Agile Product Lifecycle Management

Agile Configuration Propagation

v9.2.2.1

Part No. E11169-01
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 ready 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

Welcome to ACP

This chapter includes the following:

Overview

This guide describes the purposes, installation, and use of Agile Configuration Propagation (ACP). ACP lets you propagate the configuration of one Agile instance to another instance of the same version. "The configuration" consists of all settings content of all (Java Client) Administrator nodes in one Agile instance. The propagation may consist of the complete “Admin” data for an instance, or it may consist of a selected subset of Administration data for an instance.

Purpose of ACP

- ACP facilitates the management of configuration data across multiple Agile PLM instances.

  Agile PLM is configured by the settings in Agile Administrator. The aggregate of defined settings in Administrator is the configuration data. ACP lets you test your configuration changes or additions outside your production environment so they are fully production-ready before they “go live.”

- ACP automates some processes that the Agile administrator had to do manually.

  The ACP utility provides an automated way to apply partial or complete configurations in one Agile instance to another Agile instance. Previously, Agile PLM 9.x has allowed the administrator to export only *limited* “Admin” configuration data from one Agile instance and import that data into a second instance.

Although ACP automates the configuration process, ACP can do only what it is told to do. It relies on data in its Control File to dictate what change in one Agile instance should be propagated to another instance.

You can save your configured control files – with descriptive filenames, and in project folders – and re-use them for targeted propagations.

Improper Uses of ACP

- ACP is not an upgrade utility. ACP cannot properly complete an upgrade of Agile PLM versions.
- ACP is not a synchronization tool. ACP expects a single source of record, an instance that owns the configuration.
- ACP is not a "mass-update" tool. It is not intended to supercede Java Client behavior.

If you have questions about what ACP may or may not accomplish for your Agile PLM installation, please contact your Oracle Consulting--Agile Practice representative.

What are ACP's Basic Functions?

These are the basic functions that ACP performs:

- Exports configuration data from Agile instance to XML files
- Imports configuration data from XML files to Agile instance
- Compares configuration objects between Agile instance and XML file, or between two XML files.

Who will Use ACP?

ACP is intended for Agile administrators, IT personnel, and Oracle Consulting – Agile Practice consultants. This manual assumes that you have worked in Agile Product Lifecycle Management (Agile PLM) as an Agile administrator. The Administrator module of Agile Java Client is documented in *Oracle | Agile PLM Administrator Guide*.

Who should not Use ACP?

ACP is not meant to be used by Agile end-users, or even by users that the Agile administrator have designated as "User Administrators" (that is, who have been assigned "Admin Access for User Admin" privilege mask).

Because this tool has the potential to completely change how an Agile system functions, ACP should not be used by anyone who does not fully understand the implications of any modifications to a live PLM system.

System Requirements for ACP

This section summarizes the requirements for installation and use of Agile Configuration Propagation.

Platform Requirements

- ACP supports all operating systems that are supported by Agile PLM 9.2.*x*: Windows, Solaris, and Linux. (ACP is not certified on AIX, although it may be used on AIX.)
- ACP is compatible with Oracle, BEA WebLogic, and IBM WebSphere application servers.
- ACP expects that the source and target instances use the same Agile Server License and the same Agile PLM version.

Server License Requirements

Implementing ACP requires no particular Agile Server License. ACP is an extension of the Administrator module in Java Client.

However, ACP *does* observe the Server Licenses of both source and target Agile instances:

- ACP can export data from a source instance based on Agile licensing on the source instance;
- ACP can import data to a target instance based on Agile licensing on the target instance.

Note	ACP can only propagate licensed configuration types, licensed configuration objects, and licensed class attributes.

Environmental Requirements

The ACP Client – where ACP is installed – must be able to connect via "`http`(`s`)" to the Agile-PLM source and target instances. That is, firewalls must not prevent ACP from accessing the PLM source and target.

Using this Manual

- Chapters 1–4 of this manual answer the question "*What* is ACP?"

  Chapters 1, 2, and 3 introduce ACP's purposes, terms, and use case.

  Chapter 4, "ACP Product Information," gathers information about ACP's product features, business rules, and observations of its behavior. *This chapter is essential reading for ACP users.* It is also intended for non-users who want to understand how the product works.

- Chapters 5–8 answer the question "*How* do I use ACP?"

  These chapters cover installation, user requirements, running ACP, and configuring the control file.

- Appendices A–F provide more detailed reference material.

  The appendices cover the configuration types, Java regular expressions, ACP properties, ACP scripts, ACP exit codes, and samples of a few ACP log files.

Chapter 2

ACP Terms

This chapter includes the following:

Process Terms

- Propagate – when an Agile instance has been changed, no matter if a single change to a single object's attributes or sweeping changes across the system, "propagate" is the broad idea of effecting the same change to another Agile instance
- Delete – a propagate action; the Delete section of the control file instructs ACP with *delete* actions
- Rename – a propagate action; the Rename section of the control file instructs ACP with *rename* actions
- Copy – a propagate action; the Copy section of the control file instructs ACP with *create*, *update*, and *replace* actions

Instance Terms

Your company may have several kinds of Agile PLM instances. The instance names below are simply suggestions, these names are not required in order to use ACP. However, these names are used in this manual as defined below, notably in ACP Use Case ("Use Case" on page 9).

- Agile instance – an autonomous system of Agile PLM that maintains data in a single database
- Golden Configuration instance – a controlled environment to make configuration changes in.

  The "Golden Config" Agile PLM instance is for maintaining a clean, controlled configuration environment. Normally, it would exist with only configuration data and no business data. The critical aspect to this instance is the amount of control on it. If your installation has a practice of making configuration changes in your Production instance, you will need to update your Golden Config instance periodically with configuration information from your Production instance.
- Development instance – for developing and testing new features in PLM, and other improvements

  The "Dev" Agile PLM instance is where any configuration changes should be tested before taking them to your Production instance. By its nature, the Development instance is not a controlled instance and its configuration may therefore not be "clean". It is prudent to update

the Dev configuration from Production before starting a new project. This will help avoid lost configurations.

- Stage or Test instance – for testing in a Production-like environment the changes made in the Development instance before they are applied to the Production instance

  The Stage Agile PLM instance is a Production-like instance that can serve two purposes: (1) perform dry runs of ACP prior to going live; and (2) an instance for training your user community on the feature changes provided by the latest configuration changes.

- Production instance – contains your company's live data and business objects

  The Production Agile PLM instance is the ultimate master instance. This is where business is being conducted. It must stay pure and always be in a consistent state.

Data Terms

- configuration data, Administrator data – the content of all the settings in PLM Administrator represent the configuration data for an instance of Agile PLM. These two terms are interchangeable.

- configuration type, Administrator node – the content of the settings of one node in the Administrator UI "tree."

  These terms are also interchangeable, although there is not one-to-one correspondence. Some Admin nodes are never propagated by ACP, and some config types are not nodes in Administrator.

  So, when you propagate the data settings held in the Roles *node* in Administrator (found by opening Java Client > click Admin tab > User Settings > Roles node), you are propagating the *Roles configuration type* in ACP.

- configuration object, Administrator object – the content of the settings for a single item in an Admin node.

  So, when you propagate the data settings held in the Change Analyst *role* in Administrator (found by opening Java Client > click Admin tab > User Settings > Roles > Change Analyst role), you are propagating the *Change Analyst configuration object* in ACP.

User Terms

- Agile user – a person at your company who has been created in the Agile PLM system and assigned an Agile role that permits use of the PLM system; also called "end-user" (or simply "user," although this could be confused with the other users listed below)

- Administrator user – an Agile user who has been assigned the Administrator role (or, more specifically, the Administrator privilege mask), which enables access to the Administrator modules in Java Client and Web Client; usually referred to as "administrator" (lower-case "a"), "Agile administrator," or even the "Admin user"

- User Administrator – a specialized Agile administrator (an Agile user who has been assigned the "Admin Access for User Admin" privilege mask) whose role is limited to management of users and user groups

- ACP user – a person at your company who uses the ACP utility could be called an "ACP user,"

but formally it means “an Agile user who has been given role/privilege access to use ACP.”

Machine Terms

- Agile Server – the machine where Agile PLM is installed
- ACP Client – the machine where ACP will be run
- Agile Install Directory – this is the directory you choose to install the ACP utility. "A" indicates where you would fill in the version of Agile PLM that you are working in, for example, "9223".

  Example in Windows: `<drive>\Agile\ACPA`

  Example in Unix: `/opt/agile/acpA`
- ACP Utility – the collective set of files installed in the <ACP Install Directory>
- Project Directory – this directory contains ACP control files, which are named – or in named folders – in such a way that you know what the control file is targeted to do. "A" indicates where you would fill in the version of Agile PLM that you are working in, for example, "9223".

  Example in Windows: `<drive>\Agile\ACPWorkA`

  Example in Unix: `<user.home>/Agile/ACPWorkA`

File Terms

- ACP scripts – since ACP is a command-line–driven utility, in effect the scripts are the “User Interface” for interacting with the ACP programs.
- ACP control file – the control file (default name: config.xml) directs ACP which configuration types to process; or it may be considered an example control file
- Agile XML archive – a zip file with an extension (.agl) that contains XML files exported by ACP

Chapter 3

Use Case

This chapter includes the following:

This chapter is an overview of the administration configuration management process of Agile PLM as a whole. You may choose to do some tasks below or add tasks that are not mentioned here.

Configuration Management

The primary use case for ACP is configuration management across two or more Agile PLM instances. Individual installations may vary in how instances are used and what configuration procedures are followed.

Note	Configuring the Agile PLM solutions requires full understanding of your company's business goals and procedures. Successful configuration of PLM – initial or upgrade – often requires a representative from Oracle Consulting – Agile Practice.

Agile Configuration Propagation supports regular and frequent cycles of developing, testing, and deploying Administrator configuration of the PLM solutions. ACP allows you to propagate the entire configuration or only a subset of the configuration. Regardless of the amount of configuration data being propagated, the following tasks offer recommended best practices, at least until you become familiar with ACP and its limitations.

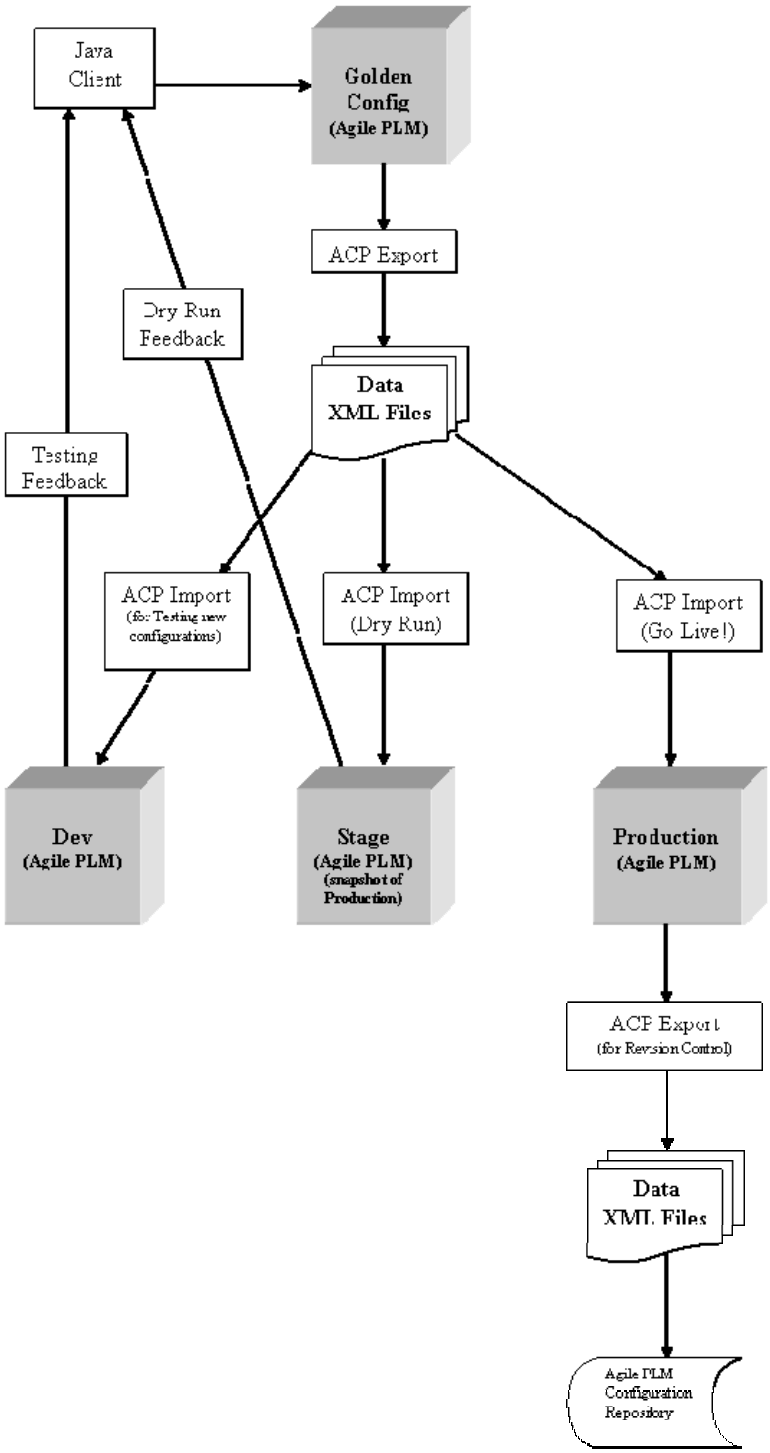
These tasks are in a suggested order to be executed, however, the order should be changed to facilitate your process.

For the discussion below, please refer to Instance Terms (on page 5) for the definitions of various kinds of Agile instances, such as Golden Config, Dev, Stage, and Production.

Configuration Tasks

The configuration tasks that are discussed in this section are also schematically depicted in the graphic below.

Schematic of PLM Configuration



Validate Agile PLM Data

Use aVerify to validate the data in an Agile PLM instance. A good practice whether or not you are using ACP is to periodically validate PLM data. Keeping your data valid ensures smooth operation of your Agile PLM instance. All Agile PLM instances should be validated before accessing it with ACP. Since ACP only propagates Admin data, you can limit your concern about data issues to Admin data. For information pertaining to aVerify, please consult *aVerify Release Notes* (on OTN, see Preface) or contact Agile Product Support.

Create a Project

Use ACP control files to denote a set of related configuration changes. Although you can propagate all configuration data in a single run with ACP, it is better to group related configuration changes. These groups can be called projects. Each project can then have a lifecycle of its own. In the control file, you can explicitly state which objects belong to the project. Projects can also be reused.

Configure Agile PLM

Use Java Client to modify Admin data in the source instance (Golden Config or Dev). Avoid making Admin data changes in your Production instance as those changes can get lost during subsequent ACP imports.

- Keep track of Admin objects changed in your project control file. This will ease the burden of trying to figure out which Admin objects need to be propagated later.
- Use a spreadsheet to keep track of actual changes made. This will aid in validating the propagation results.

Export Admin Data

Use ACP to export the Admin data related to a project from the source instance (Golden Config or Dev). Refer to Running ACP (on page 35) for more information.

Import Admin Data for Testing

Use ACP to import the Admin data that is related to a project. This task is oriented toward moving Admin data from the Golden Config instance to the Dev instance. This allows the new feature changes to be well tested before exposing the Production instance to these changes. The Dev instance is also an appropriate place to develop process extensions. The process extensions can also be tested with the modified Admin data.

- Use ACP to import the Admin data related to the project to the target instance.
- Review ACP logs for errors.
- Verify that the propagation worked as expected. Use the spreadsheet used to track changes for the project to verify that the Admin data changes are present in the target instance.

Test Admin Data Changes

Use the Agile PLM application to test the Admin data changes in the Test instance.

- Develop a formal test plan to test the changes.
- Develop a Go Live! “sanity test plan”.
- Execute the formal test plan.
- Validate the test plan results.
- Adjust the Admin data in the source instance (Golden Config or Dev) as necessary.

Prepare for Dry Run

Use ACP to prepare for a Dry Run. You want to make sure that ACP will propagate all Administrator data changes associated with a project. The way to do this is by comparing the configuration data in the source instance (Golden Config or Dev) and the target instance (Stage). This task comprises several steps:

- Create a backup of the Production instance.
- Refresh the Stage instance from the Production instance.
- Sanitize the Stage instance.
  - Turn off notifications.
  - Change Production-only Admin data to have suitable values for Stage.
- Compare the Admin Data between the source instance and the target instance.
  - Use ACP to export the Admin Data from the target instance (Stage). This can be the entire Admin data or just limited to the Admin data related to the project.
  - Use ACP to export the Admin Data from the source instance (Golden Config or Dev). This only needs to be done if you wish to compare the entire Admin data with the target instance.
  - Compare the XML files generated by ACP.
- Update the configuration data in the source instance as necessary.
- Update the project ACP control file as necessary.

Execute Dry Run

Use ACP to execute the Dry Run. Note: multiple Dry Runs may be warranted. Dry Runs typically use the Stage instance as the target instance. At this point, your project ACP control file should be configured to propagate all Admin data related to the project.

- Lock out users from logging into the target instance. This ensures your backup is valid.
- Back up target instance. It is best to back up your target instance for recovery purposes.
- Develop a Propagation Script. This script will document the exact steps used to propagate the data to the Production instance. It is used to document the order in which steps are performed

as well as any manual steps required.

- Use ACP to export the Admin data related to the project from the source instance (if not already done).
- Use ACP to import the Admin data related to the project to the target instance.
- Review ACP logs for errors.
- Verify that the propagation worked as expected. Use the spreadsheet used to track changes for the project to verify that the Admin data changes are present in the target instance.
- Execute the formal test plan.
- Adjust the Admin data in the source instance (Golden Config or Dev) as necessary.
- Re-run the Dry Run as many times as necessary to get a clean run of ACP.

Execute Go Live

Use ACP to execute the “Go Live” task. The Go Live task copies the Admin data changes from the source instance (Golden Config or Dev) to the Production instance.

- Lock out users from logging in to the Production instance. This ensures your backup is valid.
- Back up Production instance. It is best to back up your Production instance for recovery purposes.
- Follow the Propagation script created during the Dry Run.
- Use ACP to import the Admin data related to the project to the Production instance.
- Review ACP logs for errors.
- Verify that the propagation worked as expected. Use the spreadsheet used to track changes for the project to verify that the Admin data changes are present in the target instance.
- Execute the Go Live sanity test plan.
- Allow users to log in to the Production instance.

Audit Configuration Changes

Use ACP and a source control system to keep track of the changes made.

- Export entire Admin data from the Production instance.
- Place the AGL file created into any source control system.

Chapter 4

ACP Product Information

This chapter includes the following:

This chapter is mandatory reading for the ACP user. The rest of this manual builds on the information in this chapter.

"ACP Product Information" is also written to the general reader who wants an overview of ACP's product features, business rules, and observations of its behavior.

Installation of ACP

As a client application, ACP may be installed on the same machine as an Agile PLM server, or on a separate machine.

- ACP can be installed on the following operating systems: Windows, Linux, Solaris.
- ACP can work with the following application servers:
  - Oracle Application Server (OAS)
  - WebLogic Application Server (WLS)
  - Websphere Application Server (WAS)
- The ACP version must match the Agile PLM version it is interacting with. The key factor is this: the APIs that ACP uses must be available and have the same serialization.
- ACP can access an Agile instance from the same machine, over a LAN, or over a WAN.

  If the source and target instances are separated by a WAN, for better performance it is highly recommended that you copy the .agl file to the local LAN where the target instance is.

PLM Client Application

ACP is simply a client of Agile PLM.

ACP expects the source and target instance to be the same version of PLM.

(It is not required that the source and target instances be running on the same kind of application server.)

To use ACP, an Agile user must have the Administrator privilege mask for that Agile instance. There are specialized roles that can be tailored for users to have less than full access to the Administrator module of an Agile instance.

- ACP does not have “super-user” capability: it can access only those Admin objects for which the ACP-user has privileges.
- ACP adheres to the security policy assigned to the user used to connect to the Agile server. This security policy is managed through privileges. It is possible to have multiple ACP users: each user could be limited to different portions of the Administrator data.

PLM SDK Application

ACP is also considered as an Agile-PLM SDK application. As such, ACP adheres to all business rules imposed by the application. ACP interacts with Agile PLM in the same manner that Java Client interacts with the Agile application server.

ACP connects to an Agile instance using the same URL used by any SDK application. This URL is similar to the URL used to access the Agile Web Client (`http://<host name>[:<port number>]/Agile`).

- ACP is susceptible to any existing defects in the Agile server or SDK. It does not work around these defects.
- ACP does not access the Agile PLM database directly.

Propagation Tool

ACP has been expressly designed as a propagation tool. ACP’s purpose is to update or create Administrator objects in a target Agile instance just as they are configured in the source instance.

- ACP is not an upgrade utility. ACP cannot properly complete an upgrade of Agile PLM versions.
- ACP is not a synchronization tool. ACP considers only one Agile instance, the source instance, as the source of record for the configuration data: the source instance owns the configuration. It is also expected that the source Agile instance and target Agile instance are at the same product level (version).
- ACP is not a “mass-update” tool. It is not intended to supercede the Java Client behavior.

Propagation Strategies

ACP uses a “best-it-can-do” strategy: its goal is to make the propagated objects look the same in the target as it does in the source. Because PLM business rules, or unforeseen errors, can prevent ACP from fully propagating an object, sometimes this goal is not achieved; some objects may be partially propagated, or some objects may not be propagated at all.

- ACP does not propagate with an “all-or-nothing” strategy.
- If any errors are reported during the propagation, ACP cannot ‘guarantee’ that the propagation worked correctly.

Command-line User Interface

ACP is initiated via the command-line through an MS-DOS shell on the Windows operating system or one of the various shells available on the Unix operating system.

Several predefined “run” scripts are included with the ACP installation. These run scripts share variables and use command-line parameters to find:

- Connection information for connecting to Agile instances;
- Location of the control file;
- Location of an ACP XML archive;
- Log file information.

Propagation Method

ACP uses a two-step propagation method: the first step exports the objects to propagate from one Agile instance to an Agile XML archive; the second step imports objects in an Agile XML archive to an Agile instance.

This method allows you to place the Agile XML archive under source control.

This method can be used to overcome any performance issues ACP may have when used over a WAN.

Control File Drives the Propagation

The control file tells ACP what specific configuration types it should operate on and which ACP operations, or actions, will be carried out.

The control file has a defined schema, which allows you to use tools like XMLSpy to edit the control file.

The control file allows you to be as general or specific with the objects you want ACP to propagate through the use of regular expressions.

ACP does not track anything from propagation to propagation. Because it is not possible for ACP to determine what has been deleted in the source instance, the control file must explicitly direct ACP

to delete configuration items.

From reports, you determine if there are configuration items that should be excluded from the copy or configuration items that have been renamed and require a mapping.

- There is a shorthand notation for specifying all objects of an Administrator node.
- There is no limit on the number of include or exclude regular expressions that may be entered.
- Name maps must use exact names.
- Delete names must use exact names.
- Control file schema allows tools like XMLSpy to help in configuring the control file.
- The order in which Administrator nodes are configured has no bearing on the processing order.
- ACP does not propagate by a change list; Agile PLM Administrator does not support the notion of a change list.
- ACP does not automatically propagate dependent objects. All propagated objects must explicitly match one of the include patterns and none of the exclude patterns in the control file.
- ACP does automatically propagate the relationships between objects if both objects exist in the target. If only one of the objects in the relationship changed, it is only necessary to propagate that object. For example, if a role is added to a privilege, only the privilege needs to be propagated unless the role did not exist previously.

Propagation Actions

ACP supports these “actions” or operations: Copy, Rename, and Delete. The actions performed by ACP correspond to named sections in the control file.

- The Copy action allows you to propagate an object’s configuration from a source instance to the target instance using a Create, Update, or Replace operation;
- The Rename action allows you to rename objects in the target instance;
- The Delete action allows you to delete objects in the target instance.

There are two additional sections of the control file that do not direct propagation changes:

- Ignore References directs ACP to ignore references to certain objects that will not be propagated.
- Subobject Maps lets ACP make the connection between subobjects in the source and target instances;

These elements are described below, and are more fully detailed in Configuring the Control File ("Configuring the ACP Control File" on page 41).

Copy Action

The Copy action uses regular expressions to identify the objects to copy. The regular expressions come in two forms: include and exclude. If an object matches an include regular expression and does not match and exclude regular expression, then it is copied.

There is no limit to the number of expressions that can be specified. An object's name can be (or can be encoded to be) a valid regular expression itself. Therefore, you can list each object individually.

Copy contains these three “subactions”: Create, Update, and Replace. Create, Update, and Replace are implicit operations, that is, ACP decides which subaction will be performed, based on the existence of the object in the target instance, as described below.

- Create the configuration object if it is not found by name in the target instance.
  - If the source configuration object does not exist in the target instance by name, then the configuration object is inserted into target instance.
- Update the configuration object if it is found by name in the target instance.
  - If a match on name is found in both the source and the target instances, then the target configuration object is updated with the information on the source configuration object.
- Replace the configuration object if it is found by name in the target and its type in the source instance is different.
  - If the object's type differs in the target instance from the type in the source instance, Replace finds the configuration object in the target instance and deletes the object, then creates a new object in its place, thereby changing the type or meaning of an object.
  - Replace is used when the object being propagated is “strongly typed”. Some Administrator nodes are strongly typed, that is, in order to change the object’s type, it must be deleted and re-created. An example of a strongly typed Administrator node is Lists. In order to change a regular list to a cascading list, the list must be deleted and re-created.

Copy is fully detailed in Copy (<copy>) Section (on page 44).

Rename Action

The Rename action directs ACP to rename the object. The control file uses the XML term <map> to rename objects whose name was changed in the source instance. Name maps have a source name – which is the “new” name because the ACP user has changed it in the source instance – and a target name – which is the “old” name because it has not yet been changed in the target instance.

- A name map exists for the configuration object in the control file.

  If a configuration object is not found by its new name in the target instance, and it is found by its old name in the target instance, then the target configuration is updated with the new name.

Note	If an object cannot be renamed, an error is issued. If an object cannot be deleted, an error is issued.

- The names specified in the map must match the object’s name exactly; regular expressions are not appropriate here.
- There is no need to escape (or encode) regular expression special characters, for example, the asterisk ("*").
- Rename is not a kind of Update, Rename can only give an object a new name.
- Rename is an explicit operation, that is, ACP must be directed to take the Rename action.

Rename is fully detailed in Rename (<rename>) Section ("Rename (<rename>) Section" on page 49).

Delete Action

The Delete action allows objects to be deleted in the target. As a protective measure, the object name specified must match the object-to-delete's name exactly.

- A Delete directive exists for the configuration object in the control file.
- Delete is an explicit operation, that is, ACP must be directed to take the Delete action.
- Configuration objects of the particular configuration type are allowed to be deleted.
- The configuration object is found by name in the target instance.
- If the user has configured a configuration item to be deleted in the control file, the configuration item is deleted from the target instance if it exists. If the configuration item does not exist in the target instance, an error is issued.

Delete is fully detailed in Delete (<delete>) Section (on page 51).

Subobject Maps

The control file has a section for mapping names of subobjects, called Subobject Maps. "Subobject Maps" is not an ACP action.

To define subobjects in Agile PLM, consider that all PLM business objects are defined by a set of attributes. Some objects are also defined by a set of subordinate objects. These subordinate objects are objects just like the parent object they define, with the difference that subordinate objects cannot exist by themselves: they exist only because the parent object exists. An example of an object/subordinate object relationship is:

- List (object) and
  - List Entry (subordinate object).

Object Mapping is introduced in Object Matching (Mapping) (on page 22). Subobject Mapping is fully detailed in Subobject Maps ("Subobject Maps (<subobject_maps>) Section" on page 53).

Ignore References

The control file also has a section called Ignore References. "Ignore References" is not an ACP action.

The Ignore References section permits references to objects that will not be propagated to be ignored by the ACP actions. The ignorable references are limited to these kinds of Agile objects: privileges, roles, users, and user groups. Regular expressions may be used to specify which objects should be ignored.

Ignore References is fully detailed in Ignore References (<ignore references>) Section ("Ignore References (<ignore_references>) Section" on page 52).

Configuration Types

ACP propagates the data contents of Configuration Types, which are roughly the same as the

Administrator "nodes". Admin nodes are simply the named elements under Java Client's Admin tab. The Admin nodes are collectively called the "Admin tree."

Not all propagation operations are supported by each Administrator node. ACP Configuration Types (on page 57) shows the complete list of ACP-supported configuration types and the propagation actions that are supported by each type. This list is meant to be complete *for this specific release only*.

A caveat:

- An ACP user must have the appropriate Administrator privilege (privilege mask) to a given Administrator node for that node to be responsive to ACP's operations.

Administrator Nodes that are not Propagated

ACP does not propagate the following Administrator nodes:

- User Settings > User Groups > Personal User Groups
- User Settings > User Monitor – not propagated since it is a monitor
- User Settings > Deleted Users – ACP must be expressly directed to delete users
- User Settings > Deleted User Groups – ACP must be expressly directed to delete user groups
- System Settings > PCM > Currency Exchange Rates
- System Settings > PPM > UI Configuration Data
- Server Settings > Licenses
- Server Settings > LDAP
- Server Settings > Task Monitor – not propagated since it is a monitor
- Examples > Example Criteria – "read-only" out-of-the-box data
- Examples > Example Privileges – "read-only" out-of-the-box data
- Examples > Example Roles – "read-only" out-of-the-box data

Note	If you need to copy data from any of these nodes, it must be done "manually" using the Administrator Export/Import utility. See *Oracle | Agile PLM Administrator Guide*.

Admin Objects that Cannot be Propagated

The following Administrator objects cannot be propagated:

- Reports
- Sites
- Searches, that is, saved search queries (including Global and Personal searches)
- Lists with duplicate list entry names
- Classes (including base class, class, and subclass) with duplicate LifeCycle Phase names
- Workflows with duplicate workflow statuses
- Unit of measure families with duplicate UOMs
- Out-of-box character sets.

Admin Objects that do not Carry Attachments or Maintain Relationships in Propagation

The following Administrator objects can be propagated but their attachments are not propagated:

- Users
- User Groups

Also, Users and User Groups objects do not maintain Relationships in propagation.

Administrator Nodes that are in Web Client but not in Java Client

Because ACP only propagates Administrator data from Java Client, it does not ever propagate from Administrator nodes that are only found in Web Client > Tools > Administration, for instance:

- Project Summary Configuration
- Themes
- Cache Health Monitor

Note that Dashboard Configuration in Web Client corresponds to Dashboard Management in Java Client.

Object Matching (Mapping)

ACP uses object names to match (or map) objects between the source and target Agile instance. This fact is actually a limitation for ACP. Ideally, ACP would use public keys with which to match or map objects. Without public keys, ACP must match on the object name.

A public key is an identifier that uniquely identifies an object. “Key” indicates that the identifier may not be changed. “Public” indicates that the scope of the key’s uniqueness spans application and system boundaries. For purposes of Agile PLM, the term “public” is limited to a single customer; that is, the key does not need to be unique across all customers.

Unfortunately, a “name” in Agile PLM is not a good “key” because practically every kind of name can be changed in PLM. For this reason, objects whose name has been changed must be mapped in the control file.

There are Administrator nodes where objects cannot be created or deleted. For some nodes, the object cannot be deleted; in these cases, ACP leverages this fact and uses the object’s internal ID to match, which helps reduce the need for having to create name maps.

"Configuration Types and Match Fields" (in Appx. A) lists the configuration types and whether the match field is the external, changeable name of the object or the internal, unchangeable, identification number.

Limitations of Mapping

- ACP uses an object's name for matching where the customer can create or delete objects of the Administrator node.
- ACP uses an object's internal ID for matching where the customer may not create or delete

objects of the Administrator node.

- If an object is renamed and ACP uses the name for matching objects of that Administrator node, then a mapping must be specified in the control file for ACP to take the correct action.
- ACP matches subobjects by name.
- Subobject renames require a name map in the control file.
- Admin objects do not have a public key (see definition of public key in Object Mapping above)
- Flex Attributes cannot be renamed using ACP.

Processing Order in ACP

ACP performs deletes first, then renames, then copies. ACP controls the order in which Administrator nodes are processed. This order is dictated by the dependencies between each of the Administrator nodes. Deletes are processed in reverse dependency order. Renames and copies are processed in forward dependency order.

Due to some dependencies between Administrator objects, ACP may have to process the configuration type in multiple passes when copying. For example, users can belong to user groups and, conversely, user groups are populated with users. In order for ACP to process users and user groups properly, first users are created (in the target system), then user groups are created; only when users and user groups both exist in the target can they be associated with one another.

