Cisco Prime Access Registrar 8.0 Administrator Guide

Published: January 25, 2018

Last Modified: May 19, 2025

Cisco Systems, Inc.
www.cisco.com

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco website at www.cisco.com/go/offices.

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: www.cisco.com/go/trademarks. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

Any Internet Protocol (IP) addresses used in this document are not intended to be actual addresses. Any examples, command display output, and figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses in illustrative content is unintentional and coincidental.

Cisco Prime Access Registrar 8.0 User Guide
© 2018 Cisco Systems, Inc. All rights reserved.

CONTENTS

CHAPTER 1 **Overview** 1-1

Session Management 1-1

Failover by the NAS and Session Management 1-2

Cross Server Session and Resource Management 1-2

Script Processing Hierarchy 1-4

RADIUS Protocol 1-5

Steps to Connection 1-6

Obtain Documentation and Submit a Service Request 1-7

CHAPTER 2 **Configuring Cisco Prime Access Registrar** 2-1

Using **aregcmd** 2-1

General Command Syntax 2-1

aregcmd Commands 2-2

Configuring a Basic Site 2-2

Running aregcmd 2-3

Changing the Administrator's Password 2-3

Creating Additional Administrators 2-4

Configuring Prime Access Registrar Server Settings 2-4

Checking the System-Level Defaults 2-5

Checking the Server's Health 2-6

Selecting Ports to Use 2-6

Displaying the UserLists 2-7

Displaying the Default UserList 2-8

Adding Users to UserLists 2-8

Deleting Users 2-9

Displaying UserGroups 2-9

Configuring Clients 2-10

Adding a NAS 2-10

Configuring Profiles 2-11

Setting RADIUS / Diameter Attributes 2-11

Adding Multiple Cisco AV Pairs 2-12

Validating and Using Your Changes 2-12

Saving and Reloading 2-12

Testing Your Configuration 2-13

Using radclient 2-13
Troubleshooting Your Configuration 2-14
Setting the Trace Level 2-14
Configuring Accounting 2-15
Configuring SNMP 2-15
Enabling SNMP in the Cisco Prime Access Registrar Server 2-16
Stopping the Master Agent 2-16
Modifying the snmpd.conf File 2-16
Access Control 2-16
Trap Recipient 2-17
System Contact Information 2-18
Restarting the Master Agent 2-18
Configuring Dynamic DNS 2-18
Testing Dynamic DNS with radclient 2-20

CHAPTER 3 **Customizing Your Configuration** 3-1
Configuring Groups 3-1
Configuring Specific Groups 3-2
Creating and Setting Group Membership 3-2
Configuring a Default Group 3-3
Using a Script to Determine Service 3-3
Configuring Multiple UserLists 3-4
Configuring Separate UserLists 3-5
Creating Separate UserLists 3-5
Configuring Users 3-6
Populating UserLists 3-6
Configuring Services 3-6
Creating Separate Services 3-6
Creating the Script 3-7
Client Scripting 3-7
Configuring the Script 3-7
Client Scripting 3-8
Choosing the Scripting Point 3-8
Handling Multiple Scripts 3-9
Configuring a Remote Server for AA 3-9
Configuring the Remote Server 3-10
Creating a RemoteServer 3-10
Configuring Services 3-11
Creating Services 3-11

Configuring the RADIUS Server 3-12
Changing the Authentication and Authorization Defaults 3-12
Configuring Multiple Remote Servers 3-13
Configuring Two Remote Servers 3-13
Creating RemoteServers 3-14
Configuring Services 3-14
Creating the Services 3-14
Configuring the Script 3-15
Choosing the Scripting Point 3-15
Configuring Session Management 3-16
Configuring a Resource Manager 3-16
Creating a Resource Manager 3-17
Configuring a Session Manager 3-18
Creating a Session Manager 3-18
Enabling Session Management 3-18
Configuring Session Management 3-19

CHAPTER 4 **Setting the CPAR Configurable Option** 4-1
General Command Syntax 4-1
View-Only Administrator Mode 4-2
ViewOnly Property 4-3
Configuration Objects 4-3
aregcmd Command Performance 4-3
RPC Bind Services 4-4
aregcmd Commands 4-4
add 4-5
cd 4-5
delete 4-6
exit 4-6
filter 4-6
find 4-6
help 4-7
insert 4-7
login 4-7
logout 4-7
ls 4-8
next 4-8
prev 4-8
pwd 4-9

query-sessions 4-9
quit 4-9
release-sessions 4-9
reload 4-10
reset-stats 4-10
save 4-10
set 4-11
start 4-12
stats 4-12
status 4-15
stop 4-16
tacacs-stats 4-16
tacacs-reset-stats 4-16
dia-stats 4-17
dia-stats-reset 4-19
trace 4-20
trace-file-count 4-21
unset 4-21
validate 4-21
OpenSSL Commands 4-22
ecparam 4-22
req 4-22
ca 4-22
aregcmd Command Logging 4-22
aregcmd Command Line Editing 4-23
aregcmd Error Codes 4-23

CHAPTER 5 **Configuring and Monitoring the RADIUS Server** 5-1
Radius 5-2
UserLists 5-3
Users 5-4
HiddenAttributes Property 5-5
UserGroups 5-5
Policies 5-6
Clients 5-6
Vendors 5-12
Scripts 5-13
Services 5-14

Types of Services 5-15
EAP Services 5-16
Extended-EAP 5-17
File 5-17
Group 5-19
Java 5-20
LDAP 5-20
Local 5-21
ODBC 5-22
ODBC-Accounting 5-23
Prepaid Services 5-23
RADIUS 5-23
Radius Query 5-24
Diameter-RADIUS 5-28
RADIUS-Diameter 5-28
RADIUS-Session 5-29
Rex 5-29
WiMAX 5-30
Diameter 5-30
M3UA 5-37
Session Managers 5-38
Session Creation 5-41
Session Notes 5-41
Soft Group Session Limit 5-42
Session Correlation Based on User-Defined Attributes 5-43
Resource Managers 5-43
Types of Resource Managers 5-44
Group-Session-Limit 5-45
Home-Agent 5-45
Home-Agent-IPv6 5-45
IP-Dynamic 5-45
IP-Per-NAS-Port 5-46
IPX-Dynamic 5-46
Session-Cache 5-46
Subnet-Dynamic 5-47
User-Session-Limit 5-48
USR-VPN 5-48
Dynamic-DNS 5-49
Remote-IP-Dynamic 5-49
Remote-User-Session-Limit 5-49

Remote-Group-Session-Limit 5-49
Remote-Session-Cache 5-49
3GPP 5-49
Profiles 5-50
Attributes 5-50
Translations 5-51
TranslationGroups 5-51
Remote Servers 5-52
Types of Protocols 5-53
Dynamic DNS 5-53
LDAP 5-54
Map-Gateway 5-57
Sigtran 5-58
ODBC 5-59
ODBC-Accounting 5-61
OCI 5-61
OCI-Accounting 5-62
Prepaid-CRB 5-62
Prepaid-IS835C 5-62
RADIUS 5-62
Diameter 5-63
REST 5-64
SIGTRAN-M3UA 5-65
Rules 5-65
Fast Rules 5-65
Advanced 5-65
RemoteODBCSessionServer 5-81
Using the RequireNASsBehindProxyBeInClientList Property 5-82
Advance Duplicate Detection Feature 5-83
Invalid EAP Packet Processing 5-83
Ports 5-84
Interfaces 5-84
Reply Messages 5-84
Attribute Dictionary 5-86
Types 5-86
Vendor Attributes 5-87
SNMP 5-87
Diameter 5-88
Configuring Diameter Transport Management Properties 5-89

Configuring Diameter Session Management 5-91
Configuring Diameter Application 5-92
Configuring Diameter Commands 5-93
Configuring Diameter Dictionary 5-99

CHAPTER 6 **Configuring Local Authentication and Authorization** 6-1

Configuring a Local Service and UserList 6-1
Configuring a Local Service 6-2
Configuring a Userlist 6-3
Configuring Cisco Prime Access Registrar to Use the Local Service For AA 6-3
Activating the Configuration 6-4
Troubleshooting the Local Service and UserList Configuration 6-4
Verifying the Configuration 6-4
Configuring Return Attributes and Check-Items 6-6
Configuring Per User Return Attributes 6-6
Configuring Per User Check-Items 6-7
Verifying the Per User Return Attributes and Check-Items Configuration 6-7
Configuring Profiles to Group Attributes 6-8
Configuring Return Attributes and Check-Items Using UserGroup 6-9
Return Attribute Precedence 6-10
aregcmd Command Performance 6-10
UserDefined1 Property 6-11
Access-Request Logging 6-11

CHAPTER 7 **Using Extension Points** 7-1

Determining the Goal of the Script 7-2
Writing the Script 7-2
Choosing the Type of Script 7-3
Request Dictionary Script 7-3
Response Dictionary Script 7-4
Environment Dictionary Script 7-4
Adding the Script Definition 7-5
Adding the Example Script Definition 7-5
Choosing the Scripting Point 7-6
Testing the Script 7-6
About the Tcl/Tk 8.3 Engine 7-6
Cisco Prime Access Registrar Scripts 7-6
ACME 7-8

AltigaIncomingScript 7-8
AltigaOutgoingScript 7-8
ANAAAOutgoing 7-8
AscendIncomingScript 7-8
AscendOutgoingScript 7-9
AuthorizePPP 7-9
AuthorizeService 7-9
AuthorizeSLIP 7-9
AuthorizeTelnet 7-9
CabletronIncoming 7-9
CabletronOutgoing 7-9
CiscoIncoming 7-9
CiscoOutgoing 7-10
CiscoWithODAPIncomingScript 7-10
ExecCLIDRule 7-10
ExecDNISRule 7-10
ExecFilterRule 7-10
ExecNASIPRule 7-10
ExecRealmRule 7-10
ExecTimeRule 7-11
LDAPOutage 7-11
MapSourceIPAddress 7-11
ParseAAARealm 7-11
ParseAAASRealm 7-12
ParseAARealm 7-12
ParseAASRealm 7-12
ParseProxyHints 7-12
ParseServiceAndAAARealmHints 7-12
ParseServiceAndAAASRealmHints 7-12
ParseServiceAndAARealmHints 7-13
ParseServiceAndAASRealmHints 7-13
ParseServiceAndProxyHints 7-13
ParseServiceHints 7-13
ParseTranslationGroupsByCLID 7-13
ParseTranslationGroupsByDNIS 7-13
ParseTranslationGroupsByRealm 7-14
UseCLIDAsSessionKey 7-14
USRIncomingScript 7-14
USRIncomingScript-IgnoreAccountingSignature 7-14
USROutgoingScript 7-14

Internal Scripts 7-14
Blacklisting Script 7-16
IMSI-Based Blacklisting 7-16
IP-Based Blacklisting 7-17
Extension Points in Cisco Prime Access Registrar 7-17

CHAPTER 8 **Testing the RADIUS Server 8-1**
radclient Command Syntax 8-1
Working with Packets 8-2
Creating Packets 8-2
Creating CHAP Access-Request Packets 8-3
Viewing Packets 8-3
Sending Packets 8-3
Creating Empty Packets 8-4
Setting Packet Fields 8-4
Reading Packet Fields 8-5
Deleting Packets 8-5
Attributes 8-5
Creating Attributes 8-5
Setting Multivalued Attributes 8-6
Viewing Attributes 8-6
Getting Attribute Information 8-7
Deleting Attributes 8-7
Using the radclient Command 8-7
Example 1 8-7
Example 2 8-8
Example 3 8-9
Using radclient Test Commands 8-10
radclient Variables 8-10
Using timetest 8-10
Using callsPerSecond 8-11
Additional radclient Variables 8-12

CHAPTER 9 **Using Trusted ID Authorization with SESM 9-1**
Trusted ID Operational Overview 9-1
Configuration Overview 9-2
Request Processing 9-2
Session Cache Life Cycle 9-3
Configuration Restrictions 9-3

Software Requirements 9-4
Installing Cisco Prime Access Registrar 9-4
Running the TrustedIdInstall Program 9-4
Using the TrustedIdInstall.bin GUI 9-4
Using the TrustedIdInstall Command Line 9-8
Configuring Cisco Prime Access Registrar for Trusted Identity with SESM 9-12
Configuring the RADIUS Ports 9-12
Configuring NAS Clients 9-13
Configuring AAA and SPE Services 9-13
Configuration Imported by TrustedIdInstall Program 9-13
/Radius 9-14
/radius/services/spe 9-14
/radius/services/trusted-id 9-14
/Radius/SessionManagers/session-cache/ 9-14
/radius/ResourceManagers/session-cache 9-14
/radius/advanced/ 9-14
/Radius/Scripts/ChangeServiceType 9-15
Configuring EAP-MD5 Authentication 9-15
Creating the CheckEap.tcl Script 9-15
Adding the CheckEap.tcl Script 9-16
Using the CheckEap.tcl Script 9-16
Adding the EAP-MD5 Authentication Service 9-17
Adding an LDAP Remote Server 9-17
Adding an LDAP Service 9-18
Saving the Configuration and Reloading the Server 9-19
Cisco SSG VSAs in Cisco Prime Access Registrar Dictionary 9-20

CHAPTER 10 **Using the REX Accounting Script** 10-1
Building and Installing the REX Accounting Script 10-1
Configuring the Rex Accounting Script 10-2
Specifying REX Accounting Script Options 10-4
Example Script Object 10-5

Overview

Prime Access Registrar is a 3GPP-compliant, 64-bit carrier-class RADIUS (Remote Authentication Dial-In User Service)/Diameter server that enables multiple dial-in Network Access Server (NAS) devices to share a common authentication, authorization, and accounting database.

Prime Access Registrar handles the following tasks:

- Authentication—determines the identity of users and whether they can be allowed to access the network
- Authorization—determines the level of network services available to authenticated users after they are connected
- Accounting—keeps track of each user's network activity
- Session and resource management—tracks user sessions and allocates dynamic resources

This chapter contains the following sections:

- Session Management, page 1-1
- Script Processing Hierarchy, page 1-4
- RADIUS Protocol, page 1-5

Session Management

The Session Management feature requires the client (NAS or proxy) to send all RADIUS accounting requests to the Prime Access Registrar server performing session management. (The only exception is if the clients are USR/3Com Network Access Servers configured to use the USR/3Com RADIUS resource management feature.) This information is used to keep track of user sessions, and the resources allocated to those sessions.

When another accounting RADIUS server needs this accounting information, the Prime Access Registrar server performing session management might proxy it to this second server.

The **count-sessions /radius all** command helps to count the total sessions in Prime Access Registrar. The options are similar to the query-session command options. The query-session command displays cached attributes in addition to session details.

Table 1-1 describes how Prime Access Registrar handles session management.

Table 1-1 Session Management Processing

Action	Explanation
Determines whether to perform session management.	The session management defined in the Environment dictionary variable **Session-Manager**.
	The session management name referred to in **/Radius/DefaultSessionManager**.
Performs session management.	Selects Session Manager as defined in **/Radius/SessionManagers/**<*Name*>.

This section contains the following topics:

- Failover by the NAS and Session Management
- Cross Server Session and Resource Management

Failover by the NAS and Session Management

When a Network Access Server's primary RADIUS server is performing session management, and the NAS determines the server is not responding and begins sending requests to its secondary RADIUS server, the following occurs:

- The secondary server will not know about the current active sessions that are maintained on the primary server. Any resources managed by the secondary server must be distinct from those managed by the primary server, otherwise it will be possible to have two sessions with the same resources (for example, two sessions with the same IP address).
- The primary server will miss important information that allows it to maintain a correct model of what sessions are currently active (because the authentication and accounting requests are being sent to the secondary server). This means when the primary server comes back online and the NAS begins using it, its knowledge of what sessions are active will be out-of-date and the resources for those sessions are allocated even if they are free to allocate to someone else.

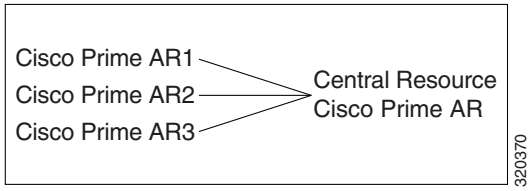
  For example, the user-session-limit resource might reject new sessions because the primary server does not know some of the users using the resource logged out while the primary server was offline. It might be necessary to release sessions manually using the **aregcmd** command **release-session**.

Note It might be possible to avoid this situation by having a disk drive shared between two systems with the second RADIUS server started up once the primary server has been determined to be offline. For more information on this setup, contact Technical Support.

Cross Server Session and Resource Management

Prime Access Registrar can manage sessions and resources across AAA Server boundaries. A session can be created by an Access-Request sent to Prime AR1, and it can be removed by an Accounting-Stop request sent to Prime AR2, as shown in Figure 1-1. This enables accurate tracking of User and Group session limits across multiple AAA Servers, and IP addresses allocated to sessions are managed in one place.

Figure 1-1 Multiple Prime Access Registrar Servers



All resources that must be shared cross multiple front line Prime Access Registrars are configured in the Central Resource Prime Access Registrar. Resources that are not shared can still be configured at each front line Prime Access Registrar.

When the front line Prime Access Registrar receives the access-request, it does the regular AA processing. If the packet is not rejected and a Central Resource Prime Access Registrar is also configured, the front line Prime Access Registrar will proxy the packet[1] to the configured Central Resource Prime Access Registrar. If the Central Resource Prime Access Registrar returns the requested resources, the process continues to the local session management (if local session manager is configured) for allocating any local resources. If the Central Resource Prime Access Registrar cannot allocate the requested resource, the packet is rejected.

When the Accounting-Stop packet arrives at the frontline Prime Access Registrar, Prime Access Registrar does the regular accounting processing. Then, if the front line Prime Access Registrar is configured to use Central Resource Prime Access Registrar, a proxy packet will be sent to Central Resource Prime Access Registrar for it to release all the allocated resources for this session. After that, any locally allocated resources are released by the local session manager.

Session-Service Service Step and Radius-Session Service

A new Service step has been added in the processing of Access-Request and Accounting packets. This is an additional step after the AA processing for Access packet or Accounting processing for Accounting packet, but before the local session management processing. The Session-Service should have a service type of radius-session.

An environment variable Session-Service is introduced to determine the Session-Service dynamically. You can use a script or the rule engine to set the Session-Service environment variable.

Configure Front Line Cisco Prime Access Registrar

To use a Central Resource server, the DefaultSessionService property must be set or the Session-Service environment variable must be set through a script or the rule engine. The value in the Session-Service variable overrides the DefaultSessionService.

The configuration parameters for a Session-Service service type are the same as those for configuring a radius service type for proxy, except the service type is ***radius-session***.

The configuration for a Session-Service Remote Server is the same as configuring a proxy server.

```
[ //localhost/Radius ]
    Name = Radius
    Description =
    Version = 7.2
    IncomingScript =
    OutgoingScript =
    DefaultAuthenticationService = local-users
```

1. The proxy packet is actually a resource allocation request, not an Access Request.

```
    DefaultAuthorizationService = local-users
    DefaultAccountingService = local-file
    DefaultSessionService = Remote-Session-Service
    DefaultSessionManager = session-mgr-1

[ //localhost/Radius/Services ]
    Remote-Session-Service/
       Name = Remote-Session-Service
       Description =
       Type = radius-session
       IncomingScript =
       OutgoingScript =
       OutagePolicy = RejectAll
       OutageScript =
       MultipleServersPolicy = Failover
       RemoteServers/
        1. central-server

[ //localhost/Radius/RemoteServers ]
    central-server/
       Name = central-server
       Description =
       Protocol = RADIUS
       IPAddress = 209.165.200.224
       Port = 1812
       ReactivateTimerInterval = 300000
       SharedSecret = secret
       Vendor =
       IncomingScript =
       OutgoingScript =
       MaxTries = 3
       InitialTimeout = 2000
       AccountingPort = 1813
```

Configure Central Prime Access Registrar

Resources at the Central Resource server are configured the same way as local resources are configured. These resources are local resources from the Central Resource server's point of view.

Script Processing Hierarchy

For request packets, the script processing order is from the most general to the most specific. For response packets, the processing order is from the most specific to the most general.

Table 1-2, Table 1-3, and Table 1-4 show the overall processing order and flow:
(1-6) Incoming Scripts, (7-11) Authentication/Authorization Scripts, and (12-17) Outgoing Scripts.

Note The client and the NAS can be the same entity, except when the immediate client is acting as a proxy for the actual NAS.

Table 1-2 Prime Access Registrar Processing Hierarchy for Incoming Scripts

Overall Flow Sequence	Incoming Scripts
1)	Radius.
2)	Vendor of the immediate client.
3)	Immediate client.
4)	Vendor of the specific NAS.
5)	Specific NAS.
6)	Service.

Table 1-3 Prime Access Registrar Processing Hierarchy for Authentication/Authorization Scripts

Overall Flow Sequence	Authentication/Authorization Scripts
7)	Group Authentication.
8)	User Authentication.
9)	Group Authorization.
10)	User Authorization.
11)	Session Management.

Table 1-4 Prime Access Registrar Processing Hierarchy for Outgoing Script

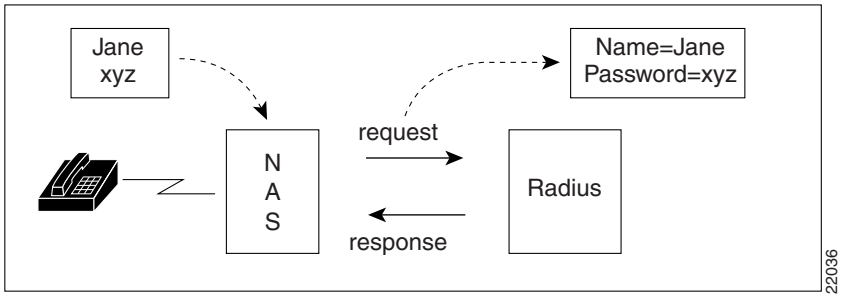
Overall Flow Sequence	Outgoing Scripts
12)	Service.
13)	Specific NAS.
14)	Vendor of the specific NAS.
15)	Immediate client.
16)	Vendor of the immediate client.
17)	Radius.

RADIUS Protocol

Prime Access Registrar is based on a client/server model, which supports AAA (authentication, authorization, and accounting). The *client* is the Network Access Server (NAS) and the *server* is Prime Access Registrar. The client passes user information on to the RADIUS server and acts on the response it receives. The *server*, on the other hand, is responsible for receiving user access requests, authenticating and authorizing users, and returning all of the necessary configuration information the client can then pass on to the user.

The protocol is a simple packet exchange in which the NAS sends a request packet to the Prime Access Registrar with a name and a password. Prime Access Registrar looks up the name and password to verify it is correct, determines for which dynamic resources the user is authorized, then returns an accept packet that contains configuration information for the user session (Figure 1-2).

Figure 1-2 Packet Exchange Between User, NAS, and RADIUS



Prime Access Registrar can also reject the packet if it needs to deny network access to the user. Or, Prime Access Registrar can issue a challenge that the NAS sends to the user, who then creates the proper response and returns it to the NAS, which forwards the challenge response to Prime Access Registrar in a second request packet.

In order to ensure network security, the client and server use a *shared secret*, which is a string they both know, but which is never sent over the network. User passwords are also encrypted between the client and the server to protect the network from unauthorized access.

This section contains the following topics:

- Steps to Connection

Steps to Connection

Three participants exist in this interaction: the user, the NAS, and the RADIUS server.

Setting Up the Connection

To describe the receipt of an access request through the sending of an access response:

Step 1 The user, at a remote location such as a branch office or at home, dials into the NAS, and supplies a name and password.

Step 2 The NAS picks up the call and begins negotiating the session.

a. The NAS receives the name and password.

b. The NAS formats this information into an Access-Request packet.

c. The NAS sends the packet on to the Prime Access Registrar server.

Step 3 The Prime Access Registrar server determines what hardware sent the request (NAS) and parses the packet.

a. It sets up the Request dictionary based on the packet information.

b. It runs any incoming scripts, which are user-written extensions to Prime Access Registrar. An incoming script can examine and change the attributes of the request packet or the environment variables, which can affect subsequent processing.

c. Based on the scripts or the defaults, it chooses a service to authenticate and/or authorize the user.

Step 4 Prime Access Registrar's authentication service verifies the username and password is in its database. Or, Prime Access Registrar delegates the authentication (as a proxy) to another RADIUS server, an LDAP, or TACACS server.

Step 5 Prime Access Registrar's authorization service creates the response with the appropriate attributes for the user's session and puts it in the Response dictionary.

Step 6 If you are using Prime Access Registrar session management at your site, the Session Manager calls the appropriate Resource Managers that allocate dynamic resources for this session.

Step 7 Prime Access Registrar runs any outgoing scripts to change the attributes of the response packet.

Step 8 Prime Access Registrar formats the response based on the Response dictionary and sends it back to the client (NAS).

Step 9 The NAS receives the response and communicates with the user, which might include sending the user an IP address to indicate the connection has been successfully established.

Related Documentation

For a complete list of Cisco Prime Access Registrar documentation, see the *Cisco Prime Access Registrar Documentation Overview*.

Note We sometimes update the documentation after original publication. Therefore, you should also review the documentation on Cisco.com for any updates.

Communications, Services, and Additional Information

- To receive timely, relevant information from Cisco, sign up at Cisco Profile Manager.
- To get the business impact you're looking for with the technologies that matter, visit Cisco Services.
- To submit a service request, visit Cisco Support.
- To discover and browse secure, validated enterprise-class apps, products, solutions and services, visit Cisco Marketplace.
- To obtain general networking, training, and certification titles, visit Cisco Press.
- To find warranty information for a specific product or product family, access Cisco Warranty Finder.

Cisco Bug Search Tool

Cisco Bug Search Tool (BST) is a web-based tool that acts as a gateway to the Cisco bug tracking system that maintains a comprehensive list of defects and vulnerabilities in Cisco products and software. BST provides you with detailed defect information about your products and software.

Configuring Cisco Prime Access Registrar

This chapter describes how to configure Cisco Prime Access Registrar 8.0. Prime Access Registrar is very flexible. You can choose to configure it in many different ways. In addition, you can write scripts that can be invoked at different points during the processing of incoming requests and/or outgoing responses.

Before you can take advantage of this flexibility, it helps to configure a simple site. This chapter describes that process. It specifically describes a site that has the following characteristics:

- Uses a single user list for all of its users
- Writes all of its accounting information to a file
- Does not use session management to allocate or track dynamic resources

This chapter contains the following sections:

- Using aregcmd, page 2-1
- Configuring a Basic Site, page 2-2
- Configuring Accounting, page 2-15
- Configuring SNMP, page 2-15

Using aregcmd

To configure Prime Access Registrar, use the **aregcmd** commands, which are command-line based configuration tools. These commands allow you to set any Prime Access Registrar configuration option, as well as, start and stop the Prime Access Registrar RADIUS server and check its statistics.

This topic contains the following sections:

- General Command Syntax, page 2-1
- aregcmd Commands, page 2-2

General Command Syntax

Prime Access Registrar stores its configuration information in a hierarchy. Using the **aregcmd** command **cd** (change directory), you can move through this information in the same manner as you would through a hierarchical file system. You can also supply full path names to these commands to affect another part of the hierarchy, and thus avoid explicitly using the **cd** command to change to that part of the tree.

The **aregcmd** commands are case *insensitive*, which means that you can use upper or lowercase letters to designate elements. In addition, when you reference existing elements in the configuration, you only need to specify enough of the element's name to distinguish it from the other elements at that level. For example, instead of entering **cd Administrators**, you can enter **cd ad** if no other element at the current level begins with *ad*.

You can use Prime Access Registrar's command completion feature to see what commands are possible from your current directory location in the Prime Access Registrar server hierarchy by pressing the Tab key. You can also press the Tab key after entering a command to see which objects you might want to manage.

The **aregcmd** commands are command-line order dependent; that is, the arguments are interpreted based on their position on the command line. To indicate an empty string as a place holder on the command line, use either two single quotes ('') or two double quotes (""). In addition, if you use any arguments that contain spaces, make sure to quote the arguments.

aregcmd Commands

The **aregcmd** commands can be grouped into the following categories:

- Navigation commands—navigates within the Prime Access Registrar hierarchy; commands include **cd**, **ls**, **pwd**, **next**, **prev**, **filter**, and **find**.
- Object commands—adds or deletes objects; commands include **add** and **delete**.
- Property commands—changes the value of properties; commands include **set**, **unset**, and **insert**.
- Server commands—manages the server; commands include **save**, **validate**, **start**, **stop**, **reload**, **status**, **stats**, **dia-stats**, and **trace**.
- Application commands—allows user access to the application; commands include **login**, **logout, exit**, **quit**, and **help**.
- Session management commands—queries the server about sessions, release active sessions, or count the number of sessions; commands include **query-sessions**, **release-sessions,** and **count-sessions**.

This chapter uses only a few of the above commands to configure the Prime Access Registrar RADIUS server. For more information about all the **aregcmd** commands, see Chapter 4, "Setting the CPAR Configurable Option."

Configuring a Basic Site

The simplest RADIUS or Diameter server configuration is a site that uses a single user list for all its users, writes its accounting information to a file, and does not use session management to allocate dynamic resources.

To configure such a site, do the following:

1. Run the **aregcmd** command on your Prime Access Registrar machine.
2. Configure the Prime Access Registrar server settings, such as the server name and the server defaults. For example, with RADIUS object.
3. Add users by copying the sample users.
4. Configure the Network Access Server (NAS) clients and proxies that communicate with Prime Access Registrar.
5. Change profile attributes as needed.

6. Save your changes and reload your Prime Access Registrar RADIUS server.

This topic contains the following sections:

- Running aregcmd, page 2-3
- Configuring Prime Access Registrar Server Settings, page 2-4
- Displaying the UserLists, page 2-7
- Displaying UserGroups, page 2-9
- Configuring Clients, page 2-10
- Configuring Profiles, page 2-11
- Validating and Using Your Changes, page 2-12
- Testing Your Configuration, page 2-13
- Troubleshooting Your Configuration, page 2-14

Running aregcmd

aregcmd is the command-line interface program used to configure the Prime Access Registrar server. The **aregcmd** program is located in **$INSTALL/bin**.

Step 1 Run the **aregcmd** command:

aregcmd

Step 2 When asked for "Cluster," press **Enter**.

Step 3 Enter your administrator name and password.

When you install Prime Access Registrar software, the installation process creates a default administrator called **admin** with the password **aicuser**.

Prime Access Registrar allows you to perform the following:

- Change the administrator's password. See Changing the Administrator's Password, page 2-3.
- Add additional administrators. See Creating Additional Administrators, page 2-4.

Changing the Administrator's Password

The administrator ID **admin** and password **aicuser** are default settings for all releases of Prime Access Registrar software. For security purposes, you should change the password for **admin** at your earliest convenience.

To change the administrator's password:

Step 1 Use the **cd** command to change to the **Administrators** level. Prime Access Registrar displays the contents of the **Administrators** object.

cd //localhost/Administrators

Step 2 Use the **cd** command to change to **admin**:

cd admin

```
[ //localhost/Administrators ]
    Entries 1 to 1 from 1 total entries
    Current filter: <all>
    admin/
```

Step 3 Use the **set** command to change the administrator's password. You enter the password on the command line in readable form, however, Prime Access Registrar displays it as encrypted.

The following example changes the password to `345`. You are asked to reenter it for confirmation.

set Password 345

Optionally, use the **set** command to change the description of the **admin** administrator.

set Description local

Step 4 Use the **ls** command to display the changed admin.

ls

Creating Additional Administrators

Use the **add** command to add additional administrators.

Step 1 Use the **cd** command to change to the **Administrators** level:

cd /Administrators

Step 2 Use the **add** command and specify the name of the administrator, an optional description, and a password.

The following example adds the administrator `jane`, description `testadmin`, and password `123`:

add jane testadmin 123

Step 3 Use the **ls** command to display the properties of the new administrator:

ls

Configuring Prime Access Registrar Server Settings

The top level of the Prime Access Registrar RADIUS server is the RADIUS object itself. It specifies the name of the server and other parameters. In configuring this site, you only need to change a few of these properties.

```
[ //localhost/Radius ]
    Name = Radius
    Description =
    Version = 7.2.0.0
```

```
IncomingScript~ =
OutgoingScript~ =
DefaultAuthenticationService~ = local-users
DefaultAuthorizationService~ = local-users
DefaultAccountingService~ = local-file
DefaultSessionService~ =
DefaultSessionManager~ = session-mgr-1
UserLists/
UserGroups/
Policies/
Clients/
Vendors/
Scripts/
Services/
SessionManagers/
ResourceManagers/
Profiles/
Rules/
Translations/
TranslationGroups/
RemoteServers/
CommandSets/
DeviceAccessRules/
GroupServers/
FastRules/
Advanced/
Replication/
```

Note For session managers, user groups, user lists, and profiles, the attributes defined must match the protocol of the incoming packet. For example, if the incoming packet is a Diameter packet, the attributes defined must be specific to Diameter or common to both RADIUS and Diameter. Similarly, if the incoming packet is a RADIUS packet, the attributes defined must be specific to RADIUS or common to both RADIUS and Diameter. Otherwise, the incoming packet will not be processed.

This topic contains the following sections:

- Checking the System-Level Defaults, page 2-5
- Checking the Server's Health, page 2-6
- Selecting Ports to Use, page 2-6

Checking the System-Level Defaults

Because this site does not use incoming or outgoing scripts, you do not need to change the scripts' properties (IncomingScript and OutgoingScript).

Since the default authentication and authorization properties specify a single user list, you can leave these unchanged as well (DefaultAuthenticationService and DefaultAuthorizationService). And because you have decided to use a file for accounting information, you can leave this property unchanged (DefaultAccountingService).

Session management, however, is on by default (DefaultSessionManager). If you do not want to use session management, you must disable it. Use the **set** command, enter *DefaultSessionManager*, then specify an empty string by entering a set of double quotes:

set DefaultSessionManager ""

Note When you do not want Prime Access Registrar to monitor resources for user sessions, you should disable session management because using it affects your server performance.

You have now configured some of the properties for the RADIUS server. The next step is to add users.

Checking the Server's Health

To check the server's health, use the **aregcmd** command **status**. The following issues decrement the server's health:

- Rejection of an Access-Request

Note One of the parameters in the calculation of the Prime Access Registrar server's health is the percentage of responses to Access-Accepts that are rejections. In a healthy environment, the rejection percentage will be fairly low. An extremely high percentage of rejections could be an indication of a Denial of Service attack.

- Configuration errors
- Running out of memory
- Errors reading from the network
- Dropping packets that cannot be read (because the server ran out of memory)
- Errors writing to the network.

Prime Access Registrar logs all of these conditions. Sending a successful response to any packet increments the server's health.

Selecting Ports to Use

By default, Prime Access Registrar uses well-known Linux ports 1812 and 1813 for TCP/IP communications. Prime Access Registrar can be configured to use other ports, if necessary. You can add them to the list of ports to use.

To configure Prime Access Registrar to use ports other than the default ports, complete the following steps:

Step 1 Change directory to **/Radius/Advanced/Ports**.

cd /Radius/Advanced/Ports

```
[ //localhost/Radius/Advanced/Ports ]
```

Step 2 Use the **add** command (twice) to add ports in pairs. (The **ls** is entered to show the results of the **add** command.)

add 1812

add 1813

ls

```
[ //localhost/Radius/Advanced/Ports ]
    Entries 1 to 2 from 2 total entries
    Current filter: <all>

    1812/
    1813/
```

Note After modifying Access Registrar's default ports setting, to continue using the existing ports, you must add them to the list of ports in **/Radius/Advanced/Ports**.

Step 3 Enter the **save** and **reload** commands to affect, validate, and save your modifications to the Prime Access Registrar server configuration.

save

```
Validating //localhost...
Saving //localhost...
```

reload

```
Reloading Server 'Radius'...
Server 'Radius' is Running, its health is 10 out of 10
```

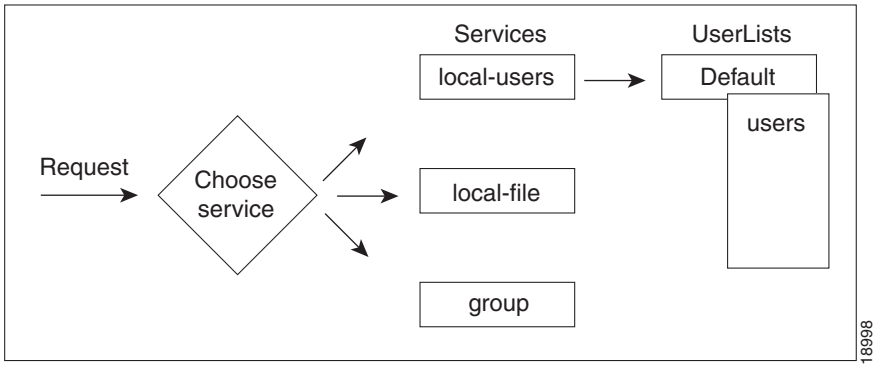
Displaying the UserLists

The first subobject in a RADIUS or Diameter hierarchy that you can configure is the Userlists. The UserLists object contains all of the individual UserLists, which in turn contain the specific users.

When Prime Access Registrar receives an Access-Request, it directs it to an authentication and/or authorization Service. If the Service has its type set to *local*, the Service looks up the user's entry in the specific **UserList**, and authenticates and/or authorizes the user.

Prime Access Registrar, by default, specifies a Service called **local-users** that has the type **local** and uses the **Default** UserList (Figure 2-1).

Figure 2-1 Choosing Appropriate Services



This topic contains the following sections:

- Displaying the Default UserList, page 2-8
- Adding Users to UserLists, page 2-8
- Deleting Users, page 2-9

Displaying the Default UserList

Step 1 Use the **cd** command to change to **UserLists/Default**:

cd /Radius/Userlists/Default

Step 2 Use the **ls -R** command to display the properties of the three users:

ls -R

Prime Access Registrar displays the three sample users:

- `bob` who is configured as a PPP user
- `jane` who is configured as a Telnet user
- `joe` who is configured as either a PPP or Telnet user depending on how he logs in.

Adding Users to UserLists

Use the **aregcmd** command **add** to create a user under a UserList.

To add a user:

Step 1 Use the **add** command to specify the name of a user and an optional description on one command line.

add jane

```
Added jane
```

Step 2 Change directory to **jane**.

cd jane

```
[ //localhost/Radius/UserLists/Default/jane ]
    Name = jane
    Description =
    Password = <encrypted>
    Enabled = TRUE
    Group~ = Telnet-users
    BaseProfile~ =
    AuthenticationScript~ =
    AuthorizationScript~ =
    UserDefined1 =
    AllowNullPassword = FALSE
    Attributes/
```

```
                CheckItems/
```

Step 3 Use the **set** command to provide a password for user **jane**.

set password jane

```
Set Password <encrypted>
```

Note When using the **aregcmd** command, you can use the **add** command and specify all of the properties, or you can use the **add** command to create the object, and then use the **set** command and property name to set the property. For an example of using the **set** command, see the "Adding a NAS" section on page 2-10.

Deleting Users

To delete the sample users, or if you want to remove a user you have added, use the **delete** command.

From the appropriate UserList, use the **delete** command, and specify the name of the user you want to delete. For example, to delete user `beth` from the Default UserList, enter:

cd /Radius/UserLists/Default

delete beth

Displaying UserGroups

The UserGroups object contains the specific UserGroups. Specific UserGroups allow you to maintain common authentication and authorization attributes in one location, and then have users reference them. By having a central location for attributes, you can make modifications in one place instead of having to make individual changes throughout your user community.

Prime Access Registrar has three default UserGroups:

- *Default*—uses the script **AuthorizeService** to determine the type of service to provide the user.
- *PPP-users*—uses the BaseProfile **default-PPP-users** to specify the attributes of PPP service to provide the user. The BaseProfile **default-PPP-users** contains the attributes that are added to the response dictionary as part of the authorization. For more information about Profiles, see the "Configuring Profiles" section on page 2-11.
- *Telnet-users*—uses the BaseProfile **default-Telnet-users** to specify the attributes of Telnet service to provide the user. The BaseProfile **default-Telnet-users** contains the attributes that are added to the response dictionary as part of the authorization.

For this basic site, you do not need to change these UserGroups. You can, however, use the **add** or **delete** commands to add or delete groups.

Configuring Clients

The Clients object contains all NAS and proxies that communicate directly with Prime Access Registrar. Each client must have an entry in the Clients list, because each NAS and proxy share a secret with the RADIUS server, which is used to encrypt passwords and to sign responses. See Adding a NAS, page 2-10, for more information on adding a NAS in Prime Access Registrar.

Note If you are just testing Prime Access Registrar with the **radclient** command, the only client you need is **localhost**. The **localhost** client is available in the sample configuration. For more information about using the **radclient** command, see the "Using radclient" section on page 2-13.

Adding a NAS

You must add your specific NAS from both ends of the connection. That is, you must add Prime Access Registrar for your NAS, and you must add your NAS for Prime Access Registrar.

To add a NAS in Prime Access Registrar:

Step 1 Use the **cd** command to change to the **Clients** level:

cd /Radius/Clients

Step 2 Use the **add** command to add the NAS: `QuickExampleNAS`:

add QuickExampleNAS

Step 3 Use the **cd** command to change directory to the **QuickExampleNAS** directory:

cd /Radius/Clients/QuickExampleNAS

Step 4 Use the **set** command to specify the description `WestOffice`, the IP address `196.168.1.92`, the shared secret of `xyz`, and the Type as `NAS`.

set Description WestOffice

set IPAddress 209.165.200.225

set SharedSecret xyz

set Type NAS

set Vendor USR

set IncomingScript ParseServiceHints

EnableDynamicAuthorization TRUE

EnableNotifications TRUE

The script, **ParseServiceHints**, checks the username for **%PPP** or **%SLIP**. It uses these tags to modify the request so it appears to the RADIUS server that the NAS requested that service.

Note When you are using a different NAS than the one in the example, or when you are adding NAS proprietary attributes, see *Chapter 5, "Configuring and Monitoring the RADIUS Server,"* for more information about configuring Client and Vendor objects.

Configure your NAS, using your vendor's documentation. Make sure both your NAS and the Client specification have the same shared secret.

Configuring Profiles

The Profiles object allows you to set specific RFC-defined attributes that Prime Access Registrar returns in the Access-Accept response. You can use profiles to group attributes that belong together, such as attributes that are appropriate for a particular class of PPP or Telnet user. You can reference profiles by name from either the UserGroup or the user properties. The sample users, mentioned earlier in this chapter, reference the following Prime Access Registrar profiles:

- **default-PPP-users**—specifies the appropriate attributes for PPP service
- **default-SLIP-users**—specifies the appropriate attributes for SLIP service
- **default-Telnet-users**—specifies the appropriate attributes for Telnet service.

This topic contains the following sections:

- Setting RADIUS / Diameter Attributes, page 2-11
- Adding Multiple Cisco AV Pairs, page 2-12

Setting RADIUS / Diameter Attributes

When you want to set an attribute to a profile, use the following command syntax:

set <attribute> <value>

This syntax assigns a new value to the named attribute. The following example sets the attribute Service-Type to Framed:

Step 1 Use the **cd** command to change to the appropriate profile and attribute.

cd /Radius/Profiles/Default-PPP-users/Attributes

Step 2 Use the **set** command to assign a value to the named attribute.

set Service-Type Framed

When you need to set an attribute to a value that includes a space, you must double-quote the value, as in the following:

set Framed-Routing "192.168.1.0/24 192.168.1.1"

Adding Multiple Cisco AV Pairs

When you want to add multiple values to the same attribute in a profile, use the following command syntax:

set <attribute> <value1> < value2> < value3>

The AV pairs cannot be added one at a time or each subsequent command will overwrite the previous value. For example, consider the following command entry:

set Cisco-AVpair "vpdn:12tp-tunnel-password=XYZ" "vpdn:tunnel-type=12tp" "vpdn:tunnel-id=telemar" "vpdn:ip-addresses=209.165.200.225"

ls

```
Cisco-Avpair = vpdn:12tp-tunnel-password=XYZ
Cisco-Avpair = vpdn:tunnel-type=12tp
Cisco-Avpair = vpdn:tunnel-id=telemar
Cisco-Avpair = vpdn:ip-addresses=209.165.200.225
```

Note The example above is for explanation only; not all attributes and properties are listed.

Validating and Using Your Changes

After you have finished configuring your Prime Access Registrar server, you must save your changes. Saving your changes causes Prime Access Registrar to validate your changes and, if there were no errors, commit them to the configuration database.

Using the **save** command, however, does not automatically update your server. To update your server you must use the **reload** command. The **reload** command stops your server if it is running, and then restarts the server, which causes Prime Access Registrar to reread the configuration database.

You must **save** and **reload** your configuration changes in order for them to take effect in the Prime Access Registrar server. For more information, see Saving and Reloading, page 2-12.

Saving and Reloading

From anywhere in the radius object hierarchy, enter the **save** and **reload** commands.

Step 1 Use the **save** command to save your changes:

save

Step 2 Use the **reload** command to reload your server.

reload

Testing Your Configuration

Now that you have configured some users and a NAS, you are ready to test your configuration. There are two ways you can test your site:

1. You can act as a user and dial in to your NAS, and check that you can successfully log in.
2. You can run the **radclient** command, and specify one of the default users when making a request. For more information, see Using radclient, page 2-13.

Using radclient

You can use the **radclient** command **simple** to create and send a packet. The following example creates an Access-Request packet for user `john` with password `john`, and the packet identifier `p001`. It displays the packet before sending it. It uses the **send** command to send the packet, which displays the response packet object identifier, `p002`. Then, the example shows how to display the contents of the response packet.

Step 1 Run the **radclient** command.

. /radclient -s

Step 2 The **radclient** command prompts you for the administrator's username and password (as defined in the Prime Access Registrar configuration). Use **admin** for the admin name, and **aicuser** for the password.

```
Cisco Prime Access Registrar 7.2.0 RADIUS Test Client
Copyright (C) 1995-2016 by Cisco Systems, Inc.  All rights reserved.
Logging in to localhost... done.
```

Step 3 Create a simple Access-Request packet for User-Name `john` and User-Password `john`. At the prompt, enter:

simple john john

```
p001
```

The **radclient** command displays the ID of the packet `p001`.

Step 4 Enter the packet identifier:

p001

```
Packet: code = Access-Request, id = 0, length = 0, attributes =
User-Name = john
User-Password = john
NAS-Identifier = localhost
NAS-Port = 0
```

Step 5 Send the request to the default host (**localhost**), enter:

p001 send

```
p002
```

Step 6 Enter the response identifier to display the contents of the Access-Accept packet:

p002

```
Packet: code = Access-Accept, id = 1,\
length = 38, attributes =
      Login-IP-Host = 196.168.1.94
      Login-Service = Telnet
      Login-TCP-Port = 541
```

Troubleshooting Your Configuration

If you are unable to receive an Access-Accept packet from the Prime Access Registrar server, you can use the **aregcmd** command **trace** to troubleshoot your problem.

The **trace** command allows you to set the trace level on your server, which governs how much information the server logs about the contents of each packet. You can set the trace levels from zero to four. The system default is zero, which means that no information is logged. For more information, see Setting the Trace Level, page 2-14.

Setting the Trace Level

Step 1 Run the **aregcmd** command.

aregcmd

Step 2 Use the **trace** command to set the trace level to 1-5.

trace 2

Step 3 Try dialing in again.

Step 4 Use the UNIX **tail** command to view the end of the **name_radius_1_trace** log.

host% **tail -f /opt/CSCOar/logs/name_radius_1_trace**

Step 5 Read through the log to see where the request failed.

Configuring Accounting

To configure Prime Access Registrar to perform accounting, you must do the following:

1. Create a service
2. Set the service's type to file
3. Set the DefaultAccountingService field in **/Radius** to the name of the service you created

After you **save** and **reload** the Prime Access Registrar server configuration, the Prime Access Registrar server writes accounting messages to the **accounting.log** file in the **/opt/CSCOar/logs** directory. The Prime Access Registrar server stores information in the **accounting.log** file until a rollover event occurs. A rollover event is caused by the **accounting.log** file exceeding a pre-set size, a period of time transpiring, or on a scheduled date.

When the rollover event occurs, the data in **accounting.log** is stored in a file named by the prefix *accounting*, a date stamp (*yyyymmdd*), and the number of rollovers for that day. For example, **accounting-20081107-14** would be the 14th rollover on November 07, 2008.

The following shows the properties for a service called Cisco Accounting:

```
[ //localhost/Radius/Services/local-file ]
    Name = local-file
    Description =
    Type = file
    IncomingScript~ =
    OutgoingScript~ =
    OutagePolicy~ = RejectAll
    OutageScript~ =
    FilenamePrefix = accounting
    MaxFileSize = "10 Megabytes"
    MaxFileAge = "1 Day"
    RolloverSchedule =
    UseLocalTimeZone = FALSE
```

Configuring SNMP

Before you perform SNMP configuration, you must first stop the operating system-specific SNMP master agent. Then, configure your local **snmpd.conf** file located in the **/cisco-ar/ucd-snmp/share/snmp** directory. Whenever snmpd.conf is modified, you must restart the Prime Access Registrar server using the following command for the new configuration to take effect:

/etc/init.d/arserver restart

This topic contains the following sections:

- Enabling SNMP in the Cisco Prime Access Registrar Server, page 2-16
- Stopping the Master Agent, page 2-16
- Modifying the snmpd.conf File, page 2-16
- Restarting the Master Agent, page 2-18

Enabling SNMP in the Cisco Prime Access Registrar Server

To enable SNMP on the Prime Access Registrar server, launch **aregcmd** and set the **/Radius/Advanced/SNMP/Enabled** property to TRUE.

aregcmd

cd /Radius/Advanced/SNMP

```
[ //localhost/Radius/Advanced/SNMP ]
    Enabled = FALSE
    TracingEnabled = FALSE
    InputQueueHighThreshold = 90
    InputQueueLowThreshold = 60
    MasterAgentEnabled = TRUE
```

set Enabled TRUE

Stopping the Master Agent

Stop the Prime Access Registrar SNMP master agent by stopping the Prime Access Registrar server.

/opt/CSCOar/bin/arserver stop

Modifying the snmpd.conf File

snmpd.conf file is placed in the **/cisco-ar/ucd-snmp/share/snmp** directory. Use **vi** (or another text editor) to edit the **snmpd.conf** file.

The three parts of this file to modify are:

- Access Control, page 2-16
- Trap Recipient, page 2-17
- System Contact Information, page 2-18

Access Control

Access control defines who can query the system. By default, the agent responds to the ***public*** community for read-only access, if run without any configuration file in place.

The following example from the default **snmpd.conf** file shows how to configure the agent so that you can change the community names, and give yourself write access as well.

To modify the **snmpd.conf** file:

Step 1 Look for the following lines in the **snmpd.conf** file for the location in the file to make modifications:

```
###############################################################################
# Access Control
###############################################################################
```

Step 2 First map the community name (COMMUNITY) into a security name that is relevant to your site, depending on where the request is coming from:

```
#       sec.name  source          community
# For IPv4
com2sec local     localhost       private
com2sec mynetwork 10.1.9.0/24      public
# For IPv6
com2sec6 local localhost6 public
com2sec6 local 2001:420:27c1:420:214:4fff:fef8:9f3e public
```

The names are tokens that you define arbitrarily.

Step 3 Map the security names into group names:

```
# sec.model  sec.name
group MyRWGroupv1local
group MyRWGroupv2clocal
group MyRWGroupusmlocal
group MyROGroupv1 mynetwork
group MyROGroupv2c mynetwork
group MyROGroupusmmynetwork
```

Step 4 Create a view to enable the groups to have rights:

```
#           incl/excl subtree                          mask
view all    included  .1                               80
```

Step 5 Finally, grant the two groups access to the one view with different write permissions:

```
#                context sec.model sec.level match  read   write  notif
access MyROGroup ""      any       noauth    exact  all    none   none
access MyRWGroup ""      any       noauth    exact  all    all    none
```

Trap Recipient

The following example provides a sample configuration that sets up trap recipients for SNMP versions v1 and v2c.

Note Most sites use a single NMS, not two as shown below.

```
# --------------------------------------------------------------------------
trapcommunity trapcom
trapsink zubat trapcom 162
trap2sink ponyta trapcom 162
##############################################################################
```

Note **trapsink** is used in SNMP version 1; **trap2sink** is used in SNMP version 2.

trapcommunity defines the default community string to be used when sending traps. This command must appear prior to **trapsink** or **trap2sink** which use this community string.

trapsink and **trap2sink** are defined as follows:

trapsink hostname community port

trap2sink hostname community port

System Contact Information

System contact information is provided in two variables through the **snmpd.conf** file, **syslocation,** and **syscontact**.

Look for the following lines in the **snmpd.conf** file:

```
##############################################################################
# System contact information
#
syslocation Your Location, A Building, 8th Floor
syscontact A. Person <someone@somewhere.org>
```

Restarting the Master Agent

Restart the Prime Access Registrar SNMP master agent by restarting the Prime Access Registrar server.

/opt/CSCOar/bin/arserver restart

Configuring Dynamic DNS

Prime Access Registrar supports the Dynamic DNS protocol providing the ability to update DNS servers. The dynamic DNS updates contain the hostname/IP Address mapping for sessions managed by Prime Access Registrar.

You enable dynamic DNS updates by creating and configuring new Resource Managers and new RemoteServers, both of type dynamic-dns. The dynamic-dns Resource Managers specify which zones to use for the forward and reverse zones and which Remote Servers to use for those zones. The dynamic-dns Remote Servers specify how to access the DNS Servers.

Before you configure Prime Access Registrar you need to gather information about your DNS environment. For a given Resource Manager you must decide which forward zone you will be updating for sessions the resource manager will manage. Given that forward zone, you must determine the IP address of the primary DNS server for that zone. If the dynamic DNS updates will be protected with TSIG keys, you must find out the name and the base64 encoded value of the secret for the TSIG key. If the resource manager should also update the reverse zone (IP address to host mapping) for sessions, you will also need to determine the same information about the primary DNS server for the reverse zone (IP address and TSIG key).

If using TSIG keys, use aregcmd to create and configure the keys. You should set the key in the Remote Server or the Resource Manager, but not both. Set the key on the Remote Server if you want to use the same key for all of the zones accessed through that Remote Server. Otherwise, set the key on the Resource Manager. That key will be used only for the zone specified in the Resource Manager.

Note For proper function of Prime Access Registrar GUI, the DNS name resolution for the server's hostname should be defined precisely.

To configure Dynamic DNS:

Step 1 Launch **aregcmd**.

Step 2 Create the dynamic-dns TSIG Keys:

cd /Radius/Advanced/DDNS/TSIGKeys

add foo.com

This example named the TSIG Key, **foo.com**, which is related to the name of the example DNS server we use. You should choose a name for TSIG keys that reflects the DDNS client-server pair (for example, **foo.bar** if the client is **foo** and the server is **bar**), but you should use the name of the TSIG Key as defined in the DNS server.

Step 3 Configure the TSIG Key:

cd foo.com

set Secret <base64-encoded string>

The Secret should be set to the same base64-encoded string as defined in the DNS server. If there is a second TSIG Key for the primary server of the reverse zone, follow these steps to add it, too.

Step 4 Use **aregcmd** to create and configure one or more dynamic-dns Remote Servers.

Step 5 Create the dynamic-dns remote server for the forward zone:

cd /Radius/RemoteServers

add ddns

This example named the remote server *ddns* which is the related to the remote server type. You can use any valid name for your remote server.

Step 6 Configure the dynamic-dns remote server:

cd ddns

set Protocol dynamic-dns

set IPAddress 10.10.10.1 (ip address of primary dns server for zone)

set ForwardZoneTSIGKey foo.com

set ReverseZoneTSIGKey foo.com

If the reverse zone will be updated and if the primary server for the reverse zone is different than the primary server for the forward zone, you will need to add another Remote Server. Follow the previous two steps to do so. Note that the IP Address and the TSIG Key will be different.

You can now use **aregcmd** to create and configure a resource manager of type dynamic-dns.

Step 7 Create the dynamic-dns resource manager:

cd /Radius/ResourceManagers

add ddns

This example named the service ddns which is the related to the resource manager type but you can use any valid name for your resource manager.

Step 8 Configure the dynamic-dns resource manager.

cd ddns

set Type dynamic-dns

set ForwardZone foo.com

set ForwardZoneServer DDNS

Finally, reference the new resource manager from a session manager. Assuming that the example configuration was installed, the following step will accomplish this. If you have a different session manager defined you can add it there if that is appropriate.

Step 9 Reference the resource manager from a session manager:

cd /Radius/SessionManagers/session-mgr-1/ResourceManagers

set 5 DDNS

Note The Property AllowAccountingStartToCreateSession must be set to TRUE for dynamic DNS to work.

Step 10 Save the changes you have made.

Testing Dynamic DNS with radclient

After the Resource Manager has been defined it must be referenced from the appropriate Session Manager. You can use **radclient** to confirm that dynamic DNS has been properly configured and is operational.

To test Dynamic DNS using **radclient**:

Step 1 Launch **aregcmd** and log into the Prime Access Registrar server.

cd /opt/CSCOar/bin

aregcmd

Step 2 Use the **trace** command to set the trace to level 4.

trace 4

Step 3 Launch **radclient**.

cd /opt/CSCOar/bin

radclient

Step 4 Create an Accounting-Start packet.

acct_request Start username

Example:

set p [acct_request Start bob]

Step 5 Add a Framed-IP-Address attribute to the Accounting-Start packet.

Step 6 Send the Accounting-Start packet.

```
$p send
```

Step 7 Check the **aregcmd** trace log and the DNS server to verify that the host entry was updated in both the forward and reverse zones.

Customizing Your Configuration

After you have configured and tested a basic site, you can begin to make changes to better address your own sites' needs. This chapter provides information that describes how to:

- Use groups to select the appropriate user service
- Use multiple user lists to separate users
- Performs authentication and authorization against data from an LDAP server
- Use a script to determine which remote server to use for authentication and authorization
- Use session management to allocate and account for dynamic resources such as the number of concurrent user sessions.

The examples in this chapter provides an introduction to many of the Cisco Prime Access Registrar 8.0 objects and their properties. See Chapter 5, "Configuring and Monitoring the RADIUS Server," for more detailed information.

This chapter consists of the following sections:

- Configuring Groups, page 3-1
- Configuring Multiple UserLists, page 3-4
- Configuring a Remote Server for AA, page 3-9
- Configuring Session Management, page 3-16

Configuring Groups

The first change you might want to make is to create distinct groups based on the type of service, and divide your user community according to these groups.

You can use Prime Access Registrar UserGroups in two ways:

- You can create separate groups for each specific type of service. For example, you can have a group for PPP users and another for Telnet users.
- You can use a default group and, depending on how the user logs in, use a script to determine which service to provide.

The default Prime Access Registrar installation provides examples of both types of groups.

Configuring Specific Groups

For users who always require the same type of service, you can create specific user groups, and then set the user's group membership to that group.

Table 3-1 provides an overview of the process. The following sections describe the process in more detail.

Table 3-1 Configuring UserGroups

Object	Action
UserGroups	Add a new UserGroup
UserLists	Set group membership

Creating and Setting Group Membership

Step 1 Run the **aregcmd** command:

aregcmd

Step 2 Use the **cd** command to change to the **UserGroups** object.

cd /Radius/UserGroups

Step 3 Use the **add** command to create a user group, specifying the name and optional description, BaseProfile, AuthenticationScript, or AuthorizationScript. The following example shows how to add the PPP-users group.

This example sets the BaseProfile to `default-PPP-users`. When you set this property to the name of a profile, Prime Access Registrar adds the properties in the profile to the response dictionary as part of the authorization process.

add PPP-users "Users who always connect using PPP" default-PPP-users

Step 4 Use the **cd** command to change to the user you want to include in this group. The following example shows how to change to the user, jean:

cd /Radius/UserLists/Default/jean

Step 5 Use the **set** command to set the user's group membership to the name of the group you have just created.

set group PPP-users

Step 6 Use the **save** command to save your changes.

save

Step 7 Use the **reload** command to reload the server.

reload

Note You must save whenever you change the configuration, either through adds, deletes, or sets. Before you exit, log out, or reload, Prime Access Registrar prompts you to save. You must reload after all saves except when you have only made changes to individual users (either adds, deletes, or sets). Unlike all other changes, Prime Access Registrar reads user records on demand; that is, when there is a request from that user.

Configuring a Default Group

If you allow users to request different Services based on how they specify their username, you can use a script to determine the type of Service to provide. For example, the user *joe* can request either PPP or Telnet Service by either logging in as `joe%PPP` or `joe%Telnet`.

This works because there are two scripts: **ParseServiceHints** and **AuthorizeService**.

- **ParseServiceHints**—checks the username suffix and if it corresponds to a service, it modifies the request so it appears as if the NAS requested that type of Service.
- **AuthorizeService**—adds a certain profile to the response based on the Service type. The script chooses the authentication and/or authorization Service, and the Service specifies the UserGroup which then specifies the UserList, which contains the user `joe`.

Table 3-2 provides an overview of the process. The following sections describe the process in more detail.

Table 3-2 Choosing Among UserGroups

Object	Action
UserGroups	Add a new UserGroup or use existing Default group.
	Set AuthorizationScript
Scripts	Add new Script.
UserLists	Set group membership.

Using a Script to Determine Service

The following instructions assume you have already created a UserGroup and you have written a script that performs this function. For some sample scripts, see *Chapter 7, "Using Extension Points."*

Step 1 Use the **cd** command to change to the UserGroup you want to associate with the script. The following example changes to the **Default** group.

cd /Radius/UserGroups/Default

Step 2 Use the **set** command to set the AuthorizationScript to the name of the script you want run. The following example sets the script to **AuthorizeService**:

set AuthorizationScript AuthorizeService

Step 3 Use the **cd** command to change to **Scripts**:

cd /Radius/Scripts

Step 4 Use the **add** command to add the new script, specifying the name, description, language (in this case `Rex` which is short for RADIUS Extension), filename and an optional entry point. When you do not specify an entry point, Prime Access Registrar uses the script's name.

add AuthorizeService "Authorization Script" Rex libAuthorizeService.so AuthorizeService

Step 5 Use the **cd** command to change to the user. The following example changes to the user `beth`:

cd /Radius/UserLists/Default/beth

Step 6 Use the **set** command to set the user's group membership to the name of that group. The following example sets `beth`'s group membership to the `Default` group.

set Group Default

Step 7 Use the **save** command to save your changes:

save

Step 8 Use the **reload** command to reload the server:

reload

Note To save your changes and reload the server after following this example, you must have an actual script. Prime Access Registrar displays a warning message when it detects missing configuration objects.

Configuring Multiple UserLists

The basic site contains a single userlist, *Default*, and uses group membership to determine the type of Service to provide each user. When all users are in the same UserList, each username must be unique.

You can, however, group your user community by department or location, and use separate UserLists to distinguish amongst them. In this case, the users names must be unique only within each UserList. Thus, you can allow a user `Jane` in the `North` UserList as well as one in the `South` UserList.

When you have more than one UserList, you must have an incoming script that Prime Access Registrar can run in response to requests. The script chooses the authentication and/or authorization Service, and the Service specifies the actual UserList (Figure 3-1).

Figure 3-1 Using a Script to Choose a UserList

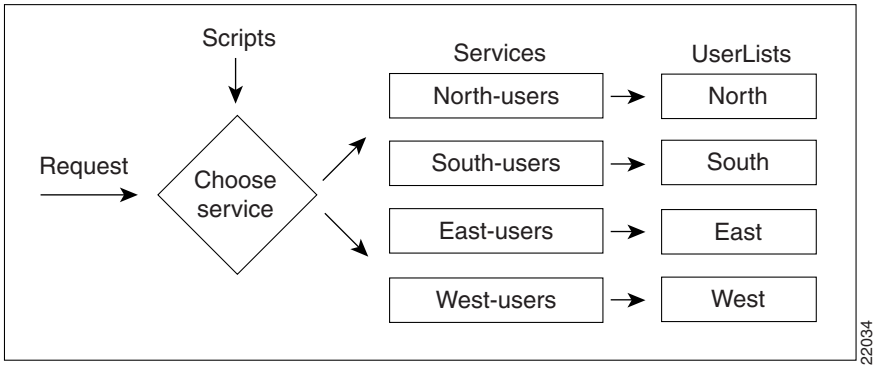


Table 3-3 provides an overview of the process. The following sections describe the process in more detail.

Table 3-3 Configuring Separate UserLists

Object	Action
UserLists	Add new UserLists.
Users	Add users.
Services	Add new Services.
	Set service type (local).
Radius	Set Incoming Script.
Scripts	Add a new Script.

Configuring Separate UserLists

Divide your site along organizational or company lines, and create a UserList for each unit.

Creating Separate UserLists

Step 1 Run the **aregcmd** command.

aregcmd

Step 2 Use the **cd** command to change to **UserLists**.

cd /Radius/UserLists

Step 3 Use the **add** command to create a UserList, specifying the name and optional description. The following example specifies the name `North` and the description `Users from the northern office`.

add North "Users from the northern office"

Step 4 Repeat for the other UserLists you want to add.

Configuring Users

After you have created multiple UserLists, you must populate them with the appropriate users.

Populating UserLists

Step 1 Use the **cd** command to change to the UserList you have created.

cd /Radius/UserLists/North

Step 2 Use the **add** command to add a user. Using the sample users as models, configure the appropriate group membership. The following example adds user `beth`, with the optional description `telemarketing`, the password `123`, Enabled set to `TRUE`, and group membership to `PPP-users`.

add beth telemarketing 123 TRUE PPP-users

Step 3 Repeat for the other users you want to add.

You can use the script, **add-100-users**, which is located in the **/opt/CSCOar/examples/cli** directory to automatically add 100 users.

Configuring Services

You must create a corresponding Service for each UserList. For example, when you create four UserLists, one for each section of the country, you must create four Services.

Note Before configuring services, ensure that you run the **save** and **reload** commands, after installation of the Prime Access Registrar kit.

Creating Separate Services

Step 1 Use the **cd** command to change to **Services**:

cd /Radius/Services

Step 2 Use the **add** command to create a Service, specifying the name and optional description. The following example specifies the name `North-users` and the description `All users from the northern branch office`:

add North-users "All users from the northern branch office"

Step 3 Use the **cd** command to change to **North-users**.

cd /Radius/Services/North-users

Step 4 Use the **set** command to set the type to *local. S*pecify the name of the UserList you want Prime Access Registrar to use. You can accept the default Outage Policy and MultipleServersPolicy or you can use the **set** command to change them. The following example sets the type to `local` and the UserList to `North`:

set type local

set UserList North

Step 5 Repeat for each Service you must create.

Creating the Script

You must write a script that looks at the username and chooses the Service to which to direct the request.

For example, you create four UserLists (`North`, `South`, `East`, and `West`), with the Service based on the origin of the user. When a user requests a Service, your script can strip off the origin in the request and use it to set the environment dictionary variables **Authentication-Service** and/or **Authorization-Service** to the name or names of the appropriate Service.

In this situation, when `beth@North.QuickExample.com` makes an Access-Request, the script will strip off the word `North` and use it to set the value of the environment variable **Authentication-Service** and/or **Authorization-Service**. Note, the script overrides any existing default authentication and/or authorization specifications.

Note For more information about writing scripts, see Chapter 7, "Using Extension Points,"and the role the dictionaries play in Prime Access Registrar, see the Cisco Prime Access Registrar 8.0 Reference Guide.

Client Scripting

Though, Prime Access Registrar allows external code (Tcl/C/C++/Java) to be used by means of a script, custom service, policy engine, and so forth, while processing request, response, or while working with the environment dictionaries, it shall not be responsible for the scripts used and will not be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of the script.

Configuring the Script

When you have multiple UserLists, you need a script to determine which UserList to check when a user makes an Access-Request. When you want the script to apply to all users, irrespective of the NAS they are using, place the script at the **Radius** level. When, on the other hand, you want to run different scripts depending on the originating NAS, place the script at the **Client** level.

Client Scripting

Though, Prime Access Registrar allows external code (Tcl/C/C++/Java) to be used by means of a script, custom service, policy engine, and so forth, while processing request, response, or while working with the environment dictionaries, it shall not be responsible for the scripts used and will not be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of the script.

Choosing the Scripting Point

Step 1 Use the **cd** command to change to the appropriate level. The following example sets the script for all requests.

cd /Radius

Step 2 Use the **set** command to set the incoming script. The following example sets the script, `ParseUserName`:

set IncomingScript ParseUserName

Step 3 Use the **cd** command to change to **Scripts**.

cd /Radius/Scripts

Step 4 Use the **add** command to add the new script, specifying the name, description, language, filename and an optional entry point. If you do not specify an entry point, Prime Access Registrar uses the script's name.

The following example specifies the name `ParseUserName`, the language `Rex` (which is RADIUS Extension), the filename `LibParseUserName.so`, and the entry point `ParseUserName`.

add ParseUserName ""Rex libParseUserName.so ParseUserName

Step 5 Use the **save** command to save your changes:

save

Step 6 Use the **reload** command to reload the server.

reload

Handling Multiple Scripts

Prime Access Registrar can run only one script from a given extension point. However, you can write a script that runs several scripts serially, one after the other. For example, the following **tcl** script, MasterScript, might look like the following:

```
## this MasterScript executes both tParseAAA and MyProcedure
# it assumes that tclscript.tcl and myscripts.tcl are in the same
#  directory as this file

source tclscript.tcl
source myscripts.tcl

proc MasterScript { request response environ } {
   tParseAAA $request $response $environ
   MyProcedure $request $response $environ
}
```

Save **tcl** scripts in the directory **/opt/CSCOar/scripts/radius/tcl**.

Configuring a Remote Server for AA

All the sites described so far in this chapter have used the Prime Access Registrar RADIUS server for authentication and authorization. You might want to delegate either one or both of those tasks to another server, such as an LDAP server or another RADIUS server.

You can specify one of the following services when you want to use a particular remote server:

- radius—authentication and/or authorization
- ldap—authentication and/or authorization
- tacacs-udp—authentication only.

Note Although these services differ in the way they handle authentication and authorization, the procedure for configuring a remote server is the same independent of its type.

Table 3-4 provides an overview of the process. The following sections describe the process in more detail.

Table 3-4 Configuring a Remote Server

Object	Action
RemoteServers	Add a new RemoteServer.
	Set the protocol (ldap).
	Set the properties.
Services	Add a new Service.
	Set the type (ldap).
	Set the RemoteServers property.
Radius	Set DefaultAuthentication.
	Set DefaultAuthorization.

Configuring the Remote Server

The RemoteServer object allows you to specify the properties of the remote server to which Services proxy requests. The remote servers you specify at this level are referenced by name from the RemoteServers list in the Services objects.

Creating a RemoteServer

Step 1 Run the **aregcmd** command:

aregcmd

Step 2 Use the **cd** command to change to the **RemoteServers** level:

cd /Radius/RemoteServers

Step 3 Use the **add** command to add the remote server you will reference in the Services level. The following example adds the remote server's hostname `QuickExample`.

add QuickExample

Step 4 Use the **cd** command to change to the **QuickExample RemoteServers** object level.

cd /Radius/RemoteServers/QuickExample

Step 5 Use the **set** command to specify the protocol `ldap`:

set protocol ldap

Step 6 Use the **set** command to specify the required LDAP properties.

At the very least you must specify:

- IPAddress—the IP address of the LDAP server (for example, `196.168.1.5`).
- Port—the port the LDAP server is listening on (for example, `389`).
- HostName—the hostname of the machine specified in the IP address field (for example, `ldap1.QuickExample.com`).
- SearchPath—the directory in the LDAP database to use as the starting point when searching for user information (for example, `o=Ace Industry, c=US`).
- Filter—the filter to use to find user entries in the LDAP database (for example, `(uid=%s)`).
- UserPasswordAttribute—the name of the LDAP attribute in a user entry that contains the user's password (for example, `userpassword`).
- BindName—specifies the distinguished name (DN) in the LDAP server for Prime Access Registrar to bind with the LDAP server (for example, `uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot`)
- BindPassword—Specifies the password for the distinguished name (for example, `cisco123`)

set IPAddress 196.168.1.5

set Port 389

set HostName ldap1.QuickExample.com

set SearchPath "o=Ace Industry, c=US"

set Filter (uid=%s)

set UserPasswordAttribute password

set BindName uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot

set BindPassword cisco123

See the "Using LDAP" chapter of the*Cisco Prime Access Registrar 8.0 User Guide* for descriptions of the other LDAP properties.

Configuring Services

To use LDAP for authorization and/or authentication, you must configure a Services object.

Creating Services

Step 1 Run the **aregcmd** command:

aregcmd

Step 2 Use the **cd** command to change to the **Services** level:

cd /Radius/Services

Step 3 Use the **add** command to add the appropriate LDAP service. The following example adds the `remote-ldap` service:

add remote-ldap "Remote LDAP Service"

Step 4 Use the **cd** command to change to the **remote-ldap** object:

cd /Radius/Services/remote-ldap

Step 5 Use the **set** command to set the type to `ldap`. You can accept the default Outage Policy and MultipleServersPolicy or you can use the **set** command to change them.

set type ldap

Step 6 Use the **cd** command to change to the **RemoteServers**:

cd /Radius/Services/remote-ldap/RemoteServers

Step 7 Use the **set** command to set the server number and name. By giving each server a number you tell Prime Access Registrar the order you want it to access each server. Prime Access Registrar uses this order when implementing the MultipleServersPolicy of Failover or RoundRobin.

The following example sets the first remote server to the server `QuickExample`:

set 1 QuickExample

The MultipleServersPolicy determines how Prime Access Registrar handles multiple remote servers.

- When you set it to `Failover`, Prime Access Registrar directs requests to the first server in the list until it determines the server is offline. At that time, Prime Access Registrar redirects all requests to the next server in the list until it finds a server that is online.
- When you set it to `RoundRobin`, Prime Access Registrar directs each request to the next server in the RemoteServers list in order to share the resource load across all the servers listed in the RemoteServers list.

Configuring the RADIUS Server

In the default Prime Access Registrar configuration, authentication and authorization are handled through the local-users Service object. This causes Prime Access Registrar to match requesting users with the names in its own database. When you select LDAP as a remote server for authentication and authorization, Prime Access Registrar looks to that server for user information.

To have Prime Access Registrar perform authentication and authorization against information from the LDAP server, you must change the DefaultAuthenticationService and DefaultAuthorizationService at the **Radius** level.

Changing the Authentication and Authorization Defaults

Step 1 Run the **aregcmd** command:

aregcmd

Step 2 Use the **cd** command to change to the **Radius** level:

cd /Radius

Step 3 Use the **set** command to change the **DefaultAuthentication**:

set DefaultAuthentication remote-ldap

Step 4 Use the **set** command to change the **DefaultAuthorization**:

set DefaultAuthorization remote-ldap

Step 5 Use the **save** command to save your changes:

save

Step 6 Use the **reload** command to reload the server:

reload

Configuring Multiple Remote Servers

All of the sites described so far in this chapter have used a single server for authentication and authorization; either the local RADIUS server or a remote LDAP server.

You can configure multiple remote servers to use the same Service, or multiple remote servers to use different Services. Figure 3-2 shows how to use multiple servers for authentication and authorization, and how to employ a script to determine which one to use.

Figure 3-2 Using a Script to Choose a Remote Server

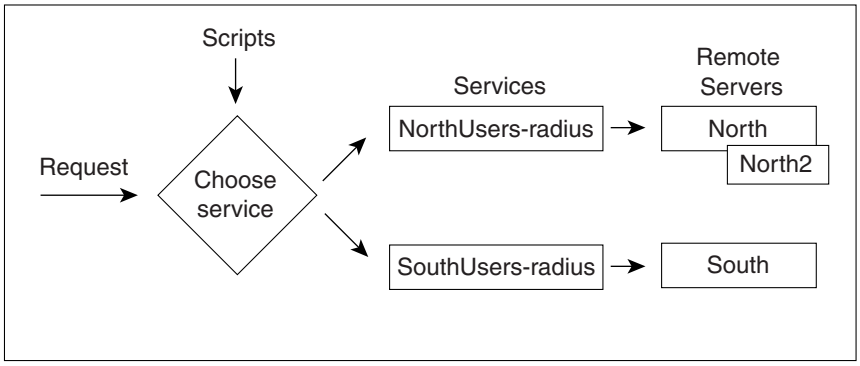


Table 3-5 provides an overview of the process. The following sections describe the process in more detail. Repeat for each RemoteServer you want to configure.

Table 3-5 Configuring Multiple Remote Servers

Object	Action
RemoteServers	Add a new RemoteServer.
	Set the protocol (radius).
	Set the shared secret.
Services	Add a new Service.
	Set the type (radius).
	Set the remote server name and number.
Scripts	Add a new Script.
Radius	Set the IncomingScript.

Configuring Two Remote Servers

Configure each remote server you want to use for authentication and authorization. The following example shows the `North` remote server.

Creating RemoteServers

Step 1 Run the **aregcmd** command:

aregcmd

Step 2 Use the **cd** command to change to the **RemoteServers** level:

cd /Radius/RemoteServers

Step 3 Use the **add** command to add the remote server you specified in the Services level. The following example adds the `North` remote server:

add North

Step 4 Use the **cd** command to change to the **North RemoteServers** level:

cd /Radius/RemoteServers/North

Step 5 Use the **set** command to specify the protocol `radius`*:*

set protocol radius

Step 6 Use the **set** command to specify the `SharedSecret 789`*:*

set SharedSecret 789

Step 7 Repeat these steps for the other remote servers.

Configuring Services

To use multiple remote servers for authorization and/or authentication you must configure the corresponding Services.

Creating the Services

Step 1 Run the **aregcmd** command:

aregcmd

Step 2 Change directory to **/Radius/Services**.

cd /Radius/Services

Step 3 Use the **add** command to add the appropriate RADIUS service. The following example adds the `NorthUsers-radius` object:

add NorthUsers-radius "NorthRemote server"

Step 4 Use the **cd** command to change the **NorthUsers-radius** object:

cd /Radius/Services/NorthUsers-radius

Step 5 Use the **set** command to set the type to `radius`:

set type radius

Step 6 Use the **set** command to set the remote server number and name. By giving each server a number, you tell Prime Access Registrar the order you want it to access each server. Prime Access Registrar uses this order when implementing the MultipleServersPolicy of Failover or RoundRobin.

The following example sets the first remote server to the server `North` and the second remote server to `North2`:

set RemoteServers/1 North

set RemoteServers/2 North2

Step 7 Create another Service (SouthUsers-radius) for the South remote server.

Configuring the Script

When you have multiple RemoteServers, you need a script that determines the authentication and/or authorization Service, which in turn specifies the RemoteServer to check when a user makes an Access-Request. If you want the script to apply to all users, irrespective of the NAS they are using, place the script at the **Radius** level.

Note See Chapter 7, "Using Extension Points," for sample scripts you can use as a basis for your own scripts.

Choosing the Scripting Point

Step 1 Run the **aregcmd** command:

aregcmd

Step 2 Use the **cd** command to change to the **Scripts** object:

cd /Radius/Scripts

Step 3 Use the **add** command to add the new script, specifying the name, description, language, filename and an optional entry point. If you do not specify an entry point, Prime Access Registrar uses the script's name.

The following example specifies the name ParseRemoteServers, the language `Rex`, the filename libParseRemoteServers.so, and the entry point ParseRemoteServers:

add ParseRemoteServers "Remote Server Script" RexlibParseRemoteServers.so ParseRemoteServers

Step 4 Use the **cd** command to change to the appropriate object level. The following example changes to the server level:

cd /Radius

Step 5 Use the **set** command to set the incoming script. The following example sets the script, **ParseRemoteServers**, at the server level:

```
set IncomingScript ParseRemoteServers
```

Step 6 Use the **save** command to save your changes:

```
save
```

Step 7 Use the **reload** command to reload the server.

```
reload
```

Configuring Session Management

You can use session management to track user sessions, and/or allocate dynamic resources to users for the lifetime of their sessions. You can define one or more Session Managers, and have each one manage the sessions for a particular group or company.

Configuring a Resource Manager

Session Managers use Resource Managers, which in turn manage a pool of resources of a particular type. The Resource Managers have the following types:

- **IP-Dynamic**—manages a pool of IP address and allows you to dynamically allocate IP addresses from that pool of addresses
- **IP-Per-NAS-Port**—allows you to associate NAS ports to specific IP addresses, and thus ensure specific NAS ports always get the same IP address
- **IPX-Dynamic**—manages a pool of IPX network addresses
- **Gateway Subobject**—includes a list of names of the Frame Relay Gateways for which to encrypt the session key.
- **Group-Session-Limit**—manages concurrent sessions for a group of users; that is, it keeps track of how many sessions are active and denies new sessions after the configured limit has been reached.
- **User-Session-Limit**—manages per-user concurrent sessions; that is, it keeps track of how many sessions each user has, and denies the user a new session after the configured limit has been reached.
- **USR-VPN**—allows you to set up a Virtual Private Network (VPN) using a US Robotics NAS. (A Virtual Private Network is a way for companies to use the Internet to securely transport private data.)
- **IP Address Pool**—allows you to manage pool of dynamic IP addresses
- **On-Demand Address Pool**—allows you to manage pool of IP dynamic subnet address
- **Session Cache**—allows you to cache additional attributes to existing session and supports the identity cache feature
- **Home-Agent**—manages a pool of on-demand IP addresses
- **Home-Agent-IPv6**—manages a pool of on-demand IPv6 addresses
- **Subnet-Dynamic**—supports the On Demand Address Pool feature

- **Dynamic-DNS**—manages the DNS servers
- **Remote-IP-Dynamic**—manages a pool of IP addresses that allows you to dynamically allocate IP addresses from a pool of addresses. It internally works with a remote ODBC database.
- **Remote-User-Session-Limit**—manages per-user concurrent sessions; that is, it keeps track of how many sessions each user has and denies the user a new session after the configured limit has been reached. It internally works with a remote ODBC database.
- **Remote-Group-Session-Limit**—manages concurrent sessions for a group of users; that is, it keeps track of how many sessions are active and denies new sessions after the configured limit has been reached. It internally works with a remote ODBC database
- **Remote-Session-Cache**—allows you to define the RADIUS attributes to store in cache. It should be used with session manager of type 'remote'.
- **3GPP**—supports HLR and/or HSS.

Each Resource Manager is responsible for examining the request and deciding whether to allocate a resource for the user, pass the request through, or cause Prime Access Registrar to reject the request.

Table 3-6 provides an overview of the process. The following sections describe the process in more detail.

Table 3-6 Configuring ResourceManagers

Object	Action
ResourceManagers	Add new ResourceManager
	Set type (Group-Session-Limit)
	Set value (100)
SessionManagers	Add new SessionManager
	Set ResourceManager
Radius	Set DefaultSessionManager

Creating a Resource Manager

You can use the default Resource Managers as models for any new Resource Managers you want to create. The following describes how to create a Resource Manager that limits the number of users to 100 or less at any one time.

Step 1 Run the **aregcmd** command:

```
aregcmd
```

Step 2 Use the **cd** command to change to the **ResourceManagers** level:

```
cd /Radius/ResourceManagers
```

Step 3 Use the **add** command to add a new ResourceManager. The following example adds the ResourceManager rm-100:

```
add rm-100
```

Step 4 Use the **cd** command to change to the ResourceManager you have just created:

cd rm-100

Step 5 Use the **set** command to set the type:

set type Group-Session-Limit

Step 6 Use the **set** command to set the number of GroupSessionLimit to 100:

set GroupSessionLimit 100

Configuring a Session Manager

After you create a Resource Manager, you must associate it with the appropriate Session Manager.

Creating a Session Manager

Step 1 Run the **aregcmd** command:

aregcmd

Step 2 Use the **cd** command to change to the **SessionManagers** level:

cd /Radius/SessionManagers

Step 3 Use the **add** command to add a new SessionManager. The following example adds the SessionManager sm-1:

add sm-1

Step 4 Use the **cd** command to change to the **SessionManager/ResourceManagers** property:

cd sm-1/ResourceManagers

Step 5 Use the **set** command to specify the ResourceManagers you want tracked per user session. Specify a number and the name of the ResourceManager. Note, you can list the ResourceManager objects in any order.

set 1 rm-100

Enabling Session Management

Prime Access Registrar, by default, comes configured with the sample SessionManagement **session-mgr-1**. You can modify it or change it to the new SessionManager you have created.

Note When you want the Session Manager to manage the resources for all Access-Requests Prime Access Registrar receives, set the RADIUS DefaultSessionManager to this Session Manager. When you want a Session Manager to manage the resources of a particular object, or to use multiple Session Managers, then use an incoming script at the appropriate level.

Configuring Session Management

Step 1 Run the **aregcmd** command:

aregcmd

Step 2 Use the **cd** command to change to the **Radius** level:

cd /Radius

Step 3 Use the **set** command to set the **DefaultSessionManager** to the name you have just created:

set DefaultSessionManager sm-1

Step 4 Use the **save** command to save your changes:

save

Step 5 Use the **reload** command to reload the Prime Access Registrar server.

reload

Setting the CPAR Configurable Option

This chapter describes how to use each of the **aregcmd** commands. The Cisco Prime Access Registrar **aregcmd** command is a command-line based configuration tool. It allows you to set any Cisco Prime Access Registrar (Prime Access Registrar) configurable option, as well as, start and stop the server and check statistics.

This chapter contains the following sections:

- General Command Syntax
- aregcmd Commands
- aregcmd Command Logging
- aregcmd Command Line Editing
- aregcmd Error Codes

General Command Syntax

Prime Access Registrar stores its configuration information in a hierarchy. Using the **aregcmd** command **cd** (change directory), you can move through this information in the same manner as you would through any hierarchical file system. Or you can supply full pathnames to these commands to affect another part of the hierarchy, and thus avoid explicitly using the **cd** command to change to that part of the tree.

- **aregcmd** command parsing is case *insensitive*, which means you can use upper or lowercase letters to designate elements. In addition, when you reference existing elements in the configuration, you need only specify enough of the element's name to distinguish it from the other elements at that level. For example, instead of entering **cd Administrators**, you can enter **cd ad** when no other element at the current level begins with **ad**.
- **aregcmd** command parsing is command-line order *dependent*; that is, the arguments are interpreted based on their position on the command line. To indicate an empty string as a place holder on the command line, use either single (') or double quotes (""). In addition, when you use any arguments that contain spaces, you must quote the arguments. For example, when you use the argument, "**Local Users**," you must enclose the phrase in quotes.

The **aregcmd** command can contain a maximum of 255 characters when specifying a parameter and 511 characters for the entire command.

The **aregcmd** command syntax is:

aregcmd [**-C** <*clustername*>] [**-N** <*adminname*>] [**-P** <*adminpassword*>] [**-V**] [**-f** <*scriptfile*>] [**-l** <*directoryname*>] [**-n**] [<*command*> [<*args*>]] [**-p**] [**-q**] [**-v**]

- **-C**—Specifies the name of the cluster to log into by default

- **-N**—Specifies the name of the administrator
- **-P**—Specifies the password
- **-V**—Specifies view-only mode
- **-f**—Specifies a file that can contain a series of commands
- **-l**—Specifies a directory where the Prime Access Registrar license file is stored and returns information about licensed components
- **-n**—Turns off prefix mode
- **-p**—Specifies prefix mode
- **-q**—Turns off verbose mode
- **-v**—Specifies verbose mode

Note The verbose (**-v**) and prefix (**-p**) modes are on by default when you run **aregcmd** interactively (for example, not entered on the command line or not running commands from a script file). Otherwise, verbose and prefix modes are off.

When you include a command (with the appropriate arguments) on the command line, **aregcmd** runs only that one command and saves any changes.

This section contains the following topics:

- View-Only Administrator Mode
- Configuration Objects
- aregcmd Command Performance

View-Only Administrator Mode

Previous releases of Prime Access Registrar provided only *super-user* administrative access. If you were able to log into **aregcmd**, you could do anything to the system, including starting and stopping the system and changing the configuration. Prime Access Registrar provides view-only administrative access. View-only access restricts an administrator to only being able to observe the system and prevents that user from making changes.

View-only access can be encountered in three ways:

- Specific administrators can be restricted to view-only access whenever they log in.
- Administrators not restricted to view-only access can choose to start **aregcmd** in a view-only mode. This might be used when an administrator wants to ensure that he or she does not make any changes.
- When an administrator who is not view-only logs in to a slave server, they will be unable to make changes to any parts of the configuration other than **/Radius/Replication**, **/Radius/Advanced/Ports, /Radius/Advanced/Interfaces** or the properties in **/Radius/Advanced**. This is because the rest of the configuration is replicated from the master server and changes directly to the slave will cause problems.

Note When a user logs in, the system determines whether a user's session is view-only or not. If the configuration is changed after a user has logged in, that change does not take effect until the affected user logs out and logs back in.

ViewOnly Property

The ViewOnly property has been added to the Administrators configuration. The default setting for the ViewOnly property is FALSE. The following shows the default setting for the **admin** user:

cd /Administrators/admin

```
[ //localhost/Administrators/admin ]
    Name = admin
    Description =
    Password = <encrypted>
    ViewOnly = FALSE
```

You can designate specific administrators to be view-only administrators by setting the new ViewOnly property to TRUE.

- If that property is set to TRUE, any time the administrator logs in to **aregcmd** the session will be in view-only mode.
- If set to FALSE, when the administrator logs in to a master server, the session will be full super-user capability.

If the administrator logs in to a slave, they only part of the configuration they will be able to modify is that part under **/Radius/Replication**, **/Radius/Advanced/Ports**, **/Radius/Advanced/Interfaces** or the properties in **/Radius/Advanced**.

When in a view-only session, the following commands will cause an error: **add**, **delete**, **set**, **unset**, **insert**, **validate**, **save**, **start**, **stop**, **reload**, **reset-stats**, **release-sessions**, and **trace**. The following error message will be displayed:

```
316 Command failed: session is ViewOnly
```

When in a slave server session, the following commands will cause an error when the object or property being operated on is not under **/Radius/Replication**, **/Radius/Advanced/Ports**, **/Radius/Advanced/Interfaces** or the properties in **/Radius/Advanced**: **add**, **delete**, **set**, **unset**, and **insert**. The following error message will be displayed:

```
317 Command failed: session is ViewOnly
```

Configuration Objects

The Prime Access Registrar **aregcmd** command lets you manipulate configuration objects, that define properties or the behavior of the RADIUS server, such as valid administrators and types of services. For descriptions of those objects, see Chapter 5, “Configuring and Monitoring the RADIUS Server.”

aregcmd Command Performance

You can impact **aregcmd** command performance and server response time by having Prime Access Registrar userlists that contain more than 10,000 users. Prime Access Registrar userlists were not designed to contain 10,000 users in any one list.

If you must provide service for groups greater than 10000 users, we recommend that you use an external data store such as an **LDAP directory** or an **Oracle database**. If you are unable to use an external data store, create multiple userlists instead, keeping each userlist under 10,000 users.

Multiple userlists require multiple services (one for each userlist), because a service cannot reference more than one userlist. The multiple services can then be combined using the Service Grouping feature with ResultRule, OR, as follows:

```
[ //localhost/Radius/Services/GroupService ]
    Name = GroupService
    Description =
    Type = group
    IncomingScript~ =
    OutgoingScript~ =
    ResultRule = OR
    GroupServices/
    1. UserService1
    2. UserService2
    3. UserService3
```

RPC Bind Services

The Prime Access Registrar server and the **aregcmd** CLI requires RPC services to be running before the server is started. If the RPC services are stopped, you must restart RPC services, then restart the Prime Access Registrar server.

Use the following commands to restart RPC services:

arserver stop

/etc/init.d/rpc start

arserver start

If RPC services are not running, the following message is displayed when you attempt to start aregcmd:

```
Login to aregcmd fails with the message:
400 Login failed
```

aregcmd Commands

This section contains the complete list of **aregcmd** commands. You can use them on the command line or insert them into scripts. The commands are listed alphabetically.

This section contains the following topics:

- add
- cd
- delete
- exit
- filter
- find
- help
- insert
- login
- logout

- ls
- next
- prev
- pwd
- query-sessions
- quit
- release-sessions
- reload
- reset-stats
- save
- set
- start
- stats
- status
- stop
- tacacs-stats
- tacacs-reset-stats
- dia-stats
- trace
- trace-file-count
- unset
- validate

add

Use the **aregcmd** command **add** to create new elements in the configuration. The **add** command is context sensitive, which means the type of element added is determined by the current context, or the path specified as the first parameter. The **add** command has one required argument; the name of the element you wish to add. You can also provide other parameters, or you can supply this information after **aregcmd** has added the new element. The optional second argument is a description of the element.

The syntax is:

add [<*path*>**/**]<*name*> *[...]*

cd

Use the **aregcmd** command **cd** to change the working context, or level in the configuration hierarchy. When you use the **cd** command without any parameters, it returns you to the root of the tree. When you use the optional path argument, you can specify a new context. To change to a higher level in the tree hierarchy, use the ".." syntax (as you would in a UNIX file system). When you change to a new context, **aregcmd** displays the contents of the new location, when you are using the command in interactive mode, or if verbose mode is on.

The syntax is:

cd [<*path*>]

delete

Use the **aregcmd** command **delete** to remove an element from the configuration hierarchy. You cannot remove properties on an element; you can only remove entire elements. The **delete** command is recursive; that is, it will remove any subelements contained within an element being removed. When the element is in the current context, you need only provide the name of the element to be deleted. You can optionally provide a complete path to an element elsewhere in the configuration hierarchy.

The syntax is:

delete [<*path*>**/**]<*name*>

exit

Use the **aregcmd** command **exit** to terminate your **aregcmd** session. If you have any unsaved modifications, Prime Access Registrar asks if you want to save them before exiting. Any modifications you don't choose to save are lost.

The syntax is:

exit

filter

Use the **aregcmd** command **filter** to display a selected view of a list. You can use the **filter** command to present only the elements of a list that have properties equal to the value you specify. You can also use the **filter** command to restore the view of the list after it has been filtered.

When using the **filter** command, you must provide a property name and a value, and you can optionally provide the path to the list. Prime Access Registrar displays a list with only those elements that have a value equal to the specified value. When you want to filter the current context, you can omit the path argument.

The **filter** command is *sticky*, in that, after you have filtered a list, you must explicitly unfilter it before you can view the complete list again. To restore the unrestricted view of the list, use the **filter** command and specify the string **all**. To restore the list in current context, you can omit the pathname.

The syntax is:

filter [<*path*>] <*property*> <*value*>

or

filter [<*path*>] **all**

find

Use the **aregcmd** command **find** to locate a specific item in a list. The **find** command takes one required argument, which is a full or partial pathname. After you use the command, Prime Access Registrar displays a page beginning with the entry that most closely matches the pathname you provided.

The syntax is:

find <*path*>

help

Use the **aregcmd** command **help** (with no argument specified) to display a brief overview of the command syntax. When you specify the name of a command, Prime Access Registrar displays help for only that command.

The syntax is:

help [<*command*>]

insert

Use the **aregcmd** command **insert** to add an item anywhere in ordered list. The required parameters are the numeric index of the position in the list in which you want to insert the new item, and the item value. When the list to which you are adding is not the current context, you can specify the complete path to the position in the list by prepending the path for the list to the numeric index. After the new value has been inserted into the list, Prime Access Registrar appropriately renumbers the list.

The syntax is:

insert [<*path*>**/**]<*index*> <*value*>

This command applies to lists of servers by index and the Resource Managers list in Session Managers.

login

Use the **aregcmd** command **login** to connect to a cluster, which contains the RADIUS server and definition of the authorized administrators. When you do not specify the cluster, admin name, and password, **aregcmd** prompts you for them.

When you are currently logged in to a cluster, the **login** command allows you to connect to another cluster. When you have changes in the current cluster that you have not saved, **aregcmd** asks if you want to save them before logging into another cluster. Any changes you do not save are lost.

After you successfully log in, and if the server is running, Prime Access Registrar displays the cluster server's health. Note, to log into a cluster, the Prime Access Registrar Server Agent for that cluster must be running.

The syntax is:

login [<*cluster*> [<*name*> [<*password*>]]]

logout

Use the **aregcmd** command **logout** to log out of the current cluster. After you log out, you have to log into make any modifications to the configuration hierarchy, or to manage the server(s). When you have any unsaved modifications, Prime Access Registrar asks if you want to save them before logging out. Any modifications you do not choose to save are lost.

The syntax is:

logout

ls

Use the **aregcmd** command **ls** to list the contents of a level in the configuration hierarchy. This command works much like the UNIX **ls** command. When you use it without any parameters, it lists the items in the current context. When you specify a path, it lists the elements found in that context. When you use the **-R** argument, it recursively lists all of the elements in and below the specified (or current) context.

For similar commands, refer to the **next** and **prev** commands.

The syntax is:

ls [**-R**] [<*path*>]

next

Use the aregcmd **next** command to review the remaining pages produced from the **ls** command. Every time you use the **cd** command, it automatically invokes the **ls** command to display the contents of the location. When the output from the **ls** command is more than one page (a page is about 24 lines) in length, Prime Access Registrar displays only the first page.

Note **ls** command retrieves only user-added objects such as Users, UserLists, and attributes.

The **next** command takes an optional path and count. The path specifies the context in which you wish to see the next page and the count specifies the number of lines you wish to see. When you use the **next** command without the path, Prime Access Registrar uses the current context. When you do not specify a count, Prime Access Registrar uses the last count value you used with the **next** or **prev** command. If you never specify a count, Prime Access Registrar uses the default value, which is 20.

Note, the current page for a context is *sticky*. This means, for example, when you use the **next** command to view entries 20 through 30, until you use the **next** or **prev** command on the same context, you will continue to see these entries even if you use the **cd** command to change to a different context, then return to the original.

The syntax is:

next [<*path*>] [<*count*>]

prev

Use the **aregcmd** command **prev** to page backwards through the output of the **ls** command. It behaves much like the **next** command, in that it takes an optional path identifying a context to display and a count parameter indicating how many lines to display.

The syntax is:

prev [<*path*>] [<*count*>]

pwd

Use the **aregcmd** command **pwd** to display the absolute pathname of the current context (level in the configuration hierarchy).

The syntax is:

pwd

query-sessions

Use the **aregcmd** command **query-sessions** to query the server about the currently active user sessions. You can request information about all of the active sessions or just those sessions that match the type you specify.

The syntax is:

query-sessions <*path*> [**all**]

or

query-sessions <*path*> **with-**<*type*> <*value*> **[send-CoA [with-profile <profile name>]]**

or

query-sessions <*path*> **with-Attribute** <*name*> <*value*> **[send-CoA [with-profile <profile name>]]**

Where <*path*> is the path to the server, Session Manager, or Resource Manager to query and **with-**<*type*> is one of the following: **with-NAS**, **with-User**, **with-IP-Address**, **with-IPX-Network**, **with-USR-VPN**, **with-Key, with-ID** or **with-Age**. The optional **[with-profile <profile name>]** parameter indicates a profile name as configured in **/Radius/Profiles**.

The **query-sessions** command with an optional [**send-CoA**] at the end causes the Prime Access Registrar server to send a Change of Authorization (CoA) request to the client. The CoA request includes the CoA attributes configured for the client. When the optional profile name is also included in the command, the Prime Access Registrar server includes the attribute-value (AV) pairs from the corresponding profile in **/Radius/Profiles** in the CoA request.

quit

Use the **aregcmd** command **quit** to terminate your **aregcmd** session. You can use it interchangeably with the **exit** command.

The syntax is:

quit

When you quit the **aregcmd** command, if you have made changes, the Prime Access Registrar server asks if you want to save the changes. Any unsaved changes are lost.

release-sessions

Use the **aregcmd** command **release-sessions** to request the server to release one or more currently active user sessions. This command might be useful, for example, in the case where you have taken a NAS offline, however, the server believes user sessions for that NAS are still active.

The syntax is one of:

release-sessions <*path*> **all**

or

release-sessions <*path*> **with-** <*type*> <*value*> [**send-pod**] [**send-notification**]

or

release-sessions <*path*> **with-Attribute** <*name*> <*value*> [**send-pod**] [**send-notification**]

Where <*path*> is the path to the server, Session Manager, or Resource Manager to query and **with-**<*type*> is one of the following: **with-NAS**, **with-User**, **with-IP-Address**, **with-IPX-Network**, **with-USR-VPN**, **with-Key,** or **with-ID**.

The optional **[send-pod <send notification>]** parameter sends the disconnect packet to the NAS to clear sessions and an Accounting-Stop notification to the client listed in the session record.

The optional **with-Attribute** parameter enables release a session based on a specific attribute and value.

reload

Use the **aregcmd** command **reload** to stop the server (when it is running), and then immediately start the server, forcing it to reread its configuration information. When you have modified the configuration hierarchy, Prime Access Registrar asks you if you want to save your changes before restarting the server. You *must* save your changes in order for the reloaded server to be able to use them.

The syntax is:

reload

reset-stats

Use the **aregcmd** command **reset-stats** to reset all server statistics displayed with the **stats** command. The **reset-stats** command also resets SNMP counters.

The **reset-stats** command provides a way of resetting the server statistics without having to reload or restart the server.

The syntax is:

reset-stats

save

Use the **aregcmd** command **save** to validate the changes you made and commit them to the configuration database, if no errors are found.

Note Using the **save** command does not automatically update the running server. To update the server, you must use the **reload** command.

The syntax is:

save

Table 4-1 lists the RADIUS server objects and the effect of Dynamic Updates upon them.

Table 4-1 Dynamic Updates Effect on RADIUS Server Objects

Object	Add	Modify or Delete
Radius	Yes	Yes
UserLists	Yes	Yes
UserGroups	Yes	Yes
Policies	Yes	Yes
Clients	Yes	Yes
Vendors	Yes	Yes
Scripts	Yes	Yes
Services	Yes	Yes
SessionManagers	Yes	No
ResourceManagers	Yes	No
Profiles	Yes	Yes
Rules	Yes	Yes
Translations	Yes	Yes
TranslationGroups	Yes	Yes
RemoteServers	Yes	No
Replication	Yes	Yes
Advanced	Yes	Yes
SNMP	No	No
Ports	No	No
Interfaces	No	No

set

Use the **aregcmd** command **set** to provide values for properties on existing configuration elements. You only need to provide the **set** command with the name of the property you wish to set (or just enough of the name to distinguish it from other properties) and the new value for that property. It also applies to the **Profiles** attribute list, the Rules attributes list, the enumeration list in the Attribute dictionary, and the **LDAPtoRadiusMappings** and **LDAPtoEnvironmentMappings** mappings.

The **set** command can also be used to order servers in a list. To specify a new position in a list for a server, use the **set** command and provide the numeric position of the server and the server's name.

The syntax is:

set [<*path*>/]<*property*> <*value*>

When the list is a list of servers by index, the syntax is:

set [<*path*>/]<*index*> <*server name*>

Note If the index is already in use, the old server name will be replaced by the new server name.

To remove a value from a property (make a property equal to NULL), use a pair of single or double quotes as the value, as shown below:

set <*property*> ""

When you need to set an attribute to a value that includes a space, you must double-quote the value, as in the following:

set Framed-Route "192.168.1.0/24 192.168.1.1"

start

Use the **aregcmd** command **start** to enable the server to handle requests. When the configuration hierarchy has been modified, Prime Access Registrar asks you if you want to save the changes before starting the server.

The syntax is:

start

stats

Use the **aregcmd** command **stats** to provide statistical information on the specified server. You can only issue this command when the server is running.

Note that **aregcmd** supports the **PAGER** environment variable. When the **aregcmd stats** command is used and the **PAGER** environment variable is set, the **stats** command output is displayed using the program specified by the **PAGER** environment variable.

The syntax is:

stats

The following is an example of the statistical information provided for remote server after you issue the **stats** command:

```
RemoteServer statistics for:ServerA, 209.165.201.1, port 1812
    active = TRUE
    maxTries = 3
    RTTAverage = 438ms
    RTTDeviation = 585ms
    TimeoutPenalty = 0ms
    totalRequestsPending = 0
    totalRequestsSent = 14
    totalRequestsOutstanding = 0
    totalRequestsTimedOut = 0
    totalRequestsAcknowledged = 14
    totalResponsesDroppedForNotInCache = 0
    totalResponsesDroppedForSignatureMismatch = 0
    totalRequestsDroppedAfterMaxTries = 0
    lastRequestTime = Mon Feb 18 17:19:46 2013
    lastAcceptTime = Mon Feb 18 17:18:11 2013
```

Table 4-2 lists the statistics displayed for the remote server by the stats command and the meaning of the values.

Table 4-2 aregcmd stats Information for RADIUS Remote Server

Stats Value	Meaning
RemoteServer statistics for:	Provides server's type, name, IP address, and port used
active	Indicates whether the server was active (not in a down state)
maxTries	Number of retry attempts to be made by the RemoteServer Object based on the RemoteServer's *maxTries* property setting
RTTAverage	Average round trip time since the last server restart
RTTDeviation	Indicates a standard deviation of the RTTAverage
TimeoutPenalty	Indicates any change made to the initial timeout default value
totalRequestsPending	Number of requests currently queued
totalRequestsSent	Number of requests sent since the last server restart **Note** totalRequestsSent should equal the sum of totalRequestsOutstanding and totalRequestsAcknowledged.
totalRequestsOutstanding	Number of requests currently proxied that have not yet returned
totalRequestsTimedOut	Number of requests that have timed out since last server restart or number requests not returned from proxy server within the [configured] initial timeout interval
totalRequestsAcknowledged	Number of responses received since last server restart
totalResponsesDroppedForNotInCache	Number of responses dropped because their ID did not match the ID of any Pending requests
totalResponsesDroppedForSignatureMismatch	Number of responses dropped because their response authenticator did not decode to the correct shared secret
totalRequestsDroppedAfterMaxTries	Number of requests dropped because no response was received after retrying the configured number of times. This value is different from totalRequestsTimedOut because using the default configuration values, no response within 2000 ms bumps the TimedOut counter, but it waits 14000 ms (2000 + 4000 + 8000) to bump this counter.

Table 4-2 aregcmd stats Information for RADIUS Remote Server (continued)

Stats Value	Meaning
lastRequestTime	Date and time of last proxy request
lastAcceptTime	Date and time of last ACCEPT response to a client

Stats for RADIUS Client

You can query the statistics of RADIUS clients using the following command:

```
stats /r/client/localhost
```

The following is an example of the statistical information provided for RADIUS clients after you issue the **stats** command:

```
Radius statistics for client: 127.0.0.1 , localhost , ipv4
TLSActiveConnectionCount = 5
totalAuthAccessRequests = 15
totalAuthDupAccessRequests = 0
totalAuthAccessAccepts = 15
totalAuthAccessRejects = 0
totalAuthAccessChallenges = 0
totalAuthMalformedAccessRequests = 0
totalAuthBadAuthenticators = 0
totalAuthPacketsDropped = 0
totalAuthUnknownTypes = 0
totalAccPacketsDropped = 0
totalAccRequests = 0
totalAccDupRequests = 0
totalAccResponses = 0
totalAccBadAuthenticators = 0
totalAccMalformedRequests = 0
totalAccNoRecords = 0
totalAccUnknownTypes = 0
```

Table 4-2 lists the statistics displayed for the RADIUS client by the stats command and the meaning of the values.

Table 4-3 aregcmd stats Information for RADIUS Client

Stats Value	Meaning
RADIUS Client statistics for:	Provides client's IP address, name, and IP address type
TLSActiveConnectionCount	Number of active TLS connections established for the RADIUS client.
totalAuthAccessRequests	Number of authentication access requests that are received by Prime Access Registrar from the client.
totalAuthDupAccessRequests	Number of duplicate authentication access requests that are received by Prime Access Registrar from the client.
totalAuthAccessAccepts	Number of authentication access requests from the client that are accepted by Prime Access Registrar.
totalAuthAccessRejects	Number of authentication access requests from the client that are rejected by Prime Access Registrar.

Table 4-3 aregcmd stats Information for RADIUS Client (continued)

Stats Value	Meaning
totalAuthAccessChallenges	Number of authentication challenges that are faced by Prime Access Registrar for the requests raised by the client.
totalAuthMalformedAccessRequests	Number of malformed authentication access requests that are received by Prime Access Registrar from the client.
totalAuthBadAuthenticators	Number of bad authentication access requests that are received by Prime Access Registrar from the client.
totalAuthPacketsDropped	Number of authentication access requests received from the client that are dropped by Prime Access Registrar. The packets, which are invalid and do not fulfill the parsing conditions, are dropped.
totalAuthUnknownTypes	Number of unknown authentication access requests that are received by Prime Access Registrar from the client.
totalAccPacketsDropped	Number of accounting access requests received from the client that are dropped by Prime Access Registrar. The packets, which are invalid and do not fulfill the parsing conditions, are dropped.
totalAccRequests	Number of accounting access requests received by Prime Access Registrar from the client.
totalAccDupRequests	Number of duplicate accounting access requests that are received by Prime Access Registrar from the client.
totalAccResponses	Number of accounting response sent by Prime Access Registrar to the client
totalAccBadAuthenticators	Number of bad accounting access requests that are received by Prime Access Registrar from the client.
totalAccMalformedRequests	Number of malformed accounting access requests that are received by Prime Access Registrar from the client.
totalAccNoRecords	Number of accounting access requests that are received with no records by Prime Access Registrar from the client.
totalAccUnknownTypes	Number of unknown accounting access requests that are received by Prime Access Registrar from the client.

status

Use the **aregcmd** command **status** to learn whether or not the specified server has been started. When the server is running, Prime Access Registrar displays its health.

The syntax is:

status

stop

Use the **aregcmd** command **stop** to cause the server to no longer accept requests.

The syntax is:

stop

tacacs-stats

Use the **aregcmd** command **tacacs-stats** to provide statistical information of TACACS+.

The syntax is:

tacacs-stats

The following is an example of the statistical information provided after you issue the **tacacs-stats** command:

```
Global Tacacs+ Statistics
    serverStartTime = Mon Apr 15 01:17:34 2013
    serverResetTime = Mon Apr 15 01:17:34 2013
    serverState = Running
    totalPacketsReceived = 60
    totalPacketsSent = 60
    totalRequests = 60
    totalResponses = 60
    totalAuthenticationRequests = 2
    totalAuthenticationAccepts = 2
    totalAuthenticationRejects = 0
    totalAuthenticationChallengeRequests = 0
    totalAuthenticationResponses = 2
    totalAuthorizationRequests = 56
    totalAuthorizationAccepts = 38
    totalAuthorizationRejects = 18
    totalAuthorizationResponses = 56
    totalAccountingRequests = 2
    totalAccountingAccepts = 2
    totalAccountingRejects = 0
    totalAccountingResponses = 2
    totalPacketsInUse = 0
    totalPacketsDropped = 0
```

See the "Using the Graphical User Interface" chapter of the *Cisco Prime Access Registrar 8.0 User Guide* for more details on TACACS statistics information.

tacacs-reset-stats

Use the **aregcmd** command **tacacs-reset-stats** to reset TACACS+ statistics displayed with the **stats** command. The **tacacs-reset-stats** command also resets SNMP counters.

The **tacacs-reset-stats** command provides a way of resetting the TACACS+ statistics without having to reload or restart the server.

The syntax is:

tacacs-reset-stats

dia-stats

Use the **aregcmd** command **dia-stats** to provide statistical information of Diameter.

The syntax is:

dia-stats

The following is an example of the statistical information provided for a Diameter remote server after you issue the following command:

```
dia-stats /Radius/RemoteServers/dia

Diameter Remote server statistics for: dia, 10.81.79.76, port 3868
    active = TRUE
    cDiaRemSvrRTTAverage = 25ms
    cDiaRemSvrRTTDeviation = 0ms
    cDiaRemSvrServerType  = Diameter
    cDiaRemSvrTotalRequestsPending = 0
    cDiaRemSvrTotalRequestsOutstanding = 0
    cDiaRemSvrTotalRequestsAcknowledged = 0
    cDiaRemSvrStatsState = Closed
    cDiaRemSvrStatsASRsIn= 0
    cDiaRemSvrStatsASRsOut= 0
    cDiaRemSvrStatsASAsIn = 0
    cDiaRemSvrStatsASAsOut = 0
    cDiaRemSvrStatsACRsIn = 0
    cDiaRemSvrStatsACRsOut = 0
    cDiaRemSvrStatsACAsIn = 0
    cDiaRemSvrStatsACAsOut = 0
    cDiaRemSvrStatsCERsIn = 0
    cDiaRemSvrStatsCERsOut = 0
    cDiaRemSvrStatsCEAsIn = 0
    cDiaRemSvrStatsCEAsOut = 0
    cDiaRemSvrStatsDWRsIn = 0
    cDiaRemSvrStatsDWRsOut = 0
    cDiaRemSvrStatsDWAsIn = 0
    cDiaRemSvrStatsDWAsOut = 0
    cDiaRemSvrStatsDPRsIn = 0
    cDiaRemSvrStatsDPRsOut = 0
    cDiaRemSvrStatsDPAsIn = 0
    cDiaRemSvrStatsDPAsOut = 0
    cDiaRemSvrStatsRARsIn = 0
    cDiaRemSvrStatsRARsOut = 0
    cDiaRemSvrStatsRAAsIn = 0
    cDiaRemSvrStatsRAAsOut = 0
    cDiaRemSvrStatsSTRsIn= 0
    cDiaRemSvrStatsSTRsOut = 0
    cDiaRemSvrStatsSTAsIn = 0
    cDiaRemSvrStatsSTAsOut = 0
    cDiaRemSvrStatsRedirectEvents = 0
    cDiaRemSvrStatsAccDupRequests = 0
    cDiaRemSvrStatsMalformedRequests = 0
    cDiaRemSvrStatsAccsNotRecorded = 0
    cDiaRemSvrStatsWhoInitDisconnect = 0
    cDiaRemSvrStatsAccRetrans = 0
    cDiaRemSvrStatsTotalRetrans= 0
    cDiaRemSvrStatsAccPendRequestsOut = 0
    cDiaRemSvrStatsAccReqstsDropped = 0
    cDiaRemSvrStatsHByHDropMessages = 0
    cDiaRemSvrStatsEToEDupMessages= 0
    cDiaRemSvrStatsUnknownTypes= 0
    cDiaRemSvrStatsProtocolErrors = 0
    cDiaRemSvrStatsTransientFailures = 0
```

```
    cDiaRemSvrStatsPermanentFailures = 0
    cDiaRemSvrStatsDWCurrentStatus= 0
    cDiaRemSvrStatsTransportDown = 0
    cDiaRemSvrStatsTimeoutConnAtmpts = 5
    cDiaRemSvrStatsMARsIn = 0
    cDiaRemSvrStatsMARsOut = 0
    cDiaRemSvrStatsMAAsIn= 0
    cDiaRemSvrStatsMAAsOut = 0
    cDiaRemSvrStatsSARsIn = 0
    cDiaRemSvrStatsSARsOut = 0
    cDiaRemSvrStatsSAAsIn = 0
    cDiaRemSvrStatsSAAsOut = 0
    cDiaRemSvrStatsRTRsIn= 0
    cDiaRemSvrStatsRTRsOut = 0
    cDiaRemSvrStatsRTAsIn = 0
    cDiaRemSvrStatsRTAsOut = 0
    cDiaRemSvrStatsPPRsIn= 0
    cDiaRemSvrStatsPPRsOut = 0
    cDiaRemSvrStatsPPAsIn = 0
    cDiaRemSvrStatsPPAsOut = 0
    cDiaRemSvrStatsDERsIn= 0
    cDiaRemSvrStatsDERsOut = 0
    cDiaRemSvrStatsDEAsIn = 0
    cDiaRemSvrStatsDEAsOut = 0
    cDiaRemSvrStatsAARsIn= 0
    cDiaRemSvrStatsAARsOut = 0
    cDiaRemSvrStatsAAAsIn = 0
    cDiaRemSvrStatsAAAsOut = 0
```

For detailed information about the Diameter remote server stats, see the "Using the Graphical User Interface" chapter of the *Cisco Prime Access Registrar 8.0 User Guide*.

The following is an example of the statistical information provided for a Diameter peer after you issue the following command:

dia-stats /Radius/Clients/vm24

```
Diameter Peer statistics for: dia, 10.81.79.76, port 3868
cdbpPeerStatsState = Closed
    cdbpPeerStatsASAsOut = 0
    cdbpPeerStatsACRsIn = 0
    cdbpPeerStatsACRsOut = 0
    cdbpPeerStatsACAsIn = 0
    cdbpPeerStatsACAsOut = 0
    cdbpPeerStatsCERsIn = 1
    cdbpPeerStatsCERsOut = 0
    cdbpPeerStatsCEAsIn = 0
    cdbpPeerStatsCEAsOut = 0
    cdbpPeerStatsDWRsIn = 0
    cdbpPeerStatsDWRsOut = 0
    cdbpPeerStatsDWAsIn = 0
    cdbpPeerStatsDWAsOut = 0
    cdbpPeerStatsDPRsIn = 0
    cdbpPeerStatsDPRsOut = 0
    cdbpPeerStatsDPAsIn = 0
    cdbpPeerStatsDPAsOut = 0
    cdbpPeerStatsRARsIn = 0
    cdbpPeerStatsRARsOut = 0
    cdbpPeerStatsRAAsIn = 0
    cdbpPeerStatsRAAsOut = 0
    cdbpPeerStatsSTRsIn= 0
    cdbpPeerStatsSTRsOut = 0
    cdbpPeerStatsSTAsIn = 0
    cdbpPeerStatsSTAsOut = 0
```

```
cdbpPeerStatsRedirectEvents = 0
cdbpPeerStatsAccDupRequests = 0
cdbpPeerStatsMalformedReqsts = 0
cdbpPeerStatsAccsNotRecorded = 0
cdbpPeerStatsWhoInitDisconnect = 0
cdbpPeerStatsAccRetrans = 0
cdbpPeerStatsTotalRetrans= 0
cdbpPeerStatsAccPendReqstsOut = 0
cdbpPeerStatsAccReqstsDropped = 0
cdbpPeerStatsHByHDropMessages = 0
cdbpPeerStatsEToEDupMessages= 0
cdbpPeerStatsUnknownTypes= 0
cdbpPeerStatsProtocolErrors = 0
cdbpPeerStatsTransientFailures = 0
cdbpPeerStatsPermanentFailures = 0
cdbpPeerStatsDWCurrentStatus= 2
cdbpPeerStatsTransportDown = 0
cdbpPeerStatsTimeoutConnAtmpts = 0
cdbpPeerStatsASRsIn= 0
cdbpPeerStatsASRsOut= 0
cdbpPeerStatsASAsIn = 0
cdbpPeerStatsDERsIn = 0
cdbpPeerStatsDERsOut = 0
cdbpPeerStatsDEAsIn = 0
cdbpPeerStatsDEAsOut = 0
cdbpPeerStatsAARsIn = 0
cdbpPeerStatsAARsOut = 0
cdbpPeerStatsAAAsIn = 0
cdbpPeerStatsAAAsOut = 0
cdbpPeerStatsMARsIn = 0
cdbpPeerStatsMARsOut = 0
cdbpPeerStatsMAAsIn = 0
cdbpPeerStatsMAAsOut = 0
cdbpPeerStatsSARsIn = 0
cdbpPeerStatsSARsOut = 0
cdbpPeerStatsSAAsIn = 0
cdbpPeerStatsSAAsOut = 0
cdbpPeerStatsRTRsIn = 0
cdbpPeerStatsRTRsOut = 0
cdbpPeerStatsRTAsIn = 0
cdbpPeerStatsRTAsOut = 0
cdbpPeerStatsPPRsIn = 0
cdbpPeerStatsPPRsOut = 0
cdbpPeerStatsPPAsIn = 0
```

For detailed information about the Diameter peer stats, see the "Using the Graphical User Interface" chapter of the *Cisco Prime Access Registrar 8.0 User Guide*.

dia-stats-reset

Use the **aregcmd** command **dia-stats-reset** to reset Diameter statistics displayed with the **stats** command. The **dia-stats-reset** command also resets SNMP counters.

The **dia-stats-reset** command provides a way of resetting the Diameter statistics without having to reload or restart the server.

The syntax is:

dia-stats-reset

trace

Use the **aregcmd** command **trace** to set the trace level in the specified server to a new value. The trace level governs how much information is displayed about the contents of a packet. When the trace level is zero, no tracing is performed. The higher the trace level, the more information displayed. The highest trace level currently used by the Prime Access Registrar server is trace level 5.

Note Although the highest **trace** level supported by the Prime Access Registrar server is **trace** level 5, an extension point script might use a higher level. There is no harm in setting the **trace** to a level higher than 5. However, increasing the trace level impacts the system performance.

The **trace** levels are inclusive, meaning that if you set **trace** to level 3, you will also get the information reported for **trace** levels 1 and 2. If you set trace level 4, you also get information reported for **trace** levels 1, 2, and 3.

When you do not specify a server, Prime Access Registrar sets the **trace** level for all of the servers in the current cluster. When you do not specify a value for the **trace** level, Prime Access Registrar displays the current value of the **trace** level. The default is 0.

The syntax for setting the **trace** level is:

trace [<*server*>] [<*level*>]

Table 4-4 lists the different **trace** levels and the information returned.

Table 4-4 Trace Levels and Information Returned

Trace Level	Information Returned by Trace Command
0	No trace performed.
1	Reports when a packet is sent or received or when there is a change in a remote server's status.
2	Indicates the following: • Which services and session managers are used to process a packet • Which client and vendor objects are used to process a packet • Detailed remote server information for LDAP and RADIUS, such as sending a packet and timing out • Details about poorly formed packets • Details included in trace level 1
3	Indicates the following: • Error traces in TCL scripts when referencing invalid RADIUS attributes. • Which scripts have been executed • Details about local UserList processing • Details included in trace levels 1 and 2

Table 4-4 Trace Levels and Information Returned (continued)

Trace Level	Information Returned by Trace Command
4	Indicates the following: • Information about advanced duplication detection processing • Details about creating, updating, and deleting sessions • Trace details about all scripting APIs called • Details included in trace levels 1, 2, and 3
5	Indicates the following: • Details about use of the policy engine including: – Which rules were run – What the rules did – If the rule passed or failed – Detailed information about which policies were called • Details included in trace levels 1, 2, 3, and 4

trace-file-count

Use the **aregcmd** command **trace-file-count** to change the trace log file count dynamically without requiring a server reload. The syntax is:

trace-file-count ***n***

Where *n* is a number that specifies the number of trace log files. This function is helpful for debugging situations when you do not want to perform a **reload**.

unset

Use the **aregcmd** command **unset** to remove items from an ordered list. Specify the numeric index of the element to remove. When the ordered list is not the current context, specify the path to the list before specifying the numeric index.

When you remove an item from the list, Prime Access Registrar renumbers the list.

The syntax is:

unset [<*path*>/]<*index*>

This command applies to lists of servers by index, the **Profiles** attribute list, the Rules Attributes list, the enumeration list in the Attribute dictionary, and the **LDAPtoRadiusMappings** and **LDAPtoEnvironmentMappings** mappings.

validate

Use the **aregcmd** command **validate** to check the consistency and validity of the specified server's configuration. If Prime Access Registrar discovers any errors, it displays them.

The syntax is:

validate

OpenSSL Commands

This section contains a list of **OpenSSL** commands. You can use them on the command line or insert them into scripts.

This section contains the following topics:

- ecparam
- req
- ca

ecparam

Use the **OpenSSL** command **ecparam** to manipulate or generate ellipitical curve (EC) parameter files.

The syntax is:

ecparam

req

Use the **OpenSSL** command **req** to create and process certificate requests.

The syntax is:

req

ca

Use the **OpenSSL** command **ca** used to sign certificate requests and generate CRLs it also maintains a text database of issued certificates and their status.

The syntax is:

ca

aregcmd Command Logging

aregcmd now records the commands that are either entered interactively, on the command line, or executed in batch mode. The recorded commands are saved in the **aregcmd_log** file, which resides in the **logs** directory within the Prime Access Registrar installation directory.

For security reasons, **aregcmd** blocks out the actual password that is entered as part of the command and replaces it with <*passwd*>.

In interactive mode, **aregcmd** logs the actions that are taking place in the exit/logout dialog box. The action can be **save** or **not save** if the configuration database has been modified after the last execution of the **save** command.

In non-interactive (batch or command-line) mode, **aregcmd** replaces the empty field with a NULL string.

aregcmd is now installed as a **setgid** program where the group is set to **staff**. This allows a non-root user to run **aregcmd** while still being able to write to the **aregcmd_log** log file. During the installation of the Prime Access Registrar software, you are prompted whether you want to install **aregcmd** with **setuid/setgid** permissions. You must reply "yes" unless you only run **aregcmd** as user **root**.

The following is the format of an entry in the exit/logout dialog box when **not save** has been specified:

```
mm/dd/yyyy HH:MM:SS aregcmd Info Configuration 0 [<clustername> <username>] ( exit )
mm/dd/yyyy HH:MM:SS aregcmd Info Configuration 0 [<clustername> <username>] ( *** New
                                                 config is not saved! ...proceed to logout.)
```

The following is sample output of an entry in the exit/logout dialog box when **not save** has been specified:

```
10/12/2013 16:18:56 aregcmd Info Configuration 0 [localhost admin] --> quit
10/12/2013 16:19:02 aregcmd Info Configuration 0 [localhost admin] --> *** New config is
                                                                       not saved! ...proceed to logout.
```

The following is the format of an entry in the exit/logout dialog box when **save** has been specified:

```
mm/dd/yyyy HH:MM:SS aregcmd Info Configuration 0 [<clustername> <username>] ( exit )
mm/dd/yyyy HH:MM:SS aregcmd Info Configuration 0 [<clustername> <username>] ( *** New
                                                   config saved!...proceed to logout.)
```

aregcmd Command Line Editing

Commands entered at the **aregcmd** prompt can be edited with a subset of the standard EMACS-style keystrokes. In addition, the command history can be accessed using the arrow keys on the keyboard. Use the Up arrow to retrieve the previous command and the Down arrow to retrieve the next command. A description of the supported key strokes are shown in Table 4-5.

Table 4-5 aregcmd Command Line Editing Keystrokes

Key Stroke	Description
Ctrl A	Go to the beginning of the line.
Ctrl B	Move back one character.
Ctrl D	Delete the character at the cursor.
Ctrl E	Go to the end of the line.
Ctrl F	Move forward one character.
Ctrl N	Retrieve the next line.
Ctrl P	Retrieve the previous line.

aregcmd Error Codes

Table 4-6 lists the error codes used in **aregcmd**.

Table 4-6 aregcmd Error Codes

Error Code	Meaning
200	OK
300	Command failed to parse; usually an unbalanced quotation
301	Unknown command; usually caused by a misspelled command
302	Ambiguous command; characters you have entered select more than one command
303	Not logged in; you have logged out of **aregcmd** and attempted another **aregcmd** command
304	Too few arguments
305	Too many arguments
306	Invalid argument
307	Object not found or path ambiguous; you have tried to set an unknown object or provided a partial name that matches multiple objects
308	Object already exists
309	Confirmation password did not match; when setting a user password, the re-entered password did not match the initial entry
310	Command failed; a generic response for a command that failed for a reason other than those listed here
311	Command is interactive; possibly due to attempting to batch mode with an interactive command
312	Validation failed; a new or modified object is not valid
313	Failed to reread; an error occurred while synchronizing changes from another **aregcmd** session; occurs only when using multiple **aregcmd** instances
314	Failed to open the pager; the PAGER environment variable is set to something that does not exist and the **stats** command is entered
315	Property is not modifiable; an administrator has attempted to modify a read-only property
316	Command failed: session is ViewOnly; an view-only administrator has attempted to modify a property
317	Command failed: server is a Replication Slave; an administrator has attempted to modify a replicated property on a slave of an SMDBR network
400	Login failed; an incorrect username or password was given during **aregcmd** log in
401	Unable to access server; **aregcmd** was unable to communicate with the **radius** process usually because the process is not running or is otherwise unresponsive
402	Login failed: version of aregcmd is incompatible with server; a new version of **aregcmd** has tried to connect with an older version of Prime Access Registrar
406	User has an active **aregcmd** session. Occurs when the user tries to login to a concurrent **aregcmd** session.
500	Internal error; a generic **aregcmd** error

Configuring and Monitoring the RADIUS Server

This chapter describes the objects you use to configure and operate your Cisco Prime Access Registrar (Prime Access Registrar) RADIUS server.

Prime Access Registrar is configured and operated through a set of *objects*. These objects are arranged in a hierarchy, with some of the objects containing subobjects; just as in a UNIX file system, in which directories can contain subdirectories. All of the objects, except those that are merely lists, contain properties that define the attributes or behavior of the object.

This chapter describes the following Prime Access Registrar objects:

- Radius— root of the configuration hierarchy
- UserLists—contains individual UserLists, which in turn contain users
- UserGroups—contains individual UserGroups
- Policies—contains individual Policies
- Clients—contains individual Clients
- Vendors—contains individual Vendors
- Scripts—contains individual Scripts
- Services—contains individual Services
- Session Managers—contains individual Session Managers
- Resource Managers—contains individual Resource Managers
- Profiles—contains individual Profiles
- Rules—contains individual Rules
- Fast Rules—contains attributes to add, modify, and delete in the request, response, and environment dictionaries.
- Translations—contains individual Translations
- TranslationGroups—contains individual Translation Groups
- Remote Servers—contains individual RemoteServers
- Advanced—contains advanced properties, Ports, Interfaces, Reply Messages, and the Attribute dictionary

Radius

The **Radius** object is the root of the hierarchy. For each installation of the Cisco Prime Access Registrar server, there is one instance of the **Radius** object. You reach all other objects in the hierarchy from the **Radius**.

The following is a listing of the RADIUS server object:

```
[ //localhost/Radius ]
    Name = Radius
    Description =
    Version = 7.2.0.0
    IncomingScript~ = query
    OutgoingScript~ =
    DefaultAuthenticationService~ = local-users
    DefaultAuthorizationService~ = local-users
    DefaultAccountingService~ = local-file
    DefaultSessionService~ =
    DefaultSessionManager~ = mgr-rad
    UserLists/
    UserGroups/
    Policies/
    Clients/
    Vendors/
    Scripts/
    Services/
    SessionManagers/
    ResourceManagers/
    Profiles/
    Rules/
    Translations/
    TranslationGroups/
    RemoteServers/
    CommandSets/
    DeviceAccessRules/
    FastRules/
    Advanced/
    Replication/
```

Table 5-1 lists the **Radius** properties. You you can set or change Radius properties using the Cisco Prime Access Registrar **aregcmd** commands.

Note When a field is listed as required, it means a value must be supplied; that is, the value must be set. You can use the default (if it is supplied) or you can change it to something else, but you cannot unset it. You *must* supply values for the required fields and for which no defaults exist.

Table 5-1 Radius Properties

Property	Description
Name	Required; must be unique in the list of servers in the cluster
Description	Optional description of the server
Version	Required; the currently installed version of Prime Access Registrar
IncomingScript	Optional; if there is a script, it is the first script Cisco Prime Access Registrar runs when it receives a request from any client and/or for any service

Table 5-1 Radius Properties (continued)

Property	Description
OutgoingScript	Optional; if there is a script, it is the last script Cisco Prime Access Registrar runs before it sends a response to any client
DefaultAuthenticationService	Optional; Cisco Prime Access Registrar uses this property when none of the incoming scripts sets the environment dictionary variable **Authentication-Service**
DefaultAuthorizationService	Optional; Cisco Prime Access Registrar uses this property when none of the incoming scripts sets the environment dictionary variable **Authorization-Service**
DefaultAccountingService	Optional; Cisco Prime Access Registrar uses this property when none of the incoming scripts sets the environment dictionary variable **Accounting-Service**
DefaultSessionService	Cisco Prime Access Registrar uses this property when none of the incoming scripts sets the environment dictionary variable **Session-Service**. This field is mandatory if you are upgrading to a later version of Prime Access Registrar.
DefaultSessionManager	Cisco Prime Access Registrar uses this property if none of the incoming scripts sets the environment dictionary variable **Session-Manager**. This field is mandatory if you are upgrading to a later version of Prime Access Registrar.

The remaining Cisco Prime Access Registrar objects are sub-objects of the **Radius** object.

UserLists

The **UserLists** object contains all of the individual UserLists, which in turn, contain the specific users stored within Cisco Prime Access Registrar. Cisco Prime Access Registrar references each specific UserList by **name** from a Service whose type is set to **local**. When Cisco Prime Access Registrar receives a request, it directs it to a Service. When the Service has its type property set to **local**, the Service looks up the user's entry in the specific UserList and authenticates and/or authorizes the user against that entry.

Note Usernames might not include the forward slash (**/**) character. If the Cisco Prime Access Registrar server receives an access request packet with a User-Name attribute containing a forward slash character and the Prime Access Registrar server uses an internal UserList to look up users, the server produces an error (AX_EINVAL) and might fail. If usernames require a forward slash, use a script to translate the slash to an acceptable, unused character.

You can have more than one UserList in the **UserLists** object. Therefore, use the **UserLists** object to divide your user community by organization. For example, you might have separate **UserLists** objects for Company A and B, or you might have separate **UserLists** objects for different departments within a company.

Using separate **UserLists** objects allows you to have the same name in different lists. For example, if your company has three people named `Bob` and they work in different departments, you could create a UserList for each department, and each Bob could use his own name. Using UserLists lets you avoid the problem of `Bob1`, `Bob2`, and so on.

If you have more than one UserList, you can have a script Cisco Prime Access Registrar can run in response to requests. The script chooses the Service, and the Service specifies the actual UserList which contains the user. The alternative is dynamic properties.

The subobjects are the Users listed by name. Table 5-2 lists the **UserLists** object properties.

Table 5-2 UserLists Properties

Property	Description
Name	Required; must be unique in UserLists.
Description	Optional description of the UserList.

Users

The **Users** object contains all of the information necessary to authenticate a user or authorize a user. Users in local UserLists can have multiple profiles. Table 5-3 lists the **Users** object properties.

Table 5-3 Users Properties

Property	Description
Name	Required; must be unique in the specific UserList.
Description	Optional description of the user.
Password	Required; length must be between 0-253 characters.
Enabled	Required; default is TRUE, which means the user is allowed access. Set to FALSE to cause Cisco Prime Access Registrar to deny the user access.
Group (Overridden by User-Group)	Optional; when you set this to the name of a UserGroup, Cisco Prime Access Registrar uses the properties specified in that UserGroup to authenticate and/or authorize the user.
BaseProfile (Overridden by User-Profile)	Optional; when you set this to the name of a Profile and the service-Type is not equal to Authenticate Only, Cisco Prime Access Registrar adds the properties in the Profile to the Response dictionary as part of the authorization.
AuthenticationScript	Optional; when you set this property to the name of a script, you can use the script to perform additional authentication checks to determine whether to accept or reject the user.
AuthorizationScript	Optional; when you set this property to the name of a script, you can use the script to add, delete, or modify the attributes of the Response dictionary.
UserDefined1	Optional; you can use this property to store notational information which you can then use to filter the UserList. This property also sets the environment variable for UserDefined1.

HiddenAttributes Property

The HiddenAttributes property in the user object provides a concatenation of all user-level reply attributes. The Prime Access Registrar server uses the HiddenAttributes property to construct and populate a virtual attributes directory.

The HiddenAttributes property is, in fact, hidden. It is not displayed and cannot be set or modified using **aregcmd**, but when an administrator issues a **save**, all values from the user's Attributes directory go into the HiddenAttributes property and the persistent storage.

The attributes are added in a replace-if-present-add-if-not manner as used in the UserGroup-Base-Profile and User-Base-Profile.

The order of application of the attributes is as follows:

- UserGroup Base Profile
- UserGroup Attributes
- User Base Profile
- User Attributes

UserGroups

The **UserGroups** objects allow you to maintain common authentication and authorization attributes in one location, and then have many users reference them. By having a central location for attributes, you can make modifications in one place instead of having to make individual changes throughout your user community.

For example, you can use several **UserGroups** to separate users by the services they use, such as a group specifying PPP and another for Telnet.

Table 5-4 lists the **UserGroups** properties.

Table 5-4 UserGroups Properties

Property	Description
Name	Required; must be unique in the **UserGroup** list.
Description	Optional description of the group.
BaseProfile	Optional; when you set this to the name of a Profile, Cisco Prime Access Registrar adds the properties in the Profile to the response dictionary as part of the authorization.
AuthenticationScript	Optional; when you set this property to the name of a Script, you can use the Script to perform additional authentication checks to determine whether to accept or reject the user.
AuthorizationScript	Optional; when you set this property to the name of a Script, you can use the Script to add, delete, or modify the attributes of the Response dictionary.

Policies

A Policy is a set of rules applied to an Access-Request. If you are using **Policies**, the first one that must be created is SelectPolicy.

Table 5-5 lists the properties required for a given **Policy**.

Table 5-5 Policies Properties

Property	Description
Name	Required; must be unique in the **Policies** list
Description	Optional description of the Policy
Grouping	Optional grouping of rules

Clients

All NASs and proxy clients that communicate directly with Cisco Prime Access Registrar must have an entry in the **Clients** list. This is required because NAS and proxy clients share a secret with the RADIUS server which is used to encrypt passwords and to sign responses. Table 5-6 lists the **Client** object properties.

Table 5-6 RADIUS Client Properties

Property	Description
Name	Required and should match the Client identifier specified in the standard RADIUS attribute, **NAS-Identifier**. The name must be unique within the Clients list.
Description	Optional description of the client.
Protocol	Required; specifies the client protocol which can be **Radius**, **Diameter**, **Radius-TLS**, or **Tacacs-and-Radius**.

Table 5-6 RADIUS Client Properties (continued)

Property	Description
IPAddress	Required; must be a valid IP address and unique in the Clients list. Cisco Prime Access Registrar uses this property to identify the Client that sent the request, either using the source IP address to identify the immediate sender or using the **NAS-IP-Address** or **NAS-IPv6-Address** attribute in the Request dictionary to identify the NAS sending the request through a proxy. When a range is configured for a Client's IPAddress property, any incoming requests whose source address belongs to the range specified, will be allowed for further processing by the server. Similarly when a wildcard (an asterisk '*' in this case) is specified, any incoming requests whose source address matches the wildcard specification will be allowed. In both the cases, the configured client properties like SharedSecret, and Vendor are used to process the requests. You can specify a range of IP addresses using a hyphen as in: 100.1.2.11-20 You can use an asterisk wildcard to match all numbers in an IP address octet as in: 100.1.2.* You can specify an IPAddress and a subnet mask together using Classless Inter-Domain Routing (CIDR) notation as in: 100.1.2.0/24 You can use the IPAddress property to set a base address and use the NetMask property to specify the number of clients in the subnet range. The IP address format is enhanced to support IPv6 apart from IPv4. You can use an asterisk wildcard to match all numbers in an IP address octet as in: 1124:1124:1124:1124:*:*:*:* **Note** The IPv6 address must be in standard notation.
SharedSecret	Required; must match the secret configured in the Client.
Type	Required; accept the default (NAS), or set it to ATM, Proxy, or NAS+Proxy.
Vendor	Optional; you can use this property when you need special processing for a specific vendor's NAS. To use this property, you must configure a **Vendor** object and include a Script. Cisco Prime Access Registrar provides five Scripts you can use: one for Ascend, Cisco, Cabletron, Altiga, and one for USR. You can also provide your own Script.
IncomingScript	Optional; you can use this property to specify a Script you can use to determine the services to use for authentication, authorization, and/or accounting.

Table 5-6 RADIUS Client Properties (continued)

Property	Description
OutgoingScript	Optional; you can use this property to specify a Script you can use to make any Client-specific modifications when responding to a particular Client.
EnableDynamicAuthorization	Optional; when set to TRUE, this property enables Change of Authorization and Packet of Disconnect features.
DynamicAuthorizationServer	This subdirectory is only present in a client with EnableDynamicAuthorization set to TRUE and contains properties required for CoA and PoD requests.
Port	Located under the DynamicAuthorizationServer subdirectory, the default port is 3799.
InitialTimeout	Located under the DynamicAuthorizationServer subdirectory, the default is 5000.
MaxTries	Located under the DynamicAuthorizationServer subdirectory, the default is 3.
DynamicAuthSharedSecret	Located under the DynamicAuthorizationServer subdirectory, this is the shared secret used for communicating CoA and PoD packets with the client.
PODAttributeGroup	This property is found under the DynamicAuthorizationServer subdirectory and points to a group of attributes to be included in a POD request sent to this client. These attribute groups are created and configured under the AttributeGroups subdirectory in **/Radius/Advanced**.
COAAttributeGroup	This property is found under the DynamicAuthorizationServer subdirectory and points to a group of attributes to be included in a CoA request sent to this client. These attribute groups are created and configured under the AttributeGroups subdirectory in **/Radius/Advanced**.
NetMask	Specifies the subnet mask used with the network address setting configured for the IPAdress property when configuring a range of IP addresses. This property is not used for a single client with an IP address only. The NetMask property is used to configure multiple clients when you configure a base IP address in the IPAddress property. You can set the NetMask property for a range of 256 clients using the following example: **set NetMask 255.255.255.0** When the NetMask property indicates a pool of 256 address (255.255.255.0), the range of addresses reserved for clients is 0-255, as in 100.1.1.0-100.1.1.255. **Note** If you set the NetMask property, validation will fail if you attempt to specify a subnet mask using CIDR notation with the IPAddress property (described above).

Table 5-6 RADIUS Client Properties (continued)

Property	Description
EnableNotifications	Required; the default value is FALSE and indicates the client is not capable of receiving Accounting-Stop notifications from the Prime Access Registrar server. When set to TRUE, the client can receive Accounting-Stop notifications from the Prime Access Registrar server and additional properties must be configured under a new sub-directory named NotificationProperties.
NotificationProperties	When the EnableNotifications property is set to TRUE, this subdirectory contains additional properties required to support the Query-Notify feature.
Port	Located under the NotificationProperties subdirectory, specifies the port used by the Prime Access Registrar server to receive Accounting-Stop packets. Required when EnableNotifications is set to TRUE; the default value is 1813.
InitialTimeout	Located under the NotificationProperties subdirectory, specifies the timeout value in milliseconds the Prime Access Registrar server waits for an Accounting-Response packet before attempting a retry (sending another Accounting-Stop packet to the client). Required when EnableNotifications is set to TRUE; the default value is 5000.
MaxTries	Located under the NotificationProperties subdirectory, specifies the number of times the Prime Access Registrar server sends an Accounting-Stop packet to a client. Required when EnableNotifications is set to TRUE; the default value is 3.
NotificationAttributeGroup	Located under the NotificationProperties subdirectory, specifies the name of an attribute group under **/Radius/Advanced/AttributeGroups** that contains the attributes to be included when sending an the Accounting-Stop packet to this client. Required when EnableNotifications is set to TRUE; there is no default value. You must provide the name of a valid AttributeGroup and the named AttributeGroup must contain at least one valid attribute, or validation will fail.
EnforceTrafficThrottling	Required; the default value is TRUE and indicates enforce traffic throttling for this client. This property is under **/Radius/Advanced/MaximumOutstanding/IncomingRequests**. When set to FALSE, the traffic throttling for the packet coming from this client is bypassed.

Table 5-7 describes the Diameter client properties.

Table 5-7 Diameter Client Properties

Property	Description
Name	Required; must be unique in the client list.
Description	Optional; description of the client.
Protocol	Required; specifies the client protocol which can be Radius or Diameter.
HostName	Required; hostname or IP address of the Diameter client.
Vendor	Optional; you can use this property when you need special processing for a specific vendor's peer.
IncomingScript	Optional; specifies a script that you can use to make client-specific modifications when a request is received from a client.
OutgoingScript	Optional; specifies a script that you can use to make any client-specific modifications when responding to a particular client.
Port	Required; port on which the client connects with Prime Access Registrar server.
SCTP-Enabled	Required, default value is False. If set to TRUE, SCTP will be used to establish the connection with the peer else TCP will be used. If SCTP is enabled, you can configure SCTP parameters for the Diameter client. For details, see the "Diameter" chapter of the *Cisco Prime Access Registrar 8.0 User Guide*.

Prime Access Registrar supports configuring clients with type as Radius-TLS (Radius over TLS). This will enable the client to send the radius request using TLS connection.

Table 5-8 describes the RADIUS-TLS client properties.

Table 5-8 RADIUS-TLS Client Properties

Property	Description
MaximumTLSConnections	This parameter is applicable for protocol of type **radius-tls**. Maximum number of TLS connections that the client can establish with Prime Access Registrar. Default value is one. Maximum number of TLS connections allowed per client is 50.
TLSOptions / RTLS Options These parameters are applicable for the following client types: • Diameter with TLS-Enabled option set as TRUE • RADIUS-TLS	
PrivateKeyPassword	The password used to protect the server's private key.

Table 5-8 RADIUS-TLS Client Properties (continued)

Property	Description
ServerKeyFile	The full pathname of the file containing the server's RSA private key. The pathname can be optionally prefixed with a special string that indicates the type of encoding used for the certificate. The valid encoding prefix is "PEM". If an encoding prefix is not present, the file is assumed to be in PEM format. The following example assumes that the subdirectory **pki** under **/cisco-ar** contains the server's certificate file. The file **server-key.pem** is assumed to be in PEM format. The file extension ***.pem*** is not significant. **set ServerKeyFile PEM:/cisco-ar/pki/server-key.pem**
ServerCertificateFile	The full pathname of the file containing the server's certificate or certificate chain used during the TLS exchange. The pathname can be optionally prefixed with a special string that indicates the type of encoding used for the certificate. The valid encoding prefix is PEM. If an encoding prefix is not present, the file is assumed to be in PEM format.
CACertificateFile	The full pathname of the file containing trusted CA certificates used for client verification. The file can contain more than one certificate, but all certificates must be in PEM format. DER encoding is not allowed.
CACertificatePath	The name of a directory containing trusted CA certificates (in PEM format) used for client verification. This parameter is optional, and if it is used there are some special preparations required for the directory it references. Each certificate file in this directory must contain exactly one certificate in PEM format. The server looks up the certificate files using the MD5 hash value of the certificate's subject name as a key. The directory must therefore also contain a set of symbolic links each of which points to an actual certificate file. The name of each symbolic link is the hash of the subject name of the certificate. For example, if a certificate file named **ca-cert.pem** is located in the CACertificatePath directory, and the MD5 hash of the subject name contained in **ca-cert.path.pem** is 1b96dd93, then a symbolic link named 1b96dd93 must point to **ca-cert.pem**. If there are subject name collisions such as multiple certificates with the same subject name, each link name must be indexed with a numeric extension as in 1b96dd93.0 and 1b96dd93.1.
PeerVerificationMode	Set to one of the following: • None—If the peer verification certificate must not be requested. • Optional—If peer verification certificate can be requested; but, verification is not required. • RequireCertificate—If peer certificate must be requested and verified.
VerificationDepth	Specifies the maximum length of the certificate chain used for client verification.

Table 5-8 RADIUS-TLS Client Properties (continued)

Property	Description
EnableAutoChaining	When set to TRUE, Prime Access Registrar sends its server certificate chain (Server-Cert -> IntermediateCA -> RootCA) while presenting the server certificate to the client for server side authentication. When set to FALSE, Prime Access Registrar sends only the server certificate (Server-Cert) to the client.
TCP Options These parameters are applicable **radius-tls** clients.	
KeepAliveIntervalTime	Time interval in seconds between individual keepalive probes.
TCPConnectionIdleTime	Time (in seconds) the connection can remain idle before TCP starts sending keepalive probes.
KeepAliveMaxtries	Maximum number of keepalive probes TCP can send before dropping the connection.

Following is a sample CLI configuration of a RADIUS-TLS client:

```
[ //localhost/Radius/Clients/radius-tls ]
Name = radius-tls
Description =
Protocol = radius-tls
IPAddress = 10.81.79.41
SharedSecret = <encrypted>
Type = NAS
Vendor =
IncomingScript~ =
OutgoingScript~ =
NetMask =
EnforceTrafficThrottling = TRUE
MaximumTLSConnections = 25
RTLSOptions/
       PrivateKeyPassword = cisco
       ServerCertificateFile = /opt/CSCOar/pki/cert.pem
       ServerKeyFile = /opt/CSCOar/pki/key.pem
       CACertificateFile = /opt/CSCOar/pki/root-cert.pem
       CACertificatePath =
       PeerVerificationMode = None/Optional/RequireCertificate
       VerificationDepth = 4
       EnableAutoChaining = True
TCPOptions/
       KeepAliveIntervalTime = 2
       TCPConnectionIdleTime = 3
       KeepAliveMaxTries = 2
```

Vendors

The **Vendor** object provides a central location for specifying all of the request and response processing a particular NAS or Proxy vendor requires. Depending on the vendor, it might be necessary to map attributes in the request from one set to another, or to filter out certain attributes before sending the response to the client. For more information about standard RADIUS attributes, see the “RADIUS Attributes” chapter of the *Cisco Prime Access Registrar 8.0 Reference Guide*.

Note When you have also set **/Radius/IncomingScript**, Cisco Prime Access Registrar runs that script before the vendor's script. Conversely, when you have set a **/Radius/Outgoing** script, Cisco Prime Access Registrar runs the vendor's script before that script.

Table 5-9 lists the **Vendor** object properties.

Table 5-9 Vendor Properties

Property	Description
Name	Required; must be unique in the Vendors list.
Description	Optional description of the vendor.
IncomingScript	Optional; when you specify an IncomingScript, Cisco Prime Access Registrar runs the script on all requests from clients that specify that vendor.
OutgoingScript	Optional; when you specify an OutgoingScript, Cisco Prime Access Registrar runs the script on all responses to the Client.

Scripts

The **Script** objects define the function Cisco Prime Access Registrar invokes whenever the **Script** is referenced by name from other objects in the configuration.

You can write three types of scripts:

- REX (RADIUS EXtension) scripts are written in C or C++, and thus are compiled functions that reside in shared libraries
- Tcl scripts are written in Tcl, and are interpreted functions defined in source files.
- Java scripts

Note For more information about how to write scripts and how to incorporate them into Cisco Prime Access Registrar, see Chapter 7, "Using Extension Points."

Note Cisco is not liable for scripts developed by clients. See Client scripting in user guide chapter 1 overview chapter.

Table 5-10 lists the **Script** object properties.

Table 5-10 Script Object Properties

Property	Description
Name	Required; must be unique in the Scripts list.
Description	Optional description of the script.
Language	Required; specify either REX, Tcl, or Java.

Table 5-10 Script Object Properties (continued)

Property	Description
Filename	Required; specifies either a relative or absolute path. When you specify a relative path, the path must be relative to the **$INSTALL/scripts/radius/$Language** directory. When you specify an absolute path, the server must be able to reach it.
EntryPoint	Optional; when not set, Prime Access Registrar uses the value specified in the **Name** property.
InitEntryPoint	Optional; if set, it must be the name of the global symbol Prime Access Registrar should call when it initializes the shared library at system start up, and just before it unloads the shared library.
InitEntryPointArg	Optional; when set, it provides the arguments to be passed to the **InitEntryPoint** in the environmental variable **Arguments**.
ClassName	For Java language scripts, the name of the class that implements the extension interface; the **.class** file should be placed in **/cisco-ar/scripts/radius/java**
InitializeArg	Optional for Java language scripts; set to a string to be passed to the Initialize method if the class implements the optional ExtensionWithInitialization interface.

The **InitEntryPoint** properties allow you to perform initialization before processing and then cleanup before stopping the server. For example, when Prime Access Registrar unloads the script (when it stops the RADIUS server) it calls the **InitEntryPoint** again to allow it to perform any clean-up operations as a result of its initialization. One use of the function might be to allow the script to close an open Accounting log file before stopping the RADIUS server.

Note When you use a Prime Access Registrar file service, Prime Access Registrar automatically closes any opened files. However, if you write scripts that manipulate files, you are responsible for closing them.

Note If you have more than one extension point script (defined under **/Radius/Scripts**) using the same Java class, only one instance of the class is created and used for all the extension point scripts.

Services

Cisco Prime Access Registrar supports authentication, authorization, and accounting (AAA) services. In addition to the variety of built-in AAA services (specified in the **Type** property), Cisco Prime Access Registrar also enables you to add new AAA services through custom shared libraries.

Table 5-11 lists the common **Services** properties. There are additional properties depending on the type of service.

Table 5-11 Common Service Properties

Property	Description
Name	Required; must be unique in the Services list.
Description	Optional description of the service.
Type	Required, must set it to a valid Prime Access Registrar service.
OutagePolicy	Required; the default is **DropPacket**. This property defines how Cisco Prime Access Registrar handles requests if all servers listed in the **RemoteServers** properties are unavailable (that is, all remote RADIUS servers are not available). You must set it to one of the following: **AcceptAll**, **DropPacket**, or **RejectAll**.
OutageScript	Optional; if you set this property to the name of a script, Cisco Prime Access Registrar runs it when an outage occurs. This property allows you to create a script that notifies you when the RADIUS server detects a failure.

Note **OutagePolicy** also applies to Accounting-Requests. If an Accounting-Request is directed to an unavailable Service, then the values in Table 5-12 apply.

Table 5-12 OutagePolicy Request Packets

Property	Description	Accounting-Request Description
AcceptAll	Continues processing the packet as if the Service was successful.	The Accounting-Request will continue through the server and a response will be sent.
DropPacket	Immediately drops the packet, no further processing, and does not send any response to the client for this packet.	The packet will be discarded and it will not be processed any further.
RejectAll	Rejects the packet, but continues processing it and sends the client a reject response.	The request will be dropped and no more processing will be done.

Types of Services

This section lists the types of services available in Prime Access Registrar with their required and optional properties. The service you specify determines what additional information you must provide.

This section contains the following topics:

- EAP Services
- Extended-EAP
- File
- Group
- Java
- LDAP

- Local
- ODBC
- ODBC-Accounting
- Prepaid Services
- RADIUS
- Radius Query
- Diameter-RADIUS
- RADIUS-Diameter
- RADIUS-Session
- Rex
- WiMAX
- Diameter
- M3UA

EAP Services

Prime Access Registrar supports Extensible Authentication Protocol (EAP) and Protected EAP (PEAP) to provide a common protocol for differing authentication mechanisms. EAP enables the dynamic selection of the authentication mechanism at authentication time based on information transmitted in the Access-Request. Prime Access Registrar provides the following EAP services:

- EAP-AKA
- EAP-AKA-PRIME (EAP-AKA')
- EAP-FAST
- EAP-GTC
- EAP-LEAP
- EAP-MD5
- EAP-MSChapV2
- EAP-Negotiate
- EAP-SIM
- EAP-Transport Level Security (TLS)
- EAP-Tunneled TLS (TTLS)
- PEAP Version 0 (Microsoft PEAP)
- PEAP Version 1 (Cisco PEAP)

See the "Extensible Authentication Protocols" chapter of the *Cisco Prime Access Registrar 8.0 User Guide* for detailed information about properties used in EAP-type services.

Extended-EAP

Extended EAP is used as an authorization service to retrieve authorization information from a remote web server using the REST interface. Prime Access Registrar processes all EAP requests, and extends the process through extended EAP service. Extended EAP is supported for the following EAP services:

- EAP-AKA
- EAP-AKA-Prime
- EAP-SIM

You can configure an extended-EAP service under /Radius/Services. When you define an extended-EAP service under /Radius/Services, you must set its type to **extended-eap.** Refer to the sample configuration given below:

```
[ //localhost/Radius/Services/extended-EAP ]
    Name = extended-EAP
    Description =
    Type = extended-eap
    IncomingScript~ =
    OutgoingScript~ =
    OutagePolicy~ = AcceptAll
    OutageScript~ = NASLIST
    NasIDList = NasList
    MultipleServersPolicy = Failover
    RemoteServers/
```

To configure a REST remote server for the extended-EAP service, see REST.

File

Specify the **file** service when you want Cisco Prime Access Registrar's RADIUS Server to perform local accounting using a specific file. Every **file** Service in your configuration will cause a file with the configured name to be created when the server is started, even if the service is not being invoked by any request packets. Table 5-13 lists the properties used for a **file** service.

Table 5-13 File Service Properties

Property	Description
Type	Required; must be set to **group** for a group service.
IncomingScript	Name of script to run when the service starts.
OutgoingScript	Name of script to run when the service ends.
OutagePolicy	Required; the default is **DropPacket**. This property defines how Cisco Prime Access Registrar handles requests if all servers listed in the **RemoteServers** properties are unavailable (that is, all remote RADIUS servers are not available). You must set it to one of the following: **AcceptAll**, **DropPacket**, or **RejectAll**.
OutageScript	Optional; if you set this property to the name of a script, Cisco Prime Access Registrar runs it when an outage occurs. This property allows you to create a script that notifies you when the RADIUS server detects a failure.

Table 5-13 File Service Properties (continued)

Property	Description
FilenamePrefix	Required; a string that specifies where Cisco Prime Access Registrar writes the account records. It must be either a relative or absolute path. When you specify a relative path, it must be relative to the **$INSTALL/logs** directory. When you specify an absolute path, the server must be able to reach it. The default is **Accounting**.
MaxFileSize	Optional; stored as a string, but is composed of two parts, a number and a units indicator (<n> <units>) in which the unit is one of: K, Kilobyte, Kilobytes, M, Megabyte, Megabytes, G, Gigabyte, Gigabytes. The default is ten megabytes.
MaxFileAge	Optional; stored as a string, but is composed of two parts, a number and a units indicator (<*n*> <*units*>) in which the unit is one of: H, Hour, Hours, D, Day, Days, W, Week, Weeks. The default is one day.
RolloverSchedule	Indicates the exact time including the day of the month or day of the week, hour and minute to roll over the accounting log file.
UseLocalTimeZone	When set to TRUE, indicates the accounting records' TimeStamp is in local time. When set to FALSE, the default, accounting records' TimeStamp is in GMT.
FileType	Indicates the file type to export the accounting records to. Could be one of the following: • **log**— Prime Access Registrar server writes accounting messages to the accounting.log file in the /opt/CSCOar/logs directory. • **csv**—Prime Access Registrar server writes accounting messages to the accounting.csv file in the /opt/CSCOar/logs directory.
EnableRollOverIntelligence	Set to TRUE to enable rollover of the accounting records based on the accounting service properties. For more information on rollover of accounting logs, see the "RADIUS Accounting" chapter of the *Cisco Prime Access Registrar 8.0 User Guide.*
Delimiter	The delimiter to use in the accounting file, if the file type is **csv**.

Cisco Prime Access Registrar opens the file when it starts the RADIUS server and closes the file when you stop the server. Prime Access Registrar flushes the accounting record to disk before it acknowledges the request.

Based on the maximum file size and age you have specified, Prime Access Registrar closes the accounting file, moves it to a new name, and reopens the file as a new file. The name Prime Access Registrar gives this accounting file depends on its creation and modification dates.

- If the file was created and modified on the same date, the filename is **FileNamePrefix-**<*yyyymmdd*>**-**<*n*>**.log**. The date is displayed as year, month, day, number.
- If the file was created on one day and modified on another, the filename is **FileNamePrefix-**<*yyyymmdd*>**-**<*yyyymmdd*>**-**<*n*>**.log**. The dates are creation, modification, and number.

Group

A group service contains a list of references to other services and specifies whether the responses from each of the services should be handled as a logical AND or a logical OR function. You specify AND or OR in the Result-Rule attribute of Group Services. The default value is AND.

Table 5-14 lists the properties used to configure a **group** service.

Table 5-14 Group Service Properties

Property	Description
Type	Required; must set it to group.
IncomingScript	Optional; name of script to run when the service starts.
OutgoingScript	Optional; name of script to run when the service ends.
ResultRule	When set to AND (the default), the response from the GroupService is positive if each of the services referenced return a positive result. The response is negative if any of the services reference return a negative result. When set to OR, the response from the GroupService is positive if any of the services referenced return a positive result. The response is negative if all the referenced services return a negative result. The settings parallel-AND or parallel-OR are similar to AND and OR settings, except that each referenced service processes requests simultaneously instead of asking each reference service sequentially to save processing time.
GroupServices	Use the GroupServices subdirectory to specify the subservices in an indexed list to provide specific ordering control of which services to apply first. Each subservice listed must be defined in the Services section of the RADIUS configuration and cannot be a of type *g*roup, eap-leap, or eap-md5.

If Result-Rule is set to AND, the response from the Group Service is positive if each of the services referenced return a positive result. The response is negative if any of the services reference return a negative result. If Result-Rule is set to OR, the response from the Group Service is positive if any of the services referenced return a positive result. The response is negative if all the referenced services return a negative result.

When the Result-Rule attribute is set to AND or OR, each referenced service is accessed sequentially, and the Group Service waits for a response from the first referenced service before moving on to the next service (if necessary). If a service takes a long time to respond, that causes a delay in sending the request to the next referenced server.

The ResultRule settings parallel-and and parallel-or are similar to the AND and OR settings except that they ask each referenced service to process the request simultaneously instead of asking each referenced server sequentially, thereby saving processing time.

A parallel-and setting might respond with its own reply as soon as it receives a negative response, but otherwise must wait for all responses before it can respond with a positive reply. Likewise, a parallel-or might respond as soon as it receives a positive response, but otherwise must wait for all responses before it can reply with a negative response.

If a service referenced from a Group Service is of type RADIUS and if Accounting-Requests are being processed by the Group Service, setting the AckAccounting property in the remote server will affect the behavior of the parallel-or Group Service. This is because if AckAccounting is set to FALSE, the RADIUS Remote Server will not wait for the response from the remote server but returns a response immediately. Since the Group Service is set to parallel-or, after it receives the response from the

RADIUS service, it is free to send a response itself. This will have the effect that a response is sent very quickly from the Group Service acknowledging the Accounting-Request and responses from the other referenced services are handled as the arrive.

Note that since AckAccounting was set to FALSE, there is no guarantee that the Remote Server successfully processed the request. Since it is a RADIUS Remote Server, the Prime Access Registrar server attempts for MaxTries to send the request to the server and to get back an acknowledgment, but if that fails, there will be no indication to the client about that event. The acknowledgment to the client has been sent long before.

Java

Specify the **java** service type when you want to create a custom service and use a script for authentication, authorization, or accounting. Table 5-15 lists the properties required to configure a java service.

A java service uses an extension point script to provide the service's functionality and handles both RADIUS and TACACS requests for authentication, authorization, and accounting.

Table 5-15 Java Service Properties

Property	Description
Type	Required; must set it to java.
IncomingScript	Name of script to run when the service starts.
OutgoingScript	Name of script to run when the service ends.
OutagePolicy	Required; the default is **DropPacket**. This property defines how Cisco Prime Access Registrar handles requests if all servers listed in the **RemoteServers** properties are unavailable (that is, all remote RADIUS servers are not available). You must set it to one of the following: **AcceptAll**, **DropPacket**, or **RejectAll**.
OutageScript	Optional; if you set this property to the name of a script, Cisco Prime Access Registrar runs it when an outage occurs. This property allows you to create a script that notifies you when the RADIUS server detects a failure.
ClassName	Set to the name of a class that implements the Extension interface.
InitializeArg	Optional; set to a string to be passed to the Initialize method if the class implements the optional ExtensionWithInitialization interface.

LDAP

Specify the **ldap** service type when you want to use a particular LDAP remote server for authentication and/or authorization. Table 5-16 lists the properties used to configure an LDAP service.

When using LDAP for authentication and a local database for authorization, ensure that the usernames in both locations are identical with regard to case sensitivity.

Table 5-16 LDAP Service Properties

Property	Description
Type	Required, must set it to **ldap**
IncomingScript	Name of script to run when the service starts.
OutgoingScript	Name of script to run when the service ends.
OutagePolicy	Required; the default is **DropPacket**. This property defines how Cisco Prime Access Registrar handles requests if all servers listed in the **RemoteServers** properties are unavailable (that is, all remote RADIUS servers are not available). You must set it to one of the following: **AcceptAll**, **DropPacket**, or **RejectAll**.
OutageScript	Optional; if you set this property to the name of a script, Cisco Prime Access Registrar runs it when an outage occurs. This property allows you to create a script that notifies you when the RADIUS server detects a failure.
MultipleServersPolicy	Required; must be set to either **Failover** or **RoundRobin**. When you set it to **Failover**, Cisco Prime Access Registrar directs requests to the first server in the list until it determines the server is offline. At which time, Cisco Prime Access Registrar redirects all requests to the next server in the list until it finds a server that is online. When you set it to **RoundRobin**, Cisco Prime Access Registrar directs each request to the next server in the RemoteServers list to share the resource load across all of the servers listed in the RemoteServers list.
RemoteServers	Required; an indexed list from 1 to <*n*>. Each entry in the list is the name of a RemoteServer.

Local

Specify **local** when you want the Cisco Prime Access Registrar server to perform the authentication and authorization using a specific UserList. For more information, see the "UserLists" section on page 5-3. Table 5-17 lists the properties used to configure a **local** service.

Table 5-17 Local Service Properties

Property	Description
Type	Required, must set it to **local**.
IncomingScript	Optional; name of script to run when the service starts.
OutgoingScript	Optional; name of script to run when the service ends.
OutagePolicy	Required; the default is **DropPacket**. This property defines how Cisco Prime Access Registrar handles requests if all servers listed in the **RemoteServers** properties are unavailable (that is, all remote RADIUS servers are not available). You must set it to one of the following: **AcceptAll**, **DropPacket**, or **RejectAll**.

Table 5-17 Local Service Properties (continued)

Property	Description
OutageScript	Optional; if you set this property to the name of a script, Cisco Prime Access Registrar runs it when an outage occurs. This property allows you to create a script that notifies you when the RADIUS server detects a failure.
UserLists	Required; this object contains all of the individual UserLists, which in turn, contain the specific users stored within Prime Access Registrar. Cisco Prime Access Registrar references each specific UserList by **name** from a Service whose type is set to **local**. When Cisco Prime Access Registrar receives a request, it directs it to a Service. When the Service has its type property set to **local**, the Service looks up the user's entry in the specific UserList and authenticates and/or authorizes the user against that entry.

ODBC

Specify **odbc** when you want to use an ODBC service for authentication, authorization and accounting through an ODBC data store. Use an ODBC service to authenticate and authorize an access requests by querying user information through ODBC and to insert accounting records into a data store through ODBC. Table 5-18 lists the properties used to configure an ODBC service.

Table 5-18 ODBC Service Properties

Property	Description
Type	Required; must set it to **odbc**.
IncomingScript	Optional; name of script to run when the service starts.
OutgoingScript	Optional; name of script to run when the service ends.
OutagePolicy	Required; the default is **DropPacket**. This property defines how Cisco Prime Access Registrar handles requests if all servers listed in the **RemoteServers** properties are unavailable (that is, all remote RADIUS servers are not available). You must set it to one of the following: **AcceptAll**, **DropPacket**, or **RejectAll**.
OutageScript	Optional; if you set this property to the name of a script, Cisco Prime Access Registrar runs it when an outage occurs. This property allows you to create a script that notifies you when the RADIUS server detects a failure.
MultipleServersPolicy	Required; must be set to either **Failover** or **RoundRobin**. When you set it to **Failover**, Cisco Prime Access Registrar directs requests to the first server in the list until it determines the server is offline. At which time, Cisco Prime Access Registrar redirects all requests to the next server in the list until it finds a server that is online. When you set it to **RoundRobin**, Cisco Prime Access Registrar directs each request to the next server in the RemoteServers list to share the resource load across all of the servers listed in the RemoteServers list.
RemoteServers	Required; an indexed list from 1 to <*n*>. Each entry in the list is the name of a RemoteServer.

ODBC-Accounting

If you use the Oracle Accounting feature, you must configure an ODBC-Accounting RemoteServer object. See the "Configuring an ODBC/OCI RemoteServer" section in the "Using Open Database Connectivity" chapter of the *Cisco Prime Access Registrar 8.0 User Guide*, for more information on ODBC-Accounting RemoteServer.

Prepaid Services

Cisco Prime Access Registrar (Prime Access Registrar) supports two types of prepaid billing, IS835C and Cisco Real-time Billing (CRB), a Cisco proprietary solution. See "Using Prepaid Billing" chapter of the *Cisco Prime Access Registrar 8.0 User Guide* for more information on Prepaid -IS835C and Prepaid-CRB.

RADIUS

Specify the **radius** service type when you want to use a particular RADIUS remote server for authentication and authorization. Table 5-19 lists the properties used to configure a RADIUS service.

Table 5-19 RADIUS Service Properties

Property	Description
Type	Required; must set it to **radius**.
IncomingScript	Optional; name of script to run when the service starts.
OutgoingScript	Optional; name of script to run when the service ends.
OutagePolicy	Required; the default is **DropPacket**. This property defines how Cisco Prime Access Registrar handles requests if all servers listed in the **RemoteServers** properties are unavailable (that is, all remote RADIUS servers are not available). You must set it to one of the following: **AcceptAll**, **DropPacket**, or **RejectAll**.
OutageScript	Optional; if you set this property to the name of a script, Cisco Prime Access Registrar runs it when an outage occurs. This property allows you to create a script that notifies you when the RADIUS server detects a failure.
MultipleServersPolicy	Required; must be set to either **Failover** or **RoundRobin**. When you set it to **Failover**, Cisco Prime Access Registrar directs requests to the first server in the list until it determines the server is offline. At which time, Cisco Prime Access Registrar redirects all requests to the next server in the list until it finds a server that is online. When you set it to **RoundRobin**, Cisco Prime Access Registrar directs each request to the next server in the RemoteServers list to share the resource load across all of the servers listed in the RemoteServers list.
RemoteServers	Required; an indexed list from 1 to <*n*>. Each entry in the list is the name of a RemoteServer.

Radius Query

Prime Access Registrar supports a new service type called radius-query that can be used to query cached data through RADIUS packets. This radius-query service contains a list of session managers to be queried from and a list of (cached) attributes to be returned in the Access-Accept packet in response to a RADIUS Query request. Prime Access Registrar also supports caching and querying of multivalued attributes.

The RADIUS Query service should be selected through an extension point script or through the Rule and Policy Engine by setting it to a new environment variable named Query-Service. The reason for this is that the RADIUS Query request comes in as an Access-Request and the server has no way of knowing whether it is a RADIUS Query request or normal authentication request. Setting the Query-Service environment variable tells the Prime Access Registrar server that the request is a RADIUS Query request so the Prime Access Registrar server can process the request with the radius-query service set in the Query-Service environment variable.

When a RADIUS Query service is selected to process an Access-Request, it queries the configured list of Session Managers for a matching record using the QueryKey value configured in the session-cache Resource Manager referenced under these Session Managers as key. If a matching record is found, an Access-Accept containing a list of cached attributes present (based on the configuration) in the matched record is sent back to the client. If the session cache contains a multivalued attribute, all values of that attribute are returned in the response as a multivalued attribute. If there is no matching record, an Access-Reject packet is sent to the client.

Prime Access Registrar introduces scripting points at the Session Manager level along with automated programmable interfaces (APIs) to access cached information present in the session record. You can use these scripting points and APIs to write extension point scrips to modify the cached information.

The following example shows the default configuration of a radius-query service:

```
[ //localhost/Radius/Services/radius-query ]
    Name = radius-query
    Description =
    Type = radius-query
    IncomingScript~ =
    OutgoingScript~ =
    SessionManagersToBeQueried/
    AttributesToBeReturned/
```

Table 5-20 lists the properties used to configure a RADIUS Query service.

Table 5-20 RADIUS Query Service Properties

Property	Description
Type	Required; must set it to **radius query**.
IncomingScript	Optional; name of script to run before this service starts processing on the request.
OutgoingScript	Optional; name of script to run after this service completes processing on the request.

Table 5-20 RADIUS Query Service Properties (continued)

Property	Description
SessionManagersToBeQueried	Lists Session Managers to be queried for the target record. If this list is empty, all Session Managers having session-cache Resource Managers will be queried for the target record. Otherwise, only those SessionManagers configured under SessionManagerToBeQueried are queried. If the targeted record is found in a Session Manager, the query stops and the response is returned to the client.
AttributesToBeReturned	Lists attributes to be returned if present in a matched record. If this list is empty, all attributes cached in a matched session are returned. If a configured attribute is not present in the matched record, that attribute is ignored. **Note** The User-Password attribute will not be returned in query responses and cannot be configured under AttributesToBeReturned.

When an Access-Request packet is received by the Prime Access Registrar server, the session-cache Resource Manager caches the configured attributes in the session with the configured QueryKey as the key to the cached data. In the TAL solution, the QueryKey will usually be Framed-IP-Address. If an Accounting-Requestor Accounting-Start packet is received for the same session, the cached data is updated if necessary. If there is a multivalued attribute in the Access-Request packet or Accounting-Request packet, the Prime Access Registrar server caches all the values of that attributes.

In TAL, when the SSG receives an IP packet originating from a user unknown to the SSG, it sends an Access-Request packet to the Prime Access Registrar server in which the User-Name and Framed-IP-Address attributes both contain the user's source IP address, and the Service-Type is set to Outbound, among other attributes. These attributes and their values distinguish RADIUS Query requests from normal authentication requests in TAL.

Note In solutions other than TAL, the criterion that distinguishes RADIUS Query requests from normal authentication requests might be different.

A new environment variable, Query-Service, can be set to the name of a radius-query service, in an extension point script, or through the Rule and Policy engine so the Prime Access Registrar server knows the current request is a RADIUS Query request and processes it with the radius-query service value set in the Query-Service environment variable.

API Calls

Prime Access Registrar provides several new API calls you can use to get, put, and delete the cached attributes present in the session record.

The entry point function changes slightly to take a fifth argument which is a pointer to a structure containing the new API calls:

```
typedef int (REXAPI * RexEntryPointFunction)
(
```

```
    int iScriptingPoint,
    rex_AttributeDictionary_t* pRequest,
    rex_AttributeDictionary_t* pResponse,
    rex_EnvironmentDictionary_t* pRadius,
    rex_SessionRecord_t* pSession
    );
```

However, you can continue to write extension point scripts with four arguments as well, for example without the pSession argument.

The following are API calls and their functionality. All these API calls fail gracefully when they are invoked from any scripting point other than the Session Manager scripting points.

const char* get

```
const char* get(
    rex_SessionRecord_t* pSession,
    const char* pszAttribute,
    int <iIndex>,
    abool_t* <pbMore>
    )
```

This API returns the value of the <iIndex>'d instance of the attribute cached in the session, represented as a string. When the session does not contain the attribute, an empty string is returned. When <pbMore> is non-zero, this method sets <pbMore> to TRUE when more instances of the same attribute exist after the one returned and to FALSE otherwise. This can be used to determine whether another call to get() method should be made to retrieve other instances of the same attribute.

abool_t put

```
abool_t put(
    rex_SessionRecord_t* pSession,
    const char* pszAttribute,
    const char* <pszValue>,
    int <iIndex>
    )
```

When <iIndex> equals the special value REX_REPLACE, this method replaces any existing instances of <pszAttribute> with a single value in the session. When <iIndex> equals the special value REX_APPEND, it appends a new instance of <pszAttribute> to the end of the list of existing instances of <pszAttribute>. When <iIndex> equals the special value REX_AUGMENT, this method only puts <pszAttribute> when it does not already exist. Otherwise, a new instance of <pszAttribute> is inserted/replaced at the position indicated. This method returns TRUE if it is able to cache the attribute successfully and FALSE otherwise.

abool_t remove

```
abool_t remove(
    rex_SessionRecord_t* pSession,
    const char* pszAttribute,
    int <iIndex>
    )
```

This method removes the <pszAttribute> from the session. When <iIndex> equals the special value REX_REMOVE_ALL, this method removes any existing instances of <pszAttribute>. Otherwise, it removes the instance of <pszAttribute> at the position indicated. It returns FALSE when <pszAttribute> is not present at any index in the session record and returns TRUE otherwise.

rex_SessionInfo_t*

rex_SessionInfo_t* getSessionInfo(rex_SessionRecord_t* pSession)

This method returns the pointer to a structure that contains the other session-related information, like Session Id, Session Start time, Session Last Accessed Time, present in the session record. The structure that holds this information will appear as follows:

```
typedef  struct rex_SessionInfo_s
{
     auint32_t iSessionId;
     auint32_t tSessionStartTime;
     auint32_t tSessionLastAccessedTime;
} rex_SessionInfo_t;
```

Tcl API calls

To use the extension point scripts written in Tcl, define the procedure at the session manager level as shown below:

```
proc test { request response environ session } {
}
```

There is a fourth argument *session* that needs to be passed to the Tcl procedure and the API calls that are intended to operate on the session record need to use this *session* dictionary.

API calls in Tcl have the same meaning with same number arguments and return values as described in Rex. The only difference is that the API getSessionInfo will not return a structure as in Rex but it will return the info as a string, as in the following example:

```
Session-ID=1, Session-Start-Time=1102099334, Session-Last-Accessed-Time=1102099334
```

Java API calls

There are two new interfaces ExtensionForSession and ExtensionForSessionWithInitialization and the customers whishing to use the extension point scripts written in Java at the session manager level needs to implement one of these interfaces.

The runExtension method of these interfaces will look as below:

```
public int runExtension
    ( int iExtensionPoint,
    AttributeDictionary request,
    AttributeDictionary response,
    EnvironmentDictionary environment,
    SessionRecord session
    );
```

API calls that are intended to operate on session record needs to use this ‘session’ dictionary.

API calls in Java have the same meaning with same number arguments and return values as described in Rex. The only difference is that the API getSessionInfo will not return a structure as in Rex but it will return the info as a string. For example:

Session-ID=1, Session-Start-Time=1102099334, Session-Last-Accessed-Time=1102099334

Existing scripts written in any of these three languages will not be affected with the introduction of the new ‘session dictionary’ argument. And the customers can use a script with any number of arguments (i.e with or without the last ‘session dictionary’ argument) at any extension point script. If there is no session to operate on, for example when the customer is trying to use session dictionary argument at an extension point other than session manager’s, the Prime Access Registrar gracefully returns an error logging the appropriate message.

The simple *replace or add if it does not exist* model can still be used for simple modifications as before without the need to write a script. If the cached attributes are updated in the IncomingScript and if customers do not want them to be touched or updated again when the processing reaches session-cache resource manager, they can set the OverwriteAttributes property of the session-cache resource manager to FALSE so that the session-cache resource manager will not operate on this packet.

Diameter-RADIUS

This service helps to translate incoming Diameter request to a RADIUS equivalent and then the RADIUS response to Diameter equivalent. Prime Access Registrar provides scripting points, which operate on the original packet and on the newly translated packet based on request and response mapping.

RADIUS-Diameter

This service helps to translate incoming RADIUS request to a Diameter equivalent and then the Diameter response to RADIUS equivalent. Prime Access Registrar provides scripting points, which operate on the original packet and on the newly translated packet based on request and response mapping.

Table 5-21 Diameter-RADIUS and RADIUS-Diameter Service Properties

The following properties are applicable for **diameter-radius** or **radius-diameter** service type.	
ProxyServiceName	The Diameter proxy service name.
DiameterApplicationID	The Diameter service application ID. This is applicable only for **radius-diameter** service type.
SendRAR-ASRToClient	Set to TRUE if the COA/POD packets received by Prime Access Registrar are to be translated and sent as Re-Auth-Request (RAR) / Abort Session Request (ASR) to a Diameter client. This is applicable only for radius-diameter service type.
ClientHostName	Hostname of the Diameter client to which the translated RAR/ASR must be sent. If the session manager is configured, the client host name can be acquired from it. This is applicable only for radius-diameter service type.
UseFor3GPPReverseAuthorizationService	Set to TRUE to enable 3GPP authorization service in the translation framework. This is applicable only for radius-diameter service type.
PreRequestTranslationScript	The scripting point to be called on the original request packet.
PostRequestTranslationScript	The scripting point to be called on the translated request packet.
PreResponseTranslationScript	The scripting point to be called on the response packet.
PostResponseTranslationScript	The scripting point to be called on the translated response packet.

The following example shows a sample configuration of translation service in Diameter:

```
[ //localhost/Radius/Services/rad-dia-trans ]
    Name = rad-dia-trans
    Description =
    Type = radius-diameter
    SendRAR-ASRToClient = true
    ClientHostName =
    DiameterApplicationId = 5
    ProxyServiceName = dia
    EnableRequestCommandMappings = true
    PreRequestTranslationScript~ = sm
    PostRequestTranslationScript~ =
    PreResponseTranslationScript~ = env
```

```
PostResponseTranslationScript~ =
RequestMapping/
    CommandMappings/
        Radius-CoA-Request = Re-Auth
        Radius-PoD-Request = Abort-Session
    AVPMappings/
        Calling-Station-Id = Session-Id
    AVPsToBeAdded/
        Re-Auth-Request-Type = AUTHORIZE_AUTHENTICATE
    EnvironmentMappings/
ResponseMapping/
    ResultCodeMappings/
        Diameter-Success = Radius-PoD-ACK
        Diameter-Unable-To-Deliver = Radius-PoD-Nak
    AVPMappings/
    AVPsToBeAdded/
    EnvironmentMappings/
```

RADIUS-Session

A new Service step has been added in the processing of Access-Request and Accounting packets. This is an additional step after the AA processing for Access packet or Accounting processing for Accounting packet, but before the local session management processing. The Session-Service should have a service type of radius-session.

An environment variable Session-Service is introduced to determine the Session-Service dynamically. You can use a script or the rule engine to set the Session-Service environment variable. See Cross Server Session and Resource Management, page 1-2 for more information on RADIUS-Session.

Rex

Specify the **rex** service type when you want to create a custom service and use a script for authentication, authorization, or accounting. Table 5-22 lists the properties required to configure a **rex** service.

Table 5-22 rex Service Properties

Property	Description
Type	Required; must be set to **rex**.
IncomingScript	Optional; name of script to run when the service starts.
OutgoingScript	Optional; name of script to run when the service ends.
OutagePolicy	Required; the default is **DropPacket**. This property defines how Cisco Prime Access Registrar handles requests if all servers listed in the **RemoteServers** properties are unavailable (that is, all remote RADIUS servers are not available). You must set it to one of the following: **AcceptAll**, **DropPacket**, or **RejectAll**.
OutageScript	Optional; if you set this property to the name of a script, Cisco Prime Access Registrar runs it when an outage occurs. This property allows you to create a script that notifies you when the RADIUS server detects a failure.
Filename	Required; must be either a relative or an absolute path to the shared library containing the Service. When the pathname is relative, it must be relative to **$INSTALL/Scripts/Radius/rex**.

Table 5-22 rex Service Properties (continued)

Property	Description
EntryPoint	Required; must be set to the function's global symbol.
InitEntryPoint	Required; must be the name of the global symbol Cisco Prime Access Registrar should call when it initializes the shared library and just before it unloads the shared library. **Note** A rex service must have an InitEntryPoint even if the service only returns REX_OK.
InitEntryPointArgs	Optional; when set, it provides the arguments to be passed to the **InitEntryPoint** in the environmental variable **Arguments**.

For more information about scripting, see Chapter 7, "Using Extension Points." For more information about using the REX Attribute dictionary, see "Cisco Prime Access Registrar Tcl, REX, and Java Dictionaries" chapter of the *Cisco Prime Access Registrar 8.0 Reference Guide.*

WiMAX

Prime Access Registrar uses the Extensible Authentication Protocol (EAP) to enable the WiMAX feature. It also caches the IP attributes and Mobility Keys that are generated during network access authentication. To enable caching of the WiMAX attributes, you must configure the respective resource managers. For more information, see the "Using WiMAX in Cisco Prime Access Registrar" chapter of the *Cisco Prime Access Registrar 8.0 Reference Guide*.

Diameter

Diameter works with the rule policy engine to perform the routing for multiple peers. The following are the multiple peer policies supported with the proxy service to route the traffic:

- RoundRobin
- FailOver
- IMSI Range Based

The following configuration is used to add Diameter proxy without Sticky session configuration:

```
[ //localhost/Radius/Services/dia-proxy ]
    Name = dia-proxy
    Description =
    Type = diameter
    IncomingScript~ =
    OutgoingScript~ =
    EnableSticky = FALSE
    MultiplePeersPolicy = RoundRobin
    PeerTimeOutPolicy = SendError
    DiaRemoteServers/
        Entries 1 to 2 from 2 total entries
        Current filter: <all>

        proxy_rm/
            Name = proxy_rm
            Metric = 1
            Weight = 0
            IsActive = TRUE
        proxy_rm1/
```

```
            Name = proxy_rm1
            Metric = 2
            Weight = 0
            IsActive = TRUE
```

The following configuration is used to add Diameter proxy with Sticky session configuration:

```
[ //localhost/Radius/Services/dia-proxy ]
Name = dia-proxy
    Description =
    Type = diameter
    IncomingScript~ =
    OutgoingScript~ =
    EnableSticky = TRUE
    StickySessionKey = User-Name#1
    StickyCreationCmdList = 265
    StickyDeletionCmdList = 275
    MultiplePeersPolicy = RoundRobin
    PeerTimeOutPolicy = SendError
    DiaRemoteServers/
        Entries 1 to 2 from 2 total entries
        Current filter: <all>

        proxy_rm/
            Name = proxy_rm
            Metric = 1
            Weight = 0
            IsActive = TRUE
        proxy_rm1/
            Name = proxy_rm1
            Metric = 2
            Weight = 0
            IsActive = TRUE
```

The following configuration is used to add Diameter proxy with IMSI range based load balancing configuration:

```
[ //localhost/Radius/Services/dia-proxy ]
    Name = dia-proxy
    Description =
    Type = diameter
    IncomingScript~ =
    OutgoingScript~ =
    EnableSticky = TRUE
    StickySessionKey = User-Name#1
    StickyCreationCmdList = 265
    StickyDeletionCmdList = 275
    MultiplePeersPolicy = IMSIRangeBased
    PeerTimeOutPolicy = SendError
    IMSIRanges/
        Entries 1 to 1 from 1 total entries
        Current filter: <all>

        112156000000001-112156001000000/
            Name = 112156000000001-112156001000000
            Description =
            Start = 112156000000001
            End = 112156001000000
            MultiplePeersPolicy = Failover
            DiaRemoteServers/
                Entries 1 to 2 from 2 total entries
                Current filter: <all>

                proxy_rm/
```

```
                    Name = proxy_rm
                    Metric = 1
                    Weight = 0
                    IsActive = TRUE
                proxy_rm1/
                    Name = proxy_rm1
                    Metric = 2
                    Weight = 0
                    IsActive = TRUE
```

The following configuration is used to add Diameter local:

```
[ //localhost/Radius/Services/dia-local ]
   Name = dia-local
   Description =
   Type = diameter
   Realm = abc.com
   Role = Local
   IncomingScript~ =
   OutgoingScript~ =
   AuthenticationService = local-users
   AccountingService = local-file
    DiaRemoteServers/
       Entries 1 to 1 from 1 total entries
       Current filter: <all>

       murdcoh-ar1/
           Name = murdcoh-ar1
           HostName = murdoch-ar1
           IsVendorSpecific = FALSE
           ApplicationID = 1
           Metric = 2
```

The following configuration is used to add Diameter relay:

```
[ //localhost/Radius/Services/dia-relay ]
   Name = dia-relay
   Description =
   Type = diameter
   Realm = abc.com
   Role = relay
    DiaRemoteServers/
       Entries 1 to 1 from 1 total entries
       Current filter: <all>

       hss1/
           Name = hss1
           HostName = 10.77.240.69
           IsVendorSpecific = FALSE
           ApplicationID = 1
           Metric = 2
```

The following configuration is used to add Diameter redirect:

```
[ //localhost/Radius/Services/dia-redirect ]
   Name = dia-relay
   Description =
   Type = diameter
   Realm = abc.com
   Role = redirect
    DiaRemoteServers/
       Entries 1 to 1 from 1 total entries
       Current filter: <all>

       redirectserver/
```

```
Name = redirectserver
HostName = 10.77.240.69
IsVendorSpecific = FALSE
ApplicationID = 1
Metric = 2
```

Table 5-23 describes the Diameter Service properties.

Table 5-23 Diameter Service Properties

Property	Description
Name	Required; name of the Diameter server.
Realm	Required; realm of the route. Must be unique for a route table.
Incoming Script	Optional; enabled when role is set to Proxy or Local. When set, must be the name of a known incoming script. Prime Access Registrar runs the IncomingScript before proxying the diameter packet to the remote diameter server.
Outgoing Script	Optional; enabled when role is set to Proxy or Local. When set, must be the name of a known outgoing script. Prime Access Registrar runs the after it receives the response from the remote Diameter server.
Description	Optional; description of the Diameter server.
Role	Required; specifies the role that the diameter entity will play in resolving messages matching the realm. The role can be any one of the following: Relay - Application acting as a Relay Agent. Redirect - Application acting as a Redirect Agent. Proxy - Application acting as a Proxy Agent. When the role is set to Proxy, the IncomingScript and OutgoingScript points are enabled. Local - Application processes the requests locally. When the role is set to Local, the AuthenticationService and AccountingService are enabled. By default, the Proxy option is selected. However, you can select another option from the drop-down list.
AuthenticationService	Required; used when service is configured to process the diameter requests locally. Set to valid service of type (local/ldap/odbc) to authenticate the user. This field is displayed when you select the role type as ‘Local’ in the Role field.
AccountingService	Required; used when service is configured to process the accounting requests locally. Set to valid accounting service of type (file/odbc-accounting) to write the accounting records. This field is displayed when you select the role type as ‘Local’ in the Role field.
Type	Required; specifies the service type.The service type ‘Diameter’ is automatically displayed in this field.
Peer Definitions tab This tab is displayed when you select the ‘Local’, ‘Relay’, or ‘Redirect’option in the Role field.	
Name	Required; name of the peer.
Host Name	Required; the hostname or IP address of the peer. The hostname must exist in the client list for the route to be active.
Metric	Required; metric value for the peer entry. The higher the value the lower the preference. The highest value of preference is 0.

Table 5-23 Diameter Service Properties (continued)

Property	Description
VendorSpecific	Required; the default is FALSE. If set to FALSE, the application is ordinary application and user is prompted to enter the ApplicationID. If set to TRUE, the application is a VendorSpecific Application. User is prompted to enter VendorSpecificApplicationID and VendorID.
VendorID	Required; specifies the VendorID for the application. Example: DIAMETER 3GPP Cx APPLICATION VendorSpecificApplicationID 16777216 VendorID 10415
VendorSpecificApplicationID	Required; specifies the integer value for the vendor specific application.
ApplicationID	Required; application used in the route. The application Id should be available in /Advanced/Diameter/Applications.
Applications tab This tab is displayed when you select the 'Proxy' option in the Role field.	
Name	Required; name of the application.
Description	The description of the application.
ApplicationID	Required; specifies the unique integer value for the application. It represents the applicationid of the Application used for load balancing the diameter messages.
EnableSticky	Required; default is FALSE. If set to True, the sticky entries for load balancing is enabled and the user is prompted to enter the values for StickySessionKey, StickyCreationCmdList, and StickyDeletionCmdList.
DeMultiplexCCTerminateRequest	Optional; default is FALSE. If set to True, Prime Access Registrar generates and sends multiple Credit Control Update (CCR-U) requests corresponding to an incoming diameter Credit Control Termination (CCR-T) request, while proxying Gy messages between the Gateway GPRS Support Node (GGSN) and Online charging system (OCS). The CCR-U requests are generated based on the number of RGs present in CCR-T message.
MultiplePeersPolicy	Required; must be set to RoundRobin, FailOver, GroupFailOver, or IMSIRangeBased. Policy used by the Prime Access Registrar server to load balance the peers.

Table 5-23 Diameter Service Properties (continued)

Property	Description
StickySessionKey	Required; used as the sticky key for mapping the sticky sessions. Set the value to a valid AVP in order to use the sticky key for maintaining diameter sessions. This ensures that Prime Access Registrar maps the request to the same server for all the subsequent messages using the sticky key. For example, set StickyAVP "Session-Id". When the Prime Access Registrar server receives the CCR-I request, Prime Access Registrar extracts the Session-Id from the request packet, maps the Session to the peer configured in the list, and forwards the request to the chosen peer. Prime Access Registrar chooses the same peer for all the subsequent messages (CCR-Update/CCR-Terminate) with same Session-Id.

StickyCreationCmdList	Required; specifies the command list to create the sticky entries.Specify the list of '||' separated command code, AVP name, and its value to create the sticky sessions.
	
	The following is the StickyCreationCmdList format:
	
	`<commandcode1>::<AVPName1=Value1> || <commandcode2<::<AVPName2=Value2>||<commandcode3>`
	
	For example, if the sticky session entries need to created based on command code '265'or based on command code '271' with Accounting-Record-Type value as 2, use the format below:
	
	`Set StickyCreationCmdList "265||271:: Accounting-Record-Type=2"`

StickyDeletionCmdList	Required; specifies the command list to delete the sticky entries.Specify the list of '||' separated command code, AVP name, and its value to delete the sticky sessions.
	
	The following is the StickyDeletionCmdList format:
	
	`<commandcode1>::<AVPName1=Value1> || <commandcode2<::<AVPName2=Value2>||<commandcode3>`
	
	For example, if the sticky session entries need to deleted based on command code '271' with Accounting-Record-Type value as 4, use the format below:
	
	`Set StickyDeletionCmdList "271:: Accounting-Record-Type=4"`

Peer Definitions Proxy tab	
Name	Required; name of the peer.
Host Name	Required; hostname or IP address of the peer. The HostName must exist in the client list for the route to be active.
Metric	Required; metric value for this peer entry. The higher the value the lower the preference. The highest value of preference is 0.

Table 5-23 Diameter Service Properties (continued)

Property	Description
Weight	Required; default value is 0. Specifies the weight percentage for which the service needs to load balance the peer. **Note** When you set the weight to a value other than 0, the weight should be in multiples of 10 and the sum of the weights configured in the peer list should be equal to 100.
IMSIRanges	Required; used for load balancing. The value is set to comma separated values of IMSI Ranges. For example, set IMSIRanges "112156000000001-112156001000000,112156010000001-112156011000000" **Note** Prime Access Registrar uses the AVP configured in StickyAVP property to check whether the IMSI is in valid range.

M3UA

Prime Access Registrar supports the M3UA service, which is used to fetch MSISDN from IMSI through RADIUS Packets. The M3UA service sends a SendRoutingInfoForLCS(SRIForLCS) request that contains the IMSI information to the remote HLR. The HLR sends the MSISDN in response. To fetch the MSISDN information from IMSI, you need to configure the SIGTRAN-M3UA remote server where Prime Access Registrar is installed. See "SIGTRAN-M3UA" chapter of the *Cisco Prime Access Registrar 8.0 User Guide* for more information.

The M3UA service checks for IMSI environment variable to fetch the MSISDN information. If there is no IMSI environment variable set, then the **User-Name** in the Radius **Access-Request** is used as IMSI to fetch the MSISDN information. The fetched MSISDN is copied to the **AuthorizationInfo** environment variable where you can write a script to copy the environment variable to any attribute of your choice. For the list of environment variables, see the "Environment Dictionary" chapter of the *Cisco Prime Access Registrar 8.0 Reference Guide.*

Note M3UA service supports fetching the MSISDN only through SIGTRAN-M3UA interface.

The following shows an example configuration of M3UA service:

```
[ //localhost/Radius/Services/FetchMSISDN ]
    Name = FetchMSISDN
    Description =
    Type = m3ua
    IncomingScript~ =
    OutgoingScript~ =
    OutageScript~ =
    OutagePolicy~ = RejectAll
    RemoteServers/
```

Table 5-24 M3UA Properties

Property	Description
Type	Required; must set to M3UA service.
IncomingScript	Optional; when set, must be the name of a known incoming script. Cisco Prime Access Registrar runs the IncomingScript after it receives the response.
OutgoingScript	Optional; when set, must be the name of a known outgoing script. Cisco Prime Access Registrar runs the just before it sends the proxy request to the remote server.
OutagePolicy	Required; the default is **DropPacket**. This property defines how Prime Access Registrar handles requests if all servers listed in the RemoteServers properties are unavailable (that is, all remote RADIUS servers are not available). You must set it to one of the following: **AcceptAll**, **DropPacket**, or **RejectAll**.
OutageScript	Optional; set this property to the name of a script. Prime Access Registrar runs the script when an outage occurs. It allows you to create a script that notifies you when the RADIUS server detects a failure.
RemoteServers	Required; an indexed list from 1 to <n>. Each entry in the list is the name of a RemoteServer of type SIGTRAN-M3UA.

Session Managers

You can use Session Managers to track user sessions. The Session Managers monitor the flow of requests from each NAS and detect the session state. When requests come through to the Session Manager, it creates sessions, allocates resources from appropriate Resource Managers, and frees and deletes sessions when users log out.

The Session Manager enables you to allocate dynamic resources to users for the lifetime of their session. You can define one or more Session Managers and have each one manage the sessions for a particular group or company.

Note Session record size is limited by the operating system (OS) paging size (4 KB in Linux). If a request triggers creation of a session that exceeds the OS paging size, the request will be dropped and the session will not be created.

Note In this release of Prime Access Registrar, the memory capacity is enhanced to store more than 4 million active sessions by storing the active session records in database server instead of storing it in the main memory. The capacity is dependent on the number of attributes that are being captured for each session.

Note If the disk partition where Prime Access Registrar stores session backing store data (usually the disk partition where Prime Access Registrar is installed, such as **/opt/CSCOar**) is full, the subsequent packets that try to create sessions will be dropped and no sessions will be created due to lack of disk space.

Session Managers use Resource Managers, which in turn, manage a pool of resources of a particular type. Table 5-25 lists the Session Manager properties.

Prime Access Registrar adds IncomingScript, OutgoingScript, and SessionKey properties. The IncomingScript is run as soon as the session is acquired. The OutgoingScript is run just before the session is written to backing store. The SessionKey property sets the session key value for the Session Manager.

Table 5-25 Session Manager Properties

Property	Description
Name	Required; must be unique in the Session Managers list.
Description	Optional description of the Session Manager.
Type	Set to local or remote. Local is the traditional session manager that maintains sessions in memory and has good performance. The remote session manager operates on a remote ODBC database, and its performance is highly dependent on the performance of the ODBC database.
IncomingScript	Optional; name of script to run when the service starts. This script is run as soon as the session is acquired in Prime Access Registrar.
OutgoingScript	Optional; script to be run just before the session is written to backing store.
SessionTimeOut	The SessionTimeOut property is optional; no value for this property means the session timeout feature is disabled. Used in conjunction with **/Radius/Advanced/SessionPurgeInterval** for the session timeout feature. Enables the session timeout feature for a Session Manager. If the SessionTimeOut property is set to a value under a session manager, all sessions that belong to that session manager will be checked for timeouts at each SessionPurgeInterval. If any sessions have timed out, they will be released, and all resources associated with those sessions are also released. The SessionTimeOut property determines the timeout for a session. If the time difference between the current time and the last update time is greater than this property's value, the session is considered to be stale. The last update time of the session is the time at which the session was created or updated. The SessionTimeOut value is comprised of a number and a units indicator, as in *n units*, where a unit is one of minutes, hours, days, or weeks. The default unit is 'days'.
AllowAccountingStartToCreateSession	Set to TRUE by default; start the session when the Prime Access Registrar server receives an Access Accept or an Accounting-Start. When set to FALSE, start the session when the Prime Access Registrar server receives an Access Accept.
Resource Managers	Ordered list of Resource Managers.

Table 5-25 Session Manager Properties (continued)

Property	Description
PhantomSessionTimeOut	Optional; no value for this property means the phantom session timeout feature is disabled. The PhantomSessionTimeOut property is used in conjunction with **/Radius/Advanced/SessionPurgeInterval** to enable the phantom session timeout feature for Session Manager. If the PhantomSessionTimeOut property is set to a value under a session manager, all sessions that belong to that session manager will be checked for receipt of an Accounting-Start packet. Sessions that do not receive an Accounting-Start packet from creation until its timeout will be released. The PhantomSessionTimeOut value comprises a number and a units indicator, as in *n* units, where a unit is one of minutes, hours, days, or weeks. The default unit is 'days'

SessionKey	SessionKey property is used to set the sessionkey value for the Session Manager.
	
	The SessionManager checks whether the environmental variable **Session-Key** is set or not. If the environmental variable is set, the server uses it as the sessionkey. If environmental variable **Session-Key** is not set then SessionManager gets the value configured in the SessionKey property under SessionManager.
	
	SessionKey can be a combination of attributes separated by a colon. The values for those attributes are obtained from the RequestDictionary. If any one of the attribute that is configured for the sessionkey is not present in the RequestDictionary, Prime Access Registrar will drop the request.
	
	However, if **Session-Key** is not set, SessionManager uses NAS-Identifier and NAS-Port to create the sessionkey. An example configuration,
	
	`--> set SessionKey "User-Name:NAS-Port"`
	The following shows the sample configuration of sessionkey for Session Manager:
	
	`[ //localhost/Radius/SessionManagers/session-mgr-1 ]`
	`Name = session-mgr-1`
	`Description =`
	`Type = local`
	`EnableDiameter = FALSE`
	`IncomingScript =`
	`OutoingScript =`
	`AllowAccountingStartToCreateSession = TRUE`
	`SessionTimeOut =`
	`PhantomSessionTimeOut =`
	`SessionKey =`
	`ResourceManagers/`

You can manage sessions with the two **aregcmd** session management commands: **query-sessions** and **release-sessions**. For more information about these two commands, see the query-sessions, page 4-9 and the release-sessions, page 4-9.

This section contains the following topics:

- Session Creation

- Session Notes
- Soft Group Session Limit
- Session Correlation Based on User-Defined Attributes

Session Creation

Cisco Prime Access Registrar Sessions can be created by two types of RADIUS packets:

- Access-Requests
- Accounting-Requests with an **Acct-Status-Type** attribute with a value of **Start**.

This allows Cisco Prime Access Registrar to monitor Sessions even when it is not allocating resources. For example, when Cisco Prime Access Registrar is being used as an "Accounting-Only" server (only receiving Accounting requests), it can create a Session for each Accounting "Start" packet it successfully processes. The corresponding Accounting "Stop" request will clean up the Session. Note, if a Session already exists for that NAS/NAS-Port/User (created by an Access-Request), Cisco Prime Access Registrar will not create a new one.

When you do not want Cisco Prime Access Registrar to create Sessions for Accounting "Start" requests, simply set the **AllowAccountingStartToCreateSession** property on the SessionManager to FALSE.

Session Notes

Session Notes are named text messages attached to a Session and are stored with the Session data, including resources allocated for a specific user session. This data, including Session Notes, can be retrieved and viewed using the **aregcmd** command **query-sessions**.

--> **query-sessions /Radius/SessionManagers/session-mgr-2**

```
sessions for /Radius/SessionManagers/session-mgr-2:
S257 NAS: localhost, NAS-Port:1, User-Name: user1, Time: 00:00:08,
  IPX 0x1, GSL 1, USL 1, NOTES: "Date" "Today is 12/14/98.", "Requested
  IP Address" "1.2.3.4", "Framed-IP-Address" "11.21.31.4"
```

Session Notes can be created by Scripts using the Environment dictionary passed into each or by the Cisco Prime Access Registrar server. When more than one Session Note is added, the **Session-Notes** entry should be a comma-separated list of entry names.

Performing a TCL Script

To perform a TCL script:

Step 1 The Script should create an Environment dictionary entry using the Session Note name as the entry name, and the Session Note text as the entry value. For example:

```
$environ put "Date" "Today is 12/15/08"
$environ put "Request IP Address" "1.2.3.4"
```

Step 2 The Script should create or set an Environment dictionary entry with the name **Session-Notes** with a value that contains the name of the entries created. For example:

```
$environ put "Session-Notes" "Date, Requested_IP_Address"
```

Performing a REX Script

To perform a REX script:

Step 1 The Script should create an Environment dictionary entry using the Session Note name as the entry name, and the Session Note text as the entry value. For example:

```
pEnviron-->put(pEnviron, Date, "Today is 12/15/08.");
pEnviron-->put(pEnviron, Request_IP_Address, "1.2.3.4");
```

Step 2 The Script should create/set an Environment dictionary entry with the name **Session-Notes** with a value that contains the name of the first entry created. For example:

```
pEnviron-->put(pEnviron, "Session-Notes", "Date, Requested_IP_Address”);
```

Note Scripts creating Session Notes must be executed before the Session Management step takes place while processing a packet.

Cisco Prime Access Registrar will automatically create a Session Note if a packet is passed to a SessionManager and it already contains a **Framed-IP-Address** attribute in the packet's Response dictionary. This IP address could come from a Profile, RemoteServer response, or from a previously executed script. For example, a Session output containing Session Notes when using the **aregcmd** command **query-session** would be as follows:

```
sessions for /Radius/SessionManagers/session-mgr-2:
  S257 NAS: localhost, NAS-Port:1, User-Name: user1, Time: 00:00:08,
  IPX 0x1, GSL 1, USL 1, NOTES: "Date" "Today is 12/14/08.", "Requested
  IP Address" "1.2.3.4", "Framed-IP-Address" "11.21.31.4"
```

Session Notes are also copied into the Environment dictionary after Session Management. The **Session-Notes** Environment dictionary entry will contain the names of all the Environment dictionary entries containing Session Notes.

In Prime Access Registrar, a major command is introduced—count-sessions. The **count-sessions /radius all** command helps to count the total sessions in Prime Access Registrar. The options are similar to the query-session command options. The query-session command displays cached attributes in addition to session details.

Soft Group Session Limit

Two new environment variables, **Group-Session-Limit** and **Current-Group-Count** (see rex.h), are set if the group session limit resource is allocated for a packet. These variables allow a script to see how close the group is to its session limit; one way to use this information is to implement a script-based soft limit. For example, you could use the Class attribute to mark sessions that have exceeded a soft limit of 80% -- as hard coded in the script (in a Tcl script called from /Radius/):

```
set softlimit [ expr 0.8 * [ $environ get Group-Session-Limit ] ]
if { [ $environ get Current-Group-Count ] < $softlimit } {
$response put Class 0
} else {
$response put Class 1
}
```

Note The soft limit itself is hard coded in the script; soft limits are not directly supported in the server. The action to be taken when the soft limit is exceeded (for example, Class = 1, and then the accounting software branches on the value of Class) is also the responsibility of the script and/or external software.

Session Correlation Based on User-Defined Attributes

All the session objects are maintained in one dictionary keyed by a string. You can define the keying material to the session dictionary through a newly introduced environment variable, **Session-Key**.

If the **Session-Key** is presented at the time of session manager process, it will be used as the key to the session object for this session. The **Session-Key** is of type string. By default, the **Session-Key** is not set. Its value should come from attributes in the incoming packet and is typically set by scripts. For example, CLID can be used to set the value of **Session-Key**.

Use the function UseCLIDAsSessionKey as defined in the script **rexscript.c** to specify that the **Calling-Station-Id** attribute that should be used as the session key to correlate requests for the same session. This is a typical case for 3G mobile user session correlation. You can provide your own script to define other attributes as the session key.

In the absence of the **Session-Key** variable, the key to the session will be created based on the string concatenated by the value of the **NAS-Identifier** and the **NAS-Port**.

There is a new option *with-key* available in **aregcmd** for query-sessions and release-sessions to access sessions by **Session-Key**.

Resource Managers

Resource Managers allow you to allocate dynamic resources to user sessions. The following lists the different types of Resource Managers.

- **IP-Dynamic**—manages a pool of IP addresses that allows you to dynamically allocate IP addresses from a pool of addresses
- **IP-Per-NAS-Port**—allows you to associate ports to specific IP addresses, and thus ensure each NAS port always gets the same IP address
- **IPX-Dynamic**—manages a pool of IPX network addresses
- **Subnet-Dynamic**—manages a pool of subnet addresses
- **Group-Session-Limit**—manages concurrent sessions for a group of users; that is, it keeps track of how many sessions are active and denies new sessions after the configured limit has been reached
- **User-Session-Limit**—manages per-user concurrent sessions; that is, it keeps track of how many sessions each user has and denies the user a new session after the configured limit has been reached
- **Home-Agent**—manages a pool of on-demand IP addresses
- **USR-VPN**—manages Virtual Private Networks (VPNs) that use USR NAS Clients.
- **Home-Agent-IPv6**—manages a pool of on-demand IPv6 addresses
- **Remote-IP-Dynamic**—manages a pool of IP addresses that allows you to dynamically allocate IP addresses from a pool of addresses. It internally works with a remote ODBC database.

- **Remote-User-Session-Limit**—manages per-user concurrent sessions; that is, it keeps track of how many sessions each user has and denies the user a new session after the configured limit has been reached. It internally works with a remote ODBC database.
- **Remote-Group-Session-Limit**—manages concurrent sessions for a group of users; that is, it keeps track of how many sessions are active and denies new sessions after the configured limit has been reached. It internally works with a remote ODBC database.
- **3GPP**—allows you to define the attribute for 3GPP authorization.

Each Resource Manager is responsible for examining the request and deciding whether to allocate a resource for the user, do nothing, or cause Cisco Prime Access Registrar to reject the request.

Table 5-26 lists the Resource Manager properties.

Table 5-26 Resource Manager Properties

Property	Description
Name	Required; must be unique in the Resource Managers list.
Description	Optional; description of the Resource Manager.
Type	Required; must be either **Dynamic-DNS**, **IP-Dynamic**, **IP-Per-NAS-Port**, **IPX-Dynamic**, **Session Cache, Subnet-Dynamic, Group-Session-Limit**, **Home-Agent**, **User-Session-Limit**, **USR-VPN, Home-Agent-IPv6, Remote-IP-Dynamic, Remote-User-Session-Limit, Remote-Group-Session-Limit** or **Remote-Session-Cache**.

Types of Resource Managers

A number of different types of Resource Managers exist that allow you to manage IP addresses dynamically or statically, limit sessions on a per group or per user basis, or manage a Virtual Private Network. See the "Cisco Prime Access Registrar Tcl, REX, and Java Dictionaries" chapter of the *Cisco Prime Access Registrar 8.0 Reference Guide* for information on how to override these individual Resource Managers.

Note Resource Manager supports the following remote type session managers: remote-ip-dynamic, remote-session-cache, home-agent, remote-user-session-limit, home-agent-ipv6 and remote-group-session-limit.

This section contains the following topics:

- Group-Session-Limit
- Home-Agent
- Home-Agent-IPv6
- IP-Dynamic
- IP-Per-NAS-Port
- IPX-Dynamic
- Session-Cache
- Subnet-Dynamic

- User-Session-Limit
- USR-VPN
- Dynamic-DNS
- Remote-IP-Dynamic
- Remote-User-Session-Limit
- Remote-Group-Session-Limit
- Remote-Session-Cache
- 3GPP

Group-Session-Limit

Group-Session-Limit allows you to manage concurrent sessions for a group of users; that is, it keeps track of how many sessions are active and denies new sessions after the configured limit has been reached.

When you use this Resource Manager, you must set the GroupSessionLimit property to the maximum number of concurrent sessions for all users.

Home-Agent

Home-Agent is a resource manager that supports dynamic HA assignment. You configure the home-agent resource manager with a list of IP addresses. The Prime Access Registrar server assigns those addresses to clients whose request dictionary has the right attributes to indicate that an assignment should be done. This is similar to the **ip-dynamic** resource manager.

Unlike the **ip-dynamic** resource manager, HAs are not exclusively allocated to an individual session but are shared among a set of sessions.

Detailed configuration information for the Home-Agent resource manager is found in the "Wireless" chapter of the Cisco Prime Access Registrar 8.0 Reference Guide. When you use this Resource Manager, you must set the Home-Agent-IPAddresses property to a single IP address or a range of IP addresses.

Home-Agent-IPv6

Home-Agent-Ipv6 is a new resource manager used to configure IPv6 address.

IP-Dynamic

IP-Dynamic allows you to manage a pool of IP addresses from which you dynamically allocate IP addresses.

When you use the IP-Dynamic Resource Manager, provide values for the properties listed in Table 5-27.

Table 5-27 IP-Dynamic Properties

Property	Description
NetMask	Required; must be set to a valid net mask.
IPAddresses	Required; must be a list of IP address ranges.

Table 5-27 IP-Dynamic Properties (continued)

Property	Description
AllowOverlappedIPAddresses	When set to TRUE, this property supports overlapping IP addresses between session managers for VPN users. Default value is FALSE.
ReuseIPForSameSessionKeyAndUser	When set to FALSE, this property does not reuse IP address resources for a session. Default value is TRUE.

IP-Per-NAS-Port

IP-Per-NAS-Port allows you to associate specific IP addresses with specific NAS ports and thus ensures each NAS port always gets the same IP address.

When you use this Resource Manager, provide values for the properties listed in Table 5-28.

Note You must have the same number of IP addresses and ports.

Table 5-28 IP-Per-NAS-Port Properties

Property	Description
NetMask	Required; if used, must be set to a valid net mask.
NAS	Required; must be the name of a known Client.This value must be the same as the NAS-Identifier attribute in the Access-Request packet.
IPAddresses	Required; must be a list of IP address ranges.
NASPorts	Required list of NAS ports.

IPX-Dynamic

An IPX-Dynamic Resource Manager allows you to dynamically manage a pool of IPX networks. When you use the IPX-Dynamic Resource Manager, you must set the Networks property to a valid set of numbers which correspond to your networks.

Note You cannot use IPX network number 0x0. If you attempt to configure a Resource Manager with an IPX network number of 0x0, validation will fail.

Session-Cache

The **session-cache** Resource Manager supports the Identity Cache feature. You use session-cache Resource Managers to define the RADIUS attributes to store in cache. Set the QueryKey property to the XML attribute you want to key on such as XML-Address-format-IPv4 and list all attributes to be cached in the AttributesToBeCached subdirectory. Use the QueryMappings subdirectory to map XML attributes to RADIUS attributes.

Table 5-29 Session-Cache Resource Manager Properties

Property	Description
QueryKey	Required; set the QueryKey to the a RADIUS attribute you want to key on, such as Framed-IP-Address. A change made in Prime Access Registrar requires that this attribute not be an XML attribute, even if this session-cache resource manager is being used for an XML query. **Note** Any existing session-cache resource managers using an XML attribute for the Query Key must be changed to a RADIUS attribute that this XML attribute is mapped to under QueryMappings.
PendingRemovalDelay	Required; length of time information remains in the cache after the session ends (defaults to 10 seconds)
AttributesToBeCached	Required; use this subdirectory to provide a list of RADIUS attributes you want to store in cache
QueryMappings	Required; list of attribute pairs, mapping the XML attributes on the left-hand side to the RADIUS attribute on the right-hand side.

Note Session record size is limited by the operating system (OS) paging size (4 KB in Linux). If a request triggers creation of a session that exceeds the OS paging size, the request will be dropped and the session will not be created.

If the disk partition where Prime Access Registrar stores session backing store data (usually the disk partition where Prime Access Registrar is installed, such as **/opt/CSCOar**) is full, the subsequent packets that try to create sessions will be dropped and no sessions will be created due to lack of disk space.

Subnet-Dynamic

The **subnet-dynamic** Resource Manager supports the On Demand Address Pool feature. You use subnet-dynamic resource managers to provide pools of subnet addresses. Following is an example of the configuration of a subnet dynamic resource manager:

```
/Radius/ResourceManagers/newResourceMgr
Name = newResourceMgr
Description =
Type = subnet-dynamic
Subnet-Mask = 255.255.255.0
SubnetAddresses/
    10.1.0.0-10.1.10.0
    11.1.0.0-11.1.10.0
```

When you use the subnet-dynamic Resource Manager, provide values for the properties listed in Table 5-30.

Table 5-30 Subnet-Dynamic Properties

Property	Description
Type	Required
Subnet mask	Required; must be set to the size of the managed subnets
SubnetAddresses	Required; must be a valid range of IP addresses

User-Session-Limit

User-Session-Limit allows you to manage per-user concurrent sessions; that is, it keeps track of how many sessions each user has and denies the user a new session after the configured limit has been reached.

When you use the user-session-limit Resource Manager, set the user-session-limit property to the maximum number of concurrent sessions for a particular user.

USR-VPN

USR-VPN allows you to set up a Virtual Private Network (VPN) using a US Robotics NAS.

When you use this Resource Manager, provide values for the properties listed in Table 5-31.

Table 5-31 USR-VPN Properties

Property	Description
General	
Identifier	Required; must be set to the VPN ID the USR NAS will use to identify a VPN.
Neighbor	Optional; if set, should be the IP address of the next hop router for the VPN.
FramedRouting	Optional; if set, should be **RIP V2 Off** or **RIP V2 On** if the USR NAS is to run RIP Version 2 for the user.
Gateway **Gateway** includes a list of names of the Frame Relay Gateways for which to encrypt the session key.	
Name	Required; must be unique in the Gateways list.
Description	Optional description of the gateway.
IPAddress	Required; IP address of the gateway.
SharedSecret	Required; must match the shared secret of the gateway.
TunnelRefresh	Optional; if specified it is the number of seconds the tunnel stays active before a secure “keepalive” is exchanged between the tunnel peers in order to maintain the tunnel open.
LocationID	Optional; if specified it is a string indicating the physical location of the gateway.

Dynamic-DNS

Prime Access Registrar supports the Dynamic DNS protocol providing the ability to update DNS servers.

When you use this Resource Manager, provide values for the properties listed in Table 5-32.

Table 5-32 DYNAMIC-DNS Properties

Fields	Description
Max DNS TTLS	Set the maximum TTL of the DNS record.
DNS Host bytes	Set the number of bytes to be used to construct the reverse zone entry.
Forward Zone Name	Set the name of the forward zone. For a given Resource Manager you must decide which forward zone you will be updating for sessions the resource manager will manage.
Reverse Zone Name	Set the name of the reverse zone.
Forward Zone Server	Set the Server IP of the forward zone
Reverse Zone Server	Set the Server IP of the reverse zone
Forward Zone TSIG KeyS	Server-wide security key to process all forward zone dynamic DNS updates. This is used if a ForwardZoneTSIGKey was not specified on the Resource Manager.
Reverse Zone TSIG Keys	Server-wide security key to process all reverse zone dynamic DNS updates. This is used if a ReverseZoneTSIGKey was not specified on the Resource Manager

Remote-IP-Dynamic

The configuration is same as IP-Dynamic but internally it works with a remote ODBC database.

Remote-User-Session-Limit

The configuration is same as User-Session-Limit but internally it works with a remote ODBC database.

Remote-Group-Session-Limit

The configuration is same as Group-Session-Limit but internally it works with a remote ODBC database.

Remote-Session-Cache

The configuration is same as Session-Cache but it should be used with session manager of type remote.

3GPP

Prime Access Registrar provides a resource manager for 3GPP authorization. When you use this Resource Manager, provide values for the properties listed in Table 5-33.

Table 5-33 3GPP Resource Manager Properties

Fields	Description
EnableRegistrationFlow	Set to TRUE to enable registration flow during 3GPP authorization.
EnableSessionTermination	Set to TRUE to enable session termination during 3GPP authorization.
ReuseExistingSession	Set to TRUE to reuse existing session during 3GPP authorization.
HSSProxyService	Required; the HSS proxy service to use in the 3GPP authorization flow.

Profiles

You use Profiles to group RADIUS attributes that belong together, such as attributes that are appropriate for a particular class of PPP or Telnet user. You can reference profiles by name from either the **UserGroup** or the **User** properties. Thus, if the specifications of a particular profile change, you can make the change in a single place and have it propagated throughout your user community.

Although you can use UserGroups or Profiles in a similar manner, choosing whether to use one rather than the other depends on your site. When you require some choice in determining how to authorize or authenticate a user session, then creating specific profiles, and creating a group that uses a script to choose among them is more flexible.

In such a situation, you might create a default group, and then write a script that selects the appropriate profile based on the specific request. The benefit to this technique is each user can have a single entry, and use the appropriate profile depending on the way they log in.

Table 5-34 lists the **Profile** properties.

Table 5-34 Profile Properties

Property	Description
Name	Required; must be unique in the Profiles list.
Description	Optional; description of the profile.
Attributes	Profiles include specific RADIUS attributes that Cisco Prime Access Registrar returns in the Access-Accept response.

Attributes

Attributes are specific RADIUS components of requests and responses defined in the Request and Response Attribute dictionaries. Use the **aregcmd** command **set** to assign values to attributes.

For a complete list of the attributes, see the "RADIUS Attributes" chapter of the *Cisco Prime Access Registrar 8.0 Reference Guide.*

When setting a value for a STRING-type attribute such as Connect-Info (which starts with an integer), you must use the hexadecimal representation of the integer. For example, to set the attribute Connect-Info to a value of 7:7, use a set command like the following:

```
set Connect-Info 37:3A:37
```

Translations

Translations add new attributes to a packet or change an existing attribute from one value to another. The **Translations** subdirectory lists all definitions of **Translations** the RADIUS server can apply to certain packets.

Under the **/Radius/Translations** directory, any translation to insert, substitute, or translate attributes can be added. The following is a sample configuration under the **/Radius/Translations** directory:

```
cd /Radius/Translations
Add T1
cd T1
Set DeleAttrs Session-Timeout,Called-Station-Id
cd Attributes
Set Calling-Station-Id 18009998888
```

DeleAttrs is the set of attributes to be deleted from the packet. Each attribute is comma separated and no spaces are allowed between attributes. All attribute value pairs under the attributes subdirectory are the attributes and values that are going to be added or translated to the packet.

Under the **/Radius/Translations/T1/Attributes** directory, inserted or translated attribute value pairs can be set. These attribute value pairs are either added to the packet or replaced with the new value.

If a translation applies to an Access-Request packet, by referencing the definition of that translation, the Prime Access Registrar server modifies the Request dictionary and inserts, filters and substitutes the attributes accordingly. You can set many translations for one packet and the Prime Access Registrar server applies these translations sequentially.

Note Later translations can overwrite previous translations.

Table 5-35 lists the Translation properties.

Table 5-35 Translations Properties

Property	Description
Name	Required; must be unique in the Translations list.
Description	Optional; description of the Translation
DeleteAttrs	Optional; lists attributes to be filtered out

TranslationGroups

You can add translation groups for different user groups under **TranslationGroups**. All Translations under the Translations subdirectory are applied to those packets that fall into the groups. The groups are integrated with the Prime Access Registrar Rule engine.

The Prime Access Registrar Administrator can use any RADIUS attribute to determine the **Translation Group**. The incoming and outgoing translation group can be different translation groups. For example, you can set one translation group for incoming translations and one for outgoing translations.

Under the **/Radius/TranslationGroups** directory, translations can be grouped and applied to certain sets of packets, which are referred to in a rule. The following is a sample configuration under the **/Radius/TranslationGroups** directory:

```
cd /Radius/TranslationGroups
Add CiscoIncoming
cd CiscoIncoming
cd Translations
Set 1 T1
```

The translation group is referenced through the Prime Access Registrar Policy Engine in the **/Radius/Rules/**<*RuleName*>**/Attributes** directory. **Incoming-Translation-Groups** are set to a translation group (for example `CiscoIncoming`) and **Outgoing-Translation-Groups** to another translation group (for example `CiscoOutgoing`). Table 5-36 lists the Translation Group properties.

Table 5-36 TranslationGroups Properties

Property	Description
Name	Required; must be unique in the Translations list.
Description	Optional; description of the Translation Group
Translations	Lists of translation

Remote Servers

You can use the **RemoteServers** object to specify the properties of the remote servers to which Services proxy requests. **RemoteServers** are referenced by name from the **RemoteServers** list in either the **radius**, **ldap** or **tacacs-udp** Services.

Note You must not configure a remote server with an IP address, which is same as that of the client. This is applicable for all types of remote servers.

Table 5-37 lists the common **RemoteServers** properties.

Table 5-37 Common RemoteServer Properties

Property	Description
Name	Required; must be unique in the RemoteServers list.
Description	Optional; description of the remote server.
Protocol	Required; specifies the remote server protocol which can be **radius**, **ldap**, or **tacacs-udp**.
IPAddress	Required; this property specifies where to send the proxy request. It is the address of the remote server. You must set it to a valid IP address. The IP address format is enhanced to support IPv6 apart from IPv4 only for the RADIUS type remote server.

Table 5-37 Common RemoteServer Properties (continued)

Property	Description
Port	Required; the port to which Cisco Prime Access Registrar sends proxy requests. You must specify a number greater than zero. If there is no default port number, you must supply the correct port number for your remote server. If you set a port to zero, Prime Access Registrar sets the port to the default value for the type of remote server being configured. For example, the following remote servers have these default port values: dynamic-dns—53 radius—1812 ldap—389 accounting—1813
ReactivateTimerInterval	Required; the amount of time (in milliseconds) to wait before retrying a remote server that was offline. You must specify a number greater than zero. The default is 300,000 (5 minutes).

Types of Protocols

The Remote Server protocol you specify determines what additional information you must provide. The following are the protocols available in Prime Access Registrar with their required and optional fields.

Prime Access Registrar provides the following RemoteServer protocol types:

- Dynamic DNS
- LDAP
- Map-Gateway
- Sigtran
- ODBC
- ODBC-Accounting
- OCI
- OCI-Accounting
- Prepaid-CRB
- Prepaid-IS835C
- RADIUS
- Diameter
- REST
- SIGTRAN-M3UA

Dynamic DNS

The **dynamic-dns** RemoteServer is used with the Dynamic DNS feature. The following is the default configuration of a dynamic-dns RemoteServer.

```
[ //localhost/Radius/RemoteServers/ddns ]
```

```
Name = ddns
Description =
Protocol = dynamic-dns
IPAddress =
Port = 53
MaxTries = 3
InitialTimeout = 2000
MaxDNSRenamingRetries = 3
TrimHostName = TRUE
ForwardZoneTSIGKey =
ReverseZoneTSIGKey =
```

Table 5-38 lists and defines the dynamic-dns RemoteServer properties.

Table 5-38 Dynamic-DNS RemoteServer Properties

Property	Description
IPAddress	The IPAddress address of the DNS server
Port	Port 53 is the port that most DNS servers will use as a default
MaxTries	Number of times the server tries to send dynamic updates to a DNS server
InitialTimeout	Time, in milliseconds, that the server waits for a response before retrying a dynamic DNS request
MaxRenamingRetries	Number of times that the dynamic-dns resource managers can try to add a host in DNS even if it detects that the host's name is already present. This controls the number of times Prime Access Registrar tries to modify a host's name to resolve a conflict on each failed update.
TrimHostName	Controls whether Prime Access Registrar trims the hostname string to the first period character (used to update dynamic DNS update records and to return the hostname option to clients). If this attribute is enabled, the hostname is truncated before the period. If disabled, the server retains the period characters in the hostname.
ForwardZoneTSIGKey	Server-wide security key to process all forward zone dynamic DNS updates. This is used if a ForwardZoneTSIGKey was not specified on the Resource Manager.
ForwardZoneTSIGKey	Server-wide security key to process all forward zone dynamic DNS updates. This is used if a ForwardZoneTSIGKey was not specified on the Resource Manager.
ReverseZoneTSIGKey	Server-wide security key to process all reverse zone dynamic DNS updates. This is used if a ReverseZoneTSIGKey was not specified on the Resource Manager.

LDAP

ldap specifies an LDAP server. When you specify the **ldap** protocol, provide the information listed in Table 5-39.

For any LDAP remote service, the server might perform the environment mappings at any time. This means that if the service is set to either authentication and authorization, authentication-only, or authorization-only, environment mappings will take place. RADIUS mappings will take place only if the service is set to perform authorization. Checkitem mappings will take place only if the service is set to

perform authentication. Previously environment mappings only occurred when the service was set for both authentication and authorization. RADIUS mappings, environment mappings, and checkitem mappings will not take place, if bind-based authentication is enabled.

Table 5-39 ldap RemoteServer Properties

Property	Description
Port	Required; defaults to port 389.
Timeout	Required; the default is 15. The timeout property indicates how many seconds the RADIUS server will wait for a response from the LDAP server. **Note** Use InitialTimeout from above as a template, except this is timeout is specified in seconds.
HostName	Required; the LDAP server's hostname or IP address.
BindName	Optional; the distinguished name (dn) to use when establishing a connection between the LDAP and RADIUS servers.
BindPassword	Optional; the password associated with the **BindName**.
SearchPath (Overridden by Search-Path environment variable)	Required; the path that indicates where in the LDAP database to start the search for user information.
Filter	Required; this specifies the search filter Cisco Prime Access Registrar uses when querying the LDAP server for user information. When you configure this property, use the notation "%s" to indicate where the user ID should be inserted. For example, a typical value for this property is "(uid=%s)," which means that when querying for information about user `joe`, use the filter `uid=joe`.
UserPasswordAttribute	Required; this specifies which LDAP field the RADIUS server should check for the user's password.
LimitOutstandingRequests	Required; the default is FALSE. Cisco Prime Access Registrar uses this property in conjunction with the **MaxOutstandingRequests** property to tune the RADIUS server's use of the LDAP server. When you set this property to TRUE, the number of outstanding requests for this RemoteServer is limited to the value you specified in **MaxOutstandingRequests**. When the number of requests exceeds this number, Cisco Prime Access Registrar queues the remaining requests, and sends them as soon as the number of outstanding requests drops to this number.
MaxOutstandingRequests	Required when you have set the **LimitOutstandingRequests** to TRUE. The number you specify, which must be greater than zero, determines the maximum number of outstanding requests allowed for this remote server.

Table 5-39 ldap RemoteServer Properties (continued)

Property	Description
MaxReferrals	Required; must be a number equal to or greater than zero. This property indicates how many referrals are allowed when looking up user information. When you set this property to zero, no referrals are allowed. Cisco Prime Access Registrar manages referrals by allowing the RADIUS server's administrator to indicate an LDAP "referral attribute," which might or might not appear in the user information returned from an LDAP query. When this information is returned from a query, Cisco Prime Access Registrar assumes it is a referral and initiates another query based on the referral. Referrals can also contain referrals. **Note** This is an LDAP v2 referral property.
ReferralAttribute	Required when you have specified a **MaxReferrals** value. This property specifies which LDAP attribute, returned from an LDAP search, to check for referral information. **Note** This is an LDAP v2 referral property.
ReferralFilter	Required when you have specified a **MaxReferral** value. This is the filter Cisco Prime Access Registrar uses when processing referrals. When checking referrals, the information Cisco Prime Access Registrar finds in the referral itself is considered to be the search path and this property provides the filter. The syntax is the same as that of the **Filter** property. **Note** This is an LDAP v2 referral property.
PasswordEncryptionStyle	The default is **None**. You can also specify **crypt, dynamic, SHA-1, and SSHA-1**.
EscapeSpecialCharInUserName	FALSE by default
DNSLookupAndLDAPRebindInterval	Specifies the timeout period after which the Prime Access Registrar server will attempt to resolve the LDAP hostname to IP address (DNS resolution); 0 by default
DataSourceConnections	Specifies the number of concurrent connections to the LDAP server. The default value is 8.
SearchScope	Specifies how deep to search within a search path; default is *SubTree* which indicates a search of the base object and the entire subtree of which the base object distinguished name is the highest object. *Base* indicates a search of the base object only. *OneLevel* indicates a search of objects immediately subordinate to the base object, but does not include the base object.

Table 5-39 ldap RemoteServer Properties (continued)

Property	Description
LDAPToRadiusMappings	Optional; a list of name/value pairs in which the name is the name of the **ldap** attribute to retrieve from the user record, and the value is the name of the RADIUS attribute to set to the value of the **ldap** attribute retrieved. For example, when the **LDAPToRadiusMappings** has the entry: **FramedIPAddress = Framed-IP-Address**, the RemoteServer retrieves the **FramedIPAddress** attribute from the **ldap** user entry for the specified user, uses the value returned, and sets the Response variable **Framed-IP-Address** to that value.
LDAPToEnvironmentMappings	Optional; a list of name/value pairs in which the name is the name of the **ldap** attribute to retrieve from the user record, and the value is the name of the Environment variable to set to the value of the **ldap** attribute retrieved. For example, when the **LDAPToEnvironmentMappings** has the entry: **group = User-Group**, the RemoteServer retrieves the **group** attribute from the **ldap** user entry for the specified user, uses the value returned, and sets the Environment variable **User-Group** to that value.
LDAPToCheckItemMappings	Optional; a list of LDAP *attribute/value* pairs which must be present in the RADIUS access request and must match, both name and value, for the check to pass. For example, when the **LDAPToCheckItemMappings** has the entry: **group = User-Group**, the Access Request must contain the attribute **group**, and it must be set to **User-Group**.
UseSSL	A boolean field indicating whether you want Cisco Prime Access Registrar to use SSL (Secure Socket Layer) when communicating with this RemoteServer. When you set it to TRUE, be sure to specify the **CertificateDBPath** field in the **Advanced** section, and be sure the port you specified for this RemoteServer is the SSL port used by the LDAP server.
UseBinaryPasswordComparison	A boolean field that enables binary password comparison for authentication. This property when set to TRUE, enables binary password comparison. By default, this property is set to FALSE.
UseBindBasedAuthentication	A boolean field that enables bind-based authentication with LDAP server. This property when set to TRUE, enables bind-based authentication. By default, this property is set to FALSE. When set to FALSE, it uses existing legacy authentication method.

Map-Gateway

The following is the default configuration of a map gateway RemoteServer.

```
[ //localhost/Radius/RemoteServers/map-gateway ]
    Name = map-gateway
    Description =
    Protocol = map-gateway
    IPAddress =
    Port = 0
```

```
    ReactivateTimerInterval = 300000
    SharedSecret =
    MaxTries = 3
    InitialTimeout = 2000
```

Sigtran

The following is the default configuration of a Sigtran RemoteServer.

```
[ //localhost/Radius/RemoteServers/rs ]
    Name = rs
    Description =
    Protocol = sigtran
    HostName =
    LocalSubSystemNumber =
    CgPAGlobalTitleAddress =
    SetOPCInCgPA =
    GlobalTitleTranslationScript~ =
    SUAConfigurationFilename =
    ReactivateTimerInterval =
    Timeout = 5000
    LimitOutstandingRequests = FALSE
    MaxOutstandingRequests = 0
```

Note The RPM packages such as lksctp-tools-1.0.10-1, lksctp-tools-doc-1.0.10-1 and lksctp-tools-devel-1.0.10-1 should be installed in Linux 5.3 before configuring sigtran remote server which eventually adds the sctp libs (libsctp.so.1.0.10).

The following files can be downloaded from http://lksctp.sourceforge.net/

- lksctp-tools-1.0.10-1.i386.rpm
- lksctp-tools-devel-1.0.10-1.i386.rpm
- lksctp-tools-doc-1.0.10-1.i386.rpm

Prime Access Registrar supports only:

- one object of Remoteserver with protocol type "sigtran"
- MAP version 3 (3GPP TS 29.002 V6.4.0 (2003-12)) and ITU Q.773 TCAP

Only one Quintets is fetched from HLR. The ITU TCAP continue message is not supported.

Table 5-40 lists and defines the Sigtran RemoteServer properties.

Table 5-40 Sigtran RemoteServer Properties

Property	Description
HostName	Required; represents the IP address of remote Signalling Gateway specified in the SUAConfiguration file.
LocalSubSystemNumber	Required; the default value for this property is 0. This represents the subsystem number used by SUA user.
CgPAGlobalTitleAddress	Required; represents the Global Title Address of CallingPartyAddress.
SetOPCInCgPA	Required; if it is set to TRUE, OPC will be used in CallingPartyAddress.

Table 5-40 Sigtran RemoteServer Properties (continued)

Property	Description
Global TitleTranslationScript	This is used to specify the name of script which is responsible for translating IMSI to GTA.
SUAConfigurationFilename	Required; used to specify the name of configuration file for SUA stack initialization.
ReactivateTimerInterval	Required; represents the reactivate time interval to re-connect after failure.
Timeout	Required; represents the how long the remote server should wait before marking the request as timedout.
LimitOutstandingRequests	Limits the outstanding request to HLR when it is set to TRUE.
MaxOutstandingRequests	This represents the maximum outstanding request to HLR.

Note You should restart the Prime Access Registrar server, if you change any SIGTRAN related configuration.

ODBC

odbc specifies an ODBC server. Cisco Prime Access Registrar provides a RemoteServer object (and a service) to support Open Database Connectivity (ODBC), an open specification that provides application developers a vendor-independent API with which to access data sources. Table 5-41 lists the **odbc** server attributes.

For any ODBC remote service, the server might perform the environment mappings at any time. This means that if the service is set to either authentication and authorization, authentication-only, or authorization-only, environment mappings will take place. RADIUS mappings will take place only if the service is set to perform authorization. Checkitem mappings will take place only if the service is set to perform authentication. Previously environment mappings only occurred when the service was set for both authentication and authorization.

Table 5-41 odbc Properties

Property	Description
Timeout	Required; the default is 15. The timeout property indicates how many seconds the RADIUS server will wait for a response from the LDAP server. **Note** Use InitialTimeout from above as a template, except this is timeout is specified in seconds.
Protocol	Must be set to **odbc**.
ReactivateTimerInterval	Required; default is 300,000 milliseconds. Length of time to wait before attempting to reconnect if a thread is not connected to a data source.

Table 5-41 odbc Properties (continued)

Property	Description
Data Source Connections	Required; default is 8. This represents the total number of connections Prime Access Registrar can open with the ODBC server; total number of threads Prime Access Registrar can create for the ODBC server.
ODBCDataSource	Required; defines all items required for the **odbc.ini** file. The Prime Access Registrar server automatically creates the **odbc.ini** file based on these settings.
SQLDefinition	SQLDefinition properties define the SQL you want to execute. Type—**query** (Prime Access Registrar supports only type **query**). SQL—SQL query used to acquire the password UserPasswordAttribute—Defines the database column name for the user's password. MarkerList—Defines all markers for the query. MarkerList uses the format UserName/SQL_DATA_TYPE.
ODBCToRadiusMappings	Optional; a list of name/value pairs in which the name is the name of the **odbc** attribute to retrieve from the user record, and the value is the name of the RADIUS attribute to set to the value of the **odbc** attribute retrieved. For example, when the **ODBCToRadiusMappings** has the entry: **FramedIPAddress = Framed-IP-Address**, the RemoteServer retrieves the **FramedIPAddress** attribute from the **odbc** user entry for the specified user, uses the value returned, and sets the Response variable **Framed-IP-Address** to that value.
ODBCToEnvironmentMappings	Optional; a list of name/value pairs in which the name is the name of the **odbc** attribute to retrieve from the user record, and the value is the name of the Environment variable to set to the value of the **odbc** attribute retrieved. For example, when the **ODBCToEnvironmentMappings** has the entry: **group = User-Group**, the RemoteServer retrieves the **group** attribute from the **odbc** user entry for the specified user, uses the value returned, and sets the Environment variable **User-Group** to that value.
ODBCToCheckItemMappings	Optional; a list of ODBC *attribute/value* pairs which must be present in the RADIUS access request and must match, both name and value, for the check to pass. For example, when the **ODBCToCheckItemMappings** has the entry: **group = User-Group**, the Access Request must contain the attribute **group**, and it must be set to **User-Group**.

ODBC-Accounting

If you use the Oracle Accounting feature, you must configure an ODBC-Accounting RemoteServer object. Table 5-42 lists and defines the ODBC-Accounting RemoteServer properties.

Table 5-42 ODBC-Accounting RemoteServer Properties

Property	Description
Name	Name of the remote server; this property is mandatory, and there is no default
Description	Optional description of server
Protocol	Must be set to odbc-accounting
ReactivateTimerInterval	Mandatory time interval (in milliseconds) to activate an inactive server; defaults to 300000 ms.
Timeout	Mandatory time interval (in seconds) to wait for SQL operation to complete; defaults to 15 seconds
DataSourceConnections	Mandatory number of connections to be established; defaults to 8
ODBCDataSource	Name of the ODBCDataSource to use and must refer to one entry in the list of ODBC datasources configured under **/Radius/Advanced/ODBCDataSources**. Mandatory; no default
KeepAliveTimerInterval	Mandatory time interval (in milliseconds) to send a keepalive to keep the idle connection active; defaults to zero (0) meaning the option is disabled
BufferAccountingPackets	Mandatory, TRUE or FALSE, determines whether to buffer the accounting packets to local file, defaults to TRUE which means that packet buffering is enabled. **Note** When set to TRUE, a constant flow of incoming accounting packets can fill the buffer backing store files in **/cisco-ar/data/odbc** beyond the size configured in MaximumBufferFileSize. Configure BackingStoreDiscThreshold in **/Radius/Advanced** when using ODBC accounting. See Advanced, page 5-65 for information about how to configure BackingStoreDiscThreshold.
MaximumBufferFileSize	Mandatory if BufferAccountingPackets is set to TRUE, determines the maximum buffer file size, defaults to 10 Megabyte)
NumberOfRetriesForBufferdPacket	Mandatory if BufferAccountingPackets is set to TRUE. A number greater than zero determines the number of attempts to be made to insert the buffered packet into Oracle. Defaults to 3.

OCI

OCI service can be used to authenticate and authorize an access request by querying user information through OCI and to insert accounting records into a data store through OCI. For more information on OCI server properties, see the “Using the Graphical User Interface” chapter of the *Cisco Prime Access Registrar 8.0 User Guide*.

OCI-Accounting

If you use the Oracle Accounting feature, you must configure an OCI-Accounting RemoteServer object. For more information on OCI accounting server properties, see the "Using the Graphical User Interface" chapter of the *Cisco Prime Access Registrar 8.0 User Guide*.

Prepaid-CRB

The following is the default configuration of a prepaid-crb RemoteServer. The Filename property is the name of the required shared library provided by the billing vendor. See the "Using Prepaid Billing" chapter of the *Cisco Prime Access Registrar 8.0 User Guide* for more information on Prepaid -CRB.

```
[ //localhost/Radius/RemoteServers/prepaid-crb ]
    Name = prepaid-crb
    Description =
    Protocol = prepaid-crb
    IPAddress =
    Port = 0
    Filename =
    Connections = 8
```

Prepaid-IS835C

The following is the default configuration of a prepaid-is835c RemoteServer. The Filename property is the name of the required shared library provided by the billing vendor. See the "Using Prepaid Billing" chapter of the *Cisco Prime Access Registrar 8.0 User Guide* for more information on Prepaid -IS835C.

```
[ //localhost/Radius/RemoteServers/prepaid-is835c ]
    Name = prepaid-is835c
    Description =
    Protocol = prepaid-is835c
    IPAddress =
    Port = 0
    Filename =
    Connections = 8
```

RADIUS

radius specifies a RADIUS server. When you specify the **radius** protocol, supply the information in Table 5-43.

Table 5-43 RADIUS Properties

Property	Description
SharedSecret	Required; the secret shared between the remote server and the RADIUS server.
IncomingScript	Optional; when set, must be the name of a known incoming script. Cisco Prime Access Registrar runs the IncomingScript after it receives the response.
OutgoingScript	Optional; when set, must be the name of a known outgoing script. Cisco Prime Access Registrar runs the just before it sends the proxy request to the remote server.

Table 5-43 RADIUS Properties (continued)

Property	Description
Vendor	Optional; when set, must be the name of a known Vendor.
MaxTries	Required; the number of times to send a proxy request to a remote server before deciding the server is offline. You must specify a number greater than zero. The default is 3.
InitialTimeout	Required: represents the number of milliseconds used as a timeout for the first attempt to send a specific packet to a remote server. For each successive retry on the same packet, the previous timeout value used is doubled. You must specify a number greater than zero. The default value is 2000 (or 2 seconds).
ACKaccounting	When ACKAccounting is TRUE, the Prime Access Registrar server waits for the Accounting-Response from the remote RADIUS server before sending the corresponding Accounting-Response to the client. When ACKAccounting is FALSE, the Prime Access Registrar server does not wait for the Accounting-Response and immediately returns an Accounting-Response to the client.
SendandForget	This field is available if the AcknowledgeAccounting option is disabled. After forwarding a proxy packet to the remote server and an initial response to the client, Prime Access Registrar maintains a buffer of the original request and a copy of the proxy request until it receives a response from the remote server or packet timeout is triggered. If SendandForget is enabled, Prime Access Registrar deletes the original and proxy requests from the buffer after sending the response to the client. This helps in reducing buffer pool exhaustion in case of a low-responding remote server.

Diameter

Diameter is a networking protocol which is derived from RADIUS protocol. You can configure a Diameter remote server using a set of parameters.

The following is a sample CLI configuration of a Diameter remote server. For details about these parameters, refer to the “Using the Graphical User Interface” chapter of the Cisco Prime Access Registrar 8.0 User Guide.

```
[ //localhost/Radius/RemoteServers/vm028 ]
    Name = vm028
    Description =
    Protocol = diameter
    HostName = 10.81.78.248
    DestinationPort = 3868
    DestinationRealm = abc.com
    ReactivateTimerInterval = 300000
    Vendor =
    IncomingScript~ =
    OutgoingScript~ =
    UserLogEnabled = FALSE
    MaxTries = 3
    MaxTPSLimit = 0
    MaxSessionLimit = 0
    InitialTimeout = 2000
```

```
DisconnectBasedOnThreshold = TRUE
DisconnectThreshold = 1
LimitOutstandingRequests = FALSE
MaxPendingPackets = 0
MaxOutstandingRequests = 0
DWatchDogTimeout = 2500
SCTP-Enabled = false
TLS-Enabled = FALSE
AdvertiseHostName =
AdvertiseRealm =
```

REST

Prime Access Registrar allows you to configure a REST remote server for extended-EAP service. Extended-EAP is used as an authorization service to retrieve authorization information from the remote web server using the REST interface. Prime Access Registrar processes all EAP requests and extends through extended EAP service. Extended-EAP is supported for the following EAP protocols:

- EAP-AKA
- EAP-AKA-PRIME
- EAP-SIM

To configure an extended-EAP service, see Extended-EAP.

Table 5-44 lists and describes the REST remote server properties.

Table 5-44 REST Remote Server Properties

Fields	Description
RESTRemoteServerProperties Tab	
Name	Required; name of the REST server.
Description	Optional; description of the REST server.
Protocol	Specify as REST.
ReactivateTimerInterval	Required; time interval, in milliseconds, to reactivate an inactive REST server. Default value is 300000.
Timeout	Required; timeout value, in milliseconds, the REST server can wait for a request or response before attempting a retry. Default value is 15. We recommend that you set the value to 1000.
MaxTimeOuts	Maximum number of timeouts allowed for the remote server.
RESTSourceConnections	Mandatory number of connections to be established towards the REST server; default value is eight.
RequestURL	Required; URL of the REST web server including port number. Ensure that you enter IMSI keyword in the URL.
UserName	Required; user name of the REST web server.
Password	Required; password of the REST web server.
KeepAliveTimerInterval	Mandatory time interval, in milliseconds, to send a keepalive to keep the idle connection active; defaults to zero (0) meaning the option is disabled.
RequestToJSONRequestMappings Tab	

Table 5-44 REST Remote Server Properties (continued)

Fields	Description
RESTAttribute	REST attribute
JSONAttribute	JSON attribute to map to the REST attribute.

The following is a sample CLI configuration of REST remote server for extended-EAP service:

```
[ //localhost/Radius/RemoteServers/REST-VM035 ]
    Name = REST-VM035
    Description =
    Protocol = rest
    ReactivateTimerInterval = 30000
    Timeout = 1000
    MaxTimeOuts = 1
    RESTSourceConnections = 1
    RequestURL = https://10.81.79.32:8443/eapauth/IMSI/getdetails
    UserName = eapAuth32TMUS
    Password = <encrypted>
    KeepAliveTimerInterval = 1000
    RequestToJSONRequestMappings/
        IpAddress = Calling-Station-Id
        nasIdentifier = NAS-Identifier
```

SIGTRAN-M3UA

Prime Access Registrar supports SIGTRAN-M3UA to fetch the authentication vectors from HLR, which is required for EAP-AKA/EAP-SIM authