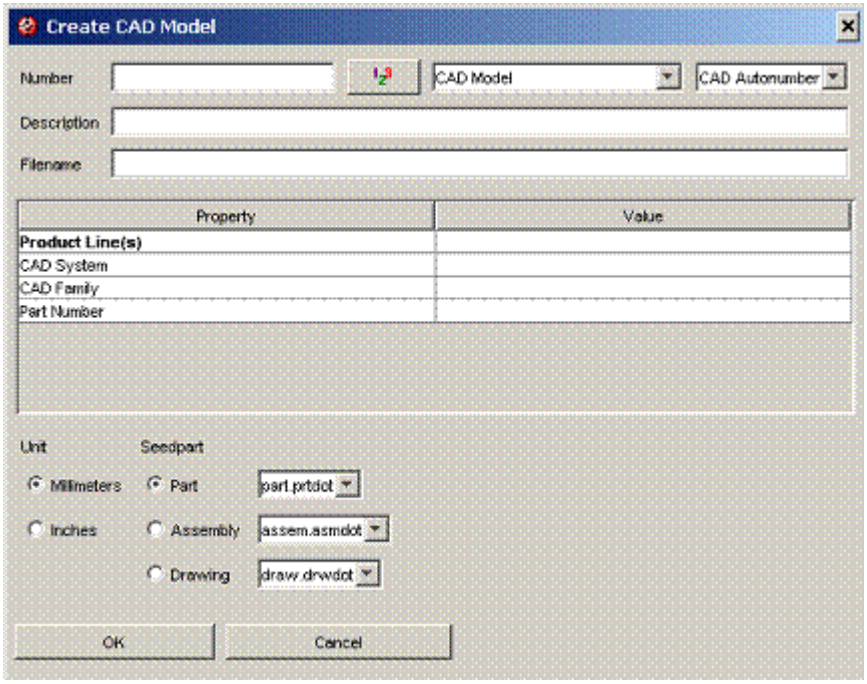
particular files being saved require user input, then the Save dialog will be opened regardless.

Save Session – This command lists the entire contents of the current CAD session in the Save dialog, rather than just the components within the active model. This is useful especially when you have multiple drawings active in session, because they can then be saved all at once.

Creating New Models

The Agile > New command allows you to create a new, empty CAD file and create the corresponding Document object in PLM. In this dialog you can select from a list of template file, if they have been configured by your administrator. Note that by using the New command, the Document object is created but the file is not yet saved into PLM (you need to use one of the Save commands to do this). The figure below shows "New" dialog.

Figure: "New" dialog



Loading from Agile

Introduction

Loading from Agile into CAD, using the Agile > Load command, is used to retrieve previously saved CAD files in order to perform CAD work. The EC Client provides a search capability to locate the desired model to load. Once the desired model is located and selected, the necessary files are extracted from Agile and placed in the designated working directory. The files extracted depend on the type of model selected, as follows:

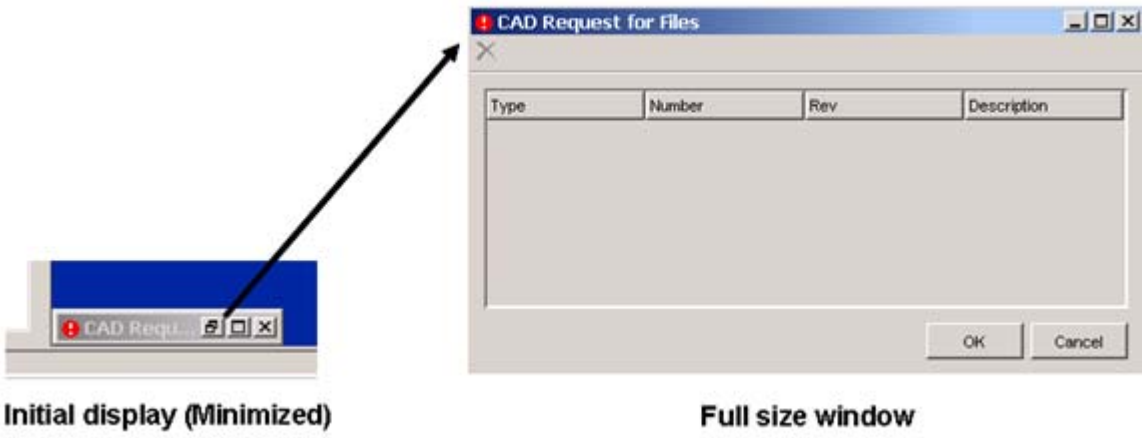
- Part – If a single part is selected, just that single part file will be loaded to the working directory.

- Assembly – If an assembly is selected, its file and all subordinate files (sub-assemblies and parts) necessary to build the assembly are extracted.
- Drawing – If a drawing is selected to load, its file and all subordinate files necessary to build the drawing (including all subordinate assemblies, subassemblies, and parts) are extracted.

Using the Load Command

To load CAD files from Agile into your CAD system, pick the Agile > Load command. This will bring the EC Client forward, and display the "CAD Request for Files" window (see figure below). This window is initially displayed minimized, so as not to interfere with other windows in the client. It is your indication that you are in "Load mode". If you don't see this window, you won't be able to load.

Figure: CAD Request for Files Window

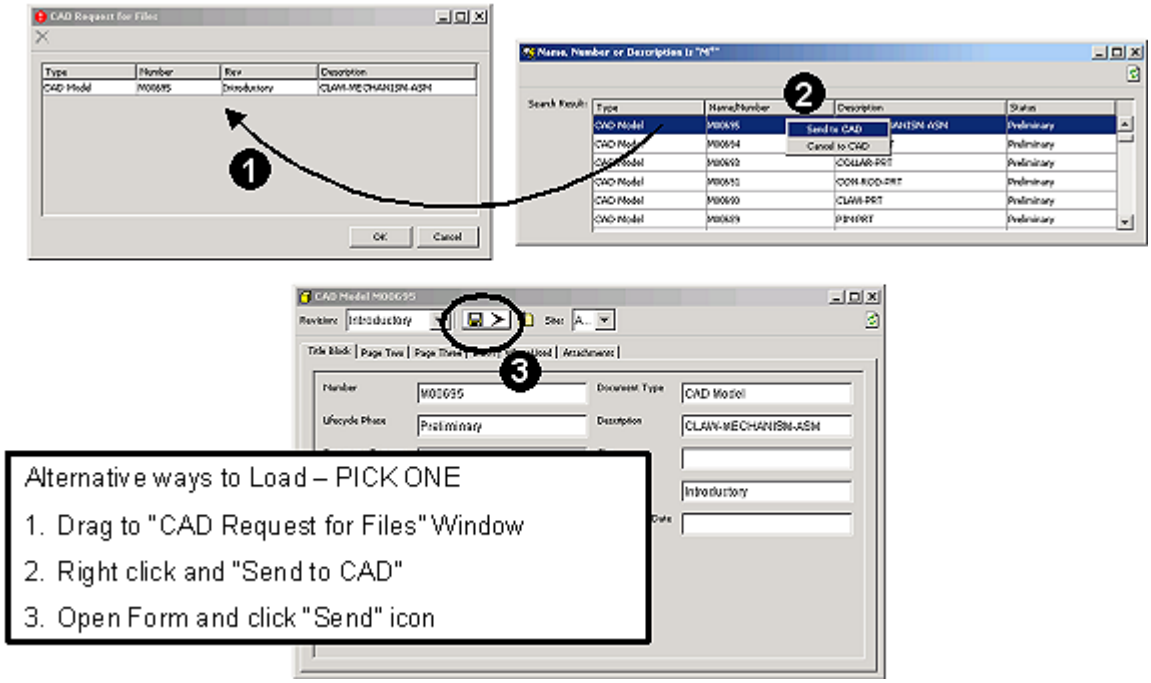


Once this window appears then you need to locate the desired CAD model that you want to load. This is performed using any of the following EC Client search options, as described in the "Engineering Collaboration Client - Search" section. When the desired model to load has been found, there are three ways that it can be loaded into CAD, as shown in the figure below:

1. Select it in the list, and drag it to the "CAD Request for Files" window and click OK
2. Right-click with the mouse and pick "Send to CAD"
3. Double-click to open the form for the model, then click on the "Send" icon (the button with the disk and arrow icons)

You only need to use ONE of these methods. Note that options 2 and 3 can be performed with the "CAD Request for Files" window still minimized.

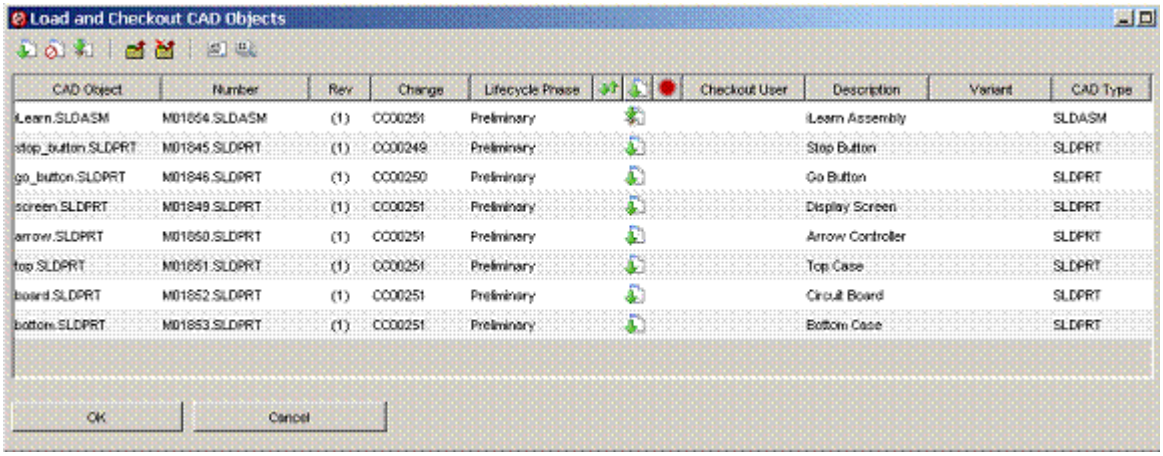
Figure: Alternative ways to load



Load Dialog

Once you have selected the object to load using one of three methods, the Load dialog appears. The main purpose of the dialog is so that you can review which files, and which revisions of the files are being loaded. Also you can see if anyone has any of the files checked out. The only user input possible in this dialog is to set the checkout reservation for the files you are loading. In addition to manually checking the box for Check Out, you can multi-select within this list, and use the right mouse button to bring up the context menu to set and unset the Check Out box.

Figure: Load Dialog



Note	Even if files are checked out by someone else, they will be loaded using “Get” (e.g. load without checkout), so that you always have full access to the files necessary to bring up the model in CAD.

Figure: Load Dialog Toolbars

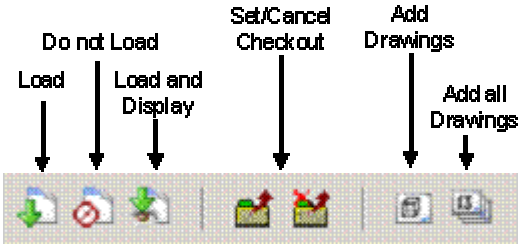


Table: Toolbar Options

Load file Do not load file Load and display file	Sets the desired load option for selected components. Determines how the file will be loaded from PLM to local disk
Set Checkout	Set checkout reservation for the selected components
Cancel Checkout	Cancel checkout reservation for the selected components
Add Drawings	Checks all selected components, and adds any related drawings to the Load dialog
Add all Drawings	Checks all components within the Load dialog, and adds any related drawings to the Load dialog

Table: Fields in the Load Dialog

CAD Object	The CAD filename
Number	Number of the Document object in Agile
Rev	Revision and ECO in Agile, if any
Change	Change object associated with Revision
Lifecycle Phase	Lifecycle phase of the specific Revision
[File Status]	Gives the status of the particular file, both in PLM and on the local disk
(blank)	- A local file does not exist
	- The local file is exactly the same as what is in PLM
	- The file has been updated in PLM, and so is more recent than what is on local disk
	- The file has been modified locally, and so is more recent than what is in PLM
	- The file has been modified both on the local disk and in PLM
	- The file status cannot be determined

[Load Option]	Determines how the file will be loaded from PLM to local disk
	- Load and display in a CAD window
	- Load but don't display
	- Do not load
Checkout User	Current checkout user, if any
Description	Description of the Document object in Agile
Variant	Used by some CAD Connectors to indicate a CAD file variant
CAD Type	Shows the CAD file extension, which can be used for sorting

Multi-Select and Context Menus

Since you will often have many items listed in the Load dialog, it is convenient to be able to set options for multiple items at a time. This is made possible by multi-select and context menus. To multi-select, simply click within any item in the window, and either hold down on the left mouse button and drag the cursor, or use Shift-click or Control-click. Once you have selected the desired items, you can use the context menu (right mouse button) to execute any of the commands listed in the following table.

Table: Load Dialog - Context Menu

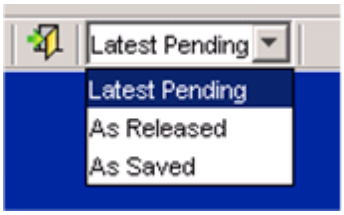
Open Form	Open the form of the selected item
Get this File	Sets the selected items to be loaded
Don't Get this File	Sets the selected items to not be loaded
CheckOut	Sets checkout reservation for currently selected items.
Cancel CheckOut	Cancels checkout reservation for currently selected items.
Change	
Open Form	Opens the form of the Change object associated with the highlighted item
Assign to Current Change	Assigns selected items to the Change object shown in the Current Change field at the top of the dialog.
Create new Change (Autonumber)	Creates a new Change object, using the defined Change sub-class, and assigns selected items to it.
Make Current Change	Sets the number of the selected item's Change object into the Current Change field at the top of the dialog. This is so that you can use it for other items.
Load Items Current Change	If an item has been assigned to a Change outside of the EC Client, you can use this option to load that Change assignment into the EC Client.
Remove from Change	Removes selected items from their current Change assignment
Set Lifecycle Phase	Sets the selected items to the current default lifecycle phase

Structure Resolution Options

The EC Client provides full capability for loading any desired revision of the CAD files. This is done through a combination of two controls within the EC Client UI:

Structure Resolution Select Box: The Structure Resolution select box has three choices, “Latest Pending”, “As Released” and “As Saved”.

Figure: Structure Resolution Select Box



The logic of these options is:

Latest Pending – Starting with the selected object to load, for each item in the structure, the pending (in work) revision will be chosen to load. If there is no pending revision for a given item, the latest released revision will be chosen instead. This is the normal option to use, when you are actively working on the CAD files.

As Released - Starting with the selected object to load, for each item in the structure, the latest released revision will be chosen to load. This option will work with old revisions as well, and it will load child parts using standard Agile BOM resolution. The logic behind this resolution is that subordinate items (parts or subassemblies) can move forward in revision level, even after the parent item is released. This is generally not appropriate for CAD designs, so mainly this option is used for loading the latest released revision, for purposes of including it in another assembly, or as the basis of a derived design.

As Saved - Starting with the selected object to load, each item in the structure will be loaded as it was saved into Agile (for that particular revision). This is considered “Fixed” BOM resolution, which is useful for loading past revisions back into CAD.

Note that with any of these options, if you want to load a past revision, you must use the third load option as shown in the figure below. This is because you need to select the revision from the form.

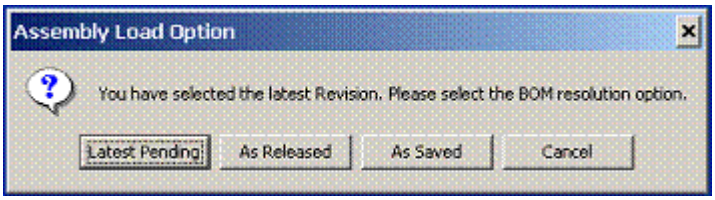
Figure: Revision Selector - For use with “As Saved”

Regardless of which load option you use, the Load dialog will always display the revision that is being loaded for each item. If it doesn’t look right, you can always cancel out of the Load dialog, and nothing will be loaded to CAD.

Depending on the configuration of EC at your site, you may also have a prompt that appears after

you click on the Load button. This prompt will ask you which resolution you want, and this will override the setting in the Structure Resolution Select Box.

Structure Resolution Prompt



Tip: The BOM tab in any Agile client never displays pending revisions for items within the BOM structure, even if pending revisions for those items do exist. You must open the object form for the object, and use the Revision Selector to verify if there is a pending revision.

Working Directory

As shown in Agile Engineering Collaboration Process, when you work with Agile Engineering Collaboration you still have a local working directory where your CAD files reside. The location of this working directory is determined by your system administrator. This is where files are copied to, when you use the Load command. If you use your CAD system's File > Save command, the files will be saved into this working directory. You should use File > Save to save periodically as you normally would, to prevent data loss during your daily work. Use the Agile > Save command on a regular basis to secure your data within Agile, and to make it available to others.

It is important to understand how the Load command interacts with the CAD system, if you have models already in CAD. If you have a file open in CAD, and try to load this same file from Agile, you will get an error and the file from Agile will not be loaded. If this file is not open in CAD, but exists in the working directory, it will be overwritten when you do the load command. When using Pro/E, you can have files that are not open in a window, but which are "in session". Files in session will not be overwritten when you do a load, which can lead to unexpected results. In this case you must use the Erase All > Not Displayed... command before loading.

When you exit from your CAD system, the contents of your working directory will not be touched (the exceptions are SolidWorks and Solid Edge, which ask you when you exit, whether or not you want to delete the contents of the working directory). You can use the File > Open command to continue working on your CAD files, from the working directory, the next time you start your CAD system.

Loading Associated Drawings

Special loading options are provided in order to load drawing files associated with your 3D part and assembly models. Since Agile PLM stores drawings as parent objects to the 3D model, the Load command does not automatically find the drawings as it traverses down the structure. One way to load the associated drawings is mentioned in the description of the Load dialog above. The buttons for "Add Drawings" and "Add all Drawings" allow you to add the necessary drawing files during the load process. In addition to this, the command Load Associated Drawings in the Agile menu will search for all drawings related to the current CAD model and load them into CAD.

Other Load Options

Two additional Load commands are available:

Quickload – This command follows the same process as the Load command, except that the Load dialog does not appear. No additional options such as creating new revisions and setting checkout are available, the files are simply loaded as-is.

Insert in Assembly – This command loads CAD files for a part or assembly from Agile into the current CAD session, inserting them into the current assembly.

Managing Change

Introduction

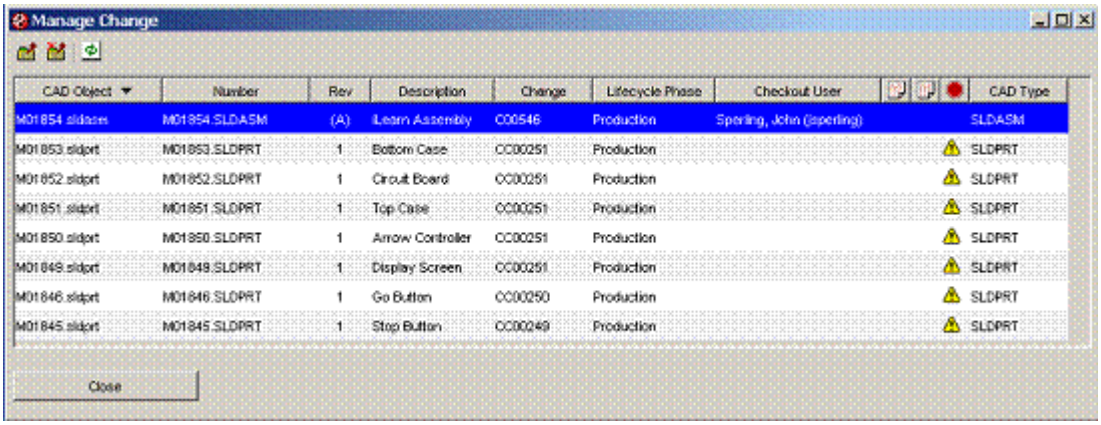
In order to control updates to CAD files and the Design Structures in Agile, the CAD Connectors provide the ability to manage the change process using the inherent capability in Agile. There are two primary components to this capability:

- Controlling the ability to update the files in Agile, using privileges, checkout reservation, and versioning.
- Controlling the change process workflow, using revisions and ECOs

Using the Manage Change Command

The Agile > Manage Change command is used to view and change the checkout status of the CAD files controlled by Agile. It is also used to create new revisions using Agile change management. The following figures and tables explain the commands available; how you use them to effectively manage the change process for the CAD designs is described following that.

Figure: Manage Change Dialog



Details of the Manage Change dialog are described in the tables below.

Table: Options of the Manage Change Dialog Toolbars

Check Out	Sets checkout reservation for currently selected items
Cancel Check Out	Cancels checkout reservation for currently selected items
Refresh Privileges	Updates from Agile the display of Revision, Checkout User, and the Stop Flag.

Table: List Fields and Controls of the Manage Change Dialog (Toolbars)

CAD Object	The CAD filename that is being saved
Number	Number of the Document object in Agile
Rev	Revision in Agile, if any
Description	The document description in Agile
Lifecycle Phase	Lifecycle phase of the specific Revsision
Change	Change object associated with Revision
Checkout User	The name of the current checkout user, if any
[Changed in CAD]	Indicates whether the file has been modified in CAD. If so, the Modified Icon will be displayed in the column.
[Changed in Agile]	Indicates whether the file has been modified in Agile. If so, the icon will be displayed in the column, meaning that what you currently have in your CAD session is out-of-date.
[Status Flag]	Indicates whether or not you have the ability to set checkout or create a new revision for this component into Agile, based on your privileges and the state of the object in Agile. If no, a Stop Sign icon will be displayed in the column. The tooltip text (when you hover the cursor over the stop sign) will indicate the specific reason. If you see a warning icon rather than a stop sign, it means you should be aware that you are creating a new pending revision from a revision that is not the latest.
CAD Type	Shows the CAD file extension, which can be used for sorting

Multi-Select and Context Menus

You can multi-select within the Manage Change dialog, to operate on more than one item at a time. Once you have selected the desired items, you can use the context menu (right mouse button) to execute any of the commands listed in the table below.

Figure: "Manage Change Dialog Capabilities

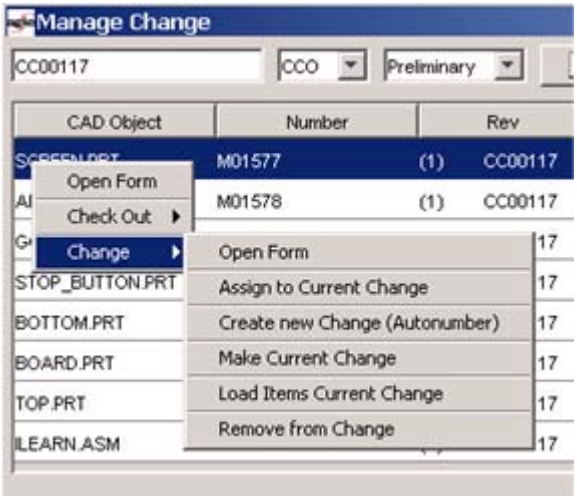


Table: Context Menus of Manage Change Dialog

Open Form	Opens the form of the highlighted item
Check Out	
Check Out	Sets checkout reservation for currently selected items
Cancel Checkout	Cancels checkout reservation without checking in
Change	
Open form	Opens the form of the Change object associated with the highlighted item
Assign to Current Change	Assigns selected items to the Change object shown in the **Current Change** field at the top of the dialog.
Create new Change (Autonumber)	Creates a new Change object, using the defined Change sub-class, and assigns selected items to it.
Make Current Change	Sets the number of the selected item's Change object into the **Current Change** field at the top of the dialog. This is so that you can use it for other items.
Load Items Current Change	If an item has been assigned to a Change outside of the EC Client, you can use this option to load that Change assignment into the EC Client.
Set Lifecycle Phase	Sets the selected items to the current default lifecycle phase

Managing the current object

The Manage Change command will interrogate the entire structure of your current CAD model, and bring it into the dialog. If you just need to manage the current object itself, not including all its structure, use the Manage Object command instead.

The Change Process

The functionality within the Manage Change dialog is designed to simplify the process of managing engineering change for your CAD designs. It uses the standard Agile change process, which involves an object called a Change Order, or ECO for short. Note that the name of the object may be different in your Agile environment. After you save your CAD models into Agile as Documents (with Document structure), you use Change objects to control revisioning of each Document. Agile Change Process gives a picture of how a document progresses through revisions, under control of the change process. There are three basic revision states, which are shown in Agile Change Process:

1. Introductory:

   This is the standard revision name that is used before any ECO has been assigned to the Document. It is perfectly acceptable to use Documents in Introductory revision, to save your CAD designs. This is useful for saving designs prior to entering a controlled change process.

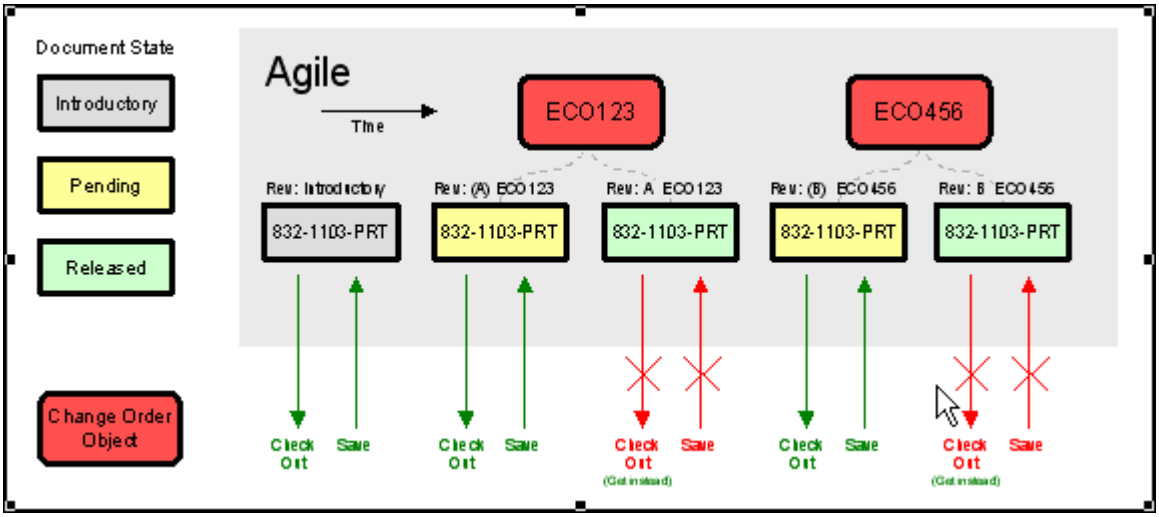
2. Pending:

   A pending revision is one that is in-work, meaning that the CAD files can be checked out and checked in.

3. Released:

   A released revision is one that has been completed and approved. CAD files belonging to released revisions cannot be checked out or checked in, although they can be accessed using “Get”, which means loading without checkout.

Figure: Agile Change Process



The main steps in the EC Client change process are listed in the following tables. The process is a little different for new designs, as compared to existing designs.

Table: Change Process for new Designs

Action	Status in Agile
Create new design files in the CAD system	(none)

Action	Status in Agile
Use Agile > Save to save the design into Agile.	This creates Documents in Introductory revision
In the Agile > Manage Change dialog, use the Create New Change command to assign the Documents to one or more Change objects. Set desired New Rev and New Lifecycle Phase for each item.	One or more Change objects are created, with the selected items as Affected Items on the change. Each Document now has a pending revision equal to the value entered for New Rev.
If necessary, make additional changes in CAD, and use Agile > Save to save into Agile.	Files attached to the pending revisions are updated, using the versioning process. Document attributes and structure may also be changed.
Using the standard Agile change process (outside the EC Client), complete the Change workflow to release the change.	The Documents are now released revisions, and no further changes can be made to the CAD files.

Note Your administrator can configure the EC Client to automatically create pending revisions during the initial Save command. If you are using this approach, you can skip the 3rd and 4th step listed above.

Table: Change Process for Existing Designs

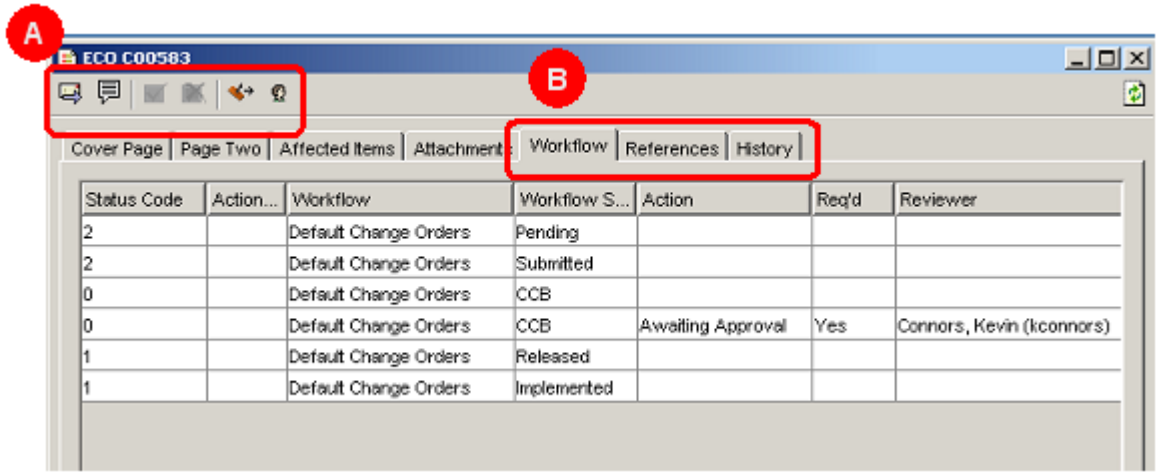
Action	Status in Agile
Use Agile > Load to load the desired design into CAD.	(none)
In the Agile > Manage Change dialog, use the Create New Change command to assign the Documents to one or more Change objects. Set desired New Rev and New Lifecycle Phase for each item.	One or more Change objects are created, with the selected items as Affected Items on the change. Each Document now has a new pending revision equal to the value entered for New Rev.
Make the desired modeling changes in CAD, and use Agile > Save to save into Agile.	Files attached to the pending revisions are updated. Document attributes and structure may also be changed.
Using the standard Agile change process (outside the EC Client), complete the Change workflow to release the change.	The Documents are now released revisions, and no further changes can be made to the CAD files.

There is a variation on these processes, which allows you to create the Change objects outside of the EC Client, using the standard Agile client. Sometimes this is desirable if you want a Change Analyst or some other non-CAD user to initiate the change. In this case, once the Change object is created, the CAD user goes into the Agile > Manage Change dialog, and uses the Change > Load Items Current Change command. This will load the previously created Change objects, plus the New Rev and New Lifecycle Phase values, into the EC Client.

Note The functionality of the EC Client permits a maximum of one pending change per Document at a time.

Note he EC Client can be configured to automatically assign different revision sequences and workflows for different Change sub-classes. Using this capability, you can establish a minor revision (e.g. 1, 2, 3, 4) and major revision (e.g. A, B, C, D) scheme, where the minor revision can be released by the designer directly and the major revision requires a more extensive workflow.

Figure: Change Form Capabilities

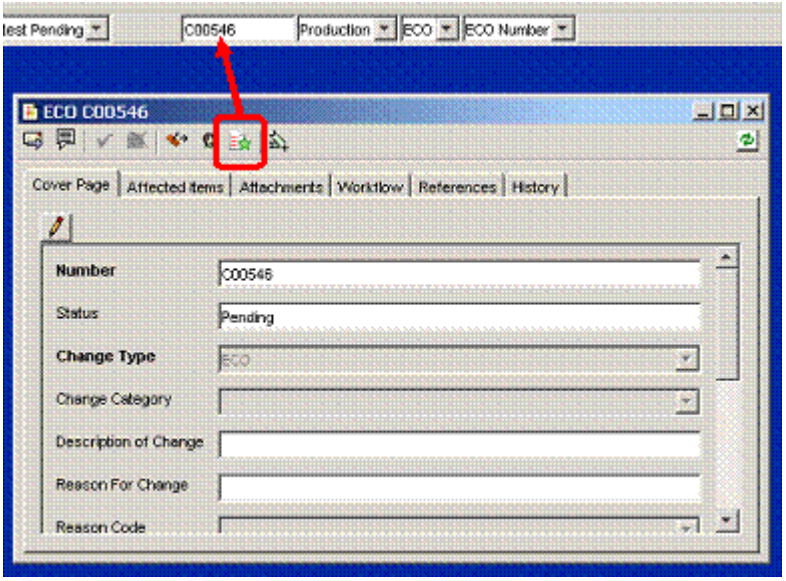


A. Change Form Buttons:

- Send - For sending notification
- Comment - For adding comments to a workflow
- Approve - For approving a workflow step
- Reject - For rejecting a workflow step
- Submit change - For submitting a workflow to the next step automatically
- Submit change manually - For submitting a workflow to the next selected step

B. Tabs for Workflow, References and History

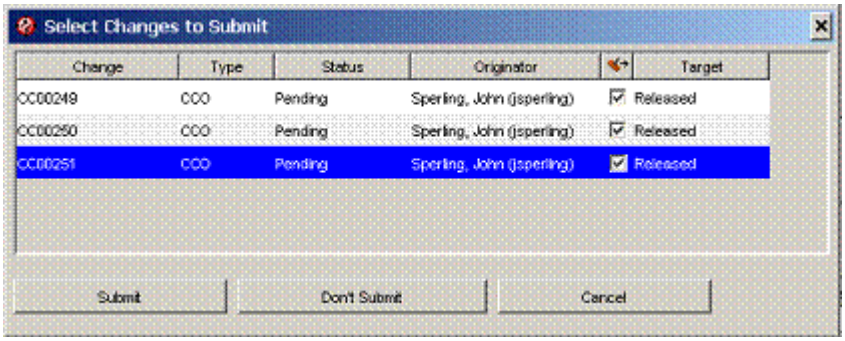
Figure: "Make Current Change" button



The “Make Current Change” button is available on the Change forms. This button loads the number of the change object into the Current Change field at the top of the EC Client. This allows you to set the current change number to then be used within other operations.

Submitting Changes

Change objects have associated workflows, and once a CAD model is done and checked in, it needs to be submitted to its workflow. This workflow could be as simple as automatically releasing the model, or it could involve many reviews and signoffs. There are two ways to submit change object to their workflows. The first is to use the “Submit Change” (or “Submit Change Manually”) button on the Change form. The second and more automated way is to use the “Submit Changes” checkbox in the Save dialog. If this is configured for your system, it allows you to automatically submit the changes for the files you are saving. If you check this option then you will get a popup dialog such as this:



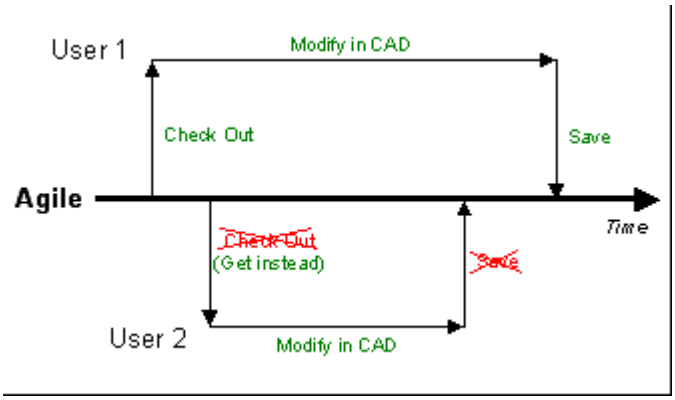
The dialog lists all change objects that are active within the current Save dialog, and shows the target workflow status (which is defined by your administrator). This target status may be different for different types of changes. You can check which changes to submit and then click the Submit button.

Concurrent Engineering

The EC Client is specifically designed to support concurrent engineering, the ability for multiple designers to work on different portions of the same overall CAD assembly at the same time. The most important consideration for concurrent engineering is that the ongoing changes by the designers be managed such that the files in the central repository (Agile) remain valid and up-to-date.

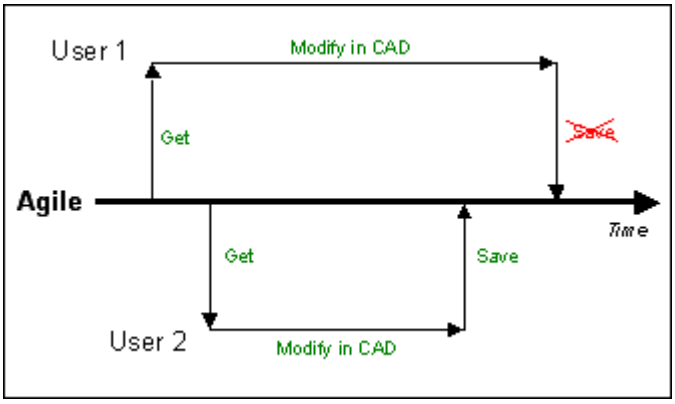
The EC Client makes use of two basic control mechanisms to manage concurrent engineering: Check Out and Timestamp. Check Out is a reservation mechanism inside Agile that is used to prevent other users from saving changes to something you are changing. Timestamp is a mechanism that relies on a timestamp value stored on each Document in Agile each time you save. If your timestamp is up-to-date, meaning that no one has made a change more recently than when you loaded the file, then you can save into Agile. The following diagrams illustrate the fundamental scenarios involving Check Out and Versioning.

Figure: Scenario 1 - Check Out Reservation



- User1 loads file from Agile using Check Out, and proceeds to modify in CAD.
- User2 attempts to load file from Agile using Check Out, but cannot since User1 already has it reserved. The file is loaded using Get instead.
- User2 modifies in CAD, then tries to save into Agile. Save is denied since User1 has checkout reservation
- User1 then tries to save, which is successful.

Figure: Versionsing



- User1 loads file from Agile using Get, and proceeds to modify in CAD.
- User2 also loads file from Agile using Get, and proceeds to modify in SolidWorks.
- User2 tries to save into Agile, which is successful (because there is no Check Out reservation)
- User1 then tries to save into Agile. Save is denied due to out-of-date version (file in Agile has been updated since User1 loaded it).

Note	Both Check Out and Version status can be checked from the Manage Change dialog. If a name is listed in the Checkout User field, then the item is checked out. If the Changed in Agile column is flagged, the item is out-of-date.

Note	All CAD files that are loaded from Agile into CAD are modifiable in the CAD system. That is, they are not loaded "Read-only". Just because you can modify the files in CAD does not mean that you will have the privileges to Save into Agile.

A configuration parameter called the SaveOption, which is set by your sytem administrator, determines how the Check Out and Version mechanisms are used to control concurrent engineering for your site. Mainly it affects under what conditions Save is allowed. Normally the value of SaveOption will be set to 2 (Medium) or 3 (High). The main difference between the two is that when it is set to 3, you are required to set the Check Out reservation in order to save; when set to 2 you do not have to set Check Out in order to save, you just have to be up-to-date.

So it is important to know what the value of SaveOption is for your site, because it determines whether it is required for you to use Check Out or not. Note that even if your site has SaveOption set to 2, you can still set Check Out reservations, in order to prevent other users from modifying files that you want to control.

Table: Agile Save Option

	Value of Save Option		
Checkout Status	1 (Low)	2 (Medium)	3 (High)
Checked out by someone else	Do not allow Save	Do not allow Save	Do not allow Save
Not checked out, and out-of-date	Allow Save	Do not allow Save	Do not allow Save
Not checked out, but up-to-date	Allow Save	Allow Save	Do not allow Save
Checked out by current user	Allow Save	Allow Save	Allow Save

Using Checkout Reservation

The mechanisms for managing the checkout reservation include the following:

Operation	Available Options
Check Out	Save, Load, Manage Change ▫ You can only check out during the load command if the Document is in an Introductory or Pending state. ▫ You can also create a new pending rev of a Document, then check it out. ▫ You will not be able to check out a released revision, you must create the pending revision first
Check In	Save ▫ By using check in during Save, the reservation is released and others now have access to the file.
Cancel Check Out	Save, Load, Manage Change ▫ Releases the reservation.

Check In and Check Out options for the Save command

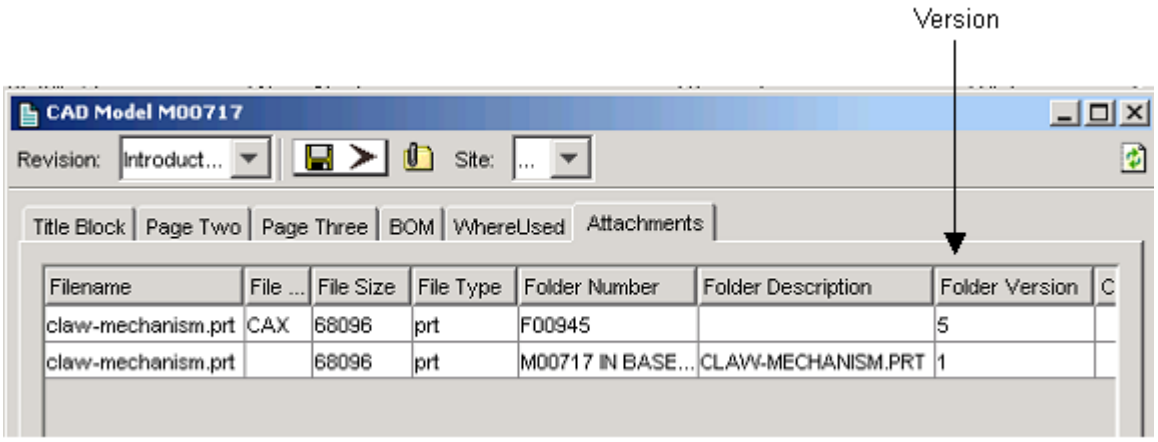
The Save dialog contains options for Check In and Check Out. These options can be combined in three ways that give you control over the reservation and visibility of each Document in Agile. These three cases are:

Version Control

Besides the Revisioning capability discussed above, the EC Client also makes use of Agile versioning capability. Each time you use the Agile > Save command to re-save a CAD file into an existing Document, it creates a new file version. Versions are tracked within the Attachments tab of the Document in Agile. Versions are actually related to the Agile "File Folder" concept, which is beyond the scope of this document. This is why in the Attachments tab it says Folder Version. The EC Client does not provide any special capability for accessing CAD files by version.

For example, you cannot select a version in the Load command, you can only Load by specific Revision. However, if you need to retrieve an earlier version of a file, you can extract it manually using the standard Agile client. This must be done one file at a time.

Figure: Agile Versioning



BOM Publishing

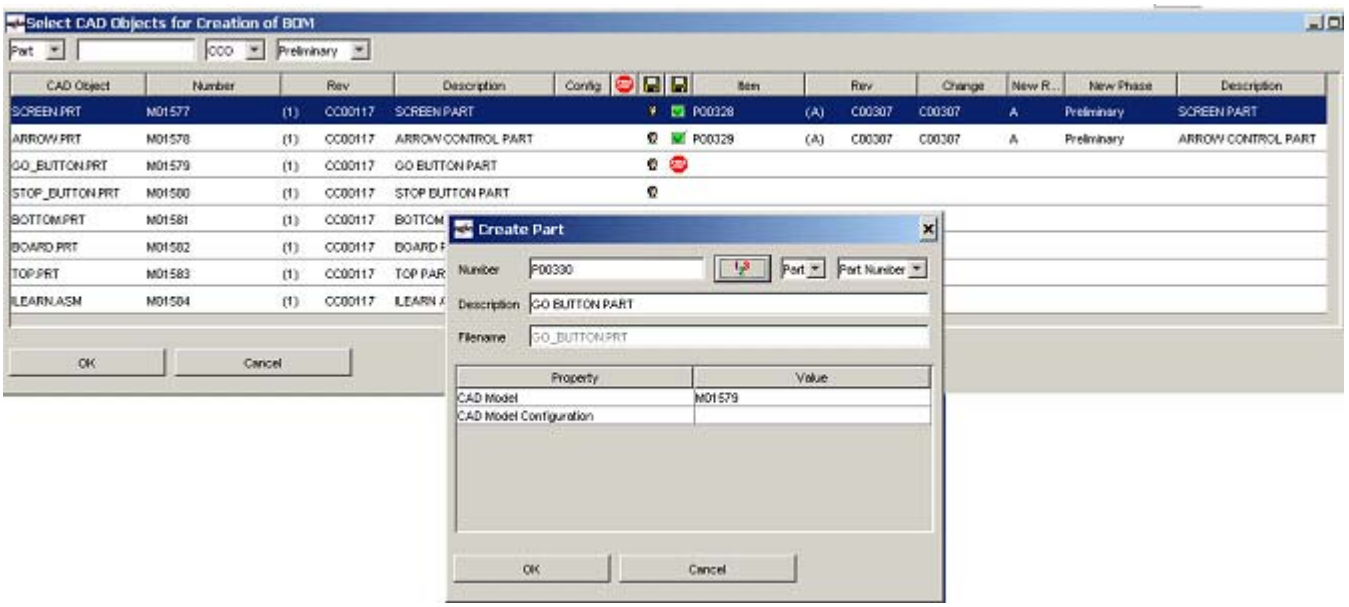
Introduction

BOM Publishing, using the Agile > Create Item/BOM command, is used to create or update Agile Product Structures based on CAD Design Structures. The Product Structure, or "Part BOM", is the definition of your product that is passed to manufacturing. Since in many cases this structure closely resembles the structure of your CAD design, the BOM Publishing step can leverage this to decrease effort and increase accuracy.

Using the Create Item/BOM Command

When you execute the Agile > Create Item/BOM command, the EC Client will pop forward, and display a dialog similar to the Save dialog, shown in the figure below.

Figure: Create Item/BOM Dialog



As with the Save command, after changing any options within the dialog (see below for details), you click the OK button to start the process. When creating an item for the first time, interactive mode is used which brings up a separate dialog window, in order to capture property information needed for the initial creation.

Details of the Save dialog are shown in the figures for the Create Item/BOM Dialog and Create Item/BOM Dialog Toolbar. They are also described below in the following two tables: Toolbar Options of the Create Item/BOM Toolbar and List Fields and Controls.

Figure: Create Item/BOM Dialog Toolbar

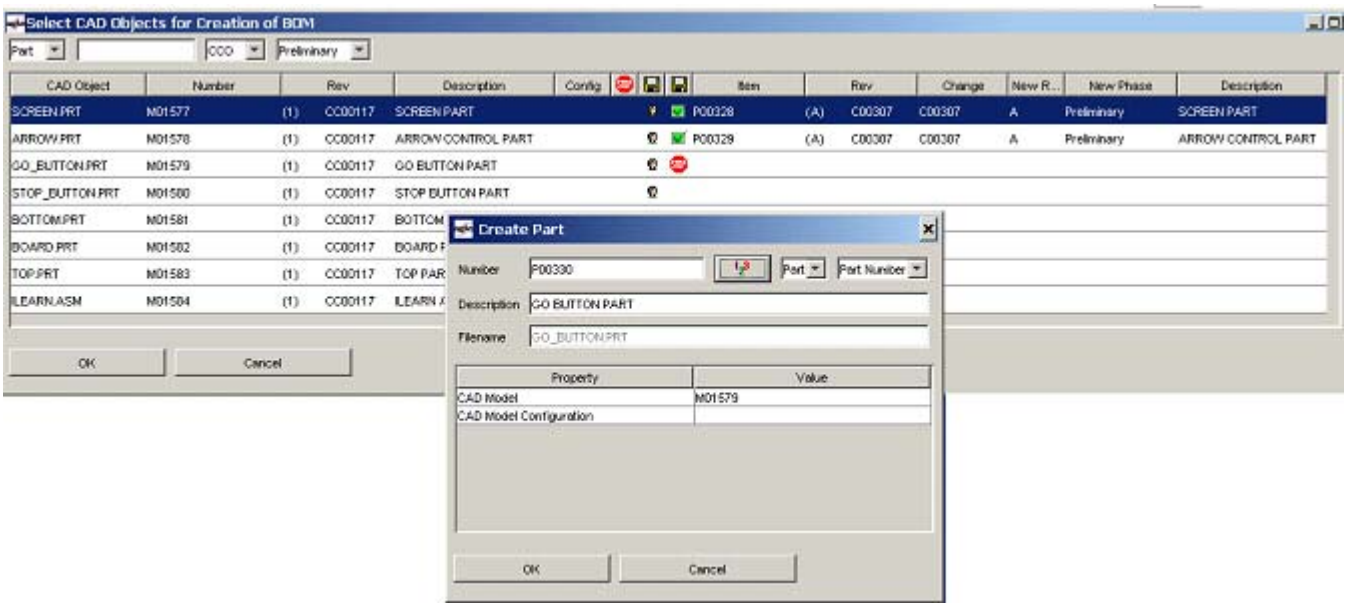


Figure: Details of Create Item/BOM Dialog

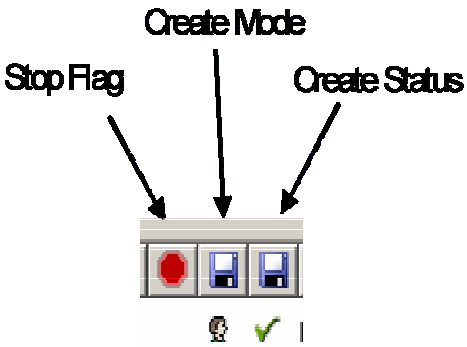


Table: Toolbar Options of the Create Item/BOM Toolbars

Part sub-class	Selector that allows you to pick the default Part sub-class to use for creating the Part BOM in Agile. Typically set to "Part". Can be overridden in the interactive dialog.
Current Change	Displays the Agile Change object that will be assigned when using the Assign to Current Change command. This is set either by using the Make Current Change command, or by manually typing in.
Change Sub-class	Selector for the Change sub-class to use when using the Create New Change command.
Default New Lifecycle Phase	Selector for the New Lifecycle Phase that gets assigned when you define a new revision for an item. Can be overridden by selecting a New Lifecycle Phase directly in the dialog (in each row).

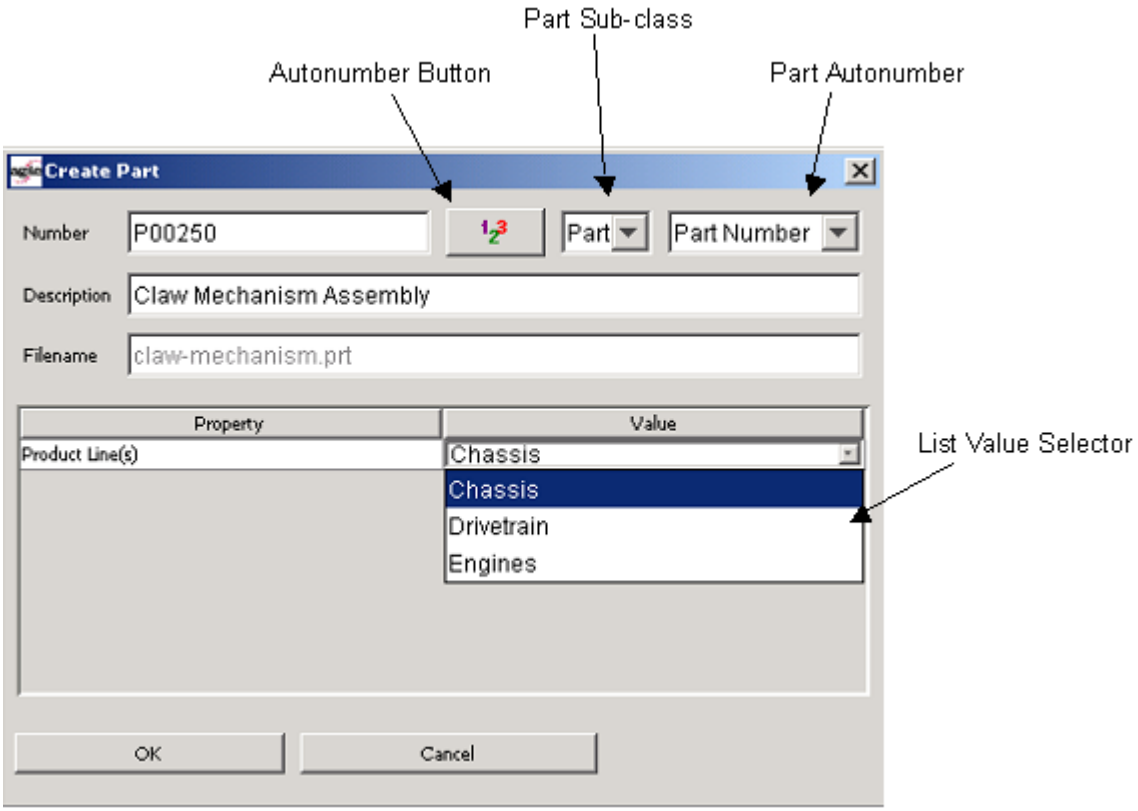
Table: List Fields and Controls

CAD Object	The CAD filename that is used as the basis of creating the Part
Number	Number of the CAD Model (Document) object in Agile
Rev	Current revision and ECO of the Document in Agile. Parentheses indicates a pending revision. The format of this field is the standard Agile revision/ECO combination, such as: (A) C12345
Description	Description of the CAD Model (Document) object in Agile.
Config	Indicates the configuration, if any, of the CAD file that is being referenced. Not supported by all CAD Connectors.
[Stop Flag]	Indicates whether or not you have the ability to create the Part object in Agile, based on your privileges. If no, a Stop Sign icon will be displayed in the column.
[Create Mode]  	Controls whether the Part is created/updated or not, and in what mode. There are three possible options, that can be selected directly in the user interface: <blank> - Do not create or update the Part - Create or update in batch mode - Create or update in interactive mode
[Create Status]  	Indicates the progress of the create/update operation as follows: - Part successfully created or updated - Create/update process is stopped at this point
Item	This field indicates the Part Number:· ▫ When the Part is initially being created, this will display the pre-defined mapping of the Part Number field, for this item. This can be overridden in the interactive dialog. ▫ Once the Part has been created, this field will show the Part Number that was used.
Rev	Current revision and ECO of the Part in Agile. Parentheses indicates a pending revision. The format of this field is the standard Agile revision/ECO combination, such as: (A) C12345
Change	This shows the ECO number that is assigned to this Part, to control the Part creation or update through the change process.
New Rev	The field allows you to change the Revision code that is assigned to the Part, for the new Revision.

New Phase	The field allows you to change the Lifecycle Phase that is assigned to the Part, for the new Revision.
Description	Description of the Part (Item) object in Agile.

Details of the Interactive Create Part dialog are shown below

Figure: Interactive Create Part Dialog



Number	The value that will become the Number assigned to the Agile Part that is being created.
Description	The value that will become the Description assigned to the Agile Part that is being created.
Filename	The CAD filename belonging to the Document, related to the Part being created.
Autonumber Button	If you click the button, it will put the next available autonumber from the selected sub-class and autonumber, into the Number field.
Part sub-class	Selector that allows you to pick the Part sub-class to use for creating this Part in Agile. This is for overriding the default value set in the main Save dialog.
Part Autonumber	Selector for the autonumber to use for creating this Part in Agile.

Property / Value Area	This area displays, and allows editing, for other properties that are being set from CAD into Agile. Properties can be either Text or List values. Text values are simply typed in, while List values are selected from a list.

Note Your site most likely has pre-defined mappings for Number, Description, and other properties. You should check with your administrator to understand the allowable values to use. Also, these properties can be set as "Required", meaning that you must enter a value before exiting the dialog. Required attributes are displayed in bold text.

Note You can cancel out of any individual interactive Part Create dialog (for example to check some other data in Agile), and the whole creation process will be paused at that point. If you click the OK button in the main Create Item/BOM dialog, it will re-start the process from where you left off.

Tip: You cannot use the Create Item/BOM command until you have first used the Save command to save the Design Structure in Agile.

Multi-Select and Context Menus

Since you will often have many items listed in the Create Item/BOM dialog, it is convenient to be able to set options for multiple items at a time. This is made possible by multi-select and context menus. To multi-select, simply click within any item in the window, and either hold down on the left mouse button and drag the cursor, or use Shift-click or Control-click. Once you have selected the desired items, you can use the context menu (right mouse button) to execute any of the commands listed in the following table.

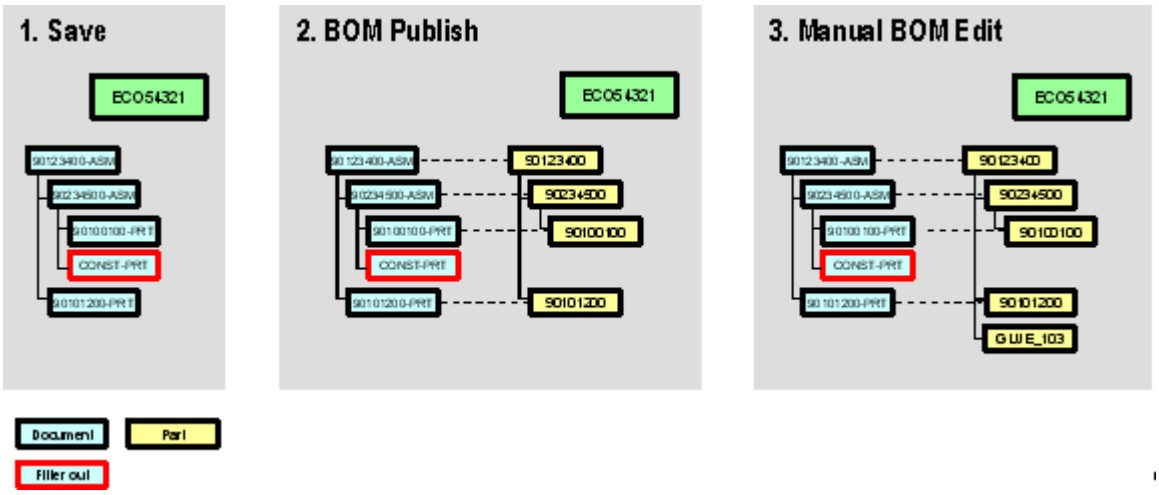
Table: Context Menus of Create Item/BOM Dialog

Assign existing item	Assign an existing item in Agile as the Part corresponding to this CAD file. If you pick this option the EC Client will allow you to select an existing Part.
Create Item Interactive	Sets all selected items to be created or updated to interactive mode.
Create Item Batch	Sets all selected items to be created or updated to batch mode.
Delete Item Reference	Disassociates the currently assigned Part, so that you can assign another one.
Create new Change (Autonumber)	Creates a new Change object, using the defined Change sub-class, and assigns selected items to it.
Make Current Change	Sets the number of the selected item's Change object into the Current Change field at the top of the dialog. This is so that you can use it for other items.
Assign to Current Change	Assigns selected items to the Change object shown in the Current Change field at the top of the dialog.

The BOM Publishing Process

The BOM Publishing process has three main steps, which are illustrated below

Figure: The BOM Publishing Process



Product Structure

Depending on how your administrator has configured the EC Client, the product structure created by the Create Item/BOM command will be one of four types:

1. Linked Structure – 3D Only

   The 3D CAD Model Document object (representing a CAD part or assembly) is linked in the BOM beneath its corresponding Agile Part object. Someone browsing the BOM structure will not immediately see the 2D CAD Model Documents (representing a CAD drawing), they can be accessed through the Where Used tab of the 3D CAD Model Documents.

2. Linked Structure – 2D & 3D

   Both the 2D (representing a CAD drawing) and 3D CAD Model Document object (representing a CAD part or assembly) are linked in the BOM beneath there corresponding Agile Part object. This can make browsing the data easier, although the BOM structure gets more cluttered.

3. Unlinked Structure

   The CAD Model Documents are not linked in the Part BOM at all. Rather, there is an attribute on the Part objects that gives the Document number of the corresponding CAD Model. Also there is an attribute on the Document giving the corresponding Part. This is not as convenient for navigation, but the Part BOM is much less cluttered.

4. Relationship linking

   The CAD Model Documents are linked in the Part BOM via Relationships. This allows the user to navigate to the Relationships tab to traverse over to the other structure. Note that this is supported in Agile 9.2.2 and later. Also note that Relationships are not revision-specific.

The figures below show the four BOM structure options, based on the originating CAD Model structure.

Figure: CAD Model Document Structure

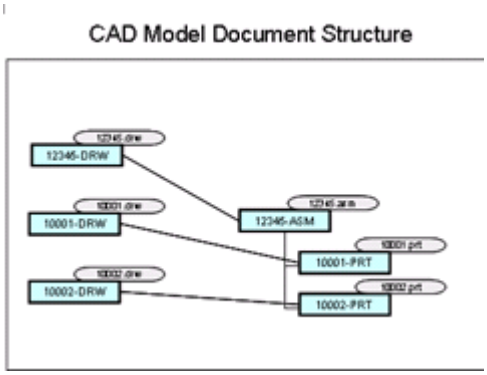
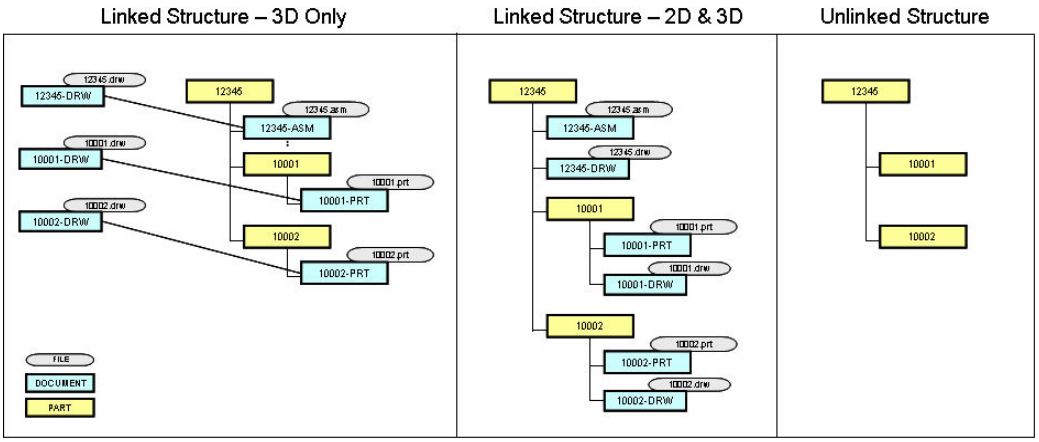
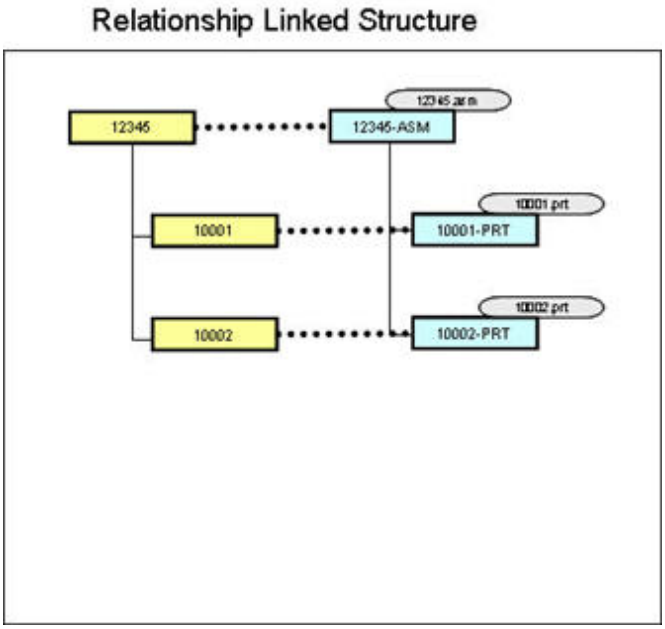


Figure: Part BOM Options





Configurations and Family Tables

Some CAD systems have special capability for dealing with families of parts. In SolidWorks this is Configurations, and in Pro/E and UG NX this is Family Tables. In each case, the EC Client will create correct BOM structures. Specifics are as follows:

SolidWorks: Since Configurations can be used for other purposes besides part families, you must flag parts and assemblies where you wish the configurations to be treated as separate parts. By default, this is by setting the Custom Property Configured = YES. When you use configurations of one of these parts or assemblies in your CAD model, using the Create Item/BOM command will generate a unique part for each unique configuration.

Pro/E and UG NX: Family table generics and instances are both maintained within the design structure. When using the Create Item/BOM command, a unique part will be generated for each instance or generic that is directly referenced within an assembly. No special parameter definition is

required.

Change Process for Parts

The Create Item/BOM command provides access to the ECO change process for parts, just like the Manage Change command provides ECO access to documents. While the process can be initiated from the EC Client, most of the workflow takes place using the standard Agile client. The configuration of the change process is very site-specific, however it is worth noting that there are two main approaches you can use. One is to the same ECO for both the documents and their corresponding parts. That is, if you authorize a change to a certain assembly, you can use a single ECO for the document (which creates the pending revision that can be modified in CAD), and for the part itself. This is the normal approach.

It is also possible to use separate ECOs for the document and for the part. In this case you would create a separate “CAD ECO” change object with a very simple workflow, just for the purpose of locking and unlocking your CAD designs.

Property Mapping

Introduction

Properties (also known as Parameters, Attributes or Metadata) are information stored as text strings that are associated with CAD data. Examples are part number, description, and author. The EC Client supports bi-directional transfer of properties between CAD and Agile. That is, you can enter a property in CAD, and have it be put into Agile, or vice versa. Properties are useful in the definition and classification of your design data, and are also useful for searching. The specific mapping of properties at your site is defined in the configuration file by your system administrator.

Types of Mapping

The figure below shows the types of mappings supported by the CAD Connectors. When mapping from CAD to Agile, there are two types of properties that can be mapped – System Properties and User Properties. System Properties are not directly defined by the user, they are things like the filename and the CAD software version number, which can be saved as properties in Agile. User Properties are defined by the user with the following commands:

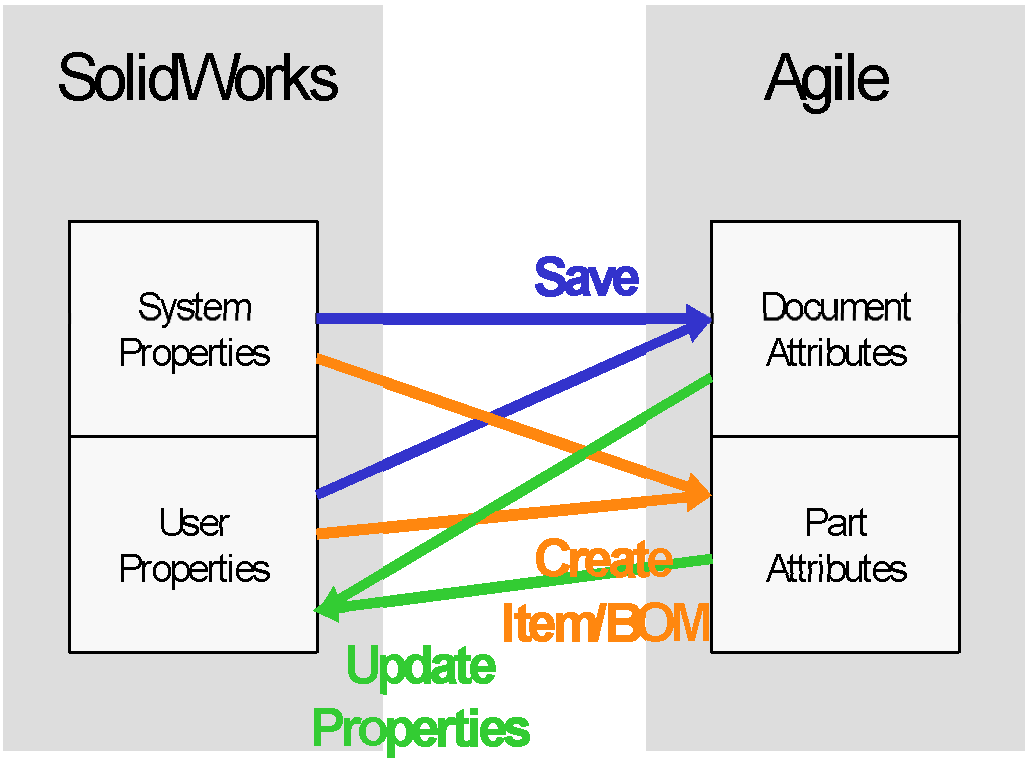
Pro/E: Tools > Parameters

SolidWorks: File > Properties

Unigraphics: File > Properties

CATIA V5: Tools > Formula

Figure: Property Mapping Options



Property mapping supports the following types of Agile attributes:

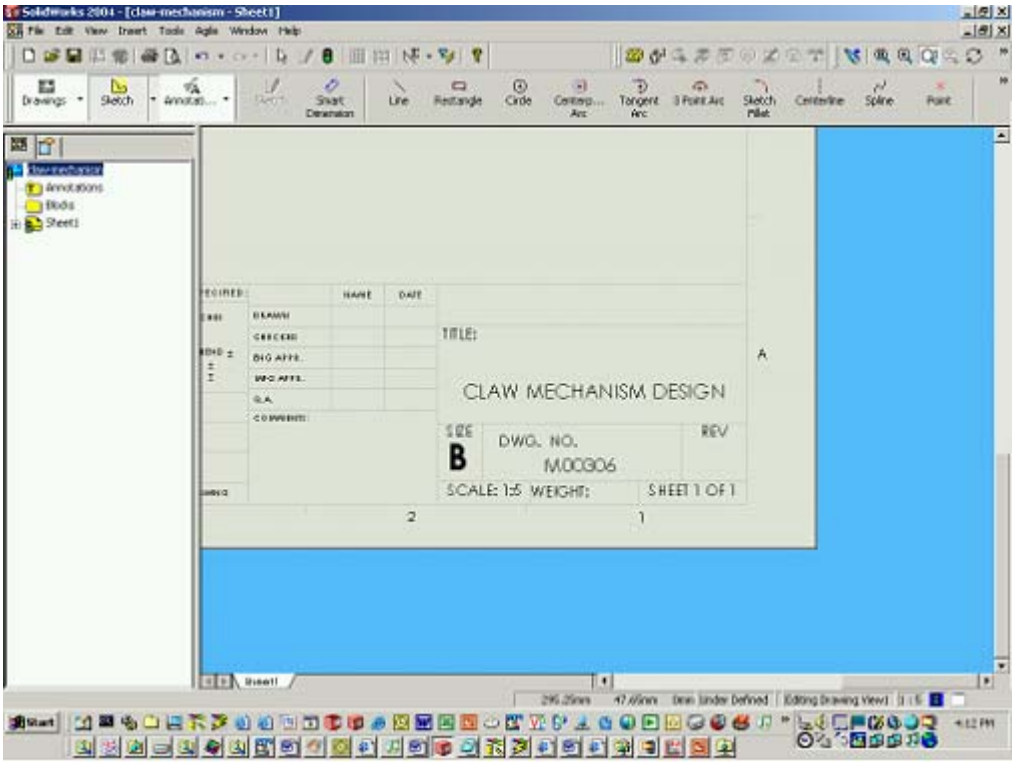
Text, MultiText, List, and MultiList.

System properties and user properties are mapped into Agile Documents as part of the Save command. They are mapped into Agile Parts as part of the Create Item/BOM command.

Mapping from Agile into CAD is done using the Update Properties command. It can also be configured to occur automatically during the Save process. Properties from both the CAD Document and the associated Part object can be mapped into CAD.

When working with drawings, there is another available command called Update Title Block. This updates properties just for the current drawing, not all subordinate models. In order to use properties within a Title Block, you need to define the text notes to be linked to properties, either within the drawing or within the part or assembly referenced on the drawing. This is standard CAD functionality. Figure below shows an example of properties used in notes within a title block.

Figure: Properties used in Title Block



CAD specific Functionality

Some CAD Connector functionality is specific to certain Connectors, because of unique capabilities in certain CAD tools. This section provides details on those specific functions.

Part Family Handling – Pro/ENGINEER and Unigraphics NX

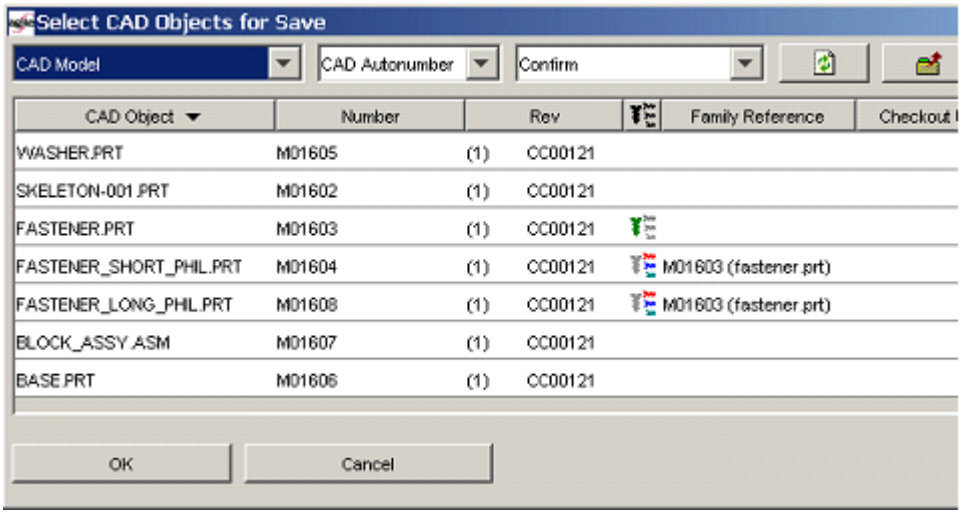
Introduction

Both Pro/ENGINEER and Unigraphics NX contain specialized functionality to manage families of parts and assemblies. This is called “Family Tables” in Pro/ENGINEER and “Part Families” in Unigraphics NX. The Agile CAD Connectors for these two tools provide additional functionality within the EC Client to display and manage part family information.

EC Client User Interface

In all four main EC Client dialogs – Save, Load, Manage Change, and Create Item/BOM, additional columns display the part family information. See Figure 3-3 for an example showing the Save dialog. If no family table parts are contained within the current CAD model, then the extra columns are not even displayed.

Figure: Additional Part Family Columns



The additional columns are:

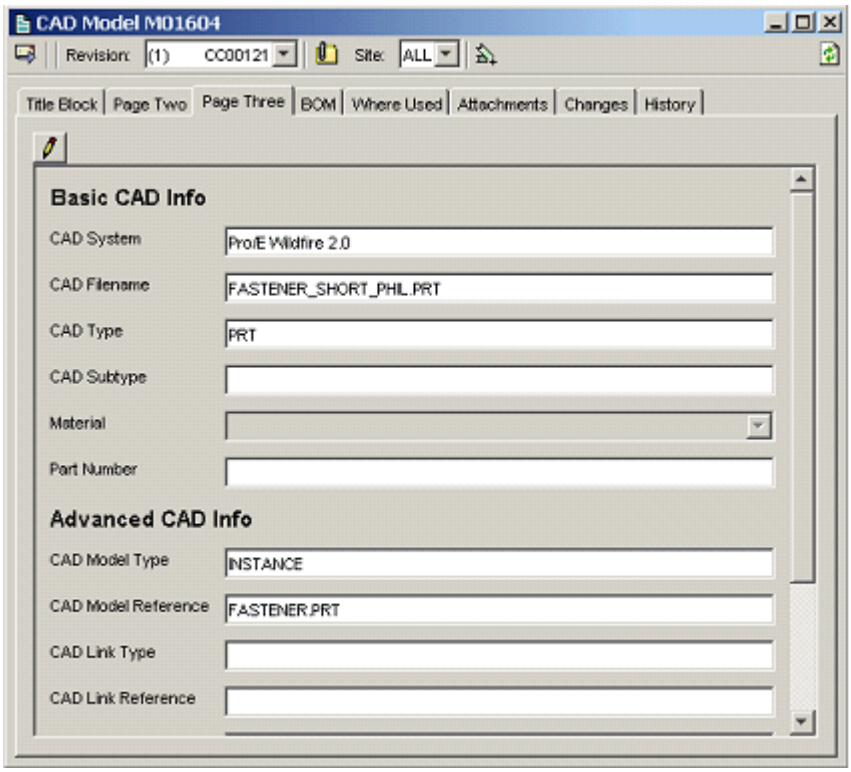
- Model Type – An icon column, where the icon indicates either an instance or a generic.
- Family Reference – Lists the Document Number and filename of the referenced model.

A Generic is indicated by the “Generic” icon in the Model Type column, and nothing in the Family Reference column.

An Instance is indicated by the “Instance” icon in the Model Type column, and the Document Number and filename of its corresponding Generic in the Family Reference column.

The information shown in these columns will also be shown on the form of each object in the EC Client, such as the “CAD Model Type” and “CAD Model Reference” shown in figure below (note that this form will look somewhat different in each customer environment). You can use these attributes to perform searches, for example on instances or generics.

Figure: CAD Model Type - CAD Model Reference



"Save Family Table" Command

In the Agile menu within the CAD tool, there is a command called Save Family Table, which allows you to save an entire part family at once into Agile. It brings up the Save dialog containing the generic and all instances of a part family. In order to use this function, you must have a part family generic part or assembly active in your CAD session. This command allows the user to save or update all instances of a part family plus the generic at once.

When using Pro/ENGINEER, this command also validates each instance and prompts you if there is an error with any of the instances. If there are any errors, the following dialog appears, and allows you to view the logfile of errors (which is also stored in your log directory for further access). If you choose to continue, the contents of the Save dialog will contain only the properly validated instances.

Figure: Save Family Warning

"Manage Family Table" Command

In the Agile menu within the CAD tool, there is a command called Manage Family Table, which allows you to manage an entire part family at once into Agile. It brings up the Manage Change dialog containing the generic and all instances of a part family. In order to use this function, you must have a part family generic part or assembly active in your CAD session. This command allows the user to set and cancel checkout, and create new pending revisions, for all instances of a part family plus the generic at once.

External Reference Handling – Pro/ENGINEER and Unigraphics NX

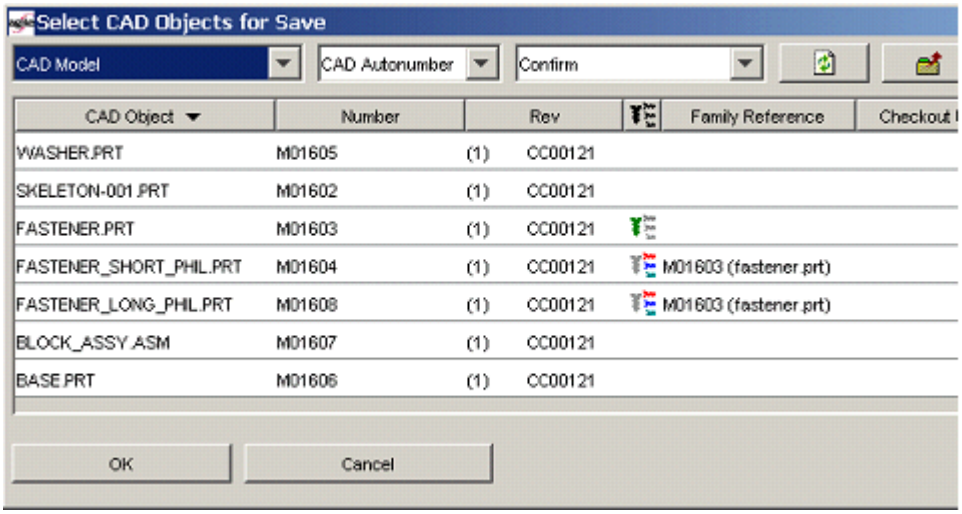
Introduction

Both Pro/ENGINEER and Unigraphics NX contain specialized functionality to manage external references. This is called "Merge References" in Pro/ENGINEER and "Wave Links" in Unigraphics NX. The Agile CAD Connectors for these two tools provide additional functionality within the EC Client to display and manage external reference information.

EC Client User Interface

In all four main EC Client dialogs – Save, Load, Manage Change, and Create Item/BOM, additional columns display the part family information. See Figure 3-3 for an example showing the Save dialog. If no family table parts are contained within the current CAD model, then the extra columns are not even displayed.

Figure: Additional Part Family Columns



The additional columns are:

- Model Type – An icon column, where the icon indicates either an instance or a generic.
- Family Reference – Lists the Document Number and filename of the referenced model.

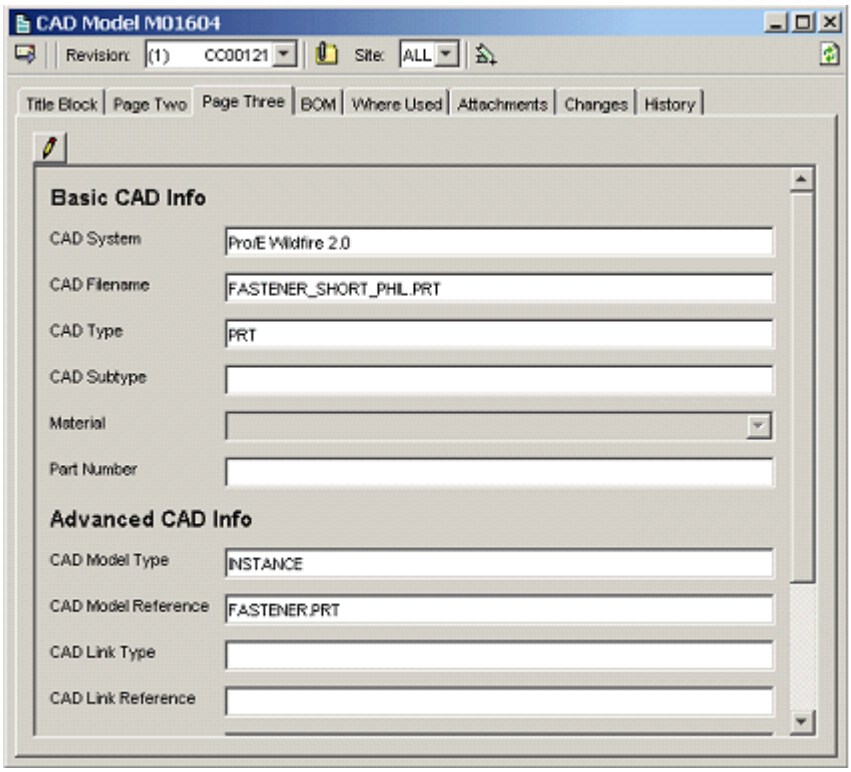
A Generic is indicated by the "Generic" icon in the Model Type column, and nothing in the Family

Reference column.

An Instance is indicated by the “Instance” icon in the Model Type column, and the Document Number and filename of its corresponding Generic in the Family Reference column.

The information shown in these columns will also be shown on the form of each object in the EC Client, such as the “CAD Model Type” and “CAD Model Reference” shown in figure below (note that this form will look somewhat different in each customer environment). You can use these attributes to perform searches, for example on instances or generics.

Figure: CAD Model Type - CAD Model Reference



CGR File Handling – CATIA V5

Introduction

Companies working with CATIA V5 commonly use CGR (CATIA Graphics Representation) files to simplify the representation of parts and assemblies that they are working with. The CGR format provides better performance when dealing with geometry that does not need to be modified (such as customer-provided assemblies that tooling is built from). The Agile CATIA V5 connector supports the management of CGR files.

Functionality Overview - Datamodel

In keeping with Agile standard methodology, which is that all representations of a given CAD model are stored together in a common Document record, both the native CATPart or CATProduct file and the corresponding CGR file will be stored in a common Document. Figure 1 shows the standard

CATProduct structure, while figure 2 shows the same structure with an added CGR file.

Figure: Standard CATProduct Structure in PLM

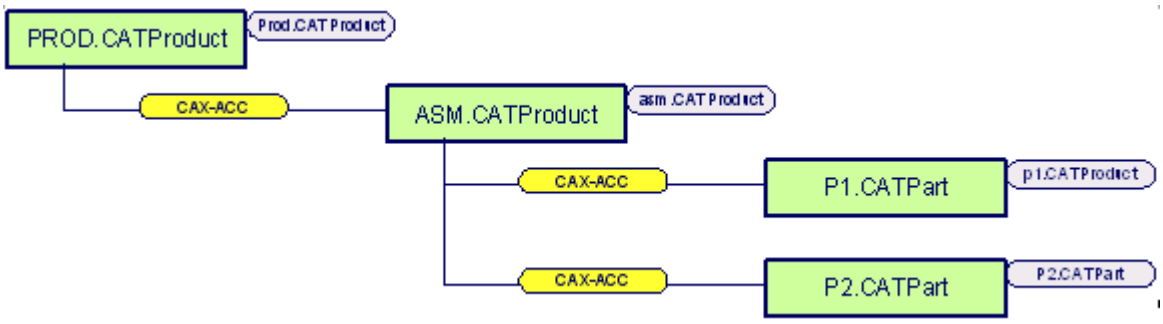
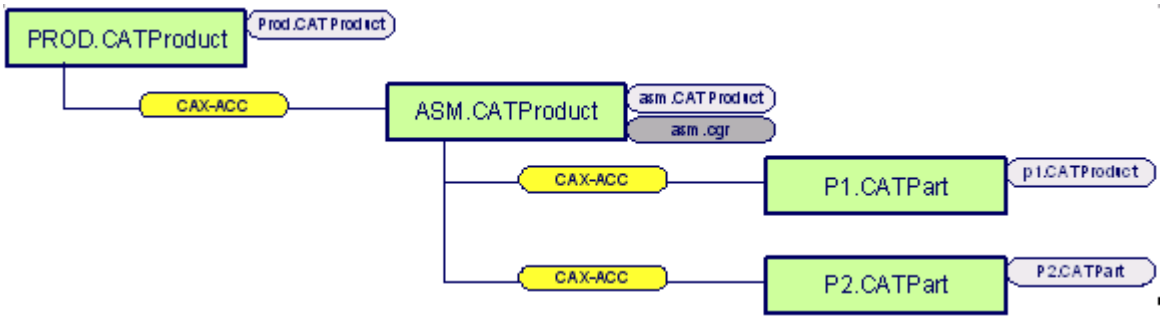


Figure 3-16: Structure with both CATProduct and CGR in PLM



The association between the CATProduct file and its corresponding CGR file is made through the new command “Save with CGR”. Once this has been done once, Agile will know that the two are associated. After that point, when the user uses the Load command to bring files from Agile into CATIA, the integration will use either the CGR or the native CATProduct, depending on whichever one was last saved (in its parent assembly) into Agile. This is “flagged” by a relationship attribute, denoted by the yellow bubble in figures 1 through 3. When the value is set to CAX-ACC by the Save command, upon subsequent loads the native CATProduct file will be loaded. When the value is set to CAX-ACC-CGR, subsequent Load commands will load the CGR instead. Note that this flag is set automatically by the Save command and should not be set manually by the user, or data corruption may result.

Figure: Assembly flagged to load CGR rather than CATProduct

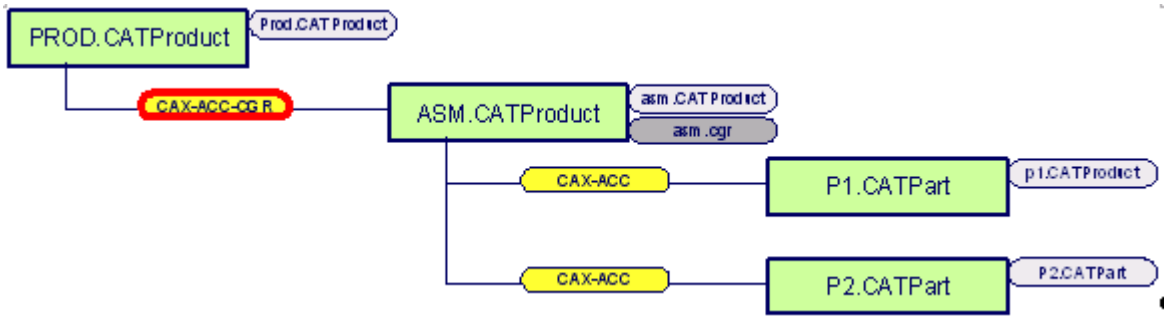
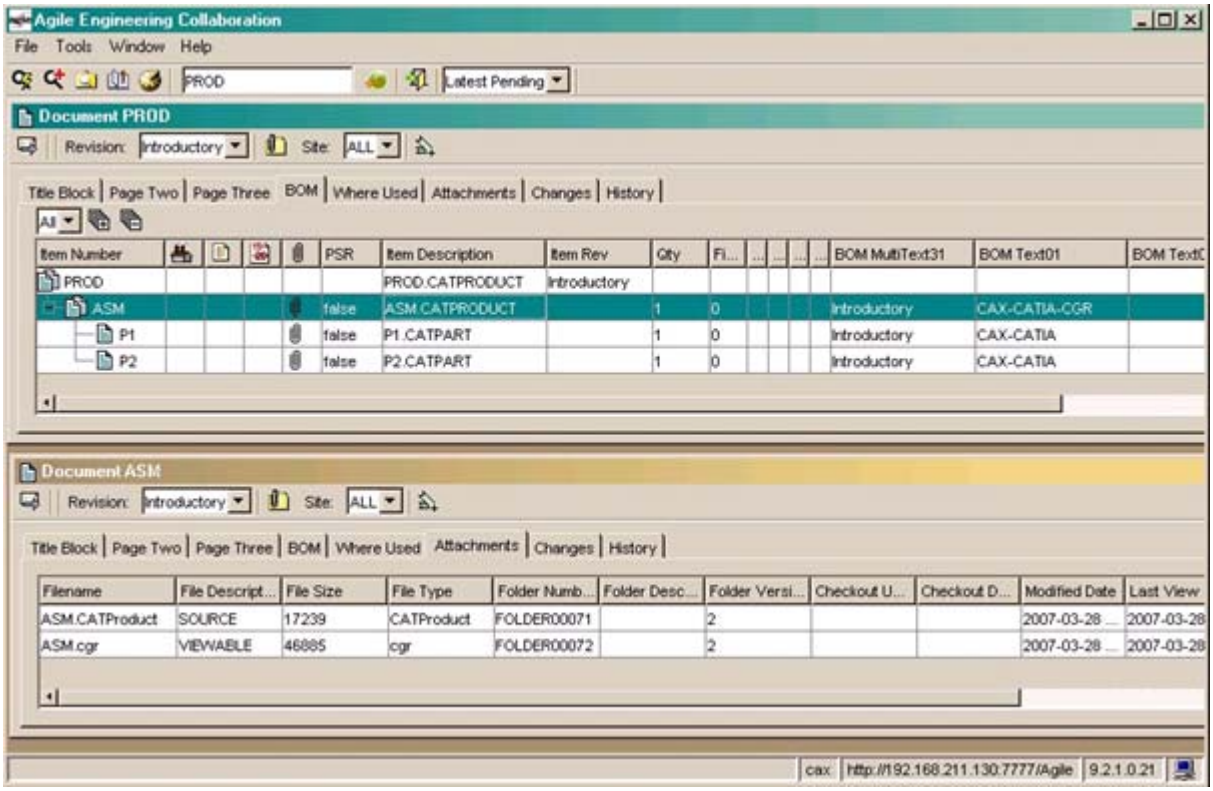


Figure: Document containing CGR file



The Catia representation is shown in the next two figures:

Figure: Screenshot from the Assembly PROD

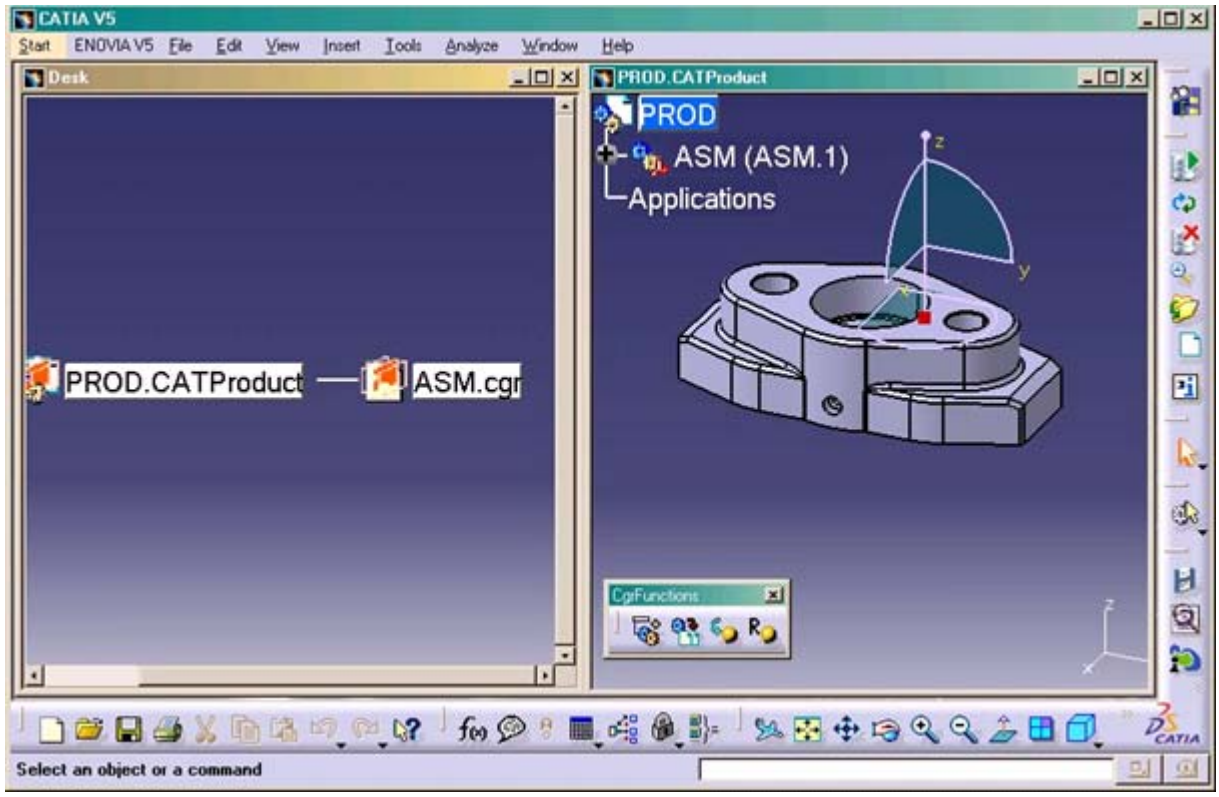
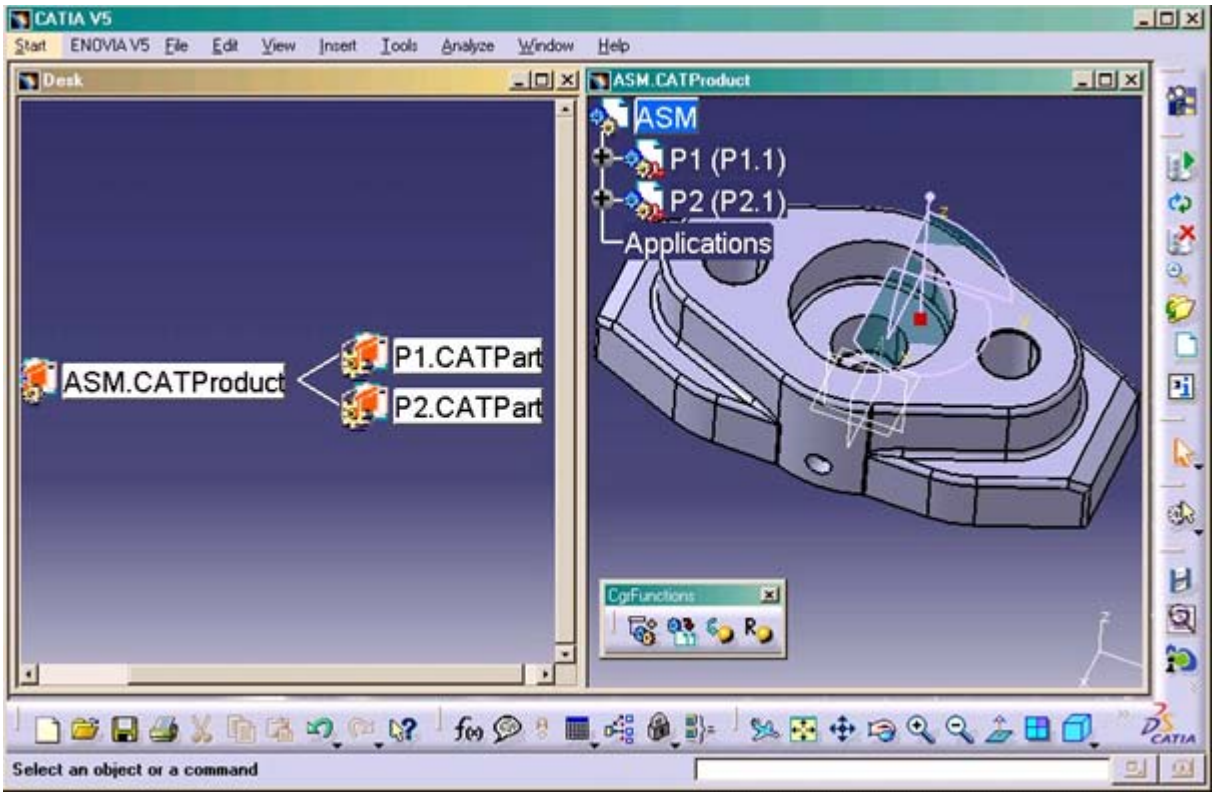


Figure: Screenshot from the Assembly ASM



In order to provide this functionality, the existing Save and Load commands are modified, and four new commands are provided as follows:

Insert CGR – Adds a CGR model to the current CATProduct

Save with CGR – Saves the CGR file along with the native CATPart or CATProduct file, and establishes the relationship between them

Open Native File – Opens the native CATPart or CATProduct files for the selected CGR file(s)

Reload CGR – Update the selected CGR file(s) with the latest version from PLM

Changes to existing commands

Save Command

The Save command is modified such that when a CGR file is saved, the relationship attribute between it and its parent is set to CAX-ACC-CGR, indicating that the CGR is active in the parent assembly.

If a Document has already an associate CGR File, the CGR File is created and checked in automatically.

Load Command

The Load command is modified such that when any assembly is loaded, the relationship attribute between each assembly and its children is checked to determine whether the CGR or the native CATProduct or CATPart will be loaded.

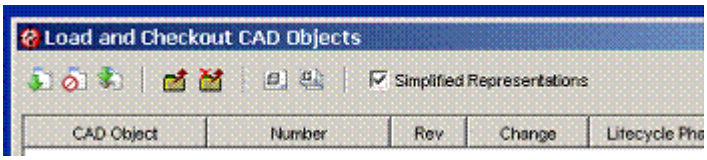
CGR commands

Table: CGR commands

Button	Command	Description
	Insert CGR (AglInsertCgr)	This command inserts an existing CGR file from PLM into the active CATProduct assembly. The functionality of the command is similar to the regular Load command, where the user is prompted to select a model to load.
	Save with CGR (AglSaveWithCgr)	This command creates or updates a CGR file from its corresponding native CATProduct or CATPart file, and then saves both files into a common PLM Document. This command associates the CGR and native files together, so that subsequent Save commands will know that the two are related.
	Open Native File (AglOpenCgrObjects)	This command allows the user to open the native CATProduct or CATPart file, that corresponds to one or more selected CGR files. This command is initiated by first selecting the CGR file(s) in the model tree, then clicking the command. The native files are opened in separate windows from the original CGR files. The user can then, for example, replace the CGR file with the native file.
	Reload CGR (AglReloadCgrObjects)	This command allows the user to update an existing CGR file with the latest version from PLM. This command is initiated by first selecting the CGR file(s) in the model tree, then clicking the command.

Simplified Representations – Pro/ENGINEER

Pro/E Simplified Reps are supported within the Load process. If you check the “Simplified Representations” box in the Load dialog, you will be prompted to select simplified reps when the files are loaded into Pro/E.



This page is blank