- ACP controls the processing order.
- ACP may process an Administrator node in multiple passes. These passes are listed separately in the log file.

Processing Order Rules

ACP defines the processing order of the configuration types based on application rules. The order is not influenced by the order the configuration types appear in the control file.

Copy Rules

- By default, all configuration objects for a configured configuration type will be copied.
- Include patterns match using Java regular expression rules.
- Java regular expressions have special characters that may need to be encoded. Direct them where to go for the list of special characters.
- XML special characters need to be encoded.
- Zero or more include patterns may be specified. There is no limit to the number of patterns you may specify.
- Exclude patterns match using Java regular expression rules.
- One or more exclude patterns may be specified. There is no limit to the number of patterns you may specify.

Rename Rules

- Zero or more name mappings may be specified.
- Name mappings use an exact match for both the old and new names.
- Regular expressions are not allowed.
- XML special characters must be encoded.
- Regular expression special characters do not need to be encoded.

Delete Rules

- By default, no configuration objects for a configured configuration type will be deleted.
- Delete names use an exact match for the name.
- Regular expressions are not allowed.
- XML special characters must be encoded.
- Regular expression special characters do not need to be encoded.

Configuration History

ACP does not propagate History information. However, since ACP uses the PLM application to propagate, a History record is created as the data is propagated. Since History items always apply to the specific user who created or modified Admin objects, it is helpful to assign propagation roles to Agile users, which simplifies tracking how the data was modified in the target instance.

Localization

ACP stores the text resources it uses in resource files, for example, labels and messages. Localized languages are:

- French
- German
- Japanese
- Simplified Chinese
- Traditional Chinese

Internationalization

ACP can be run once to propagate all language versions of the configuration data. Supported languages are:

- English
- French

- German
- Japanese
- Simplified Chinese
- Traditional Chinese

Caution If a customer has multiple languages installed, they might work in only one language. This could present a problem for ACP if, for instance, an object named "Capacitor07" is renamed to "Capacitor14" and then a new object is created and named "Capacitor07". Agile PLM allows this, but ACP may issue an error if the name already exists in a language other than the propagation language.

Chapter 5

User Requirements

This chapter includes the following:

This chapter looks at roles and privileges that give Agile users access to ACP, at the discretion of the administrator. Agile Configuration Propagation observes all business rules around Agile PLM's roles and privileges. For more information about configuring Agile PLM roles and privileges, please see the chapters "Roles" and "Privileges and Privilege Masks" in *Oracle | Agile PLM Administrator Guide*.

Standard PLM Privileges that can Access ACP

- Administrator privilege mask – permits full access to ACP functionality, as well as full access to PLM system(s)

  –OR–

- Admin Access for User Admin privilege mask – able to propagate only Users and User Groups

Remember, an ACP user must have the appropriate Administrator privilege (privilege mask) to a given Administrator node for that node to be responsive to ACP's operations.

Tailored Roles for the ACP User

Two specialized roles have been provided to permit use of ACP without permitting full Administrator access to the PLM system(s):

- (Propagation) Administrator role – contains the privileges that permit propagation of Administrator nodes
- (Propagation) User Administrator role – contains the privileges that permit propagation of Users and User Groups only.

Agile users who have been assigned either of these roles can log in to PLM with the username "propagation".

Privileges for the ACP User

The ACP user must have appropriate privilege masks to propagate configuration data from a source instance to a target instance. The privilege masks are based on the following privileges to specified object types in PLM:

- Read privilege – to the objects to be propagated from a source
- Discover, Read, Create, and Modify privileges – for all User and User Group attributes at a target instance
- Create and Modify privileges – to the objects to be created or modified at a target instance.

The AppliedTo property of all Administrator, Read, and Modify privilege masks must be correctly defined, as it allows the user to see specified Administrator nodes. The AppliedTo property is described in the chapter “Privileges and Privilege Masks” in *Oracle | Agile PLM Administrator Guide*.

Chapter 6

Installing ACP

This chapter includes the following:

Required Information

To install ACP, you must know what operating system, application server, and version of Agile PLM are being used.

Operating System

This is the operating system of the machine on which you intend to install ACP. The following operating systems work with ACP:

- Windows family
- UNIX family (for example: Solaris, Linux)

Agile PLM Version

The versions of ACP and Agile PLM that will work together must be the same release. For instance, ACP 9223.1 works with Agile PLM 9.2.2.3, but it does not work with Agile PLM 9.2.1.6, 9.2.2.1, or 9.2.2.2.

Application Server

The ACP Installer installs a specific set of files depending on the application server with which you connect to PLM. For ACP to communicate with Agile PLM, ACP and Agile PLM must be running the same application server. ACP works with the following application servers:

- Oracle Application Server (OAS)
- WebLogic Application Server (WLS)
- WebSphere Application Server (WAS)

Installation Directory

The installation directory is the directory (or folder) in which you want to install ACP. There are no

restrictions to where you may install ACP; however, the installation directory you choose should not pre-exist. If it does exist, you will be cautioned and asked if you want to proceed. If you do proceed, the existing directory will be purged.

The ACP Installer offers a default directory to install to, but you should be clear before installing exactly what directory is in line with your site installation policy.

Work Directory

The work directory is a directory where all project folders will be created. This is your directory to manage any way you see fit. The ACP Installer prompts you for your work directory so that it can place a sample project to get you started.

Prerequisites

Downloading Java Runtime Environment and the ACP Installer are prerequisites for installing ACP.

Java Runtime Environment

ACP requires Java's JRE 1.5.x to install and run ACP. You must download and install Java yourself.

ACP Installer

You can download a copy of the ACP Installer from the Oracle Support website: http://www.oracle.com/agile/support.html. Be sure to obtain the version that corresponds to the version of Agile PLM that you are using: please verify your version of PLM before selecting the version of ACP Installer. The ACP Installer is approximately 147 MB in size.

Windows Installation

To install ACP on a Windows-based system, follow the steps below.

Extract

The ACP Installer is contained in a standard ZIP file. You can unzip the files to a directory of your choice.

Run Installer

It takes two minutes or less to set up and install ACP.

Open Command Window

The ACP Installer is an (Apache) Ant-based installer. It must be run from a command window.

Go to ACP Installer Directory

Change the directory to the directory in which you unzipped the ACP Installer.

Run the Installer Script

1. Set JAVA_HOME

In order to run the ACP Installer, the environment variable JAVA_HOME must be set; it should be set to where you installed Java JRE 1.5.

```
SET JAVA_HOME=<Java JRE 1.5 Directory>
```

2. Run Installation Script

Run the specific installation script to install ACP on Windows systems:

```
install_win
```

When the script starts, the Oracle/Agile splash screen is presented, which indicates progress of the installer as it is working.

3. Answer Prompts

The installer prompts you for required information. The prompts have default responses indicated by brackets ([]). Press the Enter key to accept the default response.

a. Are you sure you want to install this version of ACP? ([y], n)

   Just above this prompt, the ACP Installer displays the version of ACP that will be installed. Please verify that this is the version you intended to install. If it is not, you must respond “n” to this prompt.

b. Select App Server Platform to use? ([oas], was, wls)

   The application server platform is based on the application server used by the Agile PLM instance to which ACP will connect. You can enter only an abbreviation to designate the application server:

   - Oracle App Server: oas
   - WebSphere App Server: was
   - WebLogic App Server: wls

c. Enter directory to install ACP? [C:\Agile\ACP<####>]

   This is the directory to which the ACP binaries and other files will be installed. The ACP Installer tries to provide a reasonable default location for you to install ACP, but you can use another location.
   “<####>” represents the version of Agile PLM with which this version of ACP works.

d. Enter the work directory to use with ACP? [C:\Agile\ACPWork<####>]

   This is the directory where you will create ACP project directories; ACP is run from a project directory. This directory can be relative to the installation directory you entered, or it can be a completely separate directory tree. “<####>” represents the version of Agile PLM with which this version of ACP works.

e. The chosen ACP install directory already exists. Do you want to continue? (y, [n])

   This prompt appears only when the install directory you specify already exists. This prompt cautions you about potentially installing over another application or overwriting an inappropriate directory. If you wish to continue, you must enter “y” as a response.

UNIX (Linux) Installation

To install ACP on a UNIX- or Linux-based system, follow the steps below.

Extract

The ACP Installer is contained in a standard ZIP file. You can unzip the files to a directory of your choice. When unzipping the ZIP file, use the -a option to make sure text files are extracted properly.

```
unzip -a acp_install.zip
```

Make Executable

For UNIX and Linux users, the ACP Installer must be prepared for execution. This is simply a matter of making the appropriate install script executable. The script exists in the directory to which you extracted the ACP Installer ZIP file.

```
chmod u+x install_unix
```

Run Installer

It takes two minutes or less to set up and install ACP.

Open Terminal Window

The ACP Installer is an (Apache) Ant-based installer. It must be run from a terminal window.

Go to ACP Installer Directory

Change the directory to the directory in which you unzipped the ACP Installer.

Run the Installer Script

1. Set JAVA_HOME

In order to run the ACP Installer, the environment variable JAVA_HOME must be set; it should be set to where you installed Java JRE 1.5.

```
JAVA_HOME=<Java JRE 1.5 Directory>
```

2. Run Script

Run the specific installation script to install ACP on UNIX/Linux systems:

```
install_unix
```

When the script starts, the Oracle/Agile splash screen is presented if X11 is running on your system. This indicates progress of the installer as it is working.

3. Answer Prompts

The installer will prompt you for the required information. The prompts have default responses indicated by brackets ([]). Press the Enter key to accept the default response.

a. Are you sure you want to install this version of ACP? ([y], n)

Just above this prompt, the ACP Installer displays the version of ACP that will be installed. Please verify that this is the version you intended to install. If it is not, you must respond "n" to this prompt.

b. Select App Server Platform to use? ([oas], was, wls)

The application server platform is based on the application server used by the Agile PLM instance to which ACP will connect. You can enter only an abbreviation to designate the application server:

- Oracle App Server: oas
- WebSphere App Server: was
- WebLogic App Server: wls

c. Enter directory to install ACP? [/opt/Agile/ACP<####>]

This is the directory to which the ACP binaries and other files will be installed. The ACP Installer tries to provide a reasonable default location for you to install ACP, but you can use another location.
"<####>" represents the version of Agile PLM with which this version of ACP works.

d. Enter the work directory to use with ACP? [/<Home>/<user>/ACPWork<####>]

This is the directory where you will create ACP project directories; ACP is run from a project directory. This directory can be relative to the installation directory you entered, or it can be a completely separate directory tree. "<####>" represents the version of Agile PLM with which this version of ACP works.

e. The chosen ACP install directory already exists. Do you want to continue? (y, [n])

This prompt appears only when the install directory you specify already exists. This prompt cautions you about potentially installing over another application or overwriting an inappropriate directory. If you wish to continue, you must enter "y" as a response.

Postinstallation Tasks

Update Security Property Files

The WebSphere Application Server requires additional authentication. This security information is stored in property files that require "environment-specific" information. The ACP installer provides a copy of these files when you choose WebSphere as the app server for ACP.

To replace WebSphere security property files:

1. Start Java Client from the machine on which you installed ACP.
2. Connect Java Client to the Agile PLM instance running on WebSphere. This downloads the necessary security property files from the Agile PLM server.
3. Copy the sas.client.props and wsjaas_client.conf properties files from `<Drive>:\Documents and Settings\<user name>\.agilecm` folder (or in UNIX family, `<user home>/agilecm`) to `<ACP Client Install Directory>\properties`.

Important	If you have an SDK license, you can obtain more information regarding Websphere Security in the "Configuring an SDK Client for Websphere" chapter in the *Agile SDK Developer Guide*.

ACP-Installed Directories

You might require multiple instances of ACP to work with multiple versions of Agile PLM, for example, if you are beginning the Upgrade process to a new PLM release but must maintain updates in the earlier, live version of PLM.

```
\Agile PLM                      <- where Agile PLM is installed
    \ACP                        <- where ACP client is installed
        \ < version >           <- level for multiple instances of ACP
            \bin                <- standard installed directories (see below)
            \classes
            \lib
            (Etc.)
```

ACP Client Installed Directory Structure

The following directories are owned by the installation. Do not modify the files in these directories, as they are subject to change without notice.

- ant – contains a subset of the Ant utility. ACP uses Apache Ant for project management.
- bin – contains scripts to run the ACP utility programs
- classes – contains text resources that may be localized (that is, translated) or customized
- lib – contains the set of libraries that are required for the ACP utility.
- properties (Websphere only) – contains the security property files required to authenticate with a Websphere app server.

  Note These files must be modified for your installation.
- schema – contains the schema file for the ACP Control File
- schema / docs – contains the online documentation for the ACP Control File, acp_control_file.html
- templates – contain template project files and property files

ACP Work Directory Structure

It is a best practice that you organize your configuration needs into "projects". Each project is maintained in a separate project folders. Project folders are contained in the work directory specified when ACP is installed.

This topic is the starting point for the next chapter, “Running ACP.”

Chapter 7

Running ACP

This chapter includes the following:

This chapter discusses the operating components in ACP, which provides a general understanding of how to run ACP. Agile Configuration Propagation consists of the following components: Projects, Properties, Control File, Scripts, Return Codes, and Log Files. Most of these components are described in further detail in succeeding chapters or appendices.

ACP Projects

Important ACP Project directories should *not* be under the installed ACP path. In order to prevent the accidental deletion of project directories, the work directory should be kept separate from the install directory.

It is a best practice that you organize your configuration needs into independent configuration projects. Each project is maintained in separate project folders. Project folders are contained in the work directory that is specified when ACP is installed.

A project is defined by its properties file and control file:

- The properties file `project.properties` provides the information that ACP needs for interacting with the environment;
- The control file `config.xml` tells ACP what objects to process.

ACP Project Directories

To help you get started, the ACP installer creates a project named “sample” for you. You can use this project or create other projects when needed.

- sample – a sample project directory created for you by the installer.
- <project> or <ACP projects>, for example – a project directory created by you; this directory does not exist until you create it. There may be multiple project directories.

Sample Project Directory

The work directory contains project folders. Projects are defined by the control file stored in the

project directory.

For example, you might create projects for List management, for Roles and Privileges, for release-based Agile Classes, and for each new feature that you want to deploy:

\Agile PLM	<- where Agile PLM is installed
\ACP	<- where ACP client is installed (ACP Project directories should *not* be under the installed ACP path)
\ACP Projects	<- "work" directory, where ACP Projects are organized
\<project_9223>	<- a directory that collects ACP projects oriented to PLM Release 9.2.2.3
\Roles_Privileges	<- control file used to propagate Admin roles and privileges
\Lists	<- control file used to propagate Admin lists
\9223_Classes	<- control file used to propagate Admin classes in PLM Rel. 9.2.2.3
\9223_Subclasses	<- control file used to propagate Admin subclasses in PLM Rel. 9.2.2.3
\New Feature_01	<- control file used to propagate a specific new feature
\New Feature_02	<- control file used to propagate a specific new feature

Important If ACP projects are placed in a work folder, the work directory should not be the same or contained in the Install directory.

As you can see, you can have as many project directories as you like, and you can name them anything you like.

Creating Projects

There are two ways in which to create an ACP project:

- You can copy an existing project; or,
- You can use the create_object script provided by ACP to create a new project.

Existing Project

This method allows you to start from an existing project. Project folders must be created in the work directory specified when installing ACP.

1. Identify the existing project folder you wish to copy.
2. Create a copy of the project folder.

   Use your favorite method for copying folders:

   - Windows: Copy/Paste
   - DOS: copy command
   - Unix: cp command.
3. Rename the copied folder.

   Recommendation: Indicate the purpose of the project in the name you choose.
4. Delete old project files.

Since you are copying a project, you will be copying files that do not apply to the new project.

Here is a list of files that you should consider deleting from the new project folder:

- Log Files (*.log, *.err)
- ACP archives (*.agl)
- Other files you may have created

The new project should only have the launch script (`acp`), the project properties file (`project.properties`) and the control file (`config.xml`).

New Project

This method allows you to start with a clean project. Project folders should be created in the work directory specified when installing ACP.

1. Choose a name for the new project folder. The folder name cannot already exist.
2. Open a terminal window (Windows: DOS window; Unix: shell)
3. Change directory to the work directory specified when ACP was installed.

   Here are examples based on the default work directories provided by the ACP Installer.

   - Windows: `cd C:\Agile\ACPWork9223`
   - Unix: `cd /<user home>/agile/acpwork9223`

   Of course you will tailor any numbered directory folders to match the specific PLM version, for example, "9216", "9221", and so forth.
4. Run the ACP create project script.

```
acp create_project <project_name>
```

ACP Properties

ACP uses properties to understand how to interact with the environment it is running in. ACP properties can be set by the program, on the command-line, in the Project Properties file, as well as common properties files. ACP properties are described in greater detail in the ACP Properties appendix ("ACP Properties" on page 67). For now, you should understand how to configure connection information for the Agile PLM instances you wish to connect to.

New projects are created with a `project.properties` file. This property file contains properties specific to the project. The sample `project.properties` file has a preconfigured list of Agile PLM instances (Golden Config, Development, QA, Stage, Training, and Production) listed in it. Each instance is assigned a nickname. For instance, the production instance might be "prod".

Here is an example of a connection that is configured in the `project.properties` file. For example purposes, we will use Acme as the name of the customer. Configure all of your connections in a similar manner.

```
prod.name              = Acme Production
prod.url               = http://www.acme.com:7777/Agile
prod.username          = propagation
prod.password          = 4DB08E9B1EBFAE
prod.password.mode     = encrypted
prod.xml               = export_prod.agl
```

In this example, "prod" is the nickname for the connection. You will specify the nickname as a

parameter to the ACP commands. Each of the properties for this particular connection must be prefixed by "prod.".

Property Name	Property Description
name	A friendly name for the connection. The application does not actually use this property. You can use this property however you would like.
url	The URL to use for accessing the Agile PLM instance. Essentially, this is the URL used to access the Agile PLM Web Client up through "/Agile".
username	The name of the administrator user to connect to the Agile PLM instance with.
password	The password for the administrator user used to connect to the Agile PLM instance.
password.mode	Indicates the form of the password. Choices are (encrypted, cleartext, or prompt).
xml	The name of the ACP archive created when exporting data from this Agile PLM instance.

Refer to ACP Properties (on page 67) for more details.

ACP Control File

ACP uses the control file `config.xml` to tell it what to propagate. The sample control file is configured to select all objects for all configuration types except for users. You can leave the control file configured as is or you can change the configured settings.

The control file is divided into five sections:

1. <copy>
2. <rename>
3. <delete>
4. <ignore_references>
5. <subobject_maps>.

The <copy> section tells ACP which objects should be created or updated.

The <rename> section tells ACP which objects should be renamed in the target.

The < delete > section tells ACP which objects should be deleted from the target.

The <ignore_references> section tells ACP which object references can be ignored if they cannot be resolved in the target instance.

The <subobject_maps> section is used to help ACP map names between the source and target for subobjects such as list entries.

See Configuring the ACP Control File (on page 41) for detailed information on how to configure the control file.

Also contained in the control file are Attributes that influence the business logic that ACP uses with regard to certain configuration types. These are also covered in the next chapter, in Business Logic Attributes in the Control File (on page 45).

ACP Scripts

ACP is run through a set of scripts. The scripts are initiated from a command line in either a DOS command window or a Unix shell. The ACP scripts are installed to the bin directory. When running the ACP scripts, the folder for the project you are currently working with needs to be your current working directory. Depending on the script you are running, you will need to pass in the nickname for the connection(s) the ACP script will be interacting with.

For convenience, a script launcher was installed to the project directory. You can use this script to launch ACP script you want to run.

1. Change directory to the project folder for the project you are working with.
2. Set the ACP_JAVA_HOME environment variable to where the JRE 1.5 has been installed.
3. Launch the desired script. Examples:
   - acp export dev
   - acp import dev prod

Refer to ACP Scripts (on page 73) for a complete list of ACP commands and their parameters.

ACP Exit Codes

ACP uses a command line interface to execute. This allows ACP to be run from background scripts that you develop. Each ACP script runs a specific ACP program. An ACP program will return a code when it exits. This "exit code" indicates whether the program completed successfully, successfully but with warnings, or with errors. Your background scripts can interrogate the exit code and take the appropriate action based on the code returned.

Refer to ACP Exit Codes (on page 77) for a list of the program return codes and corresponding text.

ACP Log Files

In addition to the data that is propagated to a target instance or exported to an ACP archive, ACP creates log files which describe what ACP did.

The process log tells you what you asked ACP to do, what was processed, and the result for each object processed.

The error log provides detailed information about warnings and errors encountered while processing.

The verbose log provides detailed information about information changed in the target instance.

Refer to ACP Program Logs (on page 79) for closer inspections (“anatomy”) and a sample of several of the log files.

Summary

ACP is driven from a command-line interface. This allows ACP to be integrated with the

configuration management process you are currently using. It does not lend itself to ad-hoc configuration changes.

Managing your configuration through projects will allow you to reuse and perform multiple configuration changes simultaneously.

You interact with ACP through a set of scripts which are installed with ACP.

Use properties to tell ACP how to interact with your environment.

Use the control file to tell ACP what objects you want propagated.

Review the work ACP has performed through the log files produced by ACP.

Interrogate the return codes provided by ACP to complete the integration with your configuration management process.

Chapter 8

Configuring the ACP Control File

This chapter includes the following:

This chapter is a conceptual overview of the contents of the control file and how its contents are interpreted by ACP. A brief section on XML formatting is included as a baseline of XML syntax when you modify the control file. Each of the main sections of the control file are discussed in this chapter.

ACP Control File

This chapter is a conceptual overview of the contents of the control file and how it is modified so ACP does what you want it to. This final "chapter" of the manual is supported by the Appendixes, notably App. A about the ACP Configuration Types and App. C, ACP Properties.

The ACP control file is how you tell ACP what to propagate. ACP knows how to propagate objects and which order objects need to be propagated in, but it does not know what to propagate. You must configure the control file to communicate to ACP what you would like it to do.

By default, the control file is named `config.xml` and is located in the project directory. This sample control file doubles as "online documentation" in that it contains the complete syntax of the tag names, attribute names, and attribute values used by ACP in propagation operations.

The name and location of the control file can be changed by setting the `control.filename` property in the Project Properties file (`project.properties`).

The control file (`config.xml`) is an XML file whose syntax must adhere to the control file schema, `acp_control_file.xsd`, located in the *<ACP Install Dir>*`/schema` directory. The control file itself is a valuable source of information: it presents the complete syntax of ACP, listing all the configuration types, business logic attributes, regular expressions, and examples of instructions to ACP, the actions that it will take.

A graphical representation of the schema can also be found in the install directory. The graphical representation is called *<ACP Install Dir>*`/schema/docs/acp_control_file.html`. This file can be viewed through any Internet browser.

XML Format

ACP's control file uses basic XML constructs. Please familiarize yourself with these basic constructs before trying to make changes to the control file.

Element

Elements have content (data). This content can be empty, simple content (text), element content, or mixed content. All elements have a start tag and an end tag. The element's content is everything specified from (including) the element's start tag to (including) the element's end tag. The name of the element is defined by its start tag. The start tag and end tag must have matching names. (XML tag names are case-sensitive.) Elements may also have attributes. Attributes are defined as part of the start tag. Here are some examples of elements:

Empty Element

An empty element is an element with nothing between its start and end tags. Since there is nothing between the start and end tags, there are actually two methods for expressing the element.

- Longhand Example: `<middle_name></middle_name>`
- Shorthand Example: `<middle_name/>`

Simple Content

An element with simple content is an element with only text between the start and end tags. Carriage returns and tabs that appear between the start and end tag are considered to be part of the text.

- Example: `<last_name>Smith</last_name>`
- Example with carriage returns:

```
<last_name>
   Smith
</last_name>
```

Element Content

An element with element content is an element with only nested elements between the start and end tags.

- Example:

```
<name>
   <first_name>Jane</first_name>
   <last_name>Doe</last_name>
</name>
```

Mixed Content

An element with mixed content has both simple content and element content.

- Example:

```
<chapter>XML Syntax
   <para>Elements must have a closing tag</para>
   <para>Elements must be properly nested</para>
</ chapter >
```

Element with Attributes

An element with attributes has attributes defined in the start tag. An element can have any number of attributes. The format of an attribute is `<attribute name>="<attribute value>"`.

- Example: `<phone type="work">1-888-555-1212</phone>`

Root Element or Document Element

All XML documents must have a root element; also known as the document element. The <agile> element is the root element for the ACP control file.

Element Tags

Element start and end tags are case sensitive and must match each other exactly. All element tags in the ACP control file are lowercase.

Attributes

XML elements may have attributes which appear as name/value pairs (<name>="<value>"). The value must be surrounded by quotes.

Comments

XML allows for comments. You can add comments to the control file that can help during re-use.

The begin comment delimiter is "`<!--`".

The end comment delimiter is "`-->`".

Comments may not be nested. In fact, comments may not contain two consecutive dashes (--) within the text.

- Example: `<!-- this is a comment -->`

Special Characters

XML uses "<" and ">" to delimit element tags. It also uses quotes ("") to delimit attribute values. For this reason, these characters cannot be used within the data (text) portion of the XML. These characters must be specially encoded to appear as data. The ampersand (&) is used to encode the data. Therefore, the ampersand (&) is also considered a special character. All encodings begin with an ampersand (&) and must end with a semicolon (;).

Here are the proper encodings for the special XML characters:

Characters	Encoding
&	&amp;
<	&lt;

Characters	Encoding
>	>
“	"

Control File Sections

The control file is divided into named sections. The Copy, Rename, and Delete sections direct ACP what to propagate. The Subobject Maps and Ignore References sections provide additional information when ACP is processing the Copy section. These are each detailed below.

In addition to the "named sections" of the control file, certain business logic attributes are used to influence the business logic that ACP uses with regard to certain configuration types. Because this subject is very important, "Business Logic Attributes in the Control File" are introduced as part of the Copy (<copy>) Section discussion.

Copy (<copy>) Section

The Copy section is marked by the <copy> element. ACP only copies objects expressly mentioned in the copy section. Each object has a configuration type, for example, List is a configuration type; Country is an object of the configuration type List. In the copy section, you will tell ACP which configuration types to process. Within each configuration type, you will tell ACP which objects to process. The list of objects configured in the copy section represents a set of objects found in the source instance.

Configuration Types in Copy Section

A complete list of configuration types that support copy (create object, update object, or replace object) and the XML tags used to denote their elements can be found in ACP Configuration Types (on page 57). In the copy section of the control file, objects to copy can be marked for inclusion or exclusion through regular expressions.

Include Patterns

Include patterns allow you to limit which objects of a specific configuration type to process. If no include pattern is configured, then ACP uses the "select all" pattern (.*) as the include pattern.

Exclude Patterns

Exclude patterns offer a second way to limit which objects of a specific configuration type to process. If no exclude pattern is configured, then ACP uses no exclude patterns. In order for an object to be excluded by an exclude pattern, the object matching the exclude pattern must also match one of the include patterns. An object is automatically excluded if it does not match one or more of the include patterns.

Regular Expressions

Regular expressions allow you to use "wild cards" in your selection of objects to include or exclude. This prevents you from having to list each object individually. Keep in mind that an object's name can be a regular expression itself. Regular expressions give special meaning to certain characters. If the special character is used in an object's name, it will need special encoding in the include/exclude pattern to be able to match the object's name exactly. Therefore, you can list objects individually as using wildcards. Regular expressions use special characters for its matching rules. These special characters may appear in your object names. See Java Regular Expressions (on page 63) for more detailed information about regular expressions and their special characters.

Business Logic Attributes in the Control File

In addition to being able to select which objects by configuration type, you can also influence the business logic that ACP uses with regard to configuration types. The amount of influence on business logic varies by configuration type. For detailed information on attribute names and their allowable values, refer to ACP Configuration Types (on page 57).

Objects per File

`objects_per_file` attribute is available to all configuration types.

When exporting objects, ACP creates separate XML files for each configuration type exported in the ACP XML archive. In addition, ACP can limit the number of objects written to an XML file. Once the limit is reached, ACP starts writing to a new XML file for that configuration type. XML files for a configuration type are sequenced as 001, 002, 003, and so forth. By default, ACP writes up to 1000 objects to an XML file for each configuration type.

File Prefix

`file_prefix` attribute is available to all configuration types.

When exporting objects, ACP creates separate XML files for each configuration type exported in the ACP XML archive. Since ACP may have to create multiple XML files per configuration type, a file prefix must be used for each file. This is the prefix before the sequence number for the file. ACP assigns a reasonable prefix to easily identify the XML files that are exported for each configuration type.

Criteria Force Update

`criteria force_update` attribute is available to only the Criteria configuration type.

Typically, criteria should not be updated once in use as this may create an undesirable change to business behavior. Java Client presents a dialog when an in-use criteria is about to be changed and requires you to confirm this change. Similarly, ACP allows you to force in-use criteria to be updated. This setting is applied to all criteria processed.

Case Sensitive List Entries

`case_sensitive_list_entries` attribute is available to only the List configuration type.

The application treats list entry names as case sensitive. That is, the following list entries in the

Yes/No list would be distinct list entries: yes, Yes, YES. In order to rename a list entry, ACP requires a subobject map to be defined for a list to allow ACP to map list entries between the source and target. This attribute provides an alternative to the subobject map when the only change to the list entry name was its case. This setting is applied to all lists processed.

Force Delete List Entry

`force_delete_list_entry` attribute is available to only the List configuration type.

This paramater addresses the problem of deletion of list entries that belong to lists that are in use by the PLM system. Although you cannot delete a list entry that is part of the configuration (an attribute default value), you can disable the value. If the `force_delete_list_entry` attribute is set to No (the default), then the list entry is disabled instead of deleted. The value remains listed as the default value. The system issues warnings when “force” behavior is invoked. For instance, the user is prompted whether to wipe out references to the entry being deleted if not used as a default value. The prompt that the user sees when deleting a list entry is shown no matter whether the list is in use. If a user creates a list, and immediately adds some values, and then tries to delete one of the values, the confirmation prompt is displayed. This attribute is available to the List configuration type.

New User Password

`new_user_password` attribute is available to only the User configuration type.

ACP does not propagate user passwords; it would be a "security hole" if it did. Since passwords cannot be propagated, ACP must assign a password when creating new users. By default, ACP assigns the password "agile" to all users that it creates. This default password may not be appropriate because of the account policy settings for minimum password length. The new user password attribute allows you to choose the password that ACP assigns to new users.

For security purposes, the new user password must be encrypted. You can use the encryptpwd script to get the encrypted version of the password you would like to use. This setting is applied to all new users created.

Process Extension Association Rule

`px_association_rule` attribute is available to only the Base Class, Class, and Subclass configuration types.

ACP works from the premise that Agile PLM configuration information is maintained in one instance and then propagated to other Agile PLM instances after it has been tested. In some situations this may not be the case.

For example, a golden configuration Agile PLM instance will not be used for testing. In this example, process extensions would not be developed and tested in the golden configuration environment.

To facilitate this multiple instance maintenance of configuration data, the propagation rule for process extension associations can be overridden. By default, the associations in the source instance are propagated to the target instance. This setting is applied to all objects of the configuration type it is used with.

User Association Rule

`user_association_rule` attribute is available to only the Role and User Group configuration types.

ACP works from the premise that Agile PLM configuration information is maintained in one instance and then propagated to other Agile PLM instances after it has been tested. In some situations this may not be the case.

For example, a golden configuration Agile PLM instance will not be used for testing. In this example, a complete set of users would not be updated in the golden configuration environment.

To facilitate this multiple instance maintenance of configuration data, the propagation rule for user associations can be overridden. By default, the associations in the source instance are propagated to the target instance. This setting is applied to all objects of the configuration type it is used with.

Summary of Business Logic Attributes

This table collects the important aspects of the business logic attributes. The control file itself can give you more information about these attributes. In the table, the default value is in italic: ACP will use this value if another value is not specified.

Attribute	Configuration type tag	What it does	Values (default value is in Bold font)
Objects Per File	`config_type objects_per_ file`	The number of objects ACP writes to a single XML file creating a new XML file for the configuration type.	1000
File Prefix	`config_type file_ prefix`	The prefix to use when naming the XML files generated by export.	Default (if not specified): configuration type–specific
Criteria Force Update	`criteria force_update`	Allows you to decide whether in-use criteria can be updated.	No. ACP does not allow an in-use criteria to be updatede.
Case-Sensitive List Entries	`case_sensitiv e_ list_entries`	In lieu of creating list entry maps for list entries whose name only changed by case, this attribute is used to treat list entries whose name only differs by case as the same list entry.	Yes or no
Force Delete List Entry Note: This attribute is not available in ACP9216.1.	`force_delete_ list_entry`	Determines whether a list entry is to be deleted or disabled	no, false, 0 all mean No = list entry is disabled, not deleted. yes, true, 1 all mean Yes = list entry is deleted, if possible.
New User Password	`user new_ user_password`	Password to assign a new user. This password is used only for new users. Passwords cannot be propagated. This value must meet any account policies you have set in your target instance. The password must be encrypted.	agile

Attribute	Configuration type tag	What it does	Values (default value is in Bold font)

Process Extension Association Rule	`base_class px_ association_r ule;`
	`class px_ association_r ule;`
	`subclass px_ association_r ule;`
		Determines how ACP will handle the process extensions associated with each class.	**replace** = The class–PX associations in the source are copied to the class in the target.
			ignore = The class–PX associations in the target are left as is.
			merge = The class–PX associations in the source are merged with the existing class-PX associations in the target.

User Association Rule	`role user_ association_r ule;` `user_group user_ association_r ule`	Determines how ACP will handle the users associated with each role or with each user group.	**replace** = The user associations for the role or user group in the source are copied to the role in the target. ignore = The user associations for the role or user group in the target are left as is. merge = The user associations for the role or user group in the source are merged with the existing role–user associations in the target.

Putting it all together in Copy section

Let's take a look at a short example of the <copy> section with some explanation of how ACP interprets what was specified.

```
<copy>
     <auto_number/>
     <base_class>
     </base_class>
     <class>
       <include>
         <pattern>.*</pattern>
       </include>
     </class>
     <privilege>
       <include>
         <pattern>\(Restricted\).*</pattern>
         <pattern>Add Phases</pattern>
         <pattern>Modify Programs &amp;amp; Gates that I am
     owner of</pattern>
         <pattern>Admin\. Privilege</pattern>
       </include>
     </privilege>
```

```
        <role user_association_rule="replace">
          <include>
            <pattern>.*</pattern>
          </include>
          <exclude>
            <pattern>\(.*\).*</pattern>
          </exclude>
        </role>
        <user>
          <exclude>
            <pattern>test.*</pattern>
          </exclude>
        </user>
    </copy>
```

In the above example, here is what ACP will process by configuration type.

- Auto Number: This is a shorthand notation. The shorthand notation is an empty element. It tells ACP to process the complete list of auto numbers found in the source data source.
- Base Class: This is similar to the shorthand notation in that it is an empty element. It tells ACP to process the complete list of base classes found in the source data source.
- Class: In this example, we explicitly use the include pattern that matches all objects. It tells ACP to process the complete list of classes found in the source data source.
- Privilege: In this example, we have listed multiple patterns. The pattern "\(Restricted\).*" matches privileges that begin with "(Restricted)". The pattern "\(Propagation\) Administrator" matches a single privilege with the name "(Propagation) Administrator".

  Notice the need to escape the parentheses as parentheses are regular expression special characters.

  The other three patterns demonstrate that it is possible to match on a specific object name. That is, the pattern can match exactly one object name.

  Notice the encoding of the ampersand (&) in "Modify Programs & Gates that I am owner of". This is necessary because the ampersand (&) is an XML special character.

  Also, notice the encoding of the period (.) in "Admin\. Privilege". This is necessary because period (.) is a regular expression special character.
- Role: This example demonstrates the use of exclude patterns. In this example, we are asking ACP to process all roles except those that begin with "(…)". For instance, this would exclude the roles that begin with "(Propagation)" and "(Restricted)".

  This example also illustrates the use of a business logic attribute. The attribute name is "user_association_rule". The attribute value is "replace". "Replace" instructs ACP to replace the users associated with the role in the target with those users associated with the role in the source.
- User: This example also demonstrates the use of exclude patterns. In this example, we omitted the include patterns. This is perfectly fine since ACP assumes that all objects of a configuration type will be processed when no include patterns are specified.

Rename (<rename>) Section

The Rename section is marked by the <rename> element. ACP uses the source object name to map it to an object in the target instance. If the name of the object changes in the source, ACP no

longer has a way to look up the object in the target instance. The rename section resolves this dilemma. You can use the rename section to rename objects in the target instance. ACP processes rename section of the control file before it processes the copy section. This allows copy to resolve any references to objects whose name has changed. Renames only get processed when the target is an Agile instance.

Configuration Types in Rename Section

A complete list of configuration types that support rename and the XML tags used to denote their elements can be found in ACP Configuration Types (on page 57). In the Rename section, objects to be renamed are specified by maps. An error is reported if the object being renamed does not exist in the target instance.

Name Maps

A name map maps an object name in the source instance to an object name in the target instance. In order for the map to have meaning, an object should exist in the source instance with the map's source name and an object should exist in the target instance with the map's target name.

Source Name

The source name is the name of the object being mapped in the source. Since this is part of a name map, the exact name of the object must be specified.

Target Name

The target name is the name of the object being mapped in the target. Since this is part of a name map, the exact name of the object must be specified.

Putting it all together in Rename Section

Let's take a look at a short example of the <rename> section with some explanation of how ACP interprets what was specified.

```
<rename>
     <list>
       <map>
         <source_name>Countries</source_name>
         <target_name>Country</target_name>
       </map>
       <map>
         <source_name>My Category</source_name>
         <target_name>Category 9 List</target_name>
       </map>
     </list>
     <privilege>
       <map>
         <source_name>(Propagation) Configuration
     Administrator</source_name>
         <target_name>(Propagation)
     Administrator</target_name>
       </map>
       <map>
         <source_name>Modify My Programs &amp;amp;
     Gates</source_name>
```

```
            <target_name>Modify Programs &amp;amp; Gates that I am
        owner of</target_name>
          </map>
          <map>
            <source_name>Administrator Privilege</source_name>
            <target_name>Admin. Privilege</target_name>
          </map>
        </privilege>
        <subclass>
          <map>
            <source_name>My ECO</source_name>
            <target_name>ECO</target_name>
          </map>
        </subclass>
    </rename>
```

In the above example, here is what ACP will process by configuration type.

- List: ACP will attempt the following renames:
  - Rename the list "Country" to "Countries";
  - Rename the list "Category 9 List" to "My Category".
- Privilege: ACP will attempt the following renames:
  - Rename the privilege "(Propagation) Administrator " to "(Propagation) Configuration Administrator"; Note that the regular expression special characters for the Open Parenthesis and the Closed Parenthesis are not encoded here: this is because the map contains exact names and not regular expressions.
  - Rename the privilege "Modify Programs & Gates that I am owner of " to "Modify My Programs & Gates".
  - Rename the privilege "Admin. Privilege" to " Administrator Privilege". Note that the regular expression special character for the Period are not encoded here: this is because the map contains exact names and not regular expressions.
- Subclass: ACP will attempt the following rename:
  - Rename the subclass "ECO" to "My ECO".

Delete (<delete>) Section

The Delete section is marked by the <delete> element. The Delete section allows you to propagate deletes objects in the target. Deletes only get processed when the target is an Agile instance. Deletes get processed before renames and copies. This allows ACP to report errors where you have asked ACP to delete an object that is being referenced by an object that was also copied.

Configuration Types in Delete Section

A complete list of configuration types that support <delete> and the XML tags used to denote their elements can be found in ACP Configuration Types (on page 57). In the delete section, objects to be deleted are specified as a list of names.

Name

The name is the name of the object to be deleted. Deletes must be done by the exact name of the object. This helps to prevent you from deleting objects unintentionally.

Putting it all together in Delete Section

Let's take a look at a short example of the Delete section with some explanation of how ACP interprets what was specified.

```
<delete>
     <list>
       <name>Category 8 List</name>
       <name>Category 9 List</name>
       <name>Category 10 List</name>
     </list>
     <privilege>
       <name>(Propagation) Administrator</name>
       <name>Modify Programs &amp;amp; Gates that I am owner
     of</name>
       <name>Admin. Privilege</name>
     </privilege>
     <subclass>
       <name>ECO</name>
     </subclass>
</delete>
```

In the above example, here is what ACP will process by configuration type.

- List: ACP will attempt to delete lists with the following names:
  - "Category 8 List";
  - "Category 9 List";
  - "Category 10 List".
- Privilege: ACP will attempt to delete privileges with the following names:
  - "(Propagation) Administrator";
  - "Modify Programs & Gates that I am owner of ";
  - "Admin. Privilege".
- Subclass: ACP will attempt to delete the subclass named "ECO".

Ignore References (<ignore_references>) Section

The Ignore References section does not actually propagate any information. Since data may be propagated from a development instance to a production data, there may be test data that will not get propagated. However, there may be configuration data that does get propagated which refers to test data. Since the test data does not get propagated, any references to test data will generate errors because the test data it references will not be present in the production instance. This is undesirable as it is writing "OK" errors to the error log which amounts to a lot of noise.

A common example where this is useful would be roles. You may have created test users in your development instance. You assigned several roles to the test users. You will want to propagate the roles, but not the test users. The Ignore References section allows you to specify "ignore references" to test users. Then, when ACP cannot resolve a user, the error message that ACP would normally issue is suppressed.

Configuration Types in Ignore References Section

A complete list of configuration types that support ignore references and the XML tags used to denote their elements can be found in ACP Configuration Types (on page 57). In the ignore references section, object references to ignore can be accomplished through regular expressions. This promotes the use of naming conventions. For instance, test users could have user logins that begin with "test".

Patterns

Patterns allow you to specify one or more objects to be ignored. Given that this is a pattern, you can use regular expressions to select multiple objects with a single pattern.

Regular Expressions

Regular expressions allow you to use wildcards in your selection of objects to include or exclude. This prevents you from having to list each object individually. Keep in mind that an object's name can be a regular expression itself. Therefore, you can list objects individually as using wildcards. Regular expressions use special characters for its matching rules. These special characters may appear in your object names. Please see Java Regular Expressions (on page 63) for detailed information about regular expressions and their special characters.

Putting it all together in Ignore References Section

Let's take a look at a short example of the <ignore_references> section with some explanation of how ACP interprets what was specified.

```
<ignore_references>
     <user>
       <pattern>test.*</pattern>
       <pattern>dev.*</pattern>
       <pattern>demo1</pattern>
     </user>
     <role>
       <name>test.*</pattern>
     </role>
</ignore_references>
```

In the above example, ACP will suppress error messages for objects that cannot be resolved that also match the patterns listed.

- User: Ignore user references to users whose user ID starts with "test" or "dev" as well as the specific user ID "demo1".
- Role: Ignore user references to roles whose name starts with "test".

Subobject Maps (<subobject_maps>) Section

The Subobject Maps section is marked by the <subobject_maps> element. A subobject is contained by an object. For example, lists contain list entries. Therefore, a list entry is a subobject of a list. Just as ACP uses an object name to map an object between the source and the target, ACP uses a subobject's name to map subobjects between the source and target. If the name of the subobject changes in the source, ACP no longer has a way to look up the subobject in the target instance.

The Subobject Maps section resolves this dilemma. You can use the <subobject_maps> section to

map subobject names between the source instance and target instance. ACP uses the contents of the Subobject Maps section when it is processing the copy section. Subobject maps only have meaning when the target is an Agile instance.

Configuration Types in Subobject Maps Section

A complete list of configuration types that support subobject maps and the XML tags used to denote their elements can be found in ACP Configuration Types (on page 57). In the Subobject Maps section, subobjects whose name has changed are documented by maps. Some configuration types may have more than one subobject type. For example, classes have LifeCycle Phases and UI tabs.

Object Reference

As the name subobject implies, a subobject is subordinate to an object. Therefore, in order for ACP to match by the subobject name, it must also have the name of the object that the subobject belongs to. What if the object name and the subobject name are changing at the same time? Since ACP processes object renames before object copies, subobject name maps must always use the new name of the object.

Subobject Type

Some objects may have multiple types of subobjects. In order for ACP to interpret the subobject name map correctly, it must know the type of subobject being mapped.

Name Maps

A name map maps a subobject name in the source instance to a subobject name in the target instance.

Source Name

The source name is the name of the subobject being mapped in the source. Since this is part of a name map, the exact name of the subobject must be specified.

Target Name

The target name is the name of the subobject being mapped in the target. Since this is part of a name map, the exact name of the subobject must be specified.

Putting it all together in Subobject Maps Section

Let's take a look at a short example of the <subobject_maps> section with some explanation of how ACP interprets what was specified.

```
<subobject_maps>
  <lists>
    <list name="Reason Code">
      <list_entries>
        <map>
          <source_name>Enhancement</source_name>
          <target_name>Product Enhancement</target_name>
        </map>
        <map>
          <source_name>Improvement</source_name>
```

```
          <target_name>Reliability
Improvement</target_name>
        </map>
      </list_entries>
    </list>
    <list name="Location||Europe">
      <list_entries>
        <map>
          <source_name>Czech Republic</source_name>
          <target_name>Czechoslovakia</target_name>
        </map>
      </list_entries>
    </list>
  </lists>
  <subclasses>
    <subclass name="Customer">
      <life_cycle_phases>
        <map>
          <source_name>Current</source_name>
          <target_name>Active</target_name>
        </map>
      </life_cycle_phases>
      <user_interface_tabs>
        <map>
          <source_name>Customer Details</source_name>
          <target_name>Page Three</target_name>
          </map>
      </user_interface_tabs>
    </subclass>
    <subclass name="ECR">
      <user_interface_tabs>
        <map>
          <source_name>ECR Details</source_name>
          <target_name>Page Three</target_name>
        </map>
      </user_interface_tabs>
    </subclass>
  </subclasses>
<subobject_maps>
```

In the above example, ACP will be able to map the following subobjects.

- List (Reason Code): The list entry "Enhancement" and "Product Enhancement" are mapped as the same. The result will be the list entry will be renamed to "Enhancement". The list entry "Improvement" and "Reliability Improvement" are mapped as the same. The result will be the list entry will be renamed to "Improvement".
- List (Location||Europe): This is an example of renaming a list entry in a cascade list. Note the way the cascade sub-list is specified. A list entry belongs to a specific list. With cascade lists, the specific list is a sub-list. The sub-list names are separated with a double pipe (||) (period could not be used since a period is valid within a list name). The list entry "Czech Republic" and "Czechoslovakia" are mapped as the same. The result will be the list entry will be renamed to "Czech Republic".
- Subclass (Customer): This example illustrates specifying multiple subobject types for a single object. In this example, the life cycle phase "Current" is mapped to "Active". The result will be the life cycle phase will be renamed to "Current". In addition, the user interface tab "Customer Details" is mapped to "Page Three". The result will be the user interface page will be renamed

to "Customer Details".

- Subclass (ECR): The user interface tab "ECR Details" is mapped to "Page Three". The result will be the user interface page will be renamed to "ECR Details".

Appendix A

ACP Configuration Types

This Appendix includes the following:

ACP propagates the data contents of configuration types, which are roughly the same as the Administrator nodes. Not all propagation operations are supported by all Administrator nodes.

The following table shows the complete list of ACP-supported configuration types and the propagation actions that are supported by each type. See “Administrator Nodes that are not Propagated” on page 4-5.

Supported ACP Configuration Types in Rel. 9.2.2.1

Actions:	Delete	Rename	Copy		
			Create	Update	Replace
Data Settings > Classes (separated into the Object Types of base classes, classes, and subclasses)					
<base_class>		X		X	
<class>		X		X	
<subclass>	X	X	X	X	
Data Settings (except Classes)					
<character_set> (Exception: out-of-box character sets are not propagated.)	X	X	X	X	
<list>	X	X	X	X	X
<process_extension>	X	X	X	X	
<auto_number>	X	X	X	X	X
<criteria>	X	X	X	X	
Workflow Settings					
<workflow>	X	X	X	X	
User Settings					
<account_policy>				X	
<user>	X	X	X	X	
<user_group>	X	X	X	X	

Actions:	Delete	Rename	Copy		
			Create	Update	Replace
<supplier_group>	X	X	X	X	
<role>	X	X	X	X	
<privilege>	X	X	X	X	X
System Settings					
<smart_rule>				X	
<viewers_and_files>				X	
<notification_template>				X	
<full_text_search>				X	
<my_assignments>				X	
<unit_of_measure_family> (UOM)	X	X	X	X	
<company_profile>				X	
<ppm_dashboard_management>	X	X	X	X	
Product Cost Management					
<pcm_ship_to_location>	X	X	X	X	
<pcm_rfq_terms_and_conditions>			X	X	
Product Portfolio Management					
<ppm_cost_status>	X	X		X	
<ppm_quality_status>	X	X		X	
<ppm_resource_status>	X	X		X	
<ppm_schedule_status>	X	X	X	X	
<ppm_default_role>		X		X	
Agile Content Service					
<acs_subscriber>	X	X	X	X	
<acs_destination>	X	X	X	X	X
<acs_event>	X	X	X	X	X
<acs_filter>	X	X	X	X	X
<acs_package_service>	X	X	X	X	
<acs_response_service>	X	X	X	X	
Product Governance & Compliance					
<pgc_signoff_message>				X	
<pgc_compliance_rollup_scheduling>				X	

Actions:	Delete	Rename	Copy		
			Create	Update	Replace
<pgc_compliance_rollup_rule_setting>				X	
<pgc_supplier_declaration> – this config. type contains all data in the **Supplier Declaration Process Extensions** node folder				X	
Server Settings					
<server_location> (**Locations** node)				X	
<server_file_manager> (**Locations** > **File Manager** tab)	X	X	X	X	
<server_portal> (**Locations** > **Portal** tab)	X	X	X	X	
<server_database> (**Database** node)				X	
<server_preference> (**Preferences** node)				X	
<server_task> (**Task Configuration** node)				X	

Configuration Types and Match Fields in Rel. 9.2.2.1

This table lists the configuration types and whether the match field is the external, changeable name of the object or the internal, unchangeable, identification number.

Configuration Type	Match Field
Classes	
<base_class>	ID
<class>	ID
<subclass>	Name
Data Settings	
<auto_number>	Name
<character_set>	Name
<criteria>	Name
<list>	Name
<process_extension>	Name

Configuration Type	Match Field
Workflow Settings	
<workflow>	Name
User Settings	
<account_policy>	ID
<privilege>	Name
<role>	Name
<supplier_group>	Name
<user>	User ID
<user_group>	Name
System Settings	
<company_profile>	ID
<ppm_dashboard_management>	ID (out of the box) / Name
<full_text_search>	ID
<my_assignments>	ID
<notification_template>	ID
<smart_rules>	ID
<unit_of_measure_family>	Name
<viewers_and_files>	Name
Product Cost Management	
<pcm_ship_to_location>	Name
<pcm_rfq_terms_and_conditions>	ID
Product Portfolio Management	
<ppm_cost_status>	Name
<ppm_quality_status>	Name
<ppm_resource_status>	Name
<ppm_schedule_status>	Name
<ppm_default_role>	Name
Agile Content Service	
<acs_destination>	Name
<acs_event>	Name
<acs_filters	Name

Configuration Type	Match Field
<acs_package_service>	Name
<acs_response_service>	Name
<acs_subscriber>	Name
Product Governance & Compliance	
<pgc_signoff_message>	ID
<pgc_compliance_rollup_scheduling>	Name
<pgc_compliance_rollup_rule_setting>	Name
<pgc_supplier_declaration>	Name
Server Settings	
<server_location>	ID
<server_file_manager>	Name
<server_portal>	Name
<server_database>	ID
<server_preference>	ID
<server_task>	ID

Renaming Subobjects

Additionally, some Admin objects have subobjects that can be renamed. ACP requires a name map for these subobjects to be able to update the subobject correctly. Without the map, ACP will consider then subobjects to be different. The subobject with the old name will be deleted and the subobject with the new name will be created.

Configuration Type	
Subobject Type	Method for Rename
Base Class	
LifeCycle Phases	Subobject Map
Class	
User Interface Tabs	Subobject Map
Attributes	Base ID

Configuration Type	
Subobject Type	Method for Rename
LifeCycle Phases	Subobject Map
Subclass	
User Interface Tabs	Subobject Map
Attributes	Base ID
LifeCycle Phases	Subobject Map
List	
List Entries	Subobject Map
Workflow	
Workflow Statuses	Subobject Map
Unit of Measure Family	
UOMs	Subobject Map

Appendix B

Java Regular Expressions

This Appendix includes the following:

You can use Java regular expressions to select ACP configuration types or other Administrator objects to propagate (object names are also valid expressions). Both regular expressions and XML have special characters. The ACP control file uses an XML format, therefore, normal XML characters such as '<' and '>' must be treated differently in a pattern so that XML is not confused. If you want to match on these special characters in a regular expression, you must encode the character so that XML or Java will not interpret the character as a special character.

Special Characters

These two tables list special characters you must be aware of. The table "Regular Expression Examples" (below) gives some examples of how to match on the special characters most likely to appear in object names.

XML Special Characters

Characters	Encoding
&	&amp;
<	&lt;
>	&gt;
"	&quot;

Java Regular Expression Special Characters

For a comprehensive reference to Java regular expressions, please refer to Sun's Website (http://java.sun.com/j2se/1.4.2/docs/api/java/util/regex/Pattern.html) for information in the section, java.util.regex Class Pattern.

Characters	Encoding
\	\\
.	\.
?	\?
*	*

Characters	Encoding
+	\+
^	\^
$	\$
[	\[
]	\]
(	\(
)	\)
{	\{
}	\}
!	\!
:	\:

Regular Expression Examples

Regular Expression	Meaning	Possible Matches
.*	All names	▫ Customers ▫ ABC ▫ Test_Object ▫ Nuts & Bolts ▫ Manufacutring Part ▫ (Restricted) Object
Test.*	Names beginning with “Test”	▫ Test ▫ Test_Object ▫ Test 123
.*Test	Names ending with “Test”	▫ Test ▫ My Object – Test ▫ Object 123 – Test
.*&.*	Names containing an ampersand	▫ Nuts & Bolts ▫ Nuts&Bolts
\(Acme\).*	Names that begin with “(Acme)”	▫ (Acme) Object ▫ (Acme) Nuts & Bolts
\(Test\) Nuts & Bolts	Name that matches “(Test) Nuts & Bolts”	▫ (Test) Nuts & Bolts
.*\[Dev\]	Names that end with “[Dev]”	▫ Object [Dev] ▫ Nuts & Bolts [Dev]
Nuts & Bolts \[Dev\]	Name that matches “Nuts & Bolts [Dev]”	▫ Nuts & Bolts [Dev]

Appendix C

ACP Properties

This Appendix includes the following:

ACP provides a set of properties for you to communicate to ACP how you would like ACP to interact with your environment. These properties include information for connecting to Agile PLM instances, log file names, and behavior manipulation. ACP offers great flexibility with setting properties. The primary use of this flexibility is to allow you to share properties across projects. This is especially useful for connections since chances are all of your projects will use the same connections.

Property Sources

Since ACP allows you to set properties in multiple places, it is important that you understand the precedence order ACP uses for deciding which value to use if the property is set in multiple places. If the same property is defined multiple times in the same source, the first occurrence is used.

Order	Property Source	Description
1.	Program Defined	These are properties set by the program that cannot be overridden.
2.	Command Line Property	These are -D options specified on the command line and passed to the program via the ACP script used.
3.	Command Line Property File	These are names of property files specified on the command line and passed to the program via the ACP script used.
4.	Project Property File	This is the property file specific to your project. It is the project.properties file located in your project folder.
5.	Common Property File	This is a property file which can be shared across all ACP projects for a single version. The name of this property file must be set in your project.properties file. This file can be located anywhere.
6.	Global Property File	This is a property file which can be shared across all ACP projects for all versions. The name of this property file must be set in your project.properties file. This file can be located anywhere.
7.	Program Default	These are properties set by the program that can be overridden.

Defining Properties

ACP uses Java-style properties. As mentioned before, ACP provides great flexibility with its properties. Properties can be defined in terms of other properties. You can create your own properties. And finally, properties can be referenced through indirection.

Java-style Property

Java-style properties syntax:

```
<property name> = <property value>
```

- Example:

```
prod.name = Acme Production
```

Property References

Properties can be defined in terms of other properties. This is very powerful as it allows for ease of automation. ACP uses the Ant-style property reference.

Reference Syntax:

```
${<property name>}
```

- Example:

```
acp.method = export
acp.run.datetime = 20071001_142442
log.process.name = ${acp.method}_${acp.run.datetime}.log

result: log.process.name = export_10071001_142442.log
```

Indirect References

To extend the power of defining properties in terms of others, ACP supports the use of indirection. This power is usually not warranted. It exists to support the specification of connections. ACP refers to connections as src or tgt. If ACP required you to use these names, you would need to remember to modify the project.properties file each you wanted to use a different connection. To simplify this, ACP allows you to define all of your connections once and give each connection a nickname.

Indirect Reference Syntax:

```
${${<property name>}}
```

- Example:

```
src.ref = prod
prod.name = Acme Production
log.process.name = ${${src.ref}.name}.log

result: log.process.name = Acme Production.log
```

Properties

Here are the properties known to ACP. You can also create your own properties. The properties you create would only be useful if used in the definition of other properties. You can find detailed information regarding the properties known to ACP in the file <ACP Install Directory>/templates/properties.txt.

Agile-owned Properties

The properties listed here are set by the program and cannot be overridden. These properties are made available to you so that you may use them in the definition of other properties.

Property Name	Description
acp.program.name	The name of the ACP program being run.
acp.program.version	The version of the ACP program being run.
acp.program.date	The date the ACP program was started (YYYYMMDD).
acp.program.time	The time the ACP program was started (HH24MISS).
acp.program.datetime	The date and time the ACP program was started (YYYYMMDD_HH24MISS).
acp.method	The method (export, import, propagate, compare) that ACP will use. The method determines the type of the source and target connections. Possible Values: export (source is Agile; target is XML) import (source is XML; target is Agile) compare_agile2xml (source is Agile; target is XML) compare_xml2agile (source is XML; target is Agile) compare_xml2xml (source is XML; target is XML)
src.ref	Provides a level of indirection for the propagation source. This allows the actual source to be determined on the command line. Refers to a nickname for a connection.
tgt.ref	Provides a level of indirection for the propagation target. This allows the actual target to be determined on the command line. Refers to a nickname for a connection.

Agile-defaulted Properties

The properties listed here are set by the program; however, their values can be overridden.

Property Name	Description
acp.debug	Indicates if stack traces should be included with error messages. Stack traces are useful for Agile Customer Care to solve issues. Possible Values: false – stack traces are included (default) true – stack traces are not included
acp.verbose	Indicates if detailed propagation information should be written to the verbose log. Possible Values: false – no verbose information is logged (default) true – verbose information is logged
objects.suppress.skipped	Indicates if the names of objects which do not match the selection regular expressions should be logged. Possible Values: false – skipped objects are logged (default) true – skipped objects are not logged
control.filename	The name of the control file. (format: [<file path>/]<file name>.xml. Default: config.xml
log.process.filename	The name of the process log file (format: [<file path>/]<file name>). If no path is specified, the log file is created in the current project directory. Default: ${acp.method}.log
log.process.open.mode	Indicates how ACP should open the process log file when it already exists. Possible Values: append – appends to the file if already exists overwrite – overwrites the file if already exists
log.error.filename	The name of the error log file (format: [<file path>/]<file name>). If no path is specified, the log file is created in the current project directory. Default: ${acp.method}.err
log.error.open.mode	Indicates how ACP should open the error log file when it already exists. Possible Values: append – appends to the file if already exists overwrite – overwrites the file if already exists
log.verbose.filename	The name of the verbose log file (format: [<file path>/]<file name>). If no path is specified, the log file is created in the current project directory. Default: ${acp.method}Verbose.log
log.verbose.open.mode	Indicates how ACP should open the verbose log file when it already exists. Possible Values: append – appends to the file if already exists overwrite – overwrites the file if already exists

Customer-owned Properties

The properties listed here are set by the program, however, their values can be overridden.

Note Connection nicknames are a concept, not a property. The nickname is used to qualify a connection property (use <nick`name>.name; for` example, `prod.name` for a connection nicknamed “prod”).

Property Name	Description
global.properties.filename	The path and name to an optional global properties file. This properties file is intended to allow you to share properties across all projects for all versions of ACP. An example of this file can be found at <ACP Install Dir>/templates/global.properties. Note: Use forward slashes (/) as the path separator.
common.properties.filename	The path and name to an optional common properties file. This properties file is intended to allow you to share properties across all projects for a single version of ACP. This is especially useful for maintaining connection information. An example of this file can be found at <ACP Install Dir>/templates/common.properties. Note: Use forward slashes (/) as the path separator.
<nickname>.name	This is used as a descriptive name for the connection. It is provided for you to use in the definition of other properties.
<nickname>.url	The Agile PLM URL for the connection. Format: http://<hostname>[:<port number>]/Agile. Essentially, this is the URL used to connect to the Agile PLM web client through "/Agile".
<nickname>.username	The login name for a valid Agile user which has Administrator privileges to connect to the Agile PLM instance referenced by this connection. Default: propagation.
<nickname>.password	The password for the Agile user used to connect to the Agile PLM instance referenced by this connection. Passwords can be specified as cleartext or encrypted text. ACP uses <nickname>.password.mode to determine how to interpret this password.
<nickname>.password.mode	This directs ACP on how to interpret the password specified by the <nickname>.password property. Possible Values: cleartext- ACP uses the password text as supplied.encrypted- ACP decrypts the password text to log in. prompt- ACP ignores the password text and prompts the user for the password.
<nickname>.xml	The name to use for the XML archive when exporting from or importing from this Agile PLM instance (format: [<absolute path>/]<Agile XML archive name>.agl).

Here are some complete examples for defining a connection, using “Acme” as the customer name.

```
dev.name=Acme Development
dev.url=http://agileplm-dev.acme.com:7777/Agile
dev.username=propagation
dev.password=4DB08E9B1EBFAE
dev.password.mode=encrypted
dev.xml=export_dev.agl

prod.name=Acme Production
prod.url=http://agileplm-prod.acme.com:7777/Agile
prod.username=propagation
prod.password=4DB08E9B1EBFAE
prod.password.mode= encrypted
prod.xml=export_prod.agl
```

Appendix D

ACP Scripts

This Appendix includes the following:

ACP is command-line driven. The ACP programs are initiated by scripts provided with ACP. All scripts are installed to the bin directory under the ACP Install Directory.

Working Directory

Run the ACP scripts from the project directory you are working with. Running ACP from the project directory provides ACP with context needed to run. Specifically, ACP looks for the project.properties file in the current directory.

```
cd <ACP Work Directory>/<project directory>
```

Java Home

ACP requires Java 1.5. You are required to install Java 1.5. You can use the ACP_JAVA_HOME environment variable to tell ACP where the JRE is installed. This allows you to use the JAVA_HOME environment variable for other uses if you wish.

- Windows: `set ACP_JAVA_HOME=<Java JRE 1.5 Directory>`
- Unix: `export ACP_JAVA_HOME=<Java JRE 1.5 Direcory>`

You may wish to add this environment variable to your login profile (Unix) or to the system environment variables (Windows).

Running Scripts

All ACP scripts are installed to the bin directory under the ACP Install Directory. You can run the scripts in a couple of ways.

ACP Launcher

For convenience, a special script is installed to the project directory. This script is named acp[.bat]. It allows you to launch an ACP script without having to specify the entire path to the bin directory

where the ACP script is located.

- Usage: `acp <script name> [<script parameters>]`

Version Script

ACP provides a script for determining the version of ACP being run.

version

The version script reports the version of ACP being run and also indicates the required version of the Agile PLM instance it can connect to.

- Usage: `acp version`
- Sample Output:

```
Agile(TM) ACP CopyConfig (Ver. ACP9223.1, Build #11)
ACP Version:                ACP9223.1, Build #11
Required Agile PLM Version: 9.2.2.3.11
```

Note	The notation for versions of ACP combines a PLM Release number (say, PLM 9.2.2.3) without separating periods (so, "ACP9223") and, following the period, the number of that version of ACP92xy.

Project Management Scripts

ACP provides a script for managing the creation of ACP projects.

create_project

The create_project script creates a clean project. This project will look identical to the sample project created by the installer. The project folder will contain three files: ACP Launcher, Project Properties, and Control File.

1. Choose a name for the new project folder. The folder name cannot already exist.
2. Open a terminal window (Windows: DOS window; Unix: shell)
3. Navigate to the work directory specified when ACP was installed. Here are examples based on the default work directories provided by the ACP Installer.
   - Windows: `cd C:\Agile\ACPWork9223`
   - Unix: `cd /<user home>/agile/acpwork9223`
4. Run the ACP create_project script. The new project name can be specified on the command line or, if it is not specified, the script prompts you for it.
   - Usage: `acp create_project [<project_name>]`

Password Encryption Script

ACP provides scripts for getting the encrypted string for a password. Typically, you will not want

clear text passwords stored in text files on your computer. ACP allows you to configure passwords as encrypted passwords.

encryptpwd

The encryptpwd script returns an encrypted string for a clear text password entered. You cannot pass the password you wish to get the encrypted form for on the command-line. The script will prompt you for the clear text password.

- Usage: `acp encryptpwd`

Object Name Comparison Companion Scripts

ACP provides several scripts for comparing lists of object names. These scripts are designed to help you make sure your control file has the proper <rename> maps for object names. These scripts compare the list of object names in the source and target as configured in the control file.

Note	These scripts do not perform a detailed comparison of selected objects.

compare_agile2xml

The compare_agile2xml script compares lists of object names between a source Agile PLM instance and a target ACP XML archive. This script uses the connection nicknames defined in the Project Properties file. It determines the name of the ACP XML archive by the <nickname>.xml property. Each connection can have its own XML archive.

- Usage: `acp compare_agile2xml <source nickname> <target nickname> [-debug]`
- Example: `acp compare_agile2xml dev prod`

compare_xml2agile

The compare_xml2agile script compares lists of object names between a source ACP XML archive and a target Agile PLM instance. This script uses the connection nicknames defined in the Project Properties file. It determines the name of the ACP XML archive by the <nickname>.xml property. Each connection can have its own XML archive.

- Usage: `acp compare_xml2agile <source nickname> <target nickname> [-debug]`
- Example: `acp compare_xml2agile dev prod`

compare_xml2xml

The compare_xml2xml script compares lists of object names between a source ACP XML archive and a target ACP XML archive. This script uses the connection nicknames defined in the Project

Properties file. It determines the name of the ACP XML archive by the <nickname>.xml property. Each connection can have its own XML archive.

- Usage: `acp compare_xml2xml <source nickname> <target nickname> [-debug]`
- Example: `acp compare_xml2xml dev prod`

Propagation Scripts

ACP provides scripts for propagating data from one Agile PLM instance to another Agile PLM instance via a target ACP XML archive.

export

The export script exports configuration data from a source Agile PLM instance to a target ACP XML archive. This script uses the connection nicknames defined in the Project Properties file. It determines the name of the ACP XML archive by the <nickname>.xml property. Each connection can have its own XML archive.

The export script only needs a single parameter because it has enough information to know how to connect to an Agile PLM instance as well as what the ACP XML archive is called. The -debug option provides stack traces in the error messages. These stack traces are only useful to Agile Customer Care.

- Usage: `acp export <source nickname> [-debug]`
- Example: `acp export dev`

import

The import script imports configuration data from a source ACP XML archive to a target Agile PLM instance. This script uses the connection nicknames defined in the Project Properties file. It determines the name of the ACP XML archive by the <nickname>.xml property. Each connection can have its own XML archive. The -debug option provides stack traces in the error messages. These stack traces are only useful to Agile Customer Care.

- Usage: `acp import <source nickname> <target nickname> [-debug]`
- Example: `acp import dev prod`

Appendix E

ACP Exit Codes

Program Code	Program Text
0	Completed with no errors or notes.
-101	Note: Copy completed with notes. See log for details.
-201	Note: Rename completed with notes. See log for details.
-301	Note: Delete completed with notes. See log for details.
-401	Note: Compare completed with notes. See log for details.
1	General error.
11	Unable to open the program's resource bundles.
21	Unable to determine user's locale.
31	Command-line is in error.
41	Unable to open the program's log files.
51	Control file is in error.
61	Unable to establish a connection with an Agile instance or an Agile XML Archive.
101	Encountered one or more errors while copying data.
201	Encountered one or more errors while renaming data.
301	Encountered one or more errors while deleting data.
401	Encountered one or more errors while comparing data.

Appendix F

ACP Program Logs

This Appendix includes the following:

The ACP log files are intended to tell the propagation story.

Verbose Log

The purpose of the verbose log is to provide information about specific changes made by ACP. The verbose log is produced only when the target data source is an Agile PLM instance. It logs the objects that are processed and reports the fields whose value has changed.

Console (stdout) Log

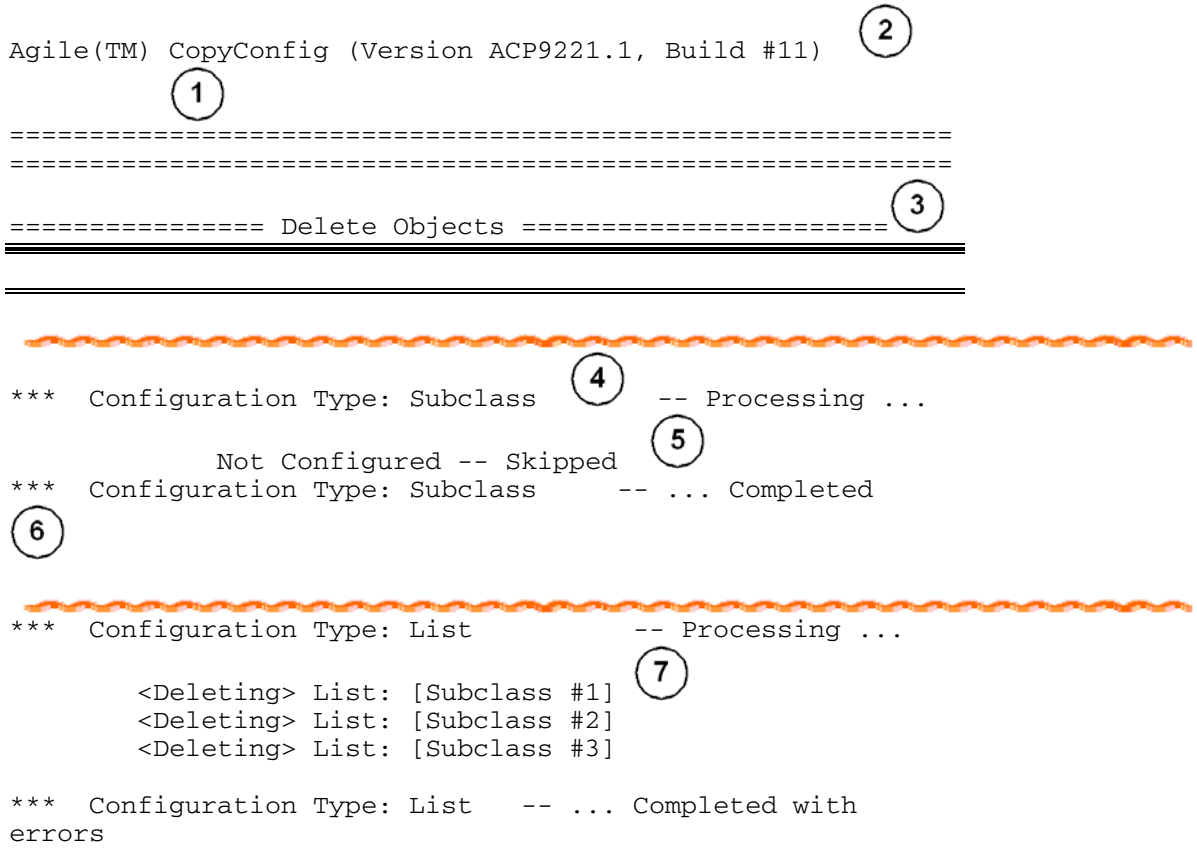
The purpose of the console output is to let you see the progress of ACP as it is running. ACP announces when it is starting and ending a configuration type. It also lets you know the status of processing for a single configuration type. As each object of a configuration type is processed, ACP logs the name of the object to the console. Objects are processed in alphabetical order. The name of the object being processed provides a rough sense of the progress ACP has made on a single configuration type.

Anatomy of Console (stdout) Log

Item #	Description
1	Name of the ACP module running.
2	Version of ACP running.
3	Delete Objects Banner. This is the start of the Delete action processing.
4	Configuration Type. This is the name of the configuration type being processed within a program action (Delete, Rename, or Copy).
5	Not Configured. If the configuration type is not configured for the action being processed, the program reports that it is not configured.

Item #	Description
6	Configuration Type Status: This is the action processing status for the configuration type. It will indicate if the processing "Completed" (no errors or warnings), "Completed with warnings" (had warnings), or "Completed with errors" had errors.
7	Delete Objects. Shows which object is currently being deleted.
8	Rename Objects Banner. This is the start of the Rename action processing.
9	Rename Objects. Shows which object is currently being renamed. It is also showing the name transformation.
10	Copy Objects Banner. This is the start of the Copy action processing.
11	Copy Objects. Shows which object is currently being copied. If it indicates "<Skipping>" before the name, this indicates that the object was not matched using the include/exclude patterns.
12	Program Completion Message. This is a human-readable message indicating if the program succeeded, failed, or completed with warnings.
13	Program Completion Code. This is the machine-readable equivalent to the Program Completion Message. Zero indicates the program completed with no errors or warnings. A positive number indicates the program completed with errors (and possibly warnings). A negative number indicates the program completed with warnings (an no errors).

Sample Console (stdout) Log



```
*****  Configuration Type: Company Profile -- Processing
...
                Not Configured -- Skipped
*****  Configuration Type: Company Profile -- ... Completed
```

```
================ Rename Objects ====================== (8)
```

```
***  Configuration Type: Company Profile (4) -- Processing
...
                Not Configured -- Skipped (5)
***  Configuration Type: Company Profile  ... Completed (6)
```

```
***  Configuration Type: List            -- Processing ...

        <Renaming> List: [Target List ==> Source List] (9)
        <Renaming> List: [Old List ==> New List]

***  Configuration Type: List    -- ... Completed with
errors
```

```
***  Configuration Type: Subclass          -- Processing
...
                Not Configured -- Skipped
***  Configuration Type: Subclass          -- ... Completed
```

```
================== Copy Objects ======================
(10)
```

```
***  Configuration Type: Company Profile (4) -- Processing
...
                Not Configured -- Skipped (5)
***  Configuration Type: Company Profile  -- ... Completed
```

```
(6)
***  Configuration Type: Class              -- Processing
...
                                                        (11)
        <Skipping> Class: [Changes.Change Orders]
        <Skipping> Class: [Changes.Change Requests]
        <Copying>  Class: [Changes.Deviations]
        <Skipping> Class: [Changes.Manufacturer Orders]
        <Skipping> Class: [Changes.Price Change Orders]

        <Skipping> Class: [User Groups.User groups]
        <Skipping> Class: [Users.users]

***  Configuration Type: Class    -- ... Completed with
errors

***  Configuration Type: Subclass     -- Processing ...

                Not Configured -- Skipped
***  Configuration Type: Subclass     -- ... Completed

CopyConfig FAILED - Encountered one or more errors while

renaming data. (201)   (12)

Error Level = 201   (13)
```

Error Log

The purpose of the error log is to provide detailed information regarding an error or warning reported by ACP. Since ACP is not an interactive process, the error information must provide additional information in its error messages. For instance, ACP provides process context. The process context provides information about what object was being processed when the error occurred. It may even provide more specific information about the object if ACP was currently processing sub-object data for the object. In addition to the process context, it tries to provide error context. The error context provides clues about exactly what ACP was trying to do at the time the error was encountered. Finally, ACP provides the cause for the error. That is, it reports the error message reported to it. Sometimes the error message will come from the Agile PLM server and other times ACP will report an error itself.

Anatomy of the Error Log

Item #	Description
1	Start of Error or Warning Message. Signifies the start of a single error or warning.
2	Process Context. The process context section provides clues about what object ACP was processing at the time of the error or warning
3	Propagation Action, Configuration Type, Configuration Object Context. These three lines of process context provide all of the information you need to know in order to know exactly which object was being processed and action was being performed.

Item #	Description
4	Additional Context. Additional context will vary with each error or warning. It attempts to describe within an object what information ACP was working with.
5	Error or Warning Context. The error or warning context provides information about what ACP was doing at the time the error or warning occurred.
6	Error or Warning Cause. The error or warning cause is the specific message issued by the application at the point where the error or warning was detected.
7	End of Error or Warning Message. Signifies the end of a single error or warning.

Sample Error Log

```
          ======== E R R O R (Begin)  ======== (1)
(2)
Process Context:
     Program Name:          CopyConfig
     Program Version:       ACP9221.1, Build #11
     Source Data Source:    Agile XML Archive: export-
golden-config.agl
     Target Data Source:    Agile Instance: URL =
http://gplasket-gx280.
agile.agilesoft.com:8888/web; User = admin
     Propagation Action:              Copy

     Configuration Type:              Criteria (3)
     Configuration Object:            All File Folders
Checked out by Me
     Configuration Object Action:     Update Object

     Object Name:              (4)    All File Folders
Checked out by Me
     Object Table Name:               Conditions
     Object Table Action:             Update


Error Context: (5)
Unable to update the "All File Folders Checked out by Me"
row of the Condition table for the configuration object.

Error Cause: (6)
Object already in use by Privileges.Checkin for File
Folders.

          ========  E R R O R (End)   ======== (7)

          ====== W A R N I N G (Begin)  ====== (1)
```

```
Process Context:  (2)
     Program Name:            CopyConfig
     Program Version:         ACP9221.1, Build #11
     Source Data Source:      Agile XML Archive: export-
golden-config.agl
     Target Data Source:      Agile Instance: URL =
http://gplasket-gx280.
  agile.agilesoft.com:8888/web; User = admin
     Propagation Action:                Copy

     Configuration Type:                List  (3)
     Configuration Object:              QCR Category
     Configuration Object Action:       Update Object

     Object Name:  (4)                   QCR Category
     List Action:                       Update List Entries
     List Entry Action:                 Delete
     List Entry Name:                   QCR Category.Audit -
External

Warning Context:  (5)
     Not Available.

Warning Cause:  (6)
The List Entry "Audit - External" could not be deleted from
the List "QCR Category". The List Entry has been disabled
instead.

          ======  W A R N I N G (End)   ======  (7)
```

Process Log

The purpose of the process log is to capture all of the information pertaining to processing. All command line parameters, program properties, and control file settings are written to the process log. This helps to validate the correctness of the settings in place. As objects are processed, an entry is logged for the object along with the action taken by ACP and the result of the action. Finally, the process log summarizes the processing in a set of statistics providing counts as well as the amount of time it took to process.

Anatomy of the Process Log

Item #	Description
1	Start of Program banner. Signifies the start of a single run of ACP.
2	Name of the ACP module running.
3	Version of ACP running.
4	The date and time that the current run of ACP was started.
5	Program Properties Section: The properties are a means to influence the business logic that ACP uses with regard to configuration types. The amount of influence on business logic varies by configuration type..

Item #	Description
6	Control Setting Section: The configuration values specified in the control file.
7	Configuration Type Name. Each configuration type has its own subsection within the Control Setting Section.
8	File Prefix Setting. This is the file prefix used for the XML files generated by ACP export. The same prefix will be used by ACP import. ACP allows this setting to be configured; however, it should never need to be changed.
9	Objects Per File Setting: This is the number of objects ACP export will write to a single XML file for the configuration type. The default for all configuration types is 1000. Should you have a configuration type that exceeds 1000 and you want to write all of the objects to a single file, you can set the value in the control file.
10	Include Patterns: The list of include patterns configured for the configuration type. These are the regular expressions used to match the object names of the current configuration type.
11	Dependent Configuration Type. Some configuration types have dependent configuration types. The dependent configuration types exist to allow a configuration type to be processed in multiple passes. Typically, dependent configuration types will have the same name as the configuration type they depend on with a suffix such as "(Associations)".
12	Not Configured (Primary Configuration Type). If a configuration type is not configured to be propagated, then it is marked as not configured to make it clear that it is not being propagated.
13	Not Configured (Dependent Configuration Type). Dependent configuration types assume the same configuration information as the configuration type they depend on. If the primary configuration type is not configured, then neither will the dependent configuration type be configured.
14	Sub-object Mappings. Some configuration types allow for sub-object mappings. The sub-object mapping section shows what type of sub-object is being mapped, the name of the object whose sub-objects are being mapped and then the list of mappings.
15	Configuration Type Attributes. Some configuration types may have attributes in addition to the common attributes (file_prefix and objects_per_file). For example, lists have the case_sensitive_list_entries.
16	Exclude Patterns. The list of exclude patterns configured for the configuration type. These are the regular expressions used to match the object names of the current configuration type. If the object name matches an exclude pattern, ACP will skip the object.
17	Name Maps. The list of name mappings for the configuration type that ACP will use to rename objects in the target instance.
18	Delete Names. The list of objects for the configuration type to delete from the target instance.
19	Ignore Reference Patterns. Some configuration types allow ignore reference patterns to be specified. These patterns are used to quash error messages when an object with a name matching one of the patterns cannot be resolved.
20	Program Actions. The program actions section simply indicates which program actions are configured. The available actions are Copy, Rename, and Delete.
21	Configuration Type Detail. Each configuration type that is propagated has a detailed section showing the action taken for each object of that configuration type. The name of the configuration type is listed at the beginning of the section. Configuration types that have not been configured will not have a detail section. A separate configuration type detail section exists for each program action (Copy, Rename, and Delete) if configured.

Item #	Description
22	Action. The actual action taken by ACP. Skipped - Was excluded by the include/exclude patterns specified. Created - The object is new in the target. Updated - The object already existed in the target. Renamed - The object's name has changed in the target. Deleted - The object has been deleted from the target.
23	Result. Indicates how the action performed. No Action Taken - The target object remains unchanged. Succeeded - The target object has been changed successfully. Warning - The target object has been changed successfully, but there were notes to be aware of. Failed - An error has occurred while processing the object. The object may or may not have been updated.
24	Configuration Object. The name of the configuration object being processed.
25	Pattern Matching Statistics. The pattern matching statistics help you understand how well your patterns (regular expressions) are matching the objects in the source instance.
26	Copy Statistics (also Rename Statistics and Delete Statistics). The copy statistics section summarizes the object processing for the configuration type. It is an easy way to find out how many objects were propagated as well as the number of objects that are available to propagate.
27	No objects found. Indicates that no objects were found for the configuration type. This provides a clear statement to this effect rather than not having anything listed at all.
28	Error Messages. A brief description of error messages appear in the Process Log. Please refer to the Error Log for more detailed information about errors.
29	End of Program banner. Signifies the end of a single run of ACP.

Sample Process Log

```
================== Start of Program ===================
(1)

Agile(TM) CopyConfig (2) (Version ACP9221.1, Build
#11)(3)

Run Date: Nov 29, 2007 9:16:34 PM (4)
================== Program Settings ======================
-debug               []
-D                   [acp.method=export, src.ref=qa_was,
                      tgt.ref=qa_was]
```

```
Property File       []

================ Program Properties ==================

     Miscellaneous Properties
global.properties.filename:                (program)
common.properties.filename:                (program)

     Program Information
acp.program.name:              ACP CopyConfig   (program)
acp.program.version:          ACP9221.1, Build #11 (program)
acp.run.date:                     20071128          (program)
acp.run.time:                     074631            (program)
acp.run.datetime:                 20071128_074631   (program)
acp.method:                       export       (command-line)
acp.debug:                        false             (program)
acp.verbose:                      false             (program)

     Objects
objects.suppress.skipped:         true
(D:\home\gplasket\Test
                                    Scripts\oas\9221\dev\
                                    project.properties)

     Source Data Source
src.ref:                          qa_was       (command-
line)
src.type:                         agile             (program)
src.name (qa_was.name):           QA Was
                         (D:\home\gplasket\Test Scripts\oas\
                         9221\dev\project.properties)
src.url (qa_was.url):          http://10.3.3.21:9080/Agile
                         (D:\home\gplasket\Test Scripts\oas\
                         9221\dev\project.properties)
src.username (qa_was.username):   propagation
                         (D:\home\gplasket\Test Scripts\oas\
                         9221\dev\project.properties)
src.password.mode (qa_was.password.mode): cleartext
                         (D:\home\gplasket\Test Scripts\oas\
                         9221\dev\project.properties)

     Target Data Source
tgt.ref:                        qa_was       (command-line)
tgt.type:                       xml               (program)
tgt.name (qa_was.name):         QA Was
                         (D:\home\gplasket\Test Scripts\oas\
                         9221\dev\project.properties)
tgt.xml (qa_was.xml):           export_qa_was.agl
                         (D:\home\gplasket\Test Scripts\oas\
                         9221\dev\project.properties)

     Control File
```

```
control.filename:               config.xml        (program)
     Process Log
log.process.filename:           export.log
(${acp.method}.log)
                                                  (program)
log.process.open.mode:          overwrite           (program)
log.process.format:             text                (program)

     Error Log
log.error.filename:             export.log
(${acp.method}.log)
                                                  (program)
log.error.open.mode:            overwrite           (program)
log.error.format:               text
(program)

     Verbose Log
log.verbose.filename:  exportVerbose.log
                    (${acp.method}Verbose.log)  (program)
log.verbose.open.mode:

================ Control Settings ===================
```

6

```
Configuration Type:   Account Policy
```

7

```
   File Prefix:                 account_policy_
```

8

```
   Objects Per File:            1000
```

9

```
   Include Patterns:            .*
```

10

```
Configuration Type:   Account Policy (Associations)
        File Prefix:                 account_policy_
        Objects Per File:            1000
        Include Patterns:            .*
```

11

```
Configuration Type:   AutoNumber    (Not Configured)
Configuration Type:   AutoNumber (Associations)
                                  (Not Configured)
```

12

13

```
Configuration Type:   Base Class
        File Prefix:                 base_class_
        Objects Per File:            1000
        Include Patterns:            .*

        Life Cycle Phase Mappings for "Manufacturers":
                      1) [Approved] ==> [Active]
```

14

```
Configuration Type:   Character Set
        File Prefix:                 character_set_
```

```
        Objects Per File:           1000
        Include Patterns:           .*

Configuration Type:   Class
        File Prefix:                class_
        Objects Per File:           1000
        Include Patterns:           .*
```

```
Configuration Type:   List
        File Prefix:                  list_
        Objects Per File:             1000

        Case Sensitive List Entries:  Yes (true)   (15)
        Include Patterns:             .*

        Exclude Patterns:             .*   (16)

         Name Mappings:
                     1) Target List ==> Source List

                     2) Old List ==> New List   (17)
        Delete Names:

                     1) Subclass #1   (18)
                     2) Subclass #2
                     3) Subclass #3
```

```
Configuration Type:   User           (Not Configured)
        Ignore Reference Patterns:  Test.*

Configuration Type:   User (Associations) (Not Configured)

Configuration Type:   User Group

        File Prefix:                user_group_
        Objects Per File:           1000
        Include Patterns:           .*

        Ignore Reference Patterns:  Test.*   (19)

Configuration Type:   User Group (Associations)

        File Prefix:                user_group_
        Objects Per File:           1000
        Include Patterns:           .*
```

```
=================== Program Actions  ======================
```

```
     Copy Objects:        Configured
     Rename Objects:      Not Configured
     Delete Objects:      Not Configured
=========================== List ======================
```

(20) (21) (22) (23) (24)

```
Action    Result                 Configuration Object
-------   -----------------      ----------------------
Create    Succeeded              Acme 1-5 List
Create    Succeeded              Acme Authoring Tool
Create    Succeeded              Acme Business Unit
Create    Succeeded              Acme Connector Gender
Create    Succeeded              Acme Eng Lib Category
Create    Succeeded              Acme FA Child/Component
Create    Succeeded              Acme FA Closure Code
Create    Succeeded              Acme FA Component
Priority
Create    Succeeded              Acme FA Configuration
Type
Create    Succeeded              Acme FA Failure Code
Create    Succeeded              Acme FA Failure Mode
Create    Succeeded              Acme FA Priority
Create    Succeeded              Acme FA Root Cause Code
Create    Succeeded              Acme FA Severity
Create    Succeeded              Acme FA Symptom Code
Update    No Action Needed       Action Status
Update    No Action Needed       AML Preferred Status
```

```
Update    No Action Needed       Priority
Update    No Action Needed       Problem Report Category
Update    Succeeded              Product Line
Update    No Action Needed       Product Line List
Update    No Action Needed       Product Lines(PSR)
Update    No Action Needed       Program Type List
Update    No Action Needed       Programs
Update    No Action Needed       Project Name
Update    No Action Needed       PSRs
Update    No Action Needed       QCR Category
Update    No Action Needed       QCRs
```



```
Update    No Action Needed       User Groups
Update    No Action Needed       Users
Update    No Action Needed       Yes/No Cost List
```

(25)

```
     Pattern Matching Statistics
---------------------------------------------------------
        .*:                          251
```

(26)

```
     Copy Statistics
---------------------------------------------------------
        Objects Attempted:           251
        Objects Skipped:               0
```

```
        Objects No Action Needed:     251
        Objects Not Licensed:           0
        Objects Created:                0
        Objects Updated:                0
        Objects with Warnings:          0
        Objects Failed:                 0
```

```
======================= User Group ========================
Action    Result                    Configuration Object
-------   -----------------------   -----------------------
```

(27)

```
No objects found.
     Pattern Matching Statistics
-------------------------------------------------------------
        .*:                             0
     Copy Statistics
-------------------------------------------------------------
        Objects Attempted:              0
        Objects Skipped:                0
        Objects No Action Needed:       0
        Objects Not Licensed:           0
        Objects Created:                0
        Objects Updated:                0
        Objects with Warnings:          0
        Objects Failed:                 0
```

```
============================ Class
=========================
Action    Result                    Configuration Object
-------   -----------------------   -----------------------
--
Update    No Action Needed          Changes.Change Orders
Update    No Action Needed          Changes.Change Requests
Update    Succeeded                 Changes.Deviations
Update    No Action Needed          Changes.Manufacturer
Orders
Update    No Action Needed          Changes.Price Change
Orders
Update    No Action Needed          Changes.Site Change
Orders
Update    Succeeded                 Changes.Stop Ships
Update    No Action Needed          Customers.customers
Update    No Action Needed
Declarations.Homogeneous Material Declarations
Update    Completed with Errors     Declarations.IPC 1752-1
Declarations
             1) ERROR: Duplicate name entered.

             2) ERROR: Duplicate name entered.
```

(28)

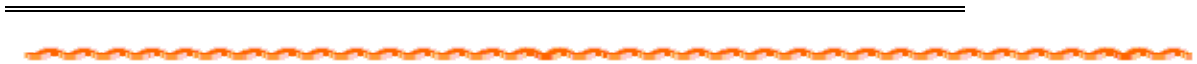
```
Update    Completed with Errors     Declarations.IPC 1752-2
Declarations
              1) ERROR: Duplicate name entered.
              2) ERROR: Duplicate name entered.

Update    No Action Needed          Declarations.JGPSSI
Declarations
Update    No Action Needed          Declarations.Part
Declarations
Update    No Action Needed          Declarations.Substance
Declarations
Update    No Action Needed          Declarations.Supplier
Declarations of Conformance
Update    No Action Needed          Discussions.discussions
Update    No Action Needed          File Folders.File
folders
Update    No Action Needed          File Folders.Historical
Report File Folders
Update    Succeeded                 Items.Documents
Update    Succeeded                 Items.Parts
Update    Succeeded                 Manufacturer
Parts.Manufacturer parts
Update    Completed with Errors
Manufacturers.manufacturers
              1) ERROR: Attribute name must not contain
dot.
              2) ERROR: Attribute name must not contain
dot.
              3) ERROR: Attribute name must not contain
dot.

Update    No Action Needed          Packages.packages
Update    No Action Needed          Part Groups.Part groups
Update    No Action Needed          Prices.Published Prices
Update    No Action Needed          Prices.Quote Histories
Update    No Action Needed          Product Service
Requests.Non-Conformance Reports
Update    No Action Needed          Product Service
Requests.Problem Reports
Update    Succeeded                 Programs.Activities
Update    Succeeded                 Programs.Gates
Update    No Action Needed          Quality Change
Requests.Audits
Update    No Action Needed          Quality Change
Requests.Corrective and Preventive Actions
Update    No Action Needed          Reports.Custom Reports
Update    No Action Needed          Reports.External
Reports
Update    No Action Needed          Reports.Standard
Reports
Update    Succeeded                 Requests for
Quote.Requests for quote
Update    No Action Needed          RFQ Responses.RFQ
responses
Update    No Action Needed          Sites.sites
Update    Completed with Errors     Sourcing
Projects.Sourcing projects
```

```
             1) ERROR: Duplicate name entered.
             2) ERROR: Duplicate name entered.

Update    No Action Needed
Specifications.specifications
Update    Completed with Errors     Substances.Materials
             1) ERROR: Duplicate name entered.
             2) ERROR: Duplicate name entered.

Update    Completed with Errors     Substances.Subparts
             1) ERROR: Duplicate name entered.
             2) ERROR: Duplicate name entered.

Update    No Action Needed          Substances.Substance
                                   Groups
Update    No Action Needed          Substances.substances
Update    Succeeded                 Suppliers.suppliers
Update    No Action Needed          Transfer
Orders.Automated
                                   Transfer Orders
Update    No Action Needed          Transfer Orders.Content
                                   Transfer Orders
Update    No Action Needed          User Groups.User groups
Update    No Action Needed          Users.users


     Pattern Matching Statistics
        .*:                        49
     Copy Statistics
        Objects Attempted:         49
        Objects Skipped:            0
        Objects No Action Needed:  34
        Objects Not Licensed:       0
        Objects Created:            0
        Objects Updated:            9
        Objects with Warnings:      0
        Objects Failed:             6


=============== User Group (Associations)  ================
Action    Result                   Configuration Object
-------   ----------------------   ----------------------
     No objects found.

     Pattern Matching Statistics
        .*:                        0
     Copy Statistics
        Objects Attempted:         0
        Objects Skipped:           0
        Objects No Action Needed:  0
        Objects Not Licensed:      0
        Objects Created:           0
        Objects Updated:           0
```

```
        Objects with Warnings:          0
        Objects Failed:                 0
```

```
=================== Role (Associations)  ==================
Action    Result                    Configuration Object
-------   ------------------------  ----------------------
Skipped                             (Propagation)
Administrator
Skipped                             (Propagation) User
                                    Administrator
Skipped                             (Restricted) Discussion
                                    Participant
Skipped                             (Restricted) Material
Provider
Skipped                             (Restricted) My User
Profile
Skipped                             (Restricted) Price
Collaborator
Skipped                             (Restricted) RFQ
Responder
Skipped                             (Restricted) Supplier
Manager
Update    Succeeded                 Acme Basic User
Update    Succeeded                 Acme Business Override
Update    Succeeded                 Acme FA CDO Manager
Update    Succeeded                 Acme FA Engineer
Update    Succeeded                 Acme FA Logistics
Update    Succeeded                 Acme FA Manager
Update    Succeeded                 Acme FA Manufacturing
QE
Update    Succeeded                 Acme FA TAC/HTTS
Creator
Update    Succeeded                 Acme IT Override
Update    Succeeded                 Acme MCAD Full Library
Viewer
Update    Succeeded                 Acme MCAD Library
Creator
Update    Succeeded                 Acme MCAD Limited
Library Viewer
Update    Succeeded                 Acme PX Role
Update    Succeeded                 Acme ReadOnly Partner
Update    Succeeded                 Acme ReadOnly
Unrestricted
Update    Succeeded                 Acme_1
Update    Succeeded                 Acme_2
Update    Succeeded                 Acme_3 [do not use]
Skipped                             Administrator
Skipped                             Approve / Reject
Skipped                             Change Analyst
Skipped                             Compliance Manager
Skipped                             Component Engineer
Skipped                             Content Manager
Skipped                             Discussion
Administrator
Skipped                             Discussion Participant
Skipped                             Enforce Field Level
Read
```

```
Skipped                                 Engineer
Skipped                                 Engineer Unrestricted
Skipped                                 Engineering
Administrator
Skipped                                 Engineering Manager
Skipped                                 Executive
Skipped                                 Folder Administrator
Skipped                                 Folder Manager
Skipped                                 Incorporator
Skipped                                 Item Content Manager
Skipped                                 Manufacturer Content
Manager
Skipped                                 My File Folder
Skipped                                 My User Profile
Skipped                                 Organization Manager
Skipped                                 Partner
Skipped                                 Portfolio Analytics
User
Skipped                                 Price Administrator
Skipped                                 Price Manager
Skipped                                 Product Content Read
Only
Skipped                                 Program Administrator
Skipped                                 Program Manager
Skipped                                 Program Team Member
Skipped                                 Quality Administrator
Skipped                                 Quality Analyst
Skipped                                 Quality Analytics User
Skipped                                 Report Manager
Skipped                                 Report User
Skipped                                 Resource Pool
Administrator
Skipped                                 Resource Pool Owner
Skipped                                 RFQ Manager
Skipped                                 Sourcing Administrator
Skipped                                 Sourcing Project
Manager
Skipped                                 User Administrator
Skipped                                 View Historical Report

     Pattern Matching Statistics
-------------------------------------------------------------------------------
        Acme.*:                         18
     Copy Statistics
-------------------------------------------------------------------------------
        Objects Attempted:              68
        Objects Skipped:                50
        Objects No Action Needed:        0
        Objects Not Licensed:            0
        Objects Created:                 0
        Objects Updated:                18
        Objects with Warnings:           0
        Objects Failed:                  0
```



```
================== End of Program ========================
```

29