

Agile Product Lifecycle Management

Agile Configuration Propagation Guide

Release 9.3.4

E52161-01

January 2015

E52161-01

Copyright © 2010, 2015, Oracle and/or its affiliates. All rights reserved.

Primary Author: Oracle Corporation

Contributing Author: Edlyn Sammanasu

Contributor:

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface	ix
Audience.....	ix
Documentation Accessibility	ix
Related Documents	ix
Conventions	ix
 1 Welcome to ACP	
Overview.....	1-1
Purpose of ACP.....	1-1
Functions of ACP	1-1
Who will use ACP?	1-2
Who should not use ACP?.....	1-2
Improper Uses of ACP	1-2
System Requirements for ACP.....	1-2
Platform Requirements.....	1-2
Licensing Requirements.....	1-3
Environmental Requirements	1-3
Using this Manual	1-3
What's New in ACP.....	1-4
New Features in ACP934	1-4
New Features in ACP933	1-4
New Features in ACP932	1-4
New Features in ACP93	1-5
ACP931/9301/9302.....	1-5
Changes to the Control File Schema (for previous ACP users).....	1-5
Server Portals not supported (Copy, Rename, Delete)	1-6
Case Sensitive List Entries attribute removed (Copy)	1-6
Changes in Rename section due to API Name/User ID	1-6
Rename by API Name.....	1-6
Rename by User ID.....	1-7
Rename no longer supported	1-7
Changes to Subobject Maps section due to API Name	1-8
Subobject Map object reference by API Name.....	1-8
Subobject Map subobject identified by API Name.....	1-8
Subobject Map no longer supported	1-9

2 ACP Terms

Instance Terms	2-1
Function Terms	2-2
Data Terms	2-2
Machine Terms	2-3
File Terms	2-3
Process Terms	2-3
User Terms	2-4

3 Use Case

Configuration Management	3-1
Configuration Tasks	3-1
Schematic of PLM Configuration	3-2
Validate Agile PLM Data	3-2
Create a Project	3-3
Configure Agile PLM	3-3
Export Admin Data	3-3
Import Admin Data for Testing	3-3
Test Admin Data Changes	3-3
Prepare for Dry Run	3-4
Execute Dry Run	3-4
Execute Go Live	3-5
Audit Configuration Changes	3-5

4 ACP Product Information

Installation of ACP	4-1
PLM Client Application	4-1
Command-line User Interface	4-2
PLM SDK Application	4-2
Propagation Tool	4-2
Propagation Strategies	4-2
Propagation Method	4-3
A Control File drives every Propagation or Comparison	4-3
API Name-based Rename and Subobject Maps	4-4
ACP Actions and Uses of the Control File	4-4
Copy Action (and section of Control File)	4-4
Rename Action (and section of Control File)	4-5
Delete Action (and section of Control File)	4-6
Subobject Maps (section of Control File)	4-6
Ignore References (section of Control File)	4-7
Name Compare and Deep Compare	4-7
Configuration Types	4-8
Nonpropagating Administrator Objects	4-8
Type Filtering on Configuration Types	4-10
Object Matching (Mapping)	4-11
Limitations of Mapping	4-11

Processing Order in ACP	4-11
Processing Order Rules	4-12
Copy Rules	4-12
Rename Rules	4-12
Delete Rules	4-12
Configuration History	4-13
Internationalization	4-13
Personalization	4-13
Tab Display Configuration	4-13
Global Views	4-13
Global Searches and Out-of-box Searches	4-14
Global Reports and Standard Reports	4-15

5 User Requirements

Standard PLM Privileges that can Access ACP	5-1
Tailored Roles for the ACP User	5-1
Privileges for the ACP User	5-1

6 Installing ACP

Required Information	6-1
Operating System	6-1
Agile PLM Version	6-1
Application Server	6-1
Installation Directory	6-1
Work Directory	6-2
Prerequisites	6-2
Java Runtime Environment	6-2
ACP Installer	6-2
Windows Installation	6-2
Extract	6-2
Run Installer	6-2
Open Command Window	6-2
Go to ACP Installer Directory	6-2
Run the Installer Script	6-2
UNIX (Linux) Installation	6-3
Extract	6-3
Make Executable	6-4
Run Installer	6-4
Open Terminal Window	6-4
Go to ACP Installer Directory	6-4
Run the Installer Script	6-4
Post-Installation Tasks	6-5
ACP-Installed Directories	6-5
ACP Client-Installed Directory Structure	6-5
ACP Work Directory Structure	6-6

7 Running ACP

ACP Projects	7-1
ACP Project Directories	7-1
Sample Project Directory	7-1
Creating Projects	7-2
Existing Project	7-2
New Project	7-3
ACP Properties	7-3
ACP Control File	7-4
ACP Scripts	7-5
ACP Exit Codes	7-5
ACP Log Files	7-5
Summary	7-6

8 Configuring the ACP Control File

ACP Control File	8-1
XML Format	8-1
Element	8-1
Empty Element	8-2
Simple-Content Element	8-2
Element-Content Element	8-2
Mixed-Content Element	8-2
Element with Attributes	8-3
Root Element or Document Element	8-3
Element Tags	8-3
Element Attributes	8-3
Comments in XML	8-3
Special Characters	8-3
Business Logic Attributes in the Control File	8-4
Objects per File	8-4
File Prefix	8-4
Criteria Force Update	8-4
Autonumber Force Update	8-5
Force Delete List Entry	8-5
New User Password	8-6
Process Extension Association Rule	8-6
User Association Rule	8-7
Control File Sections	8-8
Copy (<copy>) Section	8-8
Configuration Types in Copy Section	8-8
Include Patterns	8-8
Exclude Patterns	8-8
Regular Expressions	8-8
Putting it all together in Copy section	8-9
Rename (<rename>) Section	8-10
Configuration Types in Rename Section	8-10
Key Maps	8-10

<i>Source Key</i>	8-11
<i>Target Key</i>	8-11
Putting it all together in Rename section	8-11
Delete (<delete>) Section	8-12
Configuration Types in Delete Section	8-12
Name.....	8-12
Putting it all together in Delete section	8-12
Ignore References (<ignore_references>) Section	8-13
Configuration Types in Ignore References Section	8-13
Patterns	8-13
Regular Expressions	8-14
Putting it all together in Ignore References section	8-14
Subobject Maps (<subobject_maps>) Section	8-14
Configuration Types in Subobject Maps Section.....	8-14
Object Reference	8-15
Subobject Type	8-15
Flex Attribute Rename	8-15
Key Maps	8-15
<i>Source Key</i>	8-15
<i>Target Key</i>	8-16
Putting it all together in Subobject Maps section	8-16

A ACP Configuration Types

Supported ACP Configuration Types	A-1
Configuration Types and Match Keys	A-6
Renaming Subobjects.....	A-8
Configuration Types as Evaluated by Deep Compare.....	A-8
Log and Report Files for Deep Compare	A-15

B Regular Expressions

Java Regular Expressions	B-1
Special Characters	B-1
XML Special Characters	B-1
Java Regular Expression Special Characters	B-1
Regular Expression Examples	B-2

C Properties

Property Sources	C-1
Defining Properties.....	C-2
Java-style Property	C-2
Property References	C-2
Indirect References	C-2
Properties	C-3
Agile-owned Properties	C-3
Agile-defaulted Properties	C-3
Customer-owned Properties	C-4

D ACP Scripts

Working Directory	D-1
Java Home	D-1
Running Scripts	D-1
ACP Launcher	D-1
Propagation Scripts	D-1
export	D-2
import	D-2
Version Script.....	D-2
version	D-2
Project Management Script	D-2
create_project	D-2
Object Name Comparison Script	D-3
name_compare	D-3
Object Detail Comparison Script.....	D-3
deep_compare	D-3
Generating the Deep Compare Difference Report.....	D-3
Command.....	D-3
Console Message:	D-4
Deep Compare Difference Report	D-4
Deep Compare Report Sections and Fields	D-5

E ACP Exit Codes

F ACP Program Logs

Verbose Log.....	F-1
Console (stdout) Log	F-1
Anatomy of Console (stdout) Log	F-1
Sample Console (stdout) Log	F-3
Error Log.....	F-4
Error Messages.....	F-4
Anatomy of the Error Log.....	F-6
Sample Error Log.....	F-8
Process Log	F-8
Anatomy of the Process Log.....	F-9
Sample Process Log	F-11

Preface

Agile PLM is a comprehensive enterprise PLM solution for managing your product value chain.

Audience

This document is intended for administrators and users of the Agile PLM products.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Related Documents

Oracle's Agile PLM documentation set includes Adobe® Acrobat PDF files. The Oracle Technology Network (OTN) Web site

<http://www.oracle.com/technetwork/documentation/agile-085940.html> contains the latest versions of the Agile PLM PDF files. You can view or download these manuals from the Web site, or you can ask your Agile administrator if there is an Agile PLM Documentation folder available on your network from which you can access the Agile PLM documentation (PDF) files.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.

Convention	Meaning
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Welcome to ACP

This guide describes the purpose, installation, and use of Agile Configuration Propagation (ACP).

Overview

ACP lets you propagate the configuration of one instance of Oracle Agile PLM to another instance of the same version of PLM. The “configuration” can be thought of as the complete or partial content of Java Client Administrator settings in an Agile instance.

It is up to you, the PLM administrator, to specify exactly what you want to propagate. The propagation may consist of the complete Administration 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 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.

Although ACP automates the configuration process, ACP only propagates Administrator data that it is directed to propagate. It relies on data in its Control File to dictate what changes 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 reuse them for targeted propagations.

Functions of ACP

There are two basic functions that ACP performs: *Propagating* and *Comparing*.

Propagate Function

ACP propagates in two steps, *Export* and *Import*: A configurable Control File contains a list of the objects for ACP to propagate.

- ACP exports configuration data from Agile instance - the Source instance - to XML files.
- ACP imports configuration data from XML files to Agile instance - the Target instance.

Compare Function

When you want to discover and report differences between Agile instances, not propagate changes, ACP's Compare function reports object-specific differences.

- ACP **compares** configuration objects between XML files (from the Source instance) and Agile (the Target instance).

ACP93 enhances its previous level of comparing, called Name Compare, with a deeper level of comparing, called Deep Compare.

See ["Name Compare and Deep Compare"](#) on page 4-7.

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 *Agile PLM Administrator Guide*.

Who should not use ACP?

ACP is not intended to be used by Agile end-users.

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.

[Chapter 5, "User Requirements"](#) provides more details about appropriate roles and privileges for ACP.

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 supersede Java Client behavior.

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

System Requirements for ACP

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

Platform Requirements

- ACP is supported on all server operating systems that are supported by Agile PLM 9.3.x.

- ACP requires that the Source and Target instances use the same Agile PLM version; there is now a formal “version check”.
- ACP expects that the settings in the **Licenses** node are the same between Source and Target instances. This further discussed in the next topic.

Licensing Requirements

Since ACP is an extension of the **Administrator** module in Java Client, implementing ACP requires no additional license from Oracle.

Important: ACP should propagate only Oracle-licensed configuration types, Oracle-licensed configuration objects, and Oracle-licensed class attributes.

Regarding the settings in **Administrator > Server Settings > Licenses**, ACP expects that the settings in this node are the same between Source and Target instances; that is, ACP does not compare the **Licenses** node settings nor does it report on any discrepancies between the instances.

However, ACP *does* observe the settings in the **Licenses** node of both Source and Target Agile-instances in this way:

- ACP can export data from the Source instance based on the base class/class (to which the data “belongs”) being enabled in the **Licenses** node on the Source instance.
- ACP can import data to a Target instance based on the base class/class (to which the data “belongs”) being enabled in the **Licenses** node on the Source instance.

Again, the settings in **Licenses** node for any owned and operating Agile system should conform to the PLM solutions your company purchased from Oracle.

Environmental Requirements

The ACP Client - where ACP is installed - must be able to connect through HTTP(S) to the Agile-PLM source and target instances. That is, firewalls must not prevent ACP from accessing the PLM source and target.

Note: ACP does not need to be able to connect to both instances at the same time; however, it must be able to connect to either the Source or Target instance based on the ACP function being executed.

Using this Manual

Chapters 1-4 of this manual answer the question “What is ACP?”

- ACP's purpose, system requirements, and the Control File schema are discussed in this chapter.
- Terms used in this manual. See [Chapter 2, "ACP Terms."](#)
- Use case for ACP. See [Chapter 3, "Use Case."](#)
- [Chapter 4, "ACP Product Information"](#) gathers information about ACP's product features, business rules, and observations of its behavior. **Chapter 4 is essential reading for ACP users.** It is also useful to “non-ACP-users” who want to understand how the product works.

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

- Installation. See [Chapter 5, "User Requirements."](#)
- User requirements. See [Chapter 6, "Installing ACP."](#)
- Running ACP. See [Chapter 7, "Running ACP."](#)
- Configuring the Control File. See [Chapter 8, "Configuring the ACP Control File."](#)

Appendixes A-F provide reference material.

The appendices give more detailed information:

- ACP configuration types, match keys, renaming subobjects, and how Deep Compare evaluates the configuration types. See [Appendix A, "ACP Configuration Types."](#)
- Java regular expressions. See [Appendix B, "Regular Expressions."](#)
- ACP properties. See [Appendix C, "Properties."](#)
- ACP scripts, including propagation scripts and comparison scripts. See [Appendix D, "ACP Scripts."](#)
- ACP exit codes. See [Appendix E, "ACP Exit Codes."](#)
- ACP program log file samples. See [Appendix F, "ACP Program Logs."](#)

What's New in ACP

The following sections list new features from the current release, in addition to the past few releases.

New Features in ACP934

No new features were added to ACP in Release 9.3.4:

New Features in ACP933

This list summarizes the changes made to ACP in Release 9.3.3:

- The <InstanceName>.username property has been deprecated. This property is in the project.properties file. When this entry does not exist in this property file, ACP prompts the user to input Login User Name during run time.
- The <InstanceName>.password property has been retired. This property is in the project.properties file. During run time, ACP prompts the user to input Login User Password.
- The <InstanceName>.password.mode property has been retired. This property is in the project.properties file, but is no longer required and will be ignored even if it is set in the file.

New Features in ACP932

Agile Configuration Propagation 9.3.2 introduces support for the propagation of several new configuration types including:

- Tab Configuration
- Global Searches
- Global Views

- Global Reports
- PG & C External Rollups scheduling and rule setting
- Reference Object Application

For more information about propagating Personalization configuration for searches, tab configuration, views, and reports, "[Personalization](#)" on page 4-13.

For more information about the configuration types, "[ACP Configuration Types](#)" on page A-1.

New Features in ACP93

This list summarizes the enhancements to ACP in Release 9.3:

- API Name-based Rename and Subobject Maps - "[API Name-based Rename and Subobject Maps](#)" on page 4-4.
- Object Matching - "[Object Matching \(Mapping\)](#)" on page 4-11.
- Type Filtering on Configuration Types - "[Type Filtering on Configuration Types](#)" on page 4-10.
- Name Compare, Deep Compare, and the Deep Compare Difference Report - these topics:
 - "[Name Compare and Deep Compare](#)" on page 4-7;
 - "[Deep Compare Difference Report](#)" on page D-4;
 - "[Log and Report Files for Deep Compare](#)" on page A-15
- Autonumber Force Update - "[Autonumber Force Update](#)" on page 8-5.
- Event Management, including Create/Delete of event-based notifications - the main feature introduced in PLM 9.3, Event Management, is completely supported by ACP 9.3. This includes propagating event-based notifications (that is, those created as Notification masks, not the out-of-box default notifications)
- New Error Message Header - "[Error Messages](#)" on page F-4.
 - Error Message on:
 - * Unmatched Regular Expressions
 - * Unused Subobject Maps

ACP supports propagating data in all languages that are supported by Agile PLM.

ACP931/9301/9302

No new features were added to ACP931, ACP9301, or ACP9302.

Changes to the Control File Schema (for previous ACP users)

There were no changes to the Control File schema in PLM Releases 9.3.1, 9.3.0.1, or 9.3.0.2.

There were numerous changes to the Control File schema in PLM Release 9.3.

Some elements of the old schema syntax no longer work. This section lists those elements of the Control File that have been removed or are no longer supported or have otherwise changed since ACP 9.2.x.

These are important if you are a previous user of ACP and have customized Control Files that are still useful. The elements below should be properly changed. If any of these are present, the Control File will throw an error and not complete.

You may disregard this list of changes if:

- You are a new customer to Agile PLM, or
- You are a customer of Agile PLM 9.2.x and you did not use ACP, or
- You have used ACP 9.2.x but are not going to use old Control Files with ACP 9.3.x.

Server Portals not supported (Copy, Rename, Delete)

Agile PLM no longer supports Portlets (Agile Portlet Services). The administrator could go to **Server Settings > Locations > Portals** tab to create and manage portals. The corresponding ACP tag was `<server_portal>`.

Therefore, `<server_portal>` is no longer supported in the Copy, Rename, or Delete sections of the Control File. You should remove the tag from your existing Control Files.

Error Message: "cvc-complex-type.2.4.a: Invalid content was found starting with element 'server_portal'."

Case Sensitive List Entries attribute removed (Copy)

The `<case_sensitive_list_entries>` attribute tag has been removed from List (`<list>`) in the Copy section of the Control File.

Functional Replacement: List Entries are now matched by their API Name instead of their Name. The attribute provided an alternative to the Subobject Maps for List Entries when only the case of the List Entry name changed. Now a Subobject Map is required if any part of the API Name for the List Entry has changed, even if only the case changed.

Error Message: "cvc-complex-type.3.2.2: Attribute 'case_sensitive_list_entries' is not allowed to appear in element 'list'."

Changes in Rename section due to API Name/User ID

The following changes all apply to the Rename section of the Control File.

Rename by API Name

With the introduction of API Name to Administrator nodes, API Name is now the preferred identifier (key) for mapping Administrator nodes between Agile instances. This change should require fewer rename actions to be performed as it is expected that API Names will tend not to be changed.

The following Administrator nodes are affected by this change in the Rename section of the Control File:

- ACS Destination (`<acs_destination>`)
- ACS Event (`<acs_event>`)
- ACS Filter (`<acs_filter>`)
- ACS Package Service (`<acs_package_service>`)
- ACS Response Service (`<acs_response_service>`)

- ACS Subscriber (<acs_subscriber>)
- Auto Number (<auto_number>)
- Character Set (<character_set>)
- Criteria (<criteria>)
- List (<list>)
- PPM Dashboard Management (<ppm_dashboard_management>)
- Privilege (<privilege>)
- Process Extension (<process_extension>)
- Role (<role>)
- Server File Manager (<server_file_manager>)
- Subclass (<subclass>)
- Unit of Measure Family (<unit_of_measure_family>)
- Workflow (<workflow>)

Functional Replacement: The rename functionality still exists, however, it operates on the API Name rather than the Name. This requires two changes to the Control File on your part:

1. Use different tag names (source_apiname and target_apiname instead of source_name and target_name);
2. Specify the new and old API Name instead of the new and old Name.

Error Message: cvc-complex-type.2.4.a: Invalid content was found starting with element 'source_name'. One of '{""':source_apiname}' is expected.

Rename by User ID

With the introduction of API Name to Administrator nodes and since Users are identified by their login ID for mapping Users between Agile instances, the XML attribute name reflects what is mapped.

The **Users** (<user>) Administrator node is affected by this change in the Rename section of the Control File.

Functional Replacement: The rename functionality still exists. This requires one change to the Control File on your part: use different tag names (source_userid and target_userid instead of source_name and target_name).

Error Message: cvc-complex-type.2.4.a: Invalid content was found starting with element 'source_name'. One of '{""':source_userid}' is expected.

Rename no longer supported

With the introduction of API Name to Administrator nodes, API Name is the preferred identifier (key) for mapping Admin nodes between Agile instances. For Administrator nodes that are entirely owned by Agile, the API Name cannot be changed. In these cases, there is no longer a need to have a rename for the Administrator node.

The following Administrator nodes are affected by this change in the Rename section of the Control File:

- Base Class (<base_class>)
- Class (<class>)

- Cost Status (<ppm_cost_status>)
- Quality Status (<ppm_quality_status>)
- Resource Status (<ppm_resource_status>)
- Schedule Status (<ppm_schedule_status>)
- Database (<server_database>)

Functional Replacement: The replacement is that ACP will map these objects by their API Names. Since the API Name cannot be changed, the functionality for this feature is complete without any assistance from the Control File.

Error Message: cvc-complex-type.2.4.a: Invalid content was found starting with element '`<admin node tag name>`'.

Changes to Subobject Maps section due to API Name

The following changes all apply to the Subobject Maps section of the Control File.

Subobject Map object reference by API Name

With the introduction of API Name to Administrator nodes, API Name is now the preferred identifier (key) for mapping Administrator nodes between Agile instances. This change should require fewer subobject maps as it is expected that API Names will tend not to be changed.

The following Administrator nodes are affected by this change in the Subobject Maps section of the Control File:

- Base Class (<base_class>)
- Class (<class>)
- List (<list>)
- Subclass (<subclass>)
- Unit of Measure (<unit_of_measure_family>)
- Workflow (<workflow>)

Functional Replacement: The subobject map functionality still exists, however, the API Name is used to identify the object rather than the Name. This requires two changes to the Control File on your part:

1. Use different attribute name (apiname instead of name);
2. Specify the object's API Name as its identifier.

Error Message: cvc-complex-type.3.2.2: Attribute 'name' is not allowed to appear in element '`<Admin Node tag name>`'.

Subobject Map subobject identified by API Name

With the introduction of API Name to Administrator nodes, API Name is now the preferred identifier (key) for mapping Subobject nodes between Agile instances. This change should require fewer subobject maps as it is expected that API Names will tend not to be changed.

The following Subobject nodes are affected by this change in the Subobject Maps section of the Control File:

- Base Class (<base_class>) à **Life Cycle Phases** (<life_cycle_phases>)

- Class (<class>) à **Life Cycle Phases** (<life_cycle_phases>)
- List (<list>) à **List Entries** (<list_entries>)
- Subclass (<subclass>) à **Life Cycle Phases** (<life_cycle_phases>)
- Unit of Measure (<unit_of_measure_family>) à **Units of Measure** (<unit_of_measures>)
- Workflow (<workflow>) à **Workflow Statuses** (<workflow_statuses>)

Functional Replacement: The subobject map functionality still exists, however, the API Name is specified rather than the Name. This requires two changes to the Control File on your part: 1) use different tag names (source_apiname and target_apiname instead of source_name and target_name; 2) specify the new and old API Name instead of the new and old Name for the subobject being referenced.

Error Message: cvc-complex-type.2.4.a: Invalid content was found starting with element 'source_name'. One of '{"":source_apiname}' is expected.

Subobject Map no longer supported

With the introduction of API Name to Administrator nodes, API Name is now the preferred identifier (key) for mapping Subobject nodes between Agile instances. For Subobject nodes that are entirely owned by Agile, the API Name cannot be changed. In these cases, there is no longer a need to have a subobject map for the Subobject node.

The following Subobject nodes are affected by this change in the Subobject Maps section of the Control File:

- Class (<class>) à **User Interface Tabs** (<user_interface_tabs>)
- Subclass (<subclass>) à **User Interface Tabs** (<user_interface_tabs>)

Functional Replacement: The replacement is that ACP will map these subobjects by their API Names. Since the API Name cannot be changed, the functionality for this feature is complete without any assistance from the Control File.

Error Message: cvc-complex-type.2.4.a: Invalid content was found starting with element 'user_interface_tabs'. One of '{"":attributes, "":life_cycle_phases}' is expected.

This chapter describes terms that are used within ACP.

Instance Terms

Your company will likely have several instances of Agile PLM, which were set up for specific purposes. (These purposes are detailed in the next chapter, "Use Case" on page 3-1.

First there are two "instance" terms that simply name "From" and "To" installations when you use ACP:

- **Source instance** - the installation of Agile PLM *from* which you are propagating Administrator data with ACP
- **Target instance** - the installation of Agile PLM *to* which you are propagating Administrator data with ACP

Beyond Source and Target, the instance names below are only suggestions. They are not required to use ACP; however, these names are used in this manual as defined below, notably in "Use Case" on page 3-1.

- **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 before 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 conducted. It must stay pure and always be in a consistent state.

Function Terms

- **propagate** - when an Agile instance is changed, no matter if it is 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.

ACP propagates in two steps, Export and Import:

- **export** - ACP gathers data from the Source instance of Agile. The result of the Export function in ACP is an XML file, or archive, of the Source.
- **import** - ACP replaces data from XML (the Source archive) into the Target instance of Agile.

Note: ACP's export and import functions should not be confused with either the PLM utilities Agile Export and Import, nor Administrator Export and Import in Java Client.

- **compare** - ACP compares between Source-instance and Target-instance data so it can report object-specific differences between the two Agile instances.

There are two kinds of Compare in ACP:

- **Name Compare** - the name-based comparison identifies new objects, deleted objects, and objects whose name changed. (This is the same compare function supported by ACP 9.2.x.)
- **Deep Compare** - ACP can examine Administration objects at a deeper level, the level of object attributes. With Deep Compare, Admin objects in the Source instance are compared to those in the Target instance to capture changes in the names and properties of object Attributes.

Data Terms

- **API Name** - a required field for many Administrator objects and PLM business objects; as it tends not to change (whereas the Name attribute is much more likely to change), it is a unique systemwide identifier or key for Admin/PLM objects.
- **object key** - a business object's key is the *API Name* if the Administrator node supports it, or the object's *Name* if the node does not support API Name; for Users, the object key is *User ID*. This term is important when ACP is performing the operations listed in "Process 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.

When you propagate the data settings held in the **Roles** node in Administrator, 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.

Therefore, when you propagate the data settings held in the Change Analyst role in Administrator (**User Settings > Roles** node > **Change Analyst** role), you are propagating the *Change Analyst configuration object* in ACP.

Machine Terms

- **Agile Server** - the system where Agile PLM is installed
- **ACP Client** - the system where ACP runs
- **Agile Install Directory** - this is the directory you choose to install the ACP utility. "<version>" indicates where you would fill in the version of Agile PLM that you are working in, for example, "934".

Example in Windows: D:\Agile\ACP<version>

Example in Unix: /opt/agile/acp<version>

- **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. "<version>" indicates where you would fill in the version of Agile PLM that you are working in, for example, "934".

Example in Windows: D:\Agile\ACPwork<version>

Example in UNIX: <user.home>/Agile/ACPWork<version>

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. (It is possible to keep config.xml as an "example control file" and create working control files with meaningful names. This idea is detailed in ["Running ACP"](#) on page 7-1.)
- **Agile XMLarchive** - a ZIP file with an extension (.agl) that contains XML files exported by ACP

Process Terms

- **Delete** - a propagate action; the Delete section of the Control File instructs ACP about **delete** operations it should perform;
- **Rename** - a propagate action; the Rename section of the control file instructs ACP about **rename** operations it should perform;
- **Copy** - a propagate action; the Copy section of the control file instructs ACP about **create**, **update**, and **replace** operations it should perform.

User Terms

- **Agile user** - a person at your company who is 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 is 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 is 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 is given role/privilege access to use ACP."

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: The task of 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 enables 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.

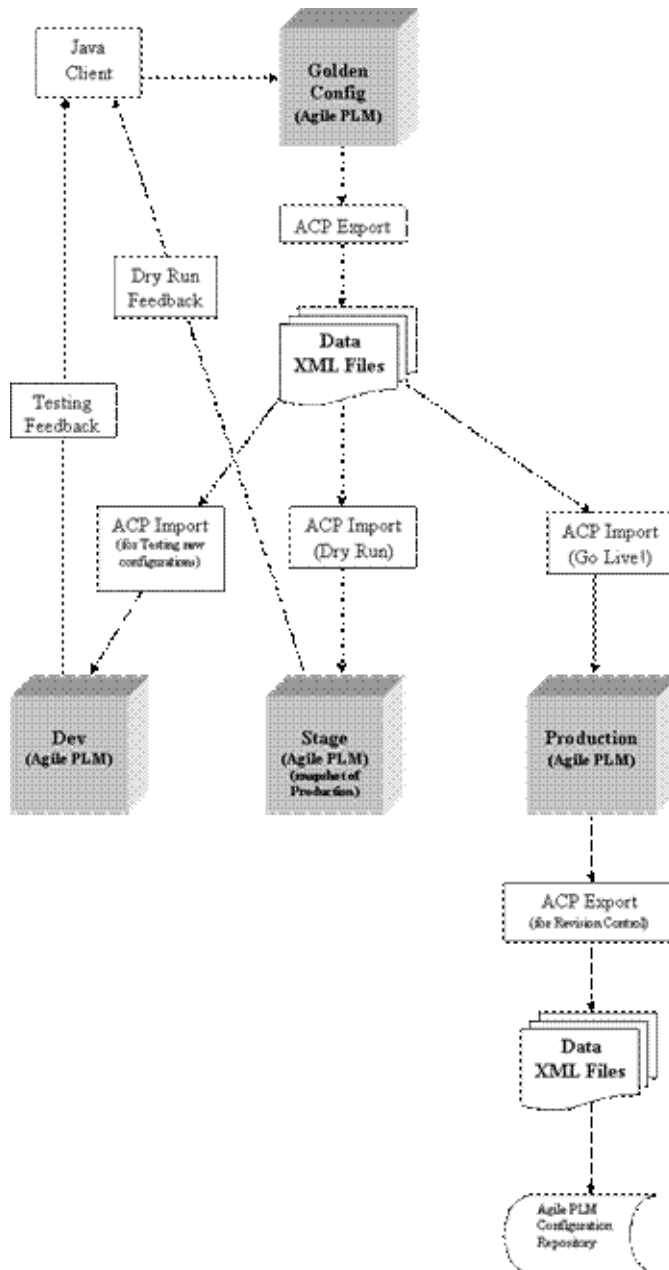
For the discussion below, refer to "[Instance Terms](#)" on page 2-1 for the definitions of various kinds of Agile instances, such as Golden Config, Dev, Stage, and Production.

Configuration Tasks

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

Schematic of PLM Configuration

Figure 3–1 Schematic of PLM Configuration



Validate Agile PLM Data

Use aVerify to validate the data in an Agile PLM instance. A good practice 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 Administrator data, you can limit your concern about data issues to Administrator data.

For information pertaining to aVerify, consult *aVerify Release Notes* (on OTN, Preface) or contact Oracle Support Services.

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 have its own lifecycle.

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 must be propagated later.
- Use a spreadsheet to keep track of actual changes made. This aids 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 7-1 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 Stage instance. New features are developed and tested on the Dev instance, but they should be tested again on the Stage instance 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.

1. Develop a formal test plan to test the changes.
2. Develop a Go Live! "sanity test plan".
3. Execute the formal test plan.
4. Validate the test plan results.

5. 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 ensure 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 must only be done 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 documents the exact steps used to propagate the data to the Production instance. Use the script to document the order in which steps are performed, and 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 the 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.

ACP Product Information

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 system as an Agile PLM server, or on a separate system.

- ACP can be installed on the following operating systems: Windows, Linux, Solaris.
- ACP can work with the following application servers: WebLogic Application Server (WLS)
- 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 system, 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 requires the source and target instance to be the same version of PLM.

ACP does not require that the source and target instances be on the same 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.

Command-line User Interface

ACP is initiated through 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.

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 is expressly designed as a propagation tool. ACP's primary purpose is to update or create Administrator objects in a target Agile instance just as they are configured in the source instance. A secondary purpose is to compare Administrator objects between source and target instances so you are certain about what is different between the two. How ACP propagates and compares is discussed below in this chapter.

As a reminder, there are several kinds of operations that cannot be accomplished by ACP:

- **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 supersede 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.

Propagation Method

ACP uses a two-step propagation method:

1. **Export** - ACP first exports the objects to propagate from one Agile instance to an Agile XML archive;
2. **Import** - ACP's second step is to import objects in an Agile XML archive to an Agile instance.

This method enables 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.

A Control File drives every Propagation or Comparison

The Control File tells ACP what specific configuration types it should examine when it is run. A Control File can be run as a “compare” - using Name Compare or Deep Compare scripts - where configuration types are specified for informational and reporting purposes. A Control File can be run as a propagation where configuration types are specified for altering the target instance using Delete, Rename, or Copy (Create, Update, Replace) actions.

The Control File has a defined schema, which enables you to use tools like XMLSpy to edit the Control File.

The Control File enables you to be as general or specific with the objects you want ACP to propagate through 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.

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.

Using Compare reports as a guide, you can determine which configuration items can be included in the propagation and which ones can be excluded from the propagation.

- 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.
- The 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 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 must be propagated unless the role did not previously exist.

API Name-based Rename and Subobject Maps

The **API Name** attribute has been added to all Administrator nodes (with some exceptions). API Name is now the preferred identifier (key) for mapping Administrator nodes between Agile instances.

Renames and Subobject Mapping are now based on the API Name attribute. This change should require fewer rename actions to be performed, and fewer subobject maps, as API Names will not be changed.

The API Name attribute is supported in ACP (for all PLM objects that support API names) with the tags <source_apiname> and <target_apiname>.

Users are renamed with the tag <source_userid>. This is only used when the User ID is changed.

There are also Administrator objects where <source_name> and <target_name> are used.

ACP Actions and Uses of the Control File

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 enables 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 enables you to rename objects in the Target instance;
- The **Delete** action enables you to delete objects in the Target instance.

There are two additional sections of the Control File that do not instruct propagation changes:

- **Ignore References** directs ACP to ignore references to certain objects that will not be propagated.
- **Subobject Maps** allows ACP to match subobjects with different names between the Source and Target instances;

These three propagation actions and two (non-propagating) sets of instructions are briefly described below, and are more fully detailed in ["Configuring the ACP Control File"](#) on page 8-1.

The two compare scripts, Name Compare and Deep Compare, are also introduced below.

Copy Action (and section of Control File)

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 an *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 *object key* in the target instance. (The object key is the API Name if the Administrator node supports it, or the Name if the node does not support API Name; for Users, the object key is User ID.)
 - 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, to change the object's type, it must be deleted and re-created. An example of a strongly typed Administrator node is **Lists**. 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 8-8.

Rename Action (and section of Control File)

The Rename action directs ACP to rename an object's key (that is, the value of the object key for that object). Renaming is now based on the **API Name** attribute, which has been added to most kinds of objects in PLM.

The Control File uses the XML term <map> to rename objects whose key was changed in the Source instance. Name maps have a Source name (key) - *which is the “new” name because the ACP user has changed it in the Source instance* - and a Target name (key) - *which is the “old” name because it has not yet been changed in the Target instance*.

- To make it more self-evident, these tag names have been introduced:
 - <source_apiname>
 - <source_name>
 - <source_userid>
 - <target_apiname>

- <target_name>
- <target_userid>
- 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.

- The names specified in the map must match the object's name exactly. Regular expressions cannot be used 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](#)" on page 8-10.

Delete Action (and section of Control File)

The Delete action allows objects to be deleted in the Target.

As a protective measure, the object name specified must exactly match the name of the object-to-be-deleted.

- 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.

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

- Not all configuration types support the Delete operation.
- The configuration object is found by name in the Target instance. The names specified in the map must match the object's name exactly. Regular expressions cannot be used here.
- 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 8-12.

Subobject Maps (section of Control File)

You will use the Subobject Maps section of the Control File to map the keys for subobjects. Subobject mapping is now based on the **API Name** attribute, which has been added to most kinds of Administrator objects.

Some Administrator objects include a set of subordinate objects. Subordinate objects ("subobjects") cannot exist by themselves, they exist only in relationship to a parent

object. An example of an object/subordinate object relationship is: List (object) and List Entry (subordinate object).

The point is that a specific List Entry object does not exist independently, it must belong to a List.

Subobject mapping can be applied to the following configuration types:

- Base Class
- Class
- Subclass
- List
- Unit of Measure
- Workflow
- My Assignments

Subobject mapping is fully detailed in "[Subobject Maps \(<subobject_maps>\) Section](#)" on page 8-14.

Object mapping is introduced in "[Object Matching \(Mapping\)](#)" on page 4-11.

Ignore References (section of Control File)

The Ignore References section of the Control File is for references to objects that will not be propagated by the ACP actions.

"Ignore References" is not an ACP action. The usefulness of Ignore References is apparent when "test Admin nodes" are created in a development environment (although these objects have a place in a Development environment, they have no place in the Production environment).

The "ignorable" references are limited to Admin nodes where test objects are likely to be created:

- Users
- User groups
- Roles
- Privileges

Regular expressions may be used to specify which objects should be ignored.

Ignore References is fully detailed in "[Ignore References \(<ignore_references>\) Section](#)" on page 8-13.

Name Compare and Deep Compare

ACP has always distinguished differences in names of Administrator objects between instances of Agile. ACP could use the Copy operations to make actual changes to object names (in the Target, based on what ACP perceived in the Source); ACP could also simply report the differences, through the Compare function.

ACP still supports the comparison of configuration objects based on their name, which is now called **Name Compare**.

ACP can now compare object attributes, both on the level of the Attribute names, and further on the names of the Properties of the Attributes. This comparison is called **Deep Compare**. With Deep Compare, each and every attribute of an object in the

Source is compared with the ones in the Target to reveal all differences in properties of the attributes.

The Compare function supports these data sources:

- in the Source instance: XML (the XML archive that is the first step to a potential propagation);
- in the Target instance: Agile (the PLM data that stands to be altered in a potential propagation).

Name Compare and Deep Compare are driven by the Control File against which they are run; only objects and configuration types specified in the Control File are compared. All other objects and configuration types are ignored regardless of whether they are present in the "XML archive" (Source) or "Agile" (Target).

Sections of the Control File that Deep Compare pays attention to:

- Copy
- Rename
- Subobject Maps
- Ignore References

Section of the Control File that Deep Compare does not pay attention to:

- Delete

A Deep Compare Report is generated through BI Publisher, and the supporting output format is a modified Microsoft Excel spreadsheet.

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 A-1 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*.

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

Nonpropagating Administrator Objects

The following Administrator objects (listed in alphabetical order) cannot be propagated.

Non-propagating Administrator objects	Where found (Java Client unless noted)	Comment
Attachments on User and User Group objects	User Settings > Users, User Groups	NA
Cache Health Monitor	Web Client > Administration	Web Client--only Admin data and Monitors are not propagated.

Non-propagating Administrator objects	Where found (Java Client unless noted)	Comment
Character Sets that are "out of the box"	Data Settings	NA
Currency Exchange Rates	System Settings	NA
Deleted Users	User Settings	ACP must be explicitly directed to delete users.
Deleted User Groups	User Settings	ACP must be explicitly directed to delete users.
Event Handler Monitor	System Settings > Event Management	Monitors are not propagated.
Example Criteria, Example Privileges, Example Roles	Examples	"Read only" out-of-the-box data
Languages on User and User Group objects tab	Server Settings > Preferences	NA
LDAP	Server Settings	NA
Licenses	Server Settings	NA
Lists with duplicate list entry names.	Data Settings	NA
Dynamic/Criteria-based lists.		
Logging Configuration	Web Client > Administration	Web Client--only Admin data and Monitors are not propagated.
Personal User Groups	(created by users, stored as Admin objects)	NA
Relationships on User and User Group objects	User Settings > Users, User Groups	NA
Report Templates	Web Client > Administration	NA
"(Restricted)" users	Roles > "(Restricted) ____" users are not propagated	NA
Sites	Classes > Sites base class	NA
Task Monitor	Server Settings	Monitors are not propagated.
Unit Of Measure families with duplicate UOMs	System Settings	NA
User Monitor	User Settings	Monitors are not propagated.
Visual Themes	Web Client > Administration	Web Client--only Admin data is not propagated.

ACP only propagates Administrator data from Java Client, and it does not distinguish Admin data that happened to be generated from Web Client. Therefore, users and user groups that are generated from Web Client can be propagated.

Note: **Dashboard Configuration** in Web Client corresponds to **Dashboard Management** in Java Client.

Type Filtering on Configuration Types

Note: The word “type” in the new feature “Type Filter” refers to filtering by Administrator nodes sub[ordinate]-types. It is pertinent to some Admin nodes, not all, as elaborated below. Do not confuse “type filtering” with either “configuration types” in ACP nor the “object types” in PLM.

The Type Filtering feature is introduced to permit more flexible object selection. For example, in the table below, the configuration type “Privilege Mask” can be filtered by “Privilege”. Therefore, you could specify that ACP look at only those privilege masks that are based on the Create privilege (in an instance of Agile PLM).

The enhancement is that you are no longer constrained to filtering by name alone.

The Type Filtering tag is <type name="xx"/>, where xx is the specific filter (in the previous example: *Create*).

In this table, where the possible values are too numerous to list, you can look at the appropriate node in Administrator to know exactly what the choice of specific Filters are.

Configuration Type	Type Filter	Possible Values
(ACS) Destination	Protocol	Agile, File, FTP, HTTP, HTTPS, JMS
(ACS) Event	(ACS) Event Type	Scheduled, Workflow
AutoNumber	Type	Standard, Custom
Class	Base Class	any base class, for example: Items, Changes, Customers
Subclass	Base Class or Class	any base class (see above) or class, for example,, Parts, Change Orders, customers.
Criteria	Object Type	any base class, class, or subclass
(Event Mgt.) Event	(Event Mgt.) Event Type	see Event Management > Event Types
Event Handler	Event Handler Type	Java PX, Script PX
Event Subscriber	(Event Mgt.) Event	any Event defined in Event Management > Events
Privilege Mask	Privilege	see User Settings > Privileges
Process Extension	(PX) Type	Internal Custom Action, URL
User Group	Subclass	any User Group subclass, for example,, User Group.
Workflow	Object Type = Routable Base Class	Changes, Declarations, File Folders, Packages, PSRs, Projects, QCRs, ATOs
Notification	Object Type	NA

Examples of Filtering

Example 4-1 To filter all ACS Destinations where the protocol is HTTP

```
<acs_destination>
  <include>
    <tyoe name="HTTP" />
  </include>
</acs_destination>
```

The following example filters all subclasses in the Items base class; all subclasses in the Change Orders class; and all subclasses that belong to the Suppliers class with the word "Manufacturer" in the name.

Example 4-2 To filter multiple subclasses

```
<subclass>
  <include>
    <type name="Items" />
    <type name="Change Orders" />
    <type name="suppliers">
      <pattern>.*Manufacturer.*</pattern>
    </type>
  </include>
</subclass>
```

Object Matching (Mapping)

ACP now uses a public key, API Name, where available. ACP93 has reduced the likelihood that an object's key will change by using the more static key API Name.

ACP recognizes User ID for users.

ACP still uses Name as the key in a few cases. "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).

Processing Order in ACP

The following sets of Processing Order Rules stem from these two facts about ACP:

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

ACP controls the order in which Administrator nodes are processed. This order is dictated by the dependencies between each of the Administrator nodes. ACP performs “Deletes” first, then “Renames”, then “Copies”. 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, in PLM, Users can belong to User Groups, and User Groups are populated with Users. In order for ACP to process the objects in these two configuration types, it first creates Users in the Target system, then it creates User Groups are created. Only when all “new” users and user groups have been created in the Target does ACP make the necessary associations.

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 is copied.
- Copy selects objects by name; users are selected by UserID.
- Include patterns match object names using Java regular expression rules.
- Java regular expressions have special characters that may must be encoded. Direct them where to go for the list of special characters.
- XML special characters must 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

- Rename selects objects by object key; users are selected by UserID.
- Zero or more key mappings may be specified.
- Name mappings use an exact match for both the old and new keys.
- Regular expressions are not allowed.
- XML special characters must be encoded.
- Regular expression special characters do not have to be encoded.

Delete Rules

- By default, no configuration objects for a configured configuration type will be deleted.
- Delete by object name: The delete names action requires an exact match to be deleted.
- Regular expressions are not allowed.

- XML special characters must be encoded.
- Regular expression special characters do not have 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.

Internationalization

ACP can be run once to propagate all language versions of the configuration data. This is true regardless of the language preference of the administrator who connects to the Agile instance during the ACP run. The target instance must have all of the languages enabled that are enabled in the source instance.

Supported languages are:

- English
- French
- German
- Japanese
- Simplified Chinese
- Traditional Chinese
- Russian
- Korean

Personalization

ACP can be run to propagate global configuration types. These include Tab Display Configuration, Global Views, Global Searches, out-of-box Searches, Global Reports, and Standard Reports.

The following sections provide more information about these configuration types. Some of the sections include usage examples, however, you should refer to the config.xml file that is provided out-of-box for general guidelines and examples.

Tab Display Configuration

You can use ACP to propagate tab display configuration from one environment to another. The short hand notation used in config.xml is `<tab_display_configuration/>`. For more information about the propagation actions available for this configuration type, see ["ACP Configuration Types"](#) on page A-1.

Global Views

You can use ACP to propagate all global views from one environment to another. The short hand notation used in config.xml is `<global_views/>`.

The following are examples of how to use the global views notation.

Example 4-3 To propagate all global views on the Relationships tab of Change Orders, except the one with name "Complete"

```
<copy>
  <global_views>
    <include>
      <pattern>Change Orders\Relationships\.*</pattern>
    </include>
    <exclude>
      <pattern>Change Orders.Relationships.Complete</ pattern >
    </exclude>
  </global_views>
</copy>
```

Example 4-4 To delete the global view named "Complete" on the Relationships tab of Change Orders

```
<delete>
  <global_views>
    <name>Change Orders.Relationships.Complete</name>
  </global_views>
</delete>
```

For more information about the propagation actions available for this configuration type, see ["ACP Configuration Types"](#) on page A-1.

Global Searches and Out-of-box Searches

You can use ACP to propagate all global and out-of-box searches from one system to another. Recycle Bin searches, Personal Searches and folders under personal searches cannot be propagated.

The short hand notation used in config.xml is <searches/>. The searches configuration type supports further refining using <Include>, <exclude> and type filtering. Folder names are used as filter criteria in type filtering. Any level folder names are valid for type filtering.

ACP creates folder structures in the target automatically, if they are not already available, to support the organization of searches and to make the target look like the source. For example, if the folder name specified is two levels deep under global searches, while propagating to the target, the same two-level deep folder organization is created in the target and then searches are placed inside it.

The following are examples of how to propagate searches.

Example 4-5 To propagate all global and out-of-box searches, except those that start with "All Items"

```
<searches>
  <include>
    <pattern>.*</pattern>
  </include>
  <exclude>
    <pattern> All Items.*</pattern>
  </exclude>
</searches>
```

Example 4-6 To use type filtering to limit what is propagated

```
<searches>
```

```

<include>
  <type name='Workflow Searches' />
  <type name='Item Searches'>
    <pattern>Items that.*</pattern>
  </type>
</include>
<exclude>
  <type name='Quality Seaches' />
  <type name='Item Searches'>
    <pattern>Scorpion Productline Items</pattern>
  </type>
</exclude>
</searches>

```

Example 4-7 To propagate all global and out-of-box searches

```
<searches/>
```

For more information about the propagation actions available for this configuration type, see ["ACP Configuration Types"](#) on page A-1.

Global Reports and Standard Reports

You can use ACP to propagate all global and standard reports from one system to another.

Note: Only General Info, Layout and Schedule tab information are supported for propagation. Historical Report tab is transactional data and is not supported. On Administrative Reports, only General Info is supported. Personal Reports and folders under Personal Reports are not supported.

The short hand notation used in config.xml is <reports/>. The reports configuration type supports further refining using <Include>, <exclude> and type filtering. Folder names are used as filter criteria in type filtering. Any level folder names are valid for type filtering.

ACP creates folder structures in the target automatically, if they are not already available, to support the organization of reports and to make the target look like the source. For example, if the folder name specified is two levels deep under global reports, while propagating to the target, the same two-level deep folder organization is created in the target and then reports are placed inside it.

The following are examples of how to propagate reports.

Example 4-8 To propagate all reports that have “BOM” somewhere in the name, except the “Manufacturer BOM Report”

```

<reports>
<include>
  <pattern>.* BOM .*</pattern>
</include>
<exclude>
  <pattern>Manufacturer BOM Report</pattern>
</exclude>
</reports>

```

Example 4–9 To use type filtering to limit what is propagated

```
<reports>
<include>
  <type name='Product Reports' />
  <type name='Compliance Reports'>
    <pattern>.* Compliance</pattern>
  </type>
</include>
<exclude>
  <type name='Quality Reports' />
  <type name='Compliance Reports'>
    <pattern>Part Compliance</pattern>
  </type>
</exclude>
</reports>
```

Example 4–10 To propagate the supported global and standard reports

```
<reports/>
```

For more information about the propagation actions available for this configuration type, see ["ACP Configuration Types"](#) on page A-1.

User Requirements

This chapter looks at roles and privileges that give Agile users access to ACP, at the discretion of the administrator. ACP observes all business rules around Agile PLM's roles and privileges. For more information about configuring Agile PLM roles and privileges, 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, and 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 (that is, the AppliedTo property in a user's Admin privilege is properly configured) to a given Administrator node for that node to be responsive to ACP's operations when an Admin user works in ACP.

Tailored Roles for the ACP User

Two specialized roles are 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 are 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 specify Administrator nodes. The chapter, “Privileges and Privilege Masks” in *Oracle | Agile PLM Administrator Guide*, describes the `AppliedTo` property.

Installing ACP

This chapter describes details regarding how to install ACP.

Required Information

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

Operating System

This is the operating system of the computer 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, ACP934 works with Agile PLM 9.3.4, but it does not work with, for example, Agile PLM 9.3.0.2.

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 WebLogic Application Server (WLS).

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 preexist. 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 ACP project to get you started.

Prerequisites

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

Java Runtime Environment

Java 7 is required to install and run ACP for the Agile PLM 9.3.4 release. You must download and install Java yourself.

ACP Installer

You can download the ACP Installer from the current Oracle MediaPack under the `Deployment_Tools/ACP` directory.

The version of ACP must correspond exactly to the version of Agile PLM that you are using.

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

Use the following steps to run the installer script.

1. Set JAVA_HOME

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

```
SET JAVA_HOME=<Java JRE 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 (I). 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. 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? ([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:

- 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. 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

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

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

2. Run Installation 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. 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?** ([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:

- 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. To continue, you must enter "y" as a response.

Post-Installation Tasks

The following sections describe post-installation tasks.

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.

Folder	Description
\Agile PLM	where Agile PLM is installed
\ACP	where ACP client is installed
ACP\ < version >	level for multiple instances of ACP
ACP\< version >\bin	standard installed directories
ACP\< version >\classes	
ACP\< version >\lib	
(and so on)	

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.
- **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 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"](#) on page 7-1.

Running ACP

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 appendixes.

ACP Projects

Caution: To prevent the accidental deletion of project directories, the **work directory** should be kept separate from the **install directory**. In any case, ACP Project directories should *not* be under the installed ACP path.

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:

Folder	Description
\AgilePLM	Agile PLM home- A directory where AgilePLM is installed
\AgilePLM\ACP	ACP home - A directory where ACP client is installed (ACP Project directories should <i>not</i> be under the installed ACP path). Note: ACP home can be a separate directory path and does not have to be under Agile PLM Home.
\AgilePLM\ACP Projects	ACP work directory where ACP projects are organized. Note: This directory should not be under the installed ACP home.
\ACPProjects\Project_934	Directory that collects ACP projects oriented to PLM Release 9.3.4.
\ACPProjects\Project_934\Roles_Privileges	Directory that contains Agile PLM Release 9.3.4 Admin Roles and Privileges specific propagation (import/export/deep compare) files such as: config.xml, project.properties, acp.bat, export_xxx.agl and logs.
\ACPProjects\Project_934\Lists	Directory that contains a control file used to propagate Admin lists
\ACPProjects\Project_934\934_Classes	Directory that contains a control file used to propagate Admin classes in PLM Release 9.3.4
\ACPProjects\Project_934\934_Subclasses	Directory that contains a control file used to propagate Admin subclasses in PLM Release 9.3.4
\ACPProjects\Project_934\New Feature_01	Directory that contains a control file used to propagate a specific new feature
\ACPProjects\Project_934\New Feature_02	Directory that contains a 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_project script provided by ACP.

Existing Project

This method enables 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 want 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.bat`), the project properties file (`project.properties`) and the control file (`config.xml`).

New Project

This method enables 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\ACPWork934`
- UNIX: `cd /<user home>/agile/acpwork934`

Of course you will tailor any numbered directory folders to match the specific PLM version, for example, "93", "9301", "931", 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, and common properties files. ACP properties are described in greater detail in "[Properties](#)" on page C-1. For now, you should understand how to configure connection information for the Agile PLM instances you want 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, Quality Analysis, 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, Acme is 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.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	(optional) The name of the administrator user to connect to the Agile PLM instance with.
xml	The name of the ACP archive created when exporting data from this Agile PLM instance.

Refer to ["Properties"](#) on page C-1 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>** The Copy section tells ACP which objects should be created or updated.
2. **<rename>** The Rename section tells ACP which objects whose keys must be renamed in the target.
3. **<delete>** The Delete section tells ACP which objects should be deleted from the target.
4. **<ignore_references>** The Ignore References section tells ACP which object references can be ignored if they cannot be resolved in the target instance.
5. **<subobject_maps>** The Subobject Maps section helps ACP map subobject keys between the source and target.

["Configuring the ACP Control File"](#) on page 8-1 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 regarding certain configuration types. These are also covered in the next chapter, in ["Business Logic Attributes in the Control File"](#) on page 8-4.

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 must 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.
 - Windows: `cd /d D:\Agile\AcpWork`
 - UNIX: `cd /export/home/joeuser/agile/acpwork`
2. Set the ACP_JAVA_HOME environment variable to where the JRE 1.6 has been installed.
 - Windows: `Set ACP_JAVA_HOME = c:\jre`
 - UNIX: `ACP_JAVA_HOME = /u01/app/jre`
3. Launch the desired script. The examples direct ACP to export the Dev instance and to import the Dev instance to the Prod instance:
 - `acp export dev`
 - `acp import dev prod`

Refer to ["ACP Scripts"](#) on page D-1 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 E-1 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 F-1 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 that are installed with ACP.

- Use ACP 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.

Configuring the ACP Control File

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 Appendix A about the ACP Configuration Types and Appendix B, “Java Regular Expressions.”

The ACP Control File is how you tell ACP what to propagate. ACP knows how to propagate objects and which order objects must 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. Familiarize yourself with these basic constructs before trying to make changes to the control file.

Element

XML elements have content (that is, data). This content can be:

- empty,
- simple content (such as text),
- element content, or
- mixed content.

Elements may also have attributes. Attributes are defined as part of the start tag.

All elements have a start tag and an end tag.

The element's content is everything specified from (and including) the element's start tag to (and 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.)

Here are some examples of XML 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 Element

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 Element

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 Element

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.

Element Attributes

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

Comments in XML

XML allows for comments. You can add comments to the control file that can help during reuse.

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 (`&`) encodes 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
<code>&</code>	<code>&amp;</code>
<code><</code>	<code>&lt;</code>
<code>></code>	<code>&gt;</code>
<code>"</code>	<code>&quot;</code>

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 regarding 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 A-1.

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.

Attribute name	Attribute 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

<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.

Attribute name	Attribute tag	What it does	Values (default value is in Bold font)
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> attribute is available to 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 enables

you to force in-use criteria to be updated. This setting is applied to all criteria processed.

Attribute name	Attribute tag	What it does	Values (default value is in Bold font)
Criteria Force Update	criteria_force_update	Enables you to choose whether in-use criteria can be updated.	No. ACP does not allow an in-use criteria to be updated.

Autonumber Force Update

<auto_number_force_update> attribute is available to the AutoNumber configuration type.

This attribute helps to control accidental updating of your company's autonumber sequences. When end-users unintentionally “upset” the autonumber sequence, and if the administrator attempts to revert the numbering, PLM may be subject to errors.

As understood by ACP, autonumbers in PLM now have two sets of properties.

- “First set” of properties are updated on a regular propagation - Name, Description, API Name
- “Second set” of properties can be controlled by Autonumber Force Update - Prefix, Suffix, Character Set, Number of Characters, Starting Number

When the “Yes” tag is used, both the first and second sets of properties are updated. When the “No” tag is used, only the first set of properties is updated. The default is set to No.

Attribute name	Attribute tag	What it does
AutoNumber Force Update	auto_number_force_update	Controls accidental updating of autonumber sequences.

Force Delete List Entry

<force_delete_list_entry> attribute is available to the List configuration type.

This parameter 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 (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 gets 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.

Attribute name	Attribute tag	What it does	Values (default value is in Bold font)
Force Delete List Entry	Determines whether a list entry is to be deleted or disabled	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

<new_user_password> attribute is available to 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 password must meet the Account Policy settings in Administrator. The <new user password> attribute enables you to choose the password that ACP assigns to new users. This setting is applied to all new users created.

Attribute name	Attribute tag	What it does	Value
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.	agile

Process Extension Association Rule

<px_association_rule> attribute is available to 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.

Attribute name	Attribute tag	What it does	Values (default value is in Bold font)
Process Extension Association Rule	base_class px_ association_rule; class px_ association_rule; subclass px_ association_rule;	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

<user_association_rule> attribute is available to 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.

Attribute name	Attribute tag	What it does	Values (default value is in Bold font)
User Association Rule	role user_ association_rule; user_group user_ association_rule	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.

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 and instructions 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 regarding certain configuration types. This is covered in ["Business Logic Attributes in the Control File"](#) on page 8-4

Copy (<copy>) Section

The Copy section is marked by the <copy> element. Both Copy and Compare use the Copy section to determine which objects to process.

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 tell ACP which configuration types to process. Within each configuration type, you 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 A-1. In the Copy section of the control file, objects to copy can be marked for inclusion or exclusion through regular expressions.

Note: The Copy section expects the object name, not the object key.

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 expressed as a regular expression that matches to itself only.

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 B-1 for more detailed information about regular expressions and their special characters.

Putting it all together in Copy section

Let's examine a short example of the Copy section with some explanation of how ACP interprets what was specified.

Example 8-1 Copy section

```
<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 & 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:** This example explicitly uses 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, there are multiple patterns listed. 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 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. This example asks 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 “actualizations”. 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. This example omits 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 key (API Name, Name, or User ID) to map it to an object in the Target instance.

If the key 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 of the Control File resolves this dilemma. You can use the Rename section to “re-key” (rename the key) objects in the Target instance.

ACP processes the Rename section of the Control File before it processes the Copy section. This allows Copy to resolve any references to objects whose key has changed. Renames are processed only 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 A-1.

In the Rename section of the Control File, objects to be renamed are specified by maps. An error is reported if the object being renamed does not exist in the Target instance.

Key Maps

A key map maps an object by its key (API Name, Name, or User ID) in the Source instance to an object's key in the Target instance.

In order for the map to have meaning, these two conditions should be met:

- an object should exist in the Source instance with the map's Source key, and
- an object should exist in the Target instance with the map's Target key.

Source Key

The Source key is the API Name/Name/User ID of the object being mapped in the Source. Since this is part of a key map, the exact API Name/Name/User ID of the object must be specified.

Target Key

The Target key is the API Name/Name/User ID of the object being mapped in the Target. Since this is part of a key map, the exact API Name/Name/User ID of the object must be specified.

Putting it all together in Rename section

Let's examine a short example of the Rename section with some explanation of how ACP interprets what was specified.

Example 8-2 Rename section

```
<rename>
  <list>
    <map>
      <source_apiname>Countries</source_apiname>
      <target_apiname>Country</target_apiname>
    </map>
    <map>
      <source_apiname>My Category</source_apiname>
      <target_apiname>Category 9 List</target_apiname>
    </map>
  </list>
  <privilege>
    <map>
      <source_apiname>(Propagation) Configuration Administrator
      </source_apiname>
      <target_apiname>(Propagation) Administrator</target_apiname>
    </map>
    <map>
      <source_apiname>Modify My Projects &
      Gates</source_apiname>
      <target_apiname>Modify Projects & Gates that I am owner of
      </target_apiname>
    </map>
    <map>
      <source_apiname>Administrator Privilege</source_apiname>
      <target_apiname>Admin. Privilege</target_apiname>
    </map>
  </privilege>
  <subclass>
    <map>
      <source_apiname>My ECO</source_apiname>
      <target_apiname>ECO</target_apiname>
    </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"; 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 Projects & Gates that I am owner of " to "Modify My Projects & Gates".
 - Rename the privilege "Admin. Privilege" to " Administrator Privilege". 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 enables you to propagate delete 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 A-1. In the Delete section, objects to be deleted are specified as a list of names.

Name

The Delete section expects object names, not object keys.

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 us examine a short example of the Delete section with some explanation of how ACP interprets what was specified.

Example 8–3 Delete section

```
<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 & 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 is marked by the <ignore_references> element.

The Ignore References section does not propagate any information. Since data may be propagated from a Development instance to a Production instance, there may be test data that does not get propagated. However, there may be configuration data that does get propagated but 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 is 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 enables 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.

The Ignore Reference section expects object names, not object keys.

Configuration Types in Ignore References Section

The “ignorable” references are limited to configuration types of those Administrator nodes where test objects are likely to be created:

- Users
- User groups
- Roles
- Privileges

In the Ignore References section of the Control File, 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 object names 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. See ["Java Regular Expressions"](#) on page B-1 for more detailed information about regular expressions and their special characters.

Putting it all together in Ignore References section

Let us examine a short example of the Ignore References section with some explanation of how ACP interprets what was specified. The Ignore References section of the Control File is honored by Copy and Deep Compare.

Example 8–4 Ignore References section

```
<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” and 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 key (API Name/Name/User ID) to map an object between the source and the target, ACP uses a subobject's key to map subobjects between the source and target. If the key 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 keys 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

The configuration types that support subobject maps are: <base_class>, <class>, <subclass>, <list>, <workflow>, <unit_of_measure_family>, and <column_assignments>.

In the Subobject Maps section of the Control File, subobjects whose names have changed are documented by maps. Some configuration types may have more than one subobject type. For example, <class> has LifeCycle Phases and Attributes. A table that

lists the corresponding subobject maps and the methods for renaming can be found in ["Renaming Subobjects"](#) on page A-8.

Object Reference

As the name subobject implies, a subobject is subordinate to an object. Therefore, in order for ACP to match by the subobject key, it must also have the name of the object that the subobject belongs to. What if the object name and the subobject key are changing at the same time? Since ACP dictates the processing order that configuration types are copied, the referenced object name is assumed to have been changed in the target already. Therefore, the subobject 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.

Flex Attribute Rename

A Subobject Map has been introduced that allows ACP to rename user-defined flex-field attributes. The XML tags are given below:

Example 8-5 Class

```
<classes>
  <class name="Items">
    <attributes>
      <map>
        <source_apiname>Mass</source_apiname>
        <target_apiname>Flex01</target_apiname>
      </map>
    </attributes>
  </class>
</classes>
```

Example 8-6 Subclass

```
<subclasses>
  <subclass name="Part">
    <attributes>
      <map>
        <source_apiname>Weight</source_apiname>
        <target_apiname>Flex02</target_apiname>
      </map>
    </attributes>
  </subclass>
</subclasses>
```

Key Maps

A key map maps a subobject key (API Name/Name) in the Source instance to a subobject key in the Target instance.

Source Key

The Source key is the API Name/Name of the subobject being mapped in the Source. Since this is part of a key map, the exact API Name/Name of the subobject must be specified.

Target Key

The Target key is the API Name/Name of the subobject being mapped in the Target. Since this is part of a key map, the exact API Name/Name of the subobject must be specified.

Putting it all together in Subobject Maps section

Let us examine a short example of the Subobject Maps section with some explanation of how ACP interprets what was specified. The Subobject Maps section of the Control File is honored by Copy and Deep Compare. Subobject maps only come into play when the key of the object matches the object key being Copied or Deep-compared.

Example 8-7 Subobject Maps section

```
<subobject_maps>
  <list apiname="ReasonCode">
    <list_entries>
      <map>
        <source_apiname>Enhancement</source_apiname>
        <target_apiname>Product Enhancement</target_apiname>
      </map>
      <map>
        <source_apiname>Improvement</source_apiname>
        <target_apiname>Reliability Improvement</target_apiname>
      </map>
    </list_entries>
  </list name>
  <list name="Location||Europe">
    <list_entries>
      <map>
        <source_apiname>Czech Republic</source_apiname>
        <target_apiname>Czechoslovakia</target_apiname>
      </map>
    </list_entries>
  </list name>
</lists>
<subclasses>
  <subclass apiname="Customer">
    <life_cycle_phases>
      <map>
        <source_apiname>Current</source_apiname>
        <target_apiname>Active</target_apiname>
      </map>
    </life_cycle_phases>
    <attributes>
      <map>
        <source_apiname>Customer Details</source_apiname>
        <target_apiname>Page Three</target_apiname>
      </map>
    </user_interface_tabs>
  </subclass>
  <subclass apiname="ECR">
    <attributes>
      <map>
        <source_apiname>Page Three||Origination Date
        </source_apiname>
        <target_apiname>Page Three||Flex Date01</target_apiname>
      </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 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 is 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 list entry is 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 life cycle phase will be renamed to “Current”. In addition, the user interface tab “Customer Details” is mapped to “Page Three”. The user interface page is renamed to “Customer Details”.
- **Subclass (ECR):** The user interface tab “ECR Details” is mapped to “Page Three”. The user interface page is renamed to “ECR Details”.

ACP Configuration Types

The following are supported ACP configuration types used to propagate data.

Supported ACP Configuration Types

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 list of ACP-supported configuration types and the propagation actions that are supported by each type. The corresponding administrator nodes are located in the Java Client.

Table A-1 Administrator > Data Settings > Classes

Configuration Types	Delete	Rename	Copy-Create	Copy-Update	Copy-Delete
<base_class>	NA	API Name is used as the key, so Base Class and Class cannot be renamed.	NA	X	NA
<class>	NA	API Name is used as the key, so Base Class and Class cannot be renamed.	NA	X	NA
<subclass>	X	X	X	X	

Table A-2 Administrator > Data Settings (except Classes) .

Configuration Types	Delete	Rename	Copy-Create	Copy-Update	Copy-Delete
<<character_set> (Exception: out-of-box character sets are not propagated.)	X	X	X	X	NA
<list>	X	X	X	X	X
<process_extension>	X	X	X	X	NA

Table A–2 (Cont.) Administrator > Data Settings (except Classes) .

Configuration Types	Delete	Rename	Copy-Create	Copy-Update	Copy-Delete
<auto_number>	X	X	X	X	X
<criteria>	X	X	X	X	NA

Table A–3 Administrator > Workflow Settings

Configuration Types	Delete	Rename	Copy-Create	Copy-Update	Copy-Delete
<workflow>	X	X	X	X	NA

Table A–4 Administrator > User Settings

Configuration Types	Delete	Rename	Copy-Create	Copy-Update	Copy-Delete
<account_policy>	NA	NA	NA	X	NA
<user>	X	X	X	X	NA
<user_group>	X	X	X	X	NA
<supplier_group>	X	X	X	X	NA
<role>	X	X	X	X	NA
<privilege>	X	X	X	X	X

Table A–5 Administrator > System Settings

Configuration Types	Delete	Rename	Copy-Create	Copy-Update	Copy-Delete
<smart_rule>	NA	NA	NA	X	NA
<viewers_and_files>	NA	NA	NA	X	NA
<notification_template>	X	X	X	X	NA
<full_text_search>	NA	NA	NA	X	NA
<column_assignments> (My Assignments)	NA	NA	NA	X	NA
<unit_of_measure_family> (UOM)	X	X	X	X	NA
<company_profile>	NA	NA	NA	X	NA
<ppm_dashboard_management>	X	X	X	X	NA

Table A-6 Administrator > System Settings > Product Cost Management

Configuration Type	Delete	Rename	Copy>Create	Copy-Update	Copy-Delete
<pcm_ship_to_location>	X	X	X	X	NA
<pcm_rfq_terms_and_conditions>	NA	NA	X	X	NA

Table A-7 Administrator > System Settings > Product Portfolio Management

Configuration Types	Delete	Rename	Copy>Create	Copy-Update	Copy-Delete
<ppm_cost_status>	X	X	NA	X	NA
<ppm_quality_status>	X	X	NA	X	NA
<ppm_resource_status>	X	X	NA	X	NA
<ppm_schedule_status>	X	X	X	X	NA
<ppm_default_role>	NA	X	NA	X	NA

Table A-8 Administrator > System Settings > Agile Content Service

Configuration Types	Delete	Rename	Copy>Create	Copy-Update	Copy-Delete
<acs_subscriber>	X	X	X	X	NA
<acs_destination>	X	X	X	X	X
<acs_event>	X	X	X	X	X
<acs_filter>	X	X	X	X	X
<acs_format_library>	X	X	X	X	NA
<acs_package_service>	X	X	X	X	NA
<acs_response_service>	X	X	X	X	NA

Table A-9 Administrator > System Settings > Product Governance & Compliance

Configuration Types	Delete	Rename	Copy>Create	Copy-Update	Copy-Delete
<pgc_signoff_message>	NA	NA	NA	X	NA

Table A–9 (Cont.) Administrator > System Settings > Product Governance & Compliance

Configuration Types	Delete	Rename	Copy-Create	Copy-Update	Copy-Delete
<pgc_compliance_rollup_scheduling>	NA	NA	NA	X	NA
<pgc_compliance_rollup_rule_setting>	NA	NA	NA	X	NA
<pgc_external_rollup_rule_setting>- in the External Rollup node folder	NA	NA	NA	X	NA
<pgc_external_rollup_scheduling> - in the External Rollup node folder	NA	NA	NA	X	NA
<pgc_supplier_declaration> - this config. type contains all data in the Supplier Declaration Process Extensions node folder	NA	NA	NA	X	NA

Table A–10 Administrator > System Settings > Event Management

Configuration Types	Delete	Rename	Copy-Create	Copy-Update	Copy-Delete
<em_event>	X	X	X	X	X
<em_event_handler>	X	X	X	X	X
<em_event_subscriber>	X	X	X	X	NA
<em_event_type>	NA	NA	NA	X	NA
<em_event_handler_type>	NA	NA	NA	X	NA

Table A–11 Administrator > System Settings > Reference Object Management

Configuration Type	Delete	Rename	Copy-Create	Copy Update	Copy-Delete
<reference_object_application>	X	X	X	X	NA

Table A-12 Administrator > Server Settings

Configuration Types	Delete	Rename	Copy>Create	Copy-Update	Copy-Delete
<server_location> (Locations node)	NA	NA	NA	X	NA
<server_file_manager> (Locations > File Manager tab)	X	X	X	X	NA
<server_database> (Database node)	NA	NA	NA	X	NA
<server_preference> (Preferences node)	NA	NA	NA	X	NA
<server_task> (Task Configuration node)	NA	NA	NA	X	NA

Table A-13 Administrator > Search

Configuration Types	Delete	Rename	Copy>Create	Copy-Update	Copy-Delete
<searches>	X	NA	X	X	NA

Table A-14 Administrator > Analytics and Reports

Configuration Types	Delete	Rename	Copy>Create	Copy-Update	Copy-Delete
<reports>	X	NA	X	X	NA

The following tables contain the configuration types that have corresponding administrator nodes in Web Client.

Table A-15 Tools and Settings > Administration

Configuration Types	Delete	Rename	Copy>Create	Copy-Update	Copy-Delete
<tab_display_configuration>	X	X	X	X	NA

Table A-16 Table View Personalization

Configuration Types	Delete	Rename	Copy>Create	Copy-Update	Copy-Delete
<global_views>	X	NA	X	X	NA

The following sections provide additional data regarding configuration types, match keys, subobjects, and deep compare.

Configuration Types and Match Keys

This table lists the configuration types and whether the match key is the key used to match objects between Agile instances. In some cases, the key value can be changed, but generally the key is intended to be static.

Configuration Type	Match Key
Classes	
<base_class>	API Name
<class>	API Name
<subclass>	API Name
Data Settings	
<auto_number>	API Name
<character_set>	API Name
<criteria>	API Name
<list>	API Name
<process_extension>	API Name
Workflow Settings	
<workflow>	API Name
User Settings	
<account_policy>	API Name
<privilege>	API Name
<role>	API Name
<supplier_group>	Name
<user>	User ID
<user_group>	Name
System Settings	
<company_profile>	API Name
<ppm_dashboard_management>	API Name
<full_text_search>	API Name
<column_assignments> (My Assignments)	API Name
<notification_template>	API Name
<smart_rules>	API Name
<unit_of_measure_family>	API Name
<viewers_and_files>	API Name
Product Cost Management	
<pcm_ship_to_location>	Name

Configuration Type	Match Key
<pcm_rfq_terms_and_conditions>	API Name
Product Portfolio Management	
<ppm_cost_status>	API Name
<ppm_quality_status>	API Name
<ppm_resource_status>	API Name
<ppm_schedule_status>	API Name
<ppm_default_role>	API Name
Agile Content Service	
<acs_destination>	API Name
<acs_event>	API Name
<acs_filters>	API Name
<acs_package_service>	API Name
<acs_response_service>	API Name
<acs_subscriber>	API Name
Product Governance & Compliance	
<pgc_signoff_message>	API Name
<pgc_compliance_rollup_scheduling>	API Name
<pgc_compliance_rollup_rule_setting>	API Name
<pgc_supplier_declaration>	API Name
Event Management	
<em_event>	API Name
<em_event_handler>	API Name
<em_event_subscriber>	API Name
<em_event_type>	API Name
<em_event_handler_type>	API Name
Server Settings	
<server_location>	API Name
<server_file_manager>	API Name
<server_database>	API Name
<server_preference>	API Name

Configuration Type	Match Key
<server_task>	API Name

Renaming Subobjects

Additionally, some Administrator objects have subobjects that can be renamed. To update the subobject correctly, ACP requires a key map for these subobjects.

Without the map, ACP considers these subobjects to be different: the subobject with the old key is deleted and the subobject with the new key is created.

Configuration Type	Subobject Type	Method for Rename
<base class>	LifeCycle Phases	Subobject Map (API Name)
<class>	Attributes	Subobject Map (API Name)
<class>	LifeCycle Phases	Subobject Map (API Name)
<subclass>	Attributes	Subobject Map (API Name)
<subclass>	LifeCycle Phases	Subobject Map (API Name)
<list>	List Entries	Subobject Map (API Name)
<workflow>	Workflow Statuses	Subobject Map (API Name)
<unit_of_measure_family> (UOM)	UOMs	Subobject Map (API Name)
<column_assignments> (My Assignments)	Column	Subobject Map (Name)

Configuration Types as Evaluated by Deep Compare

This table indicates how Deep Compare (the script and the report) evaluates the configuration types tab by tab and exactly what Deep Compare distinguishes about each configuration type. As explained in the “Compare Details” heading, attributes on the **General Information** tab of all configuration types are considered by Deep Compare, while **Where Used** and **History** tabs are not compared.

Compare Details	
Deep Compare-Supported Configuration Types	Deep Compare will take care of Rename / Subobject Mapping / Ignore References. History tabs are not considered for comparison.
<account_policy>	All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report. Each attribute of this configuration is treated as an individual object.
<acs_destination>	General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.
<acs_event>	General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.

Compare Details	
Deep Compare-Supported Configuration Types	Deep Compare will take care of Rename / Subobject Mapping / Ignore References. History tabs are not considered for comparison.
<acs_filter>	<p>General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.</p> <p>Filter tab: All attributes of the Source object are compared with all attributes of the Target object, if Source data does not match Target data, these differences are recorded in the Deep Compare report.</p>
<acs_package_service>	<p>General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.</p>
<acs_response_service>	<p>General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.</p>
<acs_subscriber>	<p>General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.</p> <p>Subscriber Details tab: Subscribers Details will be compared based on the Destination(s) match. If source Destination(s) matching with target Destination(s), then other properties, for example, Filter, Roles(s), Format, Language, and Site are compared; otherwise comparison are ignored. However, the report clearly tells that which is a source only object and target only object.</p>
<auto_number>	<p>General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.</p> <p>Where Used tab: Deep Compare has entries for each Subclass if it does not exist in the source or target.</p>
<base_class>	<p>General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.</p> <p>Life Cycle Phases tab: Life Cycle Name (primary key) is the key value to compare other Life Cycle Phases. All differences are shown in the report including Source Only and Target Only lifecycle names.</p> <p>Process Extensions tab: All names of source process extension (primary key) are the key values to compare all names of target process extension.</p> <p>Classes and Event Subscribers tabs: These tabs are not considered for comparison.</p>
<character_set>	<p>General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.</p>

Deep Compare-Supported Configuration Types	Compare Details
	Deep Compare will take care of Rename / Subobject Mapping / Ignore References. History tabs are not considered for comparison.
<class>	<p>General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.</p> <p>Life Cycle Phases tab: Life Cycle Name (primary key) is the key value to compare other Life Cycle Phases. All differences are shown in the report, including Source Only and Target Only lifecycle names.</p> <p>Process Extensions tab: All names of source process extension (primary key) are the key values to compare all names of target process extension.</p> <p>Attributes <User-Interface Tabs> tab: Each attribute of a source object is compared to each attribute of the target object by using attribute name. If a match is found, then all properties of that attribute will be compared and reported to Deep Compare if differences are found.</p> <p>If source flex attribute does not exist in the target instance or vice versa, then such difference will be noticed in Deep Compare. However, vice versa of this case is not reported.</p> <p>Only attribute name (regardless of properties change) of source and target will be reported in Deep Compare if you rename out-of-the-box attribute at target/source. However if you use Subobject Map for Attributes, then all properties of that attribute will be compared and reported same if difference found.</p> <p>Classes tab, Event Subscribers tab, and Attachments tab are not considered for comparison purposes.</p>
<company_profile>	<p>General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.</p>
<criteria>	<p>General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.</p> <p>Criteria tab: It compares at Criteria Level. If criteria attribute of source and target names are found to be same, then Adv Compare will try to find out any other (Join Operator, Relational Operator and Value) differences in the same level. If there are differences that will be shown in the report otherwise it is not shown in the report.</p> <p>However if Attributes names are different at source and target, then SOURCE ONLY and TARGET ONLY columns will have an entry with reported values.</p> <p>Where Used tab is not considered for comparison.</p>
<em_event>	<p>General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.</p> <p>Where Used tab is not considered for comparison.</p>

Deep Compare-Supported Configuration Types	Compare Details Deep Compare will take care of Rename / Subobject Mapping / Ignore References. History tabs are not considered for comparison.
<em_event_handler>	<p>General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.</p> <p>Where Used tab is not considered for comparison.</p>
<em_event_subscriber>	<p>General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.</p> <p>Where Used tab is not considered for comparison.</p>
<em_event_type>	<p>General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.</p> <p>Where Used tab is not considered for comparison.</p>
<em_event_handler_type>	<p>General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.</p> <p>Where Used tab is not considered for comparison.</p>
<full_text_search>	<p>General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.</p>
<list>	<p>General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.</p> <p>List tab:</p> <p><i>Case 1:</i> List entries of Source or Target are Empty. Deep Compare will show the List Entries/<Empty> of source/target by using delimiters. Delimiters are:</p> <p> ("pipe" character) - Separates list entries until leaf node, for example, a b c</p> <p>;(semicolon) - Separates set of list entries separated by pipe, for example, a b c;d e f</p> <p><i>Case 2:</i> List of Source and Target is cascade list. Each list entry of Source is compared with Each list entry of Target by considering the child -parent relationship hierarchy. Not only used List-Name for comparison but also used parent-child relationship hierarchy irrespective of same name existence across List unless sub objects map specification. If source has list entry A and target does not have the same entry, Deep Compare will specify "Target Only"/"Source Only" in Status column.</p> <p><i>Case 3:</i> List of Source and Target is not cascade list. Same as <i>Case 2</i> except child-parent relationship hierarchy.</p> <p><i>Case 4:</i> List of Source or Target is cascade list. If any one of Source and Target has cascade list and other has non-cascade list, Deep Compare simply names what are cascade and non-cascade lists (does not show any list entries). The difference column will NOT show a comparison.</p>

Deep Compare-Supported Configuration Types	Compare Details Deep Compare will take care of Rename / Subobject Mapping / Ignore References. History tabs are not considered for comparison.
<my_assignments> or <column_assignments>	All attributes of the Source instance are compared to all attributes of Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.
<notification_template>	General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report. Where Used tab is not considered for comparison.
<pcm_rfq_terms_and_conditions>	Text of RFQ Terms and Conditions will be compared.
<pcm_ship_to_location>	General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.
<pgc_compliance_rollup_rule_setting>	General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.
<pgc_sign_off_message>	General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.
<pgc_supplier_declaration>	It compares selected process extensions of source and target.
<ppm_cost_status>	General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.
<ppm_dashboard_management>	General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report. Tables tab: Deep Compare will try to find a source attribute in target attributes list, if there is a difference between them, then that property will be noticed in report.
<ppm_default_role>	General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.
<ppm_quality_status>	General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.
<ppm_resource_status>	General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.

Deep Compare-Supported Configuration Types	Compare Details Deep Compare will take care of Rename / Subobject Mapping / Ignore References. History tabs are not considered for comparison.
<ppm_schedule_status>	General Info tab: Over Due Type will be compared. Status tab => General Information tab: Same as above description.
<ppm_ui_configuration_data>	Content: Attribute Groups: It compares only Selected Attributes and Attribute Groups. Tables: It compares only Selected Columns and tables. Action Groups: It compares only Selected Actions and Action Groups. Layout: It compares only Form and Defined Layouts.
<privilege>	General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report. Where Used tab is also considered for comparison.
<process_extension>	General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report. Where Used tab is also considered for comparison.
<role>	General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report. Privileges tab: Compares all names of source Privileges to the all names of target Privileges, Deep compare will make an entry if there are intersected values are present. Users tab: Compares all names of source Users to the all names of target Users, Deep compare will make an entry if there are intersected values are present. User Groups tab: Compares all names of source User-Groups to the all names of target User-Groups, Deep compare will make an entry if there are intersected values are present.
<server_database>	General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.
<server_file_manager>	File Manager tab: Same as General Information tab. Vault Configuration: Deep compare will make an entry if the combination of Vault Type, Description, Base Storage Dir, Purge Dir, and Category gets changed at source or target.
<server_location>	General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.
<server_preference>	General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.

Compare Details	
Deep Compare-Supported Configuration Types	Deep Compare will take care of Rename / Subobject Mapping / Ignore References. History tabs are not considered for comparison.
<server_task>	Task Configuration tab (same as General Information tab): All attributes of the Source instance are compared to all attributes of Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.
<smart_rule>	General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.
<subclass>	Same as Class configuration.
<supplier_group>	General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.
<unit_of_measure_family>	General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report. UOM: It will compare each entry of Source UOM to the each entry of target UOM.
<user>	General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report. Preferences tab: Same as General Information tab. Escalations tab: Deep Compare will give only intersected CRITERIA name and NOTIFY USERS separated by ":" in the respective columns of the report as separate entries (Source Only and Target Only). However if the CRITERIA name is same at Source and Target but the NOTIFY USERS name(s) is (are) changed in either, then Deep Compare will treat them as a DIFFERENCE value, hence only one entry will be allocated to show the both (source and target) values. User Groups tab: Deep Compare will give only intersected USER GROUP name and STATUS separated by ":" in the respective columns of the report as separate entries (Source Only and Target Only). However if the USER GROUP name is the same at the Source and Target but the STATUS name is changed in either, then Deep Compare will treat them as a DIFFERENCE value, hence only one entry will be allocated to show the both (source and target) values Share, Relationships, Subscription and Attachments tabs: These tabs are not considered for comparison.

Compare Details	
Deep Compare-Supported Configuration Types	Deep Compare will take care of Rename / Subobject Mapping / Ignore References. History tabs are not considered for comparison.
<user_group>	<p>General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.</p> <p>Escalations tab: Deep Compare will give only intersected CRITERIA name and NOTIFY USERS separated by":" in the respective columns of the report as separate entries (Source Only and Target Only). However if the CRITERIA name is same at Source and Target but the NOTIFY USERS name(s) is (are)changed in either, then Deep Compare will treat them as a DIFFERENCE value, hence only one entry will be allocated to show the both (source and target)values</p> <p>Users tab: Deep Compare will give only intersected USER name and ROLES separated by":" in the respective columns of the report as separate entries (Source Only and Target Only). However if the USER name is same at Source and Target but the ROLES name is changed in either, then Deep Compare will treat them as a DIFFERENCE value, hence only one entry will be allocated to show the both (source and target)values.</p> <p>Share, Relationships and Attachments tabs: These tabs are not considered for comparison.</p>
<viewers_and_files>	<p>General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.</p> <p>Banners and File Association tabs: Compares all names of Source file types to all names of the Target file types. Deep Compare creates an entry if at least one name has been changed at Source or Target. It also compares selected autogenerated Thumbnails.</p> <p>Watermark tab: compares each Watermark of Source to each Watermark of target.</p>
<workflow>	<p>General Information tab: All attributes of the Source instance are compared to all attributes of the Target instance; if Source data does not match Target data, these differences are recorded in the Deep Compare report.</p> <p>Status tab: compares all statuses and its associated Criteria.</p>

Log and Report Files for Deep Compare

- **Verbose Log:** This log is really not necessary as Deep Compare does not do any update or create actions on Agile instances. This file is empty.
- **Error Log:** This log file is created in the current project directory with a default file name of deep_compare.err.
- **Console (stdout) Log:** The following message displays during execution of Deep Compare command.

ACP DeepCompare SUCCEEDED - Completed with no errors or notes.

=====DEEP-COMPARE REPORT=====

Report Generation in Progress, Please Wait...

Report Generated Successfully...

Please Check The Report @ D:\ACPWorkDir9.3.4\project2\deep_compare.xls

Error Level = 0

- **Process Log:** This log file is created in the current project directory with a default file name of `deep_compare.log`.
- **Deep Compare report:** The Excel report file is created in the current project directory with a default file name of `deep_compare.xls`.

Regular Expressions

This appendix describes regular expressions. You can use Java regular expressions to select ACP configuration types or other Administrator objects to propagate (object names are also valid expressions).

Java Regular 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. 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
&	&
<	<
>	>
"	"

Java Regular Expression Special Characters

For a comprehensive reference to Java regular expressions, refer to Oracle's website (<http://docs.oracle.com/javase/7/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 Manufacturing 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]

Properties

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 enables 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 through 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 through 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 = 20090603_142442
log.process.name = ${acp.method}_${acp.run.datetime}.log

result: log.process.name = export_20090603_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 keep things simple, ACP enables 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 by 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	<p>The method (export, import, name_compare, deep_compare) that ACP will use. The method determines the type of the source and target connections.</p> <p>Possible Values:</p> <p>export - source is Agile; target is XML</p> <p>import - source is XML; target is Agile</p> <p>name_compare and deep_compare - source is XML; target is Agile</p>
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	<p>Indicates if stack traces should be included with error messages. Stack traces are useful for Agile Customer Care to solve issues.</p> <p>Possible Values:</p> <p>false - stack traces are included (default)</p> <p>true - stack traces are not included</p>

Property Name	Description
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. ACP uses the nickname to qualify a connection property (use <nickname>.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	(optional) 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>.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.username=propagation
dev.xml=export_dev.agl
```

```
prod.name=Acme Production
prod.url=http://agileplm-prod.acme.com:7777/Agile
prod.xml=export_prod.agl
```

ACP Scripts

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 7. You are required to install Java 7. You can use the ACP_JAVA_HOME environment variable to tell ACP where the JRE is installed. This enables you to use the JAVA_HOME environment variable for other uses.

Windows: set ACP_JAVA_HOME=<Java 7 JRE/JDK Directory>

UNIX: export ACP_JAVA_HOME=<Java 7 JRE/JDK Directory>

For convenience, you can add this environment variable to 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.

ACP Launcher

For convenience, a special script is installed to the project directory. This script is named acp[.bat]. It enables 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>]

Propagation Scripts

ACP provides scripts for propagating data from one Agile PLM instance to another Agile PLM instance through 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, and also know 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`

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: `version`

Sample Output:

```
Agile(TM) ACP CopyConfig (Version ACP 9.3.4 (Build 63))
```

```
ACP Version: ACP 9.3.4 (Build 63)
```

```
Required Agile PLM Version: 9.3.4 (Build 63)
```

Project Management Script

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\ACPWork`
 - UNIX: `cd /<user home>/agile/acpwork`
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>]`

Object Name Comparison Script

Name Compare provides a script to compare object names. This script is designed to help you ensure your Control File has the proper Rename maps for object names.

Note: The `name_compare` script compares object names only. Use the `deep_compare` script to perform a detailed object comparison.

name_compare

The `name_compare` 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 name_compare <source nickname> <target nickname> [-debug]`
- Example: `acp name_compare dev prod`

Object Detail Comparison Script

Deep Compare provides a detailed comparison of an object's definition or configuration.

deep_compare

The `deep_compare` script compares lists of object attributes 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 deep_compare <source nickname> <target nickname> [-debug]`
- Example: `acp deep_compare dev prod`

Generating the Deep Compare Difference Report

This section provides details about how to generate the deep compare difference report.

Command

```
<ACPWorkDir>acp deep_compare      srcName tgtName
```

Example D-1 Example using command to run the deep compare difference report

```
D:\ACPWorkDir\project>acp deep_compare ACP934_SRC ACP934_TRG
```

Console Message:

This message indicates the status of Deep Compare:

```
ACP DeepCompare SUCCEEDED - Completed with no errors or notes.
```

```
=====DEEP-COMPARE REPORT=====
```

```
Report Generation in Progress, Please Wait...
```

```
Report Generated Successfully...
```

```
Please Check The Report @ D:\ACPWorkDir\project2\deep_compare.xls
```

```
Error Level = 0
```

Note: If you run Deep Compare and open the report in Microsoft Excel, and then you run Deep Compare again, ACP will report an error. The script will not be able to update the report because "it is in use by another application". Simply close the Difference Report before re-running Deep Compare.

Deep Compare Difference Report

The Deep Compare Difference Report contains many fields of information. Following the screenshot of a typical report, a table lists all the fields and what information they convey.

Note: There is a valid condition in Java Client criteria in which the Value field is null (is not populated with a value); ACP will report this as "Invalid Criteria Condition: Invalid value specified." You may ignore this error message.

Figure D–1 Sample Output of Deep Compare Report

Deep Compare Report

COMPARE RUN SUMMARY INFORMATION

File Name:	ACP93_export.agl
File Connection Info:	URL: http://localhost:8080/ncsp/User.s
Run Date and Time:	8/10/2020 11:41
File Compare:	0/0/0/22/501
Num of Objects:	11
Num of Differences:	23
Num of Errors:	0

OBJECT COMPARE SUMMARY

Iteration Type	Objects Available at Source	Objects Available at Target	Objects Compared	Objects With Differences	Objects With Errors
2	2	2	2	1	0
1	1	1	1	1	0
0	0	0	0	1	0

SOURCE ONLY OBJECTS

Iteration Type	Name
----------------	------

TARGET ONLY OBJECTS

Iteration Type	Name
----------------	------

DIFFERENCES DETAIL INFORMATION

Status	Configuration Type	Object	Context	Attribute	Property	Source Value	Target Value
	Base Class	Customize	Life Cycle Phase/Inactive	<NA>	Enabled	No	Yes
	Base Class	Customize	Life Cycle Phase/Inactive	<NA>	Status Strap Color	Navy	Purple
	Base Class	Customize	Life Cycle Phase/Inactive	<NA>	Description	Inactive!	Inactive
	Base Class	Customize	Life Cycle Phase	<NA>	Name	d	<Empty>
	Base Class	Customize	Life Cycle Phase	<NA>	Name	<Empty>	c
	Class	Customize/customize	Life Cycle Phase	<NA>	Name	b	<Empty>
	Class	Customize/customize	Life Cycle Phase	<NA>	Name	<Empty>	a
	Class	Customize/customize	Life Cycle Phase	<NA>	Name	<Empty>	a
	Class	Customize/customize	User Interface Tab/Page Two/Attribute	Data01	Name	Data01	Data01
	Class	Customize/customize	User Interface Tab/Page Two/Attribute	a	Description	<Empty>	ssssssssssssssssss
	Class	Customize/customize	User Interface Tab/Page Two/Attribute	a	API Name	a	ssssssssd
	Class	Customize/customize	User Interface Tab/Page Two/Attribute	b	Name	<Empty>	b

Important: Because Criteria and Privileges are both "business critical", and small changes to specific criteria and privileges (either general or single-user) in the Production instance can have consequences, a Best Practice is to always compare all Privileges and Criteria when running Deep Compare.

Deep Compare Report Sections and Fields

Deep Compare Report Sections and Fields

Explanation

COMPARE RUN SUMMARY INFORMATION

High-level and summary information.

Source XML Filename

Name of the Source (.agl) file used for Deep Compare, for example, ACP93_export.agl

Deep Compare Report Sections and Fields	Explanation
Target Agile Connection Info	<p>This information will talk about the URL or Target instance being used for comparison purpose. This field also tells who (User ID) has compared the instances.</p> <p>For example, URL = http://chandrakantdesk:8888/web; User = admin</p>
Compare Run Date and Time	Date and Time of Deep Compare report being generated, for example, 3/11/2009 2:34:24 PM
Duration of Compare	<p>How much time does ACP consume to compare the SOURCE and TARGET instances?</p> <p>For example, 00:00:01:5.907</p>
Total Number of Objects Compared	<p>How many objects have been compared?</p> <p>This gives the total number of SOURCE objects being mapped or matched exactly with TARGET objects (including RENAME mapping if it is enabled).</p> <p>Total Number of Objects Compared =? Objects Compared (OBJECT COMPARE SUMMARY) of all configurations, for example, 12.</p>
Total Number of Differences	<p>How many differences found?</p> <p>Total number of Differences is calculated based on the attribute level comparison after considering RENAME MAP, SUBOBJECTMAP and IgnoreRef (if you have enabled them in the Control file).</p> <p>Total Number of Differences =? Objects With Differences (OBJECT COMPARE SUMMARY) of all configurations, for example, 120.</p>
Total Number of Errors	<p>How many Objects are failed in this comparison?</p> <p>Total Number of Errors =? Objects With Errors (OBJECT COMPARE SUMMARY) of all configurations + Processing errors, for example, 11.</p>
OBJECT COMPARE SUMMARY	
<i>Statistics of the configuration level.</i>	
Configuration Type	Type of Configurations being compared. Refer enabled Configurations List for Deep Compare, for example, Company Profile.
Objects Available At Source	Total number of Objects available at Source. This is based on pattern matching at control file (config.xml), for example, 110.
Objects Available At Target	Total number of Objects available at Target. This is based on pattern matching at control file (config.xml), for example, 110.

Deep Compare Report Sections and Fields	Explanation
Objects Compared	<p>How many objects have been compared in this configuration?</p> <p>This gives the total number of SOURCE objects being mapped or matched exactly with TARGET objects (including RENAME mapping if it is enabled), for example, 34.</p>
Objects With Differences	<p>How many differences found in this configuration?</p> <p>Total number of Differences is calculated based on the attribute level comparison after considering RENAME MAP, SUBOBJECTMAP and IgnoreRef (if you have enabled them in the Control file), for example, 23.</p>
Objects With Errors	<p>How many Objects are failed in this configuration?</p> <p>Total number of objects failed to compare, for example, 02.</p>
SOURCE ONLY OBJECTS	
<i>"Source only" objects: not found on Target instance. This is based on pattern matching at control file (config.xml).</i>	
Configuration Type	Type of Configuration being compared. Refer to enabled Configurations List for Deep Compare, for example, Class.
Name	Name of the Object, for example, ECO.
TARGET ONLY OBJECTS	
<i>"Source only" objects: not found on Source instance. This is based on pattern matching at control file (config.xml)</i>	
Configuration Type	Type of Configuration being compared. Refer to enabled Configurations List for Deep Compare, for example, Class.
Name	Name of the Object, for example, ECO1.
DIFFERENCES DETAIL INFORMATION	
<i>Detailed information about attribute values.</i>	
Status	<p>Status can be Difference, Source Only, Target Only (Do not confuse with above Source Only and Target only, this difference is at attribute level), Not Comparable, and so on.</p> <p>This field also talks about Rename Map, Sub Object Map and Ignore References if it is enabled and matched at the both instances, for example, Difference.</p>
Configuration Type	Type of Configuration being compared. Refer to enabled Configurations List for Deep Compare, for example, Subclass.
Object	Name of the Object, for example, ECO.
Context	If Configurations are having UI Tabs and differences are existed in those Tabs, then details of accessing path will be given, for example, General Information.

Deep Compare Report Sections and Fields	Explanation
Attribute	Name of the Attribute, for example, Date01.
Property	Name of the property, for example, Description.
Source Value	Physical value at source instance, for example, ABC.
Target Value	Physical value at target instance, for example, XYZ.

ACP Exit Codes

This section lists the exit codes used in ACP.

Program Code	Program Text
0	Completed with no errors or notes.
-101	Note: Copy completed with notes. log for details.
-201	Note: Rename completed with notes. log for details.
-301	Note: Delete completed with notes. log for details.
-401	Note: Compare completed with notes. 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.
32	One or more program properties are in error. Source Name or Target name is misspelled at command-line.
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.

ACP Program Logs

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

Verobse 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 monitor 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.
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.

Item #	Description
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 computer-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 (and no errors).

Sample Console (stdout) Log

```

Agile(TM) CopyConfig (Version ACP 9.3.3 (Build 49))
===== ① ===== ② =====
=====

===== Delete Objects = ③ =====
*** Configuration Type: Subclass ④ Processing ...

        Not Configured -- Skipped ⑤
*** Configuration Type: Subclass      -- ... Completed ⑥
*** Configuration Type: List          -- Processing ...
    <Deleting> List: [Subclass #1] ⑦
    <Deleting> List: [Subclass #2]
    <Deleting> List: [Subclass #3]

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

**** 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 ⑥ Completed
*** Configuration Type: List          -- Processing ⑨
    <Renaming> List: [Target List ==> Source List]
    <Renaming> List: [Old List ==> New List]

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

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

```

```

*** Configuration Type: Company Profile-- Processing ...

===== Copy Objects (10)=====
Not Configured -- Skipped (4)
*** Configuration Type: Company Profile (5) Completed
*** Configuration Type: Class (6) Processing ...
    <Skipping> Class: [Changes.Change Orders] (11)
    <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 (12)

CopyConfig FAILED - Encountered one or more errors while renaming data. (201)
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 which object was being processed when the error occurred. It may even provide more specific information about the object if ACP was currently processing subobject 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.

Error Messages

This section lists FileLoad error messages you may encounter with applicable descriptions:

Attachment tab is invisible

Attachment tab of objects to which you want to add files is not active in Administrator. Attachment tab must be marked visible by Agile PLM administrator in Java Client.

Cannot find the file to be attached

Revise and correct the path and spelling of the attachment file name in the Index file entry. If you attached a URL, the supported protocols are FTP, HTTP, file, and HTTPS and there is no verification of the address.

Cannot find object <primary key>

The target object to receive the attachment cannot be located in the database. The object is the combination of the first three row fields: *ObjectType*, *PrimaryKey*, and *SecondaryKey*. Verify that the values for these fields are entered correctly in the Bad file and that an object of that name exists in the database. Add the object or modify the Index file as necessary.

Cannot find specified revision

The Item revision identifier in the Index file does not exist in the database, and **Attach to latest revision** was not selected in the FileLoad Options. Processing Attachments.

Cannot find the Index file or the Index file is empty

The path of the Index file is not correct (if you entered the Index file path manually) or the Index file you located is empty.

Cannot find the list value

All or some list values for flex List or Multilist fields are not valid.

Cannot perform this operation because the attachment is checked out

The file folder for the specified attachment is checked out, so the file is not uploaded. Make sure the file folder is checked in, and then use FileLoad again to upload the file.

Cannot perform this operation because the object <Object number> has been incorporated

The attachment table is read-only because the Item is incorporated by an ECO. To add attachments to incorporated Items, make sure **Attach Files To Incorporated Items** is checked in the FileLoad Options dialog box.

Failed to attach files

Reason for rejection unknown or the result of multiple errors. Make sure the object exists in the database, check the Index file entry, and retry. Ensure that you have the appropriate privileges.

Empty files are not valid to be added.

Attached files are 0 (zero) in size.

File Servers may be down. Please check the File Server Configuration

The Agile File Manager server is down. Go to the system where File Manager is installed and start it. On Windows, this involves starting the Apache Tomcat service.

Invalid date format

The date format in flex Date fields is not in accordance with the Date/Time formats in the user's profile.

Invalid number

The value for flex Numeric fields is not numeric (for example, it could include a letter string).

Invalid object type

Be sure the Index file row starts with a valid object type keyword. Object Type Keywords.

Invisible attribute fields

The specified flex field is not visible. Flex fields must be enabled (that is, made visible) in the Administrator panel.

Not a legal attachment type. Must be FILE or URL or INPLACE.

If you use FileLoad in Web Client, the AttachType (attachment type) field value must be FILE or URL. If you use FileLoad in Java Client, a third attachment type, INPLACE, is supported.

Number of fields provided is less than the minimum required.

Each Index file row must have at least six delimited fields. Empty fields must be marked by a double set of delimiter characters.

Note: When using the tab as the delimiter, be sure there is no tab after the last field in each row.

Primary Key (2nd) field empty

The second field - **PrimaryKey** - is always required. It must always contain, depending on the base class, a valid object number or name. Primary and Secondary Keys.

The attribute fieldname was not found in the Attachments tab.

This message indicates that the specified flex field name cannot be found. Flex fields must be enabled (that is, made visible) in the Administrator panel. Also, if the **Attribute Name-Value** separator in Index files is not correct, the field name cannot be properly identified.

The number length exceeds Maxlength.

The string length for flex Numeric fields exceeds the Maxlength defined in Administrator.

The number value is not between Min Value and Max Value.

The value for flex Numeric fields is not between Min Value and Max Value defined in Administrator.

The string length is not between 0 and Maxlength.

The string length for flex Text fields exceeds the Maxlength defined in Administrator

The user has insufficient privileges.

The user has insufficient privileges to attach files to objects.

Unable to connect to Agile Application Server.

Make sure the Application Server is running and you are logged in with the Checkin privilege.

Anatomy of the Error Log

Item #	Description
Header	Program name, version, and data sources.
1	Start of Error or Warning Message. Signifies the start of a single error or warning.

Item #	Description
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 exactly which object was being processed and which action was being performed.
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

```

===== Program Properties =====
Program Version: ACP 9.3.3 (Build 49)

Program Name:          ACP CopyConfig
Source Data Source:    Agile XML Archive: export_dev.agl
Target Data Source:    Agile Instance:
                        URL = http://localhost:8888/web;

User = admin
===== E R R O R (Begin) ===== ①
  Process Context: ②
  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 Con
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

  Propagation Action:          Copy
  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 and also 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 regarding configuration types. The amount of influence on business logic varies by configuration type.
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.

Item #	Description
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.
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) match 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 and also 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. 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 ACP 9.3.3 (Build 49)) ③
Run Date: Sept. 03, 2013 9:16:34 PM ④
===== Program Settings =====
-debug          []
-D              [acp.method=export, src.ref=qa_was,
                tgt.ref=qa_was]
Property File   []

===== Program Properties == ⑤=====
Miscellaneous Properties
global.properties.filename: (program)
common.properties.filename: (program)

Program Information
acp.program.name:          ACP CopyConfig (program)
acp.program.version:       Version ACP 9.3.3 (Build 49) (program)
acp.run.date:              20100903 (program)
acp.run.time:              074631 (program)
acp.run.datetime:         20100903_074631 (program)
acp.method:                export (command-line)
acp.debug:                 false (program)
acp.verbose:               false (program)

Objects
objects.suppress.skipped:  true D:\Agile\ACPWork\project\
                           project.properties)

Source Data Source
src.ref:                   qa_was (command-line)
src.type:                   agile (program)
src.name (qa_was.name):    QA Was

src.url [qa_was.url]:      (D:\Agile\ACPWork\project\project.properties)
                           http://10.3.3.21:9080/Agile
src.username (qa_was.username): propagation
                           (D:\Agile\ACPWork\project\project.properties)

Target Data Source
tgt.ref:                    qa_was (command-line)

```

```

1  tgt.type:                xml                (program)
   tgt.name (qa_was.name):   QA Was
                               (D:\Agile\ACPWork\project\project.properties)
   tgt.xml (qa_was.xml):     export_qa_was.agl
                               (D:\Agile\ACPWork\project\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: (9) 1000
  Include Patterns: (10) .*

Configuration Type:  Account Policy (Associations) (11)
  File Prefix:       account_policy_
  Objects Per File:  1000
  Include Patterns:  .*

Configuration Type:  AutoNumber (Not Configured) (12)
Configuration Type:  AutoNumber (Associations)
                     (Not Configured) (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.* (19)

Configuration Type: User (Associations) (Not Configured)

Configuration Type: User Group

File Prefix: user_group_
Objects Per File: 1000
Include Patterns: .*
Ignore Reference Patterns: Test.*

Configuration Type: User Group (Associations)

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

===== Program Actions =====

Copy Objects: Configured (20)
 Rename Objects: Not Configured
 Delete Objects: Not Configured

== (22) ===== List ===== (21)

Action	Result (23)	Configuration Object (24)
-----	-----	-----
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

Pattern Matching Statistics (25)

.*: 251

Copy Statistics (26)

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.	(28)
	2) ERROR: Duplicate name entered.	
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		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) =====