Radia Client Automation Enterprise Policy Server

For the Windows® operating system

Software Version: 9.00

Reference Guide

Document Release Date: April 2013

Software Release Date: April 2013

PERSISTENT

Legal Notices

Warranty

The only warranties for products and services are set forth in the express license or service agreements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Persistent Systems shall not be liable for technical or editorial errors or omissions contained herein. The information contained herein is subject to change without notice.

Restricted Rights Legend

Copyright Notice

Trademark Notices

Microsoft®, Windows®, Windows® XP, and Windows Vista® are U.S. registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark of The Open Group.

Acknowledgements

This product includes software developed by the Apache Software Foundation (http://www.apache.org/).

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com).

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/).

This product includes software written by Tim Hudson (tjh@cryptsoft.com).

This product includes software written by Daniel Stenberg (daniel@haxx.se).

This product includes OVAL language maintained by The MITRE Corporation (oval@mitre.org).

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to:

http://support.persistentsys.com/

This site requires that you register for a Persistent Passport and sign in. Register online at the above address.

For more details, contact your Persistent sales representative.

Support

Persistent Software support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by being able to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Submit enhancement requests online
- Download software patches
- Look up Persistent support contacts
- Enter into discussions with other software customers
- Research and register for software training

To access the Self-solve knowledge base, visit the Persistent Support home page.

Note: Most of the support areas require that you register as a Persistent Support user and sign in. Many also require an active support contract. More information about support access levels can be found on the Persistent Support site.

To register for a Persistent Support ID, go to: Persistent Support Registration.

Contents

Chapter 1

Introduction

Use this reference guide as an extension to the *Radia Client Automation Enterprise User Guide*. Its main goals are the following:

- To provide you with a better understanding of the concepts behind Policy Server management
- To allow you to perform more customized tasks that cannot be performed directly through the RCA Console

Overview of the Policy Server Service

The Policy Server Service is used by the RCA Console for administration purposes such as mapping services to users in the directory tree. It provides integration and extended enterprise functionality with your directory services. Policy method connections in the RCA Configuration Server Database (CSDB) are used to determine what services should be distributed and managed for the user that is currently logged on by querying the Policy Server.

The Policy Server leverages your investment in directory services while using RCA for software management. This greatly reduces the total cost of ownership of your environment. In other words, directory services handle policy management and RCA manages services. This saves you time because you do not have to define or maintain lists of users in the RCA Configuration Server Database.

The Policy Server integrates with existing Lightweight Directory Access Protocol (LDAP) directory servers and SQL databases in a customer's enterprise to enable single source points of control for user authentication, access policies, and subscriber entitlement.

Policy Server Processing

The Policy Server acts as a bridge between the Configuration Server and a directory server. It is a separate component from the Configuration Server. The Policy Server is installed as part of your Core and Satellite installation. A Policy Server can communicate with several LDAP directory services.

The following provides an overview of Policy Server Processing:

1. The RCA agent connects to the Configuration Server to resolve its **desired state**. The desired state embodies the content that RCA manages for a device. The desired state for each device is dynamically created by the Configuration Server based on information in the Configuration Server Database.
2. The Configuration Server contacts the Policy Server to perform policy resolution, and builds the agent's desired state using the policy information. The RCA administrator *must* entitle an agent device to minimum one service. If the Policy Server does not find any entitlements defined for an agent device, it checks the value for the parameter `POLICY_DEFAULT` in the MGR_POLICY section of the `edmprof.dat` file. Set the `POLICY_DEFAULT` parameter to a

valid ZSERVICE, such that if entitlements are not defined for an agent device, the ZSERVICE is resolved by default.

3. The policy method, LDAP_RESOLVE, handles the resolution requests, converting the requests into HTTP queries to the Policy Server, and treats the results as a set of objects and attributes to be incorporated into the desired state of the connected agent.
4. The Configuration Server completes resolution of the desired state and returns the information to the RCA agent.

The Policy Server can maintain a persistent connection to multiple LDAP directory servers and responds to policy requests by performing a policy resolution against the policy database and returns the set of objects resolved as the result set of the HTTP query.

Abbreviations and Variables

Abbreviations Used in this Guide

Abbreviation	Definition
RCA	Radia Client Automation
Core and Satellite	RCA Enterprise environment consisting of one Core server and one or more Satellite servers.
CSDB	Configuration Server Database

Variables Used in this Guide

Variable	Description	Default Values
InstallDir	Location where the RCA server is installed	For a 32-bit OS: `C:\Program Files\Hewlett-Packard\HPCA` For a 64-bit OS: `C:\Program Files(x86)\Hewlett-Packard\HPCA`
SystemDrive	Drive label for the drive where the RCA server is installed	C:

Chapter 2

Configuring Policy

You can configure policy using the RCA Console as described in the chapters in the *Radia Client Automation Enterprise User Guide*.

The sections in this chapter are useful if you must perform customized tasks that cannot be performed directly through the console.

These may include the following:

- Adding client automation policy attributes if you are not using an Active Directory LDAP directory service as described in "Adding Client Automation Policy Attributes" below.
- Editing the Policy Server configuration file or the LDAP configuration files to customize LDAP connection settings as described in "Connection to LDAP" on page 11.
- Configuring the LDAP method in the CSDB Editor if you need to modify the LDAP_RESOLVE method as described in "Configuring the LDAP Method" on page 14.
- Connecting to the LDAP method in the CSDB Editor if you want to resolve policy based on a different LDAP method as described in "Connecting to the LDAP Method" on page 17.

After completing these steps, you can administer policy using the RCA Console.

Adding Client Automation Policy Attributes

The Policy Server requires that the LDAP schema of an existing directory implementation be modified before it can be used to manage policy if the directory is not Active Directory (AD).

Caution: For AD in your Core and Satellite environment, there is no need to manually add the LDAP policy attributes to the AD directory services as discussed below. Policy Server now automatically generates `ldif` schema configuration files for each AD directory service configured in RCA. These files can be used to automatically add the necessary schema data to your AD. For more information, see the *Radia Client Automation Enterprise User Guide* topics on Policy configuration.

These attributes are used to manage policy scope, relationships, and assignments. Consult your directory service documentation and your enterprise's directory service administrator to make these changes. Be sure to back up your directory schema before any modifications.

Note: Changes to the LDAP schema can be risky because modifications to many directory services are not reversible. Be sure you type correctly. Check and double check the values you are entering before saving the changes to each value entered into the directory schema. Consult your directory services administrator and documentation.

Add the following required attributes:

- Add edmFlags as a single-valued, integer attribute with an object ID of 1.3.6.1.4.1.2133.2.1.1. It controls the scope of your policy. This is added as an optional attribute of the nvdObject class.
- Add edmLink as a multi-valued, case-sensitive string with an object ID of 1.3.6.1.4.1.2133.2.1.2. This attribute allows you to create a connection to a group that is not part of the user's LDAP group membership. This is added as an optional attribute of the nvdObject class.
- Add edmPolicy as a multi-valued, case-sensitive string with an object ID of 1.3.6.1.4.1.2133.2.1.3. Use edmPolicy to assign services to users and groups. This is added as an optional attribute of the nvdObject class.

The following attributes are not mandatory, but you may want to add them.

- Add edmPolicyDefault as a multi-valued, case-exact string with an object ID of 1.3.6.1.4.1.2133.2.1.4. Use edmPolicyDefault to assign policy defaults. This is added as an optional attribute of the nvdObject class.
- Add edmPolicyOverride as a multi-valued, case-exact string with an object ID of 1.3.6.1.4.1.2133.2.1.5. Use edmPolicyOverride to define policy overrides. This is added as an optional attribute of the nvdObject class.

> **Note:** Previous versions of this documentation incorrectly switched the object ID values for edmPolicyDefault and edmPolicyOverride. However, there is no need to correct existing values because the object IDs have no affect on Policy Server performance.

Adding the nvdObject Class

Some directory services, such as Microsoft Active Directory, do not allow adding of attributes to the *top* class. This is the highest level in the schema. If you cannot add attributes to the top class, create a class that will hold the required edmLink, edmFlags, and edmPolicy attributes, and inherit the values included in the *top* class. EdmPolicyOverride and EdmPolicyDefault are not required, but may be added for additional functionality. By creating this class, including its inherited values, we can modify the areas needed to apply RCA policies to specific areas of the directory tree. If you can add the attributes to the *top* class, policies can be placed anywhere in the tree.

If you need to create a class, name the class nvdObject. Create it as an auxiliary class with top as its parent class. Set the object ID to 1.3.6.1.4.1.2133.2.1. After creating the nvdObject class, you must add the edmFlags, edmLink, and edmPolicy. To proceed, you must reload your directory schema. Consult your directory service's documentation for instructions on how to do this.

Modifying classes with nvdObject

Once the schema has been re-loaded, the values entered above will show up as a selection, and you can add the nvdObject class to areas of your directory affected by the Policy Server.

To complete the modification for Microsoft Active Directory, nvdObject must be added as an Auxiliary class on the **Relationships** tab to all of the Active Directory classes listed below.

- Person
- Container
- DomainDNS

- Organizational Unit
- Group

You have now completed the necessary modifications to your directory schema. See "Connection to LDAP" below for instructions on how to connect the Policy Server to your directory services.

Connection to LDAP

The Policy Server obtains information about the LDAP directory services configured in RCA by communicating with the Portal periodically. The Policy Server communicates with the Portal by using information found in the `pm.cfg` file as discussed in "The Policy Server Configuration File" below section.

Based on the information obtained from the portal, the Policy Server then generates a configuration file (`.cfg` file) for each LDAP directory service that is configured. These files contain the necessary options for the Policy Server to be able to communicate with the directory services and resolve policies as discussed in "LDAP Configuration Files" on next page section.

The Policy Server Configuration File

The Policy Server configuration file, `pm.cfg`, is located in the `<InstallDir>\PolicyServer\etc` folder. It contains necessary information for Policy Server to Portal communication. It also contains some global configuration settings as discussed in "Support for Multiple LDAP Connections" below.

You may want to edit this file (with a text editor such as Notepad) to enable the directory services (`ENABLE_DS`) option and to configure the directory services synchronization interval (`DSSYNC_INTERVAL`). When the directory services option is enabled, the Policy Server will communicate with the Portal every `DSSYNC_INTERVAL` number of seconds to obtain a list of directory services. The default synchronization interval is 300 seconds. You can adjust this setting depending on how dynamic your specific environment is.

It is highly recommended that you do not change the `RMP_*` default settings the Policy Server configuration file without consulting with Persistent support.

Note: If you make manual changes to `pm.cfg`, you will need to restart the RCA Policy Server service that is hosting the Policy Server.

Support for Multiple LDAP Connections

The Policy Server supports multiple concurrent LDAP queries to a single LDAP directory at one time. You can configure the number of concurrent LDAP queries in the Policy Server's configuration file, `pm.cfg`, The table below describes which parameters apply. Note as stated previously, when you make changes to `pm.cfg`, you will need to restart the RCA Policy Server service.

Configurable Values for Multiple LDAP Queries

Value	Default	Description
N_	4	Specifies number of parallel LDAP directory connections to be created.

Value	Default	Description
workers		
PolicyUrl	/policy/ldap	Registers the URL of the Policy Server's LDAP. This is required to use the N_WORKERS parameter. The parameter name is case sensitive. If you do not have this line in your `pm.cfg`, then you will need to add it.

LDAP Configuration Files

As previously noted, Policy Server generates an LDAP configuration file for each LDAP directory service configured in RCA. These files are located in the `<InstallDir>\PolicyServer\etc\ldap` folder.

Each configuration file contains two distinct sections, the `DsConfig` section and the `DsConfigOverride` section. The override section contains settings that can be overridden, such as adding a list of hostnames for failover, policy prefix changes (edm instead of nvd), and other settings that may require an override. When a change is detected in the Portal relating to the directory service, the `.cfg` file will automatically be updated. However, the overrides section will remain intact.

Options for DsConfig Section

The following table shows the options that are specified in the `DsConfig` section of the LDAP configuration file:

DsConfig Options

Option	Description
BASE_DN	Specifies the base domain. Example: dc=asdfoods, dc=com.
BIND_DN	Specifies the fully qualified name of the account that has Active Directory Schema Permissions on the Directory. Example: cn=Administrator , cn=Users, dc=asdfoods, dc=com or administrator@asdfoods.com
BIND_PW	Specifies password associated with BASE_DN
TYPE	Specifies type of LDAP connection. It is either 'ldap' or 'ldaps'. 'ldap' is used for non SSL connection with LDAP server and 'ldaps' is used for SSL connection.
HOST	Specifies the host name of the LDAP server
PORT	Specifies the LDAP server port number
SSL_ CACERTDIR	Specifies the directory that hosts Certificate Authority certificate file

Option	Description
SSL_ CACERTFILE	Specifies full path for Certificate Authority certificate file
LDAP_ DEBUG	Specifies if LDAP debugging should be enabled. Values are either 1 or 0.
LDAP_ LOGFILE	Specifies the log file in which debugging messages are logged if LDAP_ DEBUG is enabled
DISABLED	Specifies if this LDAP configuration can be used for Policy Resolution. Values are either 0 or 1 based on if this configuration file can be used or cannot be used for Policy Resolution

Options for DsConfigOverride Section

The following table shows the options that are specified in the `DsConfigOverride` section of the LDAP configuration file:

DsConfigOverride Options

Field	Default	Description
HOST	Not Set	Specifies coma separated host names for failover support; e.g. host1,host2,host3
LDAP_ DEBUG	0	Specifies if LDAP debugging should be enabled. Values are either 1 or 0.
LDAP_ LOGFILE	Not Set	Specifies the log file in which debugging messages are logged if LDAP_ DEBUG is enabled
DISABLED	Not Set	Specifies if this LDAP configuration can be used for Policy Resolution. Values are either 0 or 1 based on if this configuration file can or cannot be used for Policy Resolution.
VERSION	3	Specifies LDAP Protocol version to use (2 or 3)
CACHE	1	Specifies if caching is enabled. Values are either 0 or 1.
FLUSH_ FREQ	"60*60"	Specifies the delay in seconds between each flush of the cache
RETRY	1	Specifies the number of attempts to issue the LDAP request before marking the directory as unavailable. If this occurs, a reconnection attempt will be made when the next ping is performed. This is also the number of attempts to reconnect to the directory if the ping detects the directory to be offline. If HOST is specified as a comma separated list of host names, RETRY should have a value of at least the number of host names.
TIMEOUT	120	Specifies Timeout (in seconds) for LDAP request

Field	Default	Description
PREFIX	"edm"	Specifies prefix of RCA attributes and objectclass specified in LDAP schema
VIEW	Not Set	See the "Controlling Policy Scope Globally" on page 20 section.

Policy Section in the Configuration Server Profile

To understand how the Configuration Server communicates with the Policy Server, review the information contained the Configuration Server profile file, `edmprof.dat` located in the *<InstallDir>*`\ConfigurationServer\bin` folder. It contains a MGR_POLICY section that specifies the host name and port number of the Policy Server as shown in the following example:

```
* Manager Policy Section                                    *
*  HTTP_HOST      = Host name of Policy Server          *
*         Multiple hosts may be specified (space or comma *
*         separated) for fail over                         *
*  HTTP_PORT      = IP Port number of Policy Server      *
*         NO restart required                              *
*-----------------------------------------------------------*
[MGR_POLICY]
HTTP_HOST = XXX.XXX.XXX.XXX
HTTP_PORT = 3466
```

You should not modify this file without contacting Persistent Support.

Configuring the LDAP Method

If you are using LDAP, you will have to prepare your Configuration Server Database (CSDB) to use Policy Server.

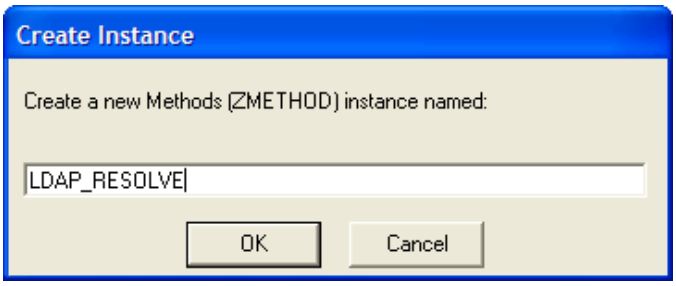
You must:

- Create a connection to the LDAP method in the CSDB as discussed in this section
- Connect the users to the LDAP method as discussed in the next section, "Connecting to the LDAP Method" on page 17

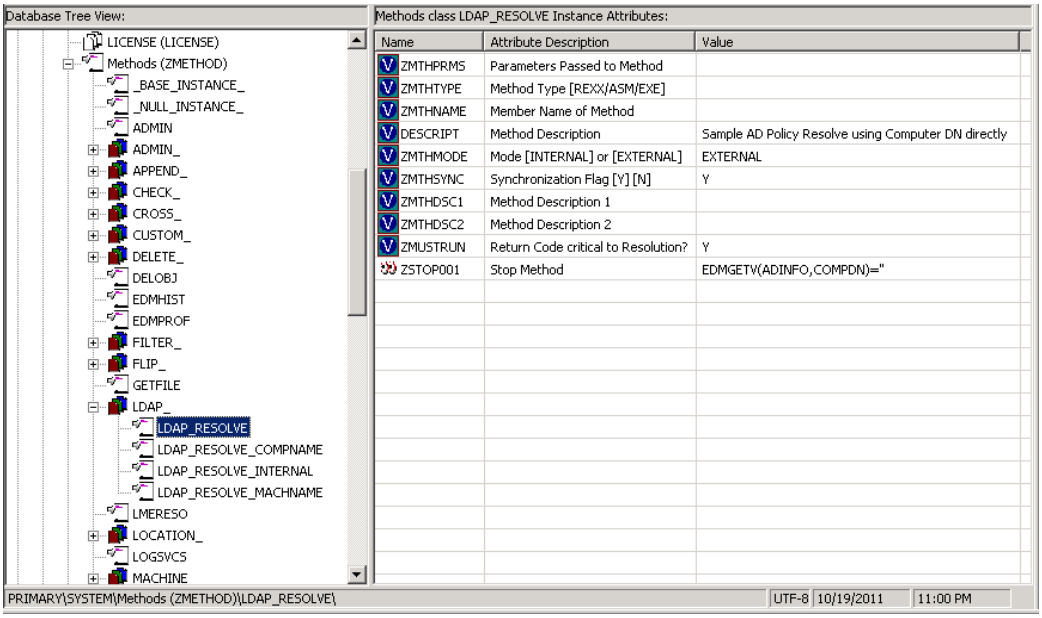
> **Note:** Your RCA Core and Satellite installation includes templates for LDAP ZMETHOD. Modify these templates as necessary using the information in this section.

To create the LDAP method in the Configuration Server Database (CSDB):

1. Open the RCA Admin CSDB Editor, and go to PRIMARY.SYSTEM.ZMETHOD.
2. Right-click **Methods (ZMETHOD)**.
   A shortcut menu opens.
3. From the shortcut menu, select **New Instance**.
   The Create Instance dialog box opens.



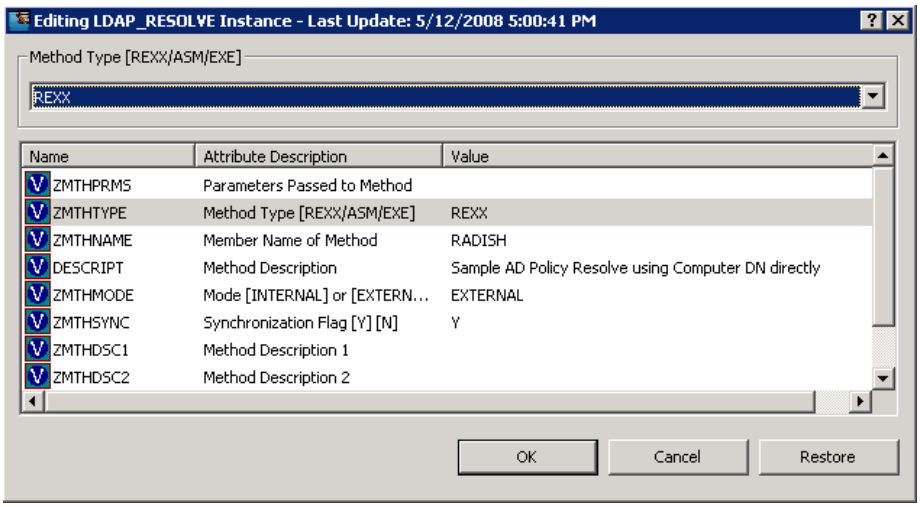
4. Type `LDAP_RESOLVE` in the text box, and click **OK**.
   The RCA Admin CSDB Editor window opens.
5. Double-click **LDAP_**.
   The tree expands.
6. Double-click **LDAP_RESOLVE** in the tree view.
   The attributes of LDAP_RESOLVE appear in the list view.



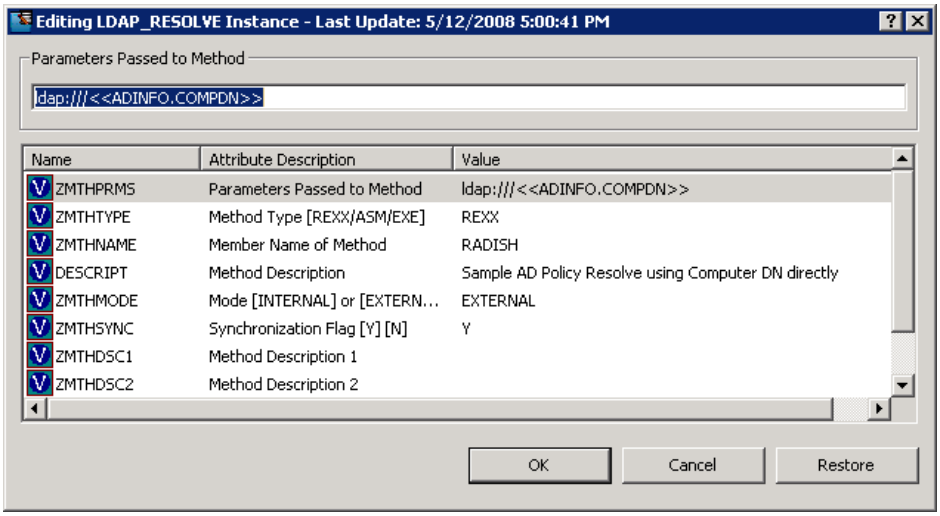
7. Double-click the **ZMTHNAME** attribute in the list view.
   The Editing Instance dialog box opens.
8. In the **Member Name of Method** field, type `RADISH`.

> **Note:** Configuration Servers running on UNIX® platforms are case sensitive and require RADISH entered in upper case.

9. Click **ZMTHTYPE**.



10. In the **Method Type** drop-down list, select **REXX**.
11. Click **ZMTHPRMS**.



12. In the **Parameters Passed to Method** text box, use the following values.
For HTTP:

`http:///policy/ldap?dn=<<ZDN>>&&os=<<ZOS>>`

For Microsoft Active Directory:

- To manage policies by machine (preferred method for single directory service):
  - `ldap:///dc=`*`domainname`*`,dc=`*`forestname`*`,dc=com??sub?samaccountname=<<ZUSERID>>$` (If the client uses $MACHINE as the ZUSERID)
  - `ldap:///<ADINFO.COMPDN>`
  - `http:///policy/ldap?dn=<<COMPDN>>`

  For example,

```
ldap:///dc=asdfoods,dc=com??sub?samaccountname=<<ZUSERID>>$
```

- To manage policies by machine (preferred method for multiple directory services):
  `ldap:///<ADINFO.COMPDN>`
- To manage policies by user:
  `ldap:///dc=`*`domainname`*`,dc=`*`forestname`*`,dc=com??sub?samaccountname=<<LOCALUID>>`

  For example,

  `ldap:///dc=asdfoods,dc=com??sub?samaccountname=<<LOCALUID>>`

For Novell Directory Services (NDS):

- To search the entire NDS tree for policy, type:
  ldap:///o=organization??sub?cn=<<ZNTUSER>>

  For example,

  `ldap:///o=cert??sub?cn=<<ZNTUSER>>`
- To search NDS with a specified Distinguished Name, type:
  `http:///policy/ldap?dn=<<ZMASTER.DN>>`

For Netscape iPlanet:

- To manage policies by user type:
  `ldap:///dc=com??sub?uid=<<ZUSERID>>`

13. Click **OK**.
    The Instance Edit Confirmation dialog box opens.
14. Click **Yes** to confirm the changes. The RCA Admin CSDB Editor window opens.

Now, whenever a managed device connects to the Configuration Server, the null instance calls the policy method, and will point to the appropriate services for that user.

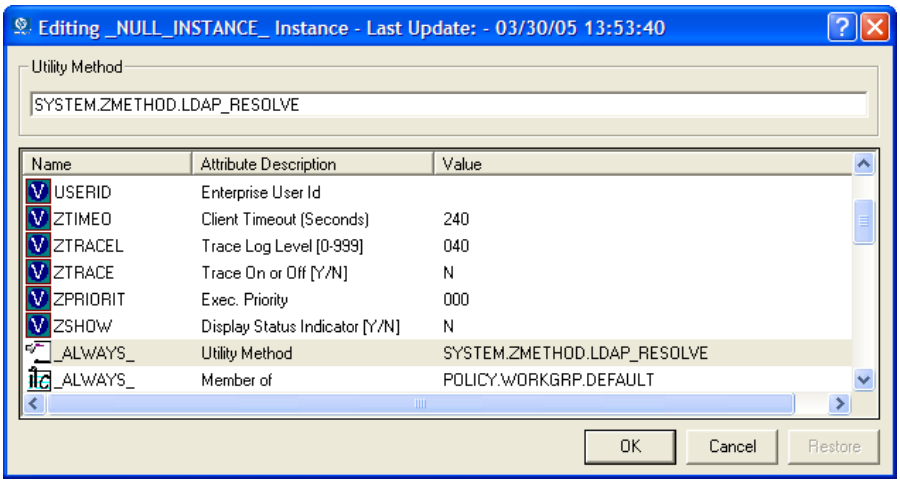
Connecting to the LDAP Method

You must connect the LDAP method to an instance in the POLICY domain for policy resolution.

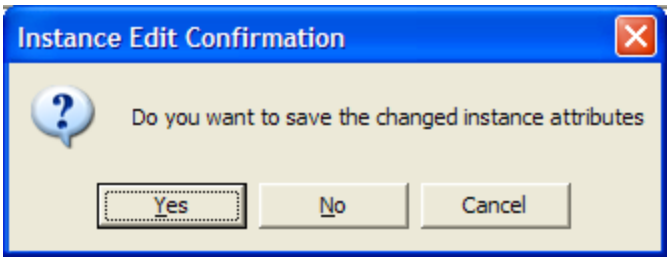
To connect the user to the LDAP method:

1. Open the RCA Admin CSDB Editor.
2. Navigate to PRIMARY.POLICY.USER.
3. Double-click the null instance.

   Note: If the null instance is connected to the Default workgroup, change the name of the instance from Default to _NONE_.
4. In the list view, double-click on the **_ALWAYS_ Utility Method** line.
   The Editing Instance dialog box opens.



5. In the **Utility Method** text box, type `SYSTEM.ZMETHOD.LDAP_RESOLVE`.
6. Click **OK**.
   The Instance Edit Confirmation opens.



7. Click **Yes**.

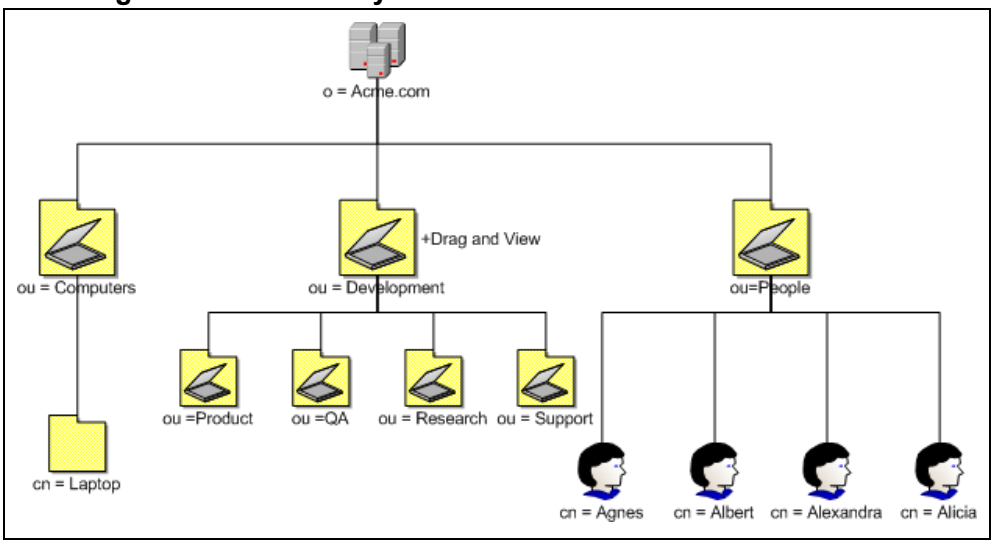
The LDAP_RESOLVE method is connected to the Null User instance.

Chapter 3

Policy Scope and Resolution

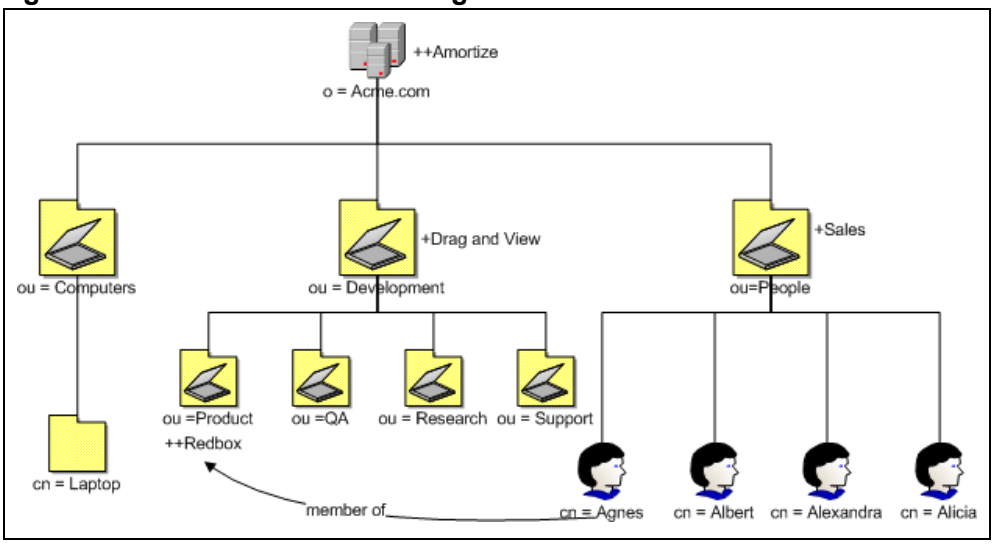
By default, a subscriber inherits the policy from the parent of any groups it is linked to. This link can be through either the subscriber's directory service membership or through the use of the edmLink attribute. The figure Acme organization directory structure shows a part of the Acme organization. It has three organizational units, Computers, Development, and People. **Computers** holds the Laptop container. **Development** includes the Product, QA, Research, and Support organizational units. **People** includes the actual users of the enterprise.

Acme organization directory structure



In this figure, Agnes will inherit the policy of the People organizational unit and the Acme organization.

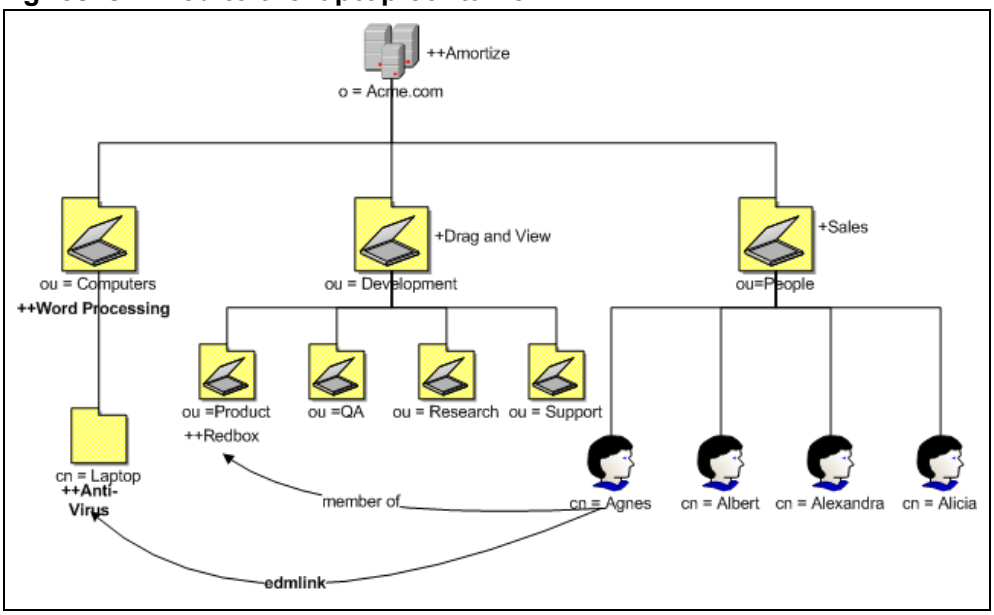
Agnes is a member of Product organizational unit



If Agnes is a member of the Product organizational unit, she will also inherit the policy from that unit and the Development organizational unit. In the figure Agnes is a member of Product organizational unit, Agnes would get Sales and Amortize because she is a part of the People organizational unit. Because Agnes is a member of the Product organizational unit, she would *also* inherit Redbox *and* Drag and View.

Suppose that you need Agnes to receive the services associated with the laptop container, but she is not linked to that container through directory services. Use edmLink to connect her to that container.

Agnes is linked to the laptop container



In the figure Agnes is linked to the laptop container, Agnes will receive Anti-Virus because she has been linked to the laptop container. Since laptop is part of the Computers organizational unit, she will also get Word Processing. Now, she has a total of six applications.

Managing Policy Scope

If you do *not* want to inherit the policy from the parent objects, you can limit the Policy Server's scope of resolution. You can do this either globally for the entire directory structure or for only specific objects. Manage the scope globally by modifying the Policy Server configuration file. Control policy scope for one object by using the edmFlags attribute.

Caution: Be sure that you have a thorough understanding of your directory structure. When designing a change to the scope of policy resolution, anticipate the result of your modifications *before* making the modifications.

Controlling Policy Scope Globally

The VIEW option allows you to control whether or not to continue up the directory tree to assign policy. Modify the VIEW option in the `DsConfigOverride` section of the individual LDAP configuration file to control the scope. See "LDAP Configuration Files" on page 12.

The syntax for the VIEW option is:

```
VIEW {
<attr> {view}
}
```

Where *attr* is one of the attributes listed in the LINKS (`edmLink`, `groupmembership`, `memberof`, `aliasedobjectname`) configuration option in `pm.cfg`, and *view* is a list of `LINKS` the Policy Server is allowed to see. An empty list means that there is no view when visiting an object from the specified attribute. This would result in following that link and not continuing. You can list as many or as few attributes as needed.

The default values for the `LINKS` configuration option are `edmLink`, `memberof`, `groupmembership`, and `aliasedobjectname`. When you look at a particular object such as a group or user through the Policy Server interface, you will see only these attributes for that object. If you do not want Policy Server to inherit the policy for any parents of an `edmLink` attribute, modify the `VIEW` option in the `DsConfigOverride` section of the individual LDAP configuration file like this:

```
VIEW {
edmLink {  }
}
```

This configuration with the empty brackets tells Policy Server to follow `edmLink`, but not to inherit from any parents or any links contained within the object from that branch of the directory tree.

Looking back at the Acme organization example, suppose you want Agnes to receive policy for the laptop container, but not inherit any policy from the Computers organizational unit. In the figure Agnes is linked to the laptop container Agnes will receive Anti-Virus because she has been linked to the laptop container, but she will not inherit Word Processing when `edmLink` is configured with empty brackets to not inherit from any parents.

Similarly, if we wanted to follow a `memberof` attribute, and then not inherit from the parent objects, we would replace `edmLink` with `memberof`. The `VIEW` option would look like this:

```
VIEW {
memberof {  }
}
```

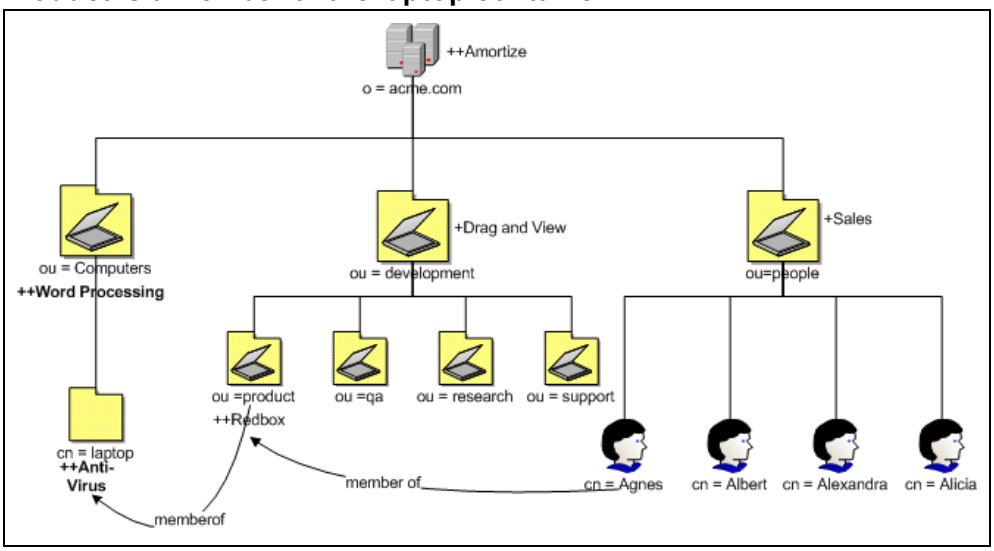
This configuration with the empty brackets tells Policy Server to follow `memberof`, but not to inherit from any parents from that branch of the directory tree or any links contained within the object.

Finally, suppose that we only want to follow `memberof` relationships. The `VIEW` option would look like this:

```
VIEW {
memberof {memberof}
}
```

This configuration with the `memberof` in quotes tells Policy Server to follow `memberof`, but not to inherit from any parents from that branch of the directory tree. When we follow a `memberof` relationship, we will continue to follow `memberof` relationships until we reach an object that does not contain a `memberof` relationship. In the figure Product is a member of the laptop container, Agnes will get Sales, Amortize. Then she will get Redbox because she is a member of Product. Since Product is a `memberof` laptop, she will get Anti-Virus. If Laptop had any `memberof` relationships, she would follow those relationships, too. Agnes will not follow any relationships other than `memberof`.

Product is a member of the laptop container



Optimization for Single-Service Policy Resolution

To optimize the resolution of the policies for a single service, you can include domain names in the CSDB, which will be used in the case of a single service policy resolution. These domains are skipped for a detailed policy server resolution.

Caution: Domains thus specified will lose the ability to pass policy attributes into the resolution model.

The default behavior in policy resolution is to walk the tree for all calls.

To create domain names in CSDB for single-service policy resolution:

1. Create instances of the POLICY.POLPRMS class with the name of the domain.
   The actual domain name skipped for single service resolution is the value of the variable XDOMAIN in the instance. For example, if you create an instance of the POLICY.POLPRMS class with the name, PATCHMGR, the XDOMAIN variable in this instance will automatically get the name PATCHMGR. You may specify any number of instances as required. The XDOMAIN variable can also contain the star (*) character as a wildcard to skip all the domains with matching names.

   The rules for the wildcard in the XDOMAIN variable are the following:

   - Only one star maximum can be present in the XDOMAIN.
   - It can be placed anywhere in the XDOMAIN.

2. Modify or create SYSTEM.PROCESS.PREFACE.
   Add a *class* connection (not a method connection) to POLICY.POLPRMS.* instances.

   As an alternative, you can pass up the POLPRMS instances from the client.

Appendix A

LDAP Discussion

This appendix provides more information on directory services for policy administrators needing additional information. It also includes descriptions of LDAP terminology, the use of substitution and expressions, and URLs used for RCA Policy Server.

LDAP Background

An LDAP directory is a hierarchically named tree of objects, where each object has a class (type) or classes, and contains potentially many named attributes, appropriate to its classes. Each attribute may contain multiple values.

It is outside the scope of this document to describe in any detail what an LDAP directory means. As a rapidly growing force in the systems management industry, many excellent sources exist for further background.

The Policy Server is not concerned with such differences in interpretation—our only requirement is that the directory supports either the LDAP v2 or LDAP v3 protocols.

RCA Policy Server and LDAP

The LDAP Policy Extension, in conjunction with the RCA Policy Server, is intended to provide a scalable policy infrastructure, leveraging your existing investment in directories. The LDAP Policy Extension was developed to provide "Low Cost of Entry" to policy-based management, allowing you to start with a very simple policy model and incrementally grow the model as your policies mature. The LDAP Policy Extension provides a clean integration with the standard repository for enterprise management information (LDAP), and allows an organization to leverage the information represented in its directories to deliver sophisticated policy-management to the many computing devices in its enterprise.

The Policy Server is aimed at customers who have a detailed understanding of LDAP/X.500 directories, and an established directory infrastructure. The Policy Server uses the LDAP protocol (version 2 or 3) (over TCP/IP) to speak to the customer's directory. This protocol encompasses all major directory products on the market, including the latest offerings from companies such as Novell, Microsoft, and Netscape.

The LDAP Policy Extension extends the Policy Server with a number of features that enable you to represent your software management policy within your existing directory infrastructure and have this policy drive your RCA infrastructure to provide a comprehensive and sophisticated software management solution.

The extension makes policy resolution available via a URL utilizing the standard Policy Server policy framework. It maintains a persistent LDAP connection to your corporate directory, and provides online HTML documentation. From the RCA Console, there are a number of interactive tools for discovering or diagnosing the policy outcome for target objects (typically users or machines) in your directory.

It is anticipated, but not required, that the Policy Server hosting this extension be co-located on or near the directory to keep network latency to a minimum and enhance performance and manageability.

The LDAP Policy Extension understands the standard relationships that exist in a directory between different objects (parent-child, memberOf). In addition to these standard relationships, three additional attributes may be used:

- `<pfx>Flags`
  controls various subtle aspects of the policy resolution.
- `<pfx>Link`
  allows you to specify additional, potentially dynamic or conditional relationships.
- `<pfx>Policy`
  allows you to define resultant strings that will be netted out during policy resolution.

By default the prefix used is `edm`, but alternatives may be used to allow your directory to support multiple concurrent policy frameworks for different purposes.

The LDAP Policy Extension starts at the specified DN, and walks the entire tree of relationships that the object has with other objects, accumulating policy attributes. Then it evaluates all conditional policies, and finally resolves any conflicting policies, using a straightforward should/may, grant/deny model.

Terminology

Before using directory-based policy management with Policy Server, it is important to establish some terminology that is used throughout this discussion.

- **Should**
  This is used to describe a mandatory or **required** policy.
- **May**
  This is used to describe a desired or **advisory** policy.
- **Policy**
  This is a string that is used to **represent** a **desired** outcome. The Policy Server does not impose any particular interpretation upon this. When used in conjunction with the LDAP Adapter, the adapter will interpret this as the name of an application defined within Radia Client Automation.
- **Relationship (link)**
  Two directory objects are said to be **related** if one can be reached from the other, directly or indirectly. Examples of relationship include parent-child, and group membership (a user is **related** to the group he is a member of). Relationships are unidirectional.
- **MemberOf**
  This is used to describe a **relationship** between two objects. Many common directories support an attribute called **memberOf** that embodies this relationship, typically between users and groups.

Substitution

Two forms of substitution are provided:

- Current Object Attributes: <<nameOfAttr>, or
- Inbound Object Attributes: <<in.nameOfAttr>>

The former allows you to construct expressions based upon the value of another attribute in the current object (same one that contains the edmLink or edmPolicy), for example,

```
edmLink: cn=<<homePC>>, cn=Computes, o=Acme.
edmLink: cn=wnt001, cn=Computers, o=Acme ; <<homePC>> == "wnt001".
```

The latter allows you to reference attributes that were supplied as input to the policy resolution, for example:

```
edmPolicy: ++SOFTWARE/STRATUS_PAD; <<in.hostname>> == "XKEZ01$"
```

Currently the minimum attributes that will exist are listed in the following table:

Default Inbound Object Attributes <<in.nameOfAttr>>

Attribute Name	Sample Value
context	{}
dn	{cn=su61er, cn=computers, dc=acme, dc=com}
dname	software
domain	{ACMEWEST\XKEZ01$}
hostname	{XKEZ01$}
ipaddress	192.168.0.100
mtime	{2007-06-22 18:54:35}
nvdipnetworknumber	192.168.0.0
nvdsubnet	255.255.255.0
smenclosureserialnumber	CNU0123456
smsystemproductname	{HP Compaq nc6000 (DU655C)}
smsystemmanufacturer	Hewlett-Packard
smsystemserialnumber	CNU0123456
smsystemuuid	27494E2D171E11DB09906D9908020929

Expressions

The expressions are implemented as Tcl (www.scriptics.com) expressions, where instead of using $myVar you would use <<myAttribute>>. A simplified summary of valid expressions is provided below. Most of the standard C language expression operators are valid.

Expressions

Expression	Meaning
A && B	Logical AND
A || B	Logical OR
!A	Logical NOT
<<myAttr>> == "Hello"	Test for equality (case-sensitive)
<<myAttr>> != "Hello"	Test for inequality
<<myAttr>> < 55	Numerical comparison for less than
<<myAttr>> >= "Hello"	Dictionary comparison for greater than or equal to (C locale)

There are also a small number of specialized functions.

Specialized Function Example

Example	Meaning
[memberOf "ou=Accounting, o=Acme"]	Yields TRUE if the DN specified is part of your policy model.
[parent <<dn>>] == <<aSpecialDN>>	Yields TRUE if the parent DN of the current object is the same as the "aSpecialDN".

The LDAP Extension URL Namespace

The LDAP extension provides the following special purpose URLs:

LDAP Extension URL Namespace

URL	Description
/policy/ldap?<x-url encoded query>	Perform machine-readable policy resolution. The query arguments should be an attribute value list of inbound attributes, formatted in accordance with the X-URL encoding specification. The following attributes are currently supported and interpreted by the LDAP Policy Extension: • **dn** - the distinguished name or LDAP URL to perform policy resolution upon. (REQUIRED) • **phase** - the value may be specified as "1", "2", or "3", to view the intermediate stages of policy resolution. (default=3) • **prefix** - the value is the prefix to use when searching the directory for policy related attributes, i.e., <pfx>Policy or <pfx>Link. (default=edm) • **debug** - the value is the log level to use for this single query, a value of 9 or above will generate detailed logging in the Policy Server log file. (no default)

URL	Description
/status/ldap	Return an overview of the current status of the extension.
/status/ldap/all	Return all available status information on extension.
/status/ldap/cache	Return information on cache.
/status/ldap/stats	Return statistics on usage of extension.

Appendix B

Domain Filtering

If you are using the RCA Policy Server to create entitlements in your enterprise, you can filter out which domains the Policy Server will assign services from based on connect parameters.

If you are using the Policy Server with RCA Patch Management, you will want to separate resolution of regular software services from those for Patch Management. The Policy Server filters services based on the dname passed on the radskman command line. The Policy Server configuration file, `pm.cfg`, contains filter settings in the format:

```
DNAME=<DOMAIN NAME>   { rule }
```

Where the DOMAIN NAME is the value passed in dname by RADISH. In the case of a Patch Management agent, this will be the dname parameter of radskman. Dname should be patch. If the filter name passed in dname is not found in `pm.cfg`, then the filter DNAME=* will be used.

```
The default configuration for these filters is shown below:

DNAME=*          { * !PATCHMGR !OS !SECURITY !AUDIT }

DNAME=PATCH      { PATCHMGR }

DNAME=OS         { OS }

DNAME=VM         { SECURITY }

DNAME=SECURITY   { SECURITY }

DNAME=AUDIT      { AUDIT }
```

In this configuration the default rule (*) will ignore PATCHMGR, OS, SECURITY and AUDIT domains and allow everything else as denoted by the use of an exclamation point (!). PATCH, OS, and AUDIT rules allow only policies for PATCH, OS, and AUDIT domains, respectively.

If the dname parameter passed by radskman is either VM or SECURITY, the rules allow only policies for SECURITY.

If, for instance, we wanted to allow any policies for AUDIT resolution we would change the last filter to: DNAME=USAGE {*}.

We appreciate your feedback!

If an email client is configured on this system, by default an email window opens when you click here.

If no email client is available, copy the information below to a new message in a web mail client, and then send this message to radiadocfeedback@persistent.co.in.

Product name and version: Radia Client Automation Enterprise Policy Server, 9.00

Document title: Reference Guide

Feedback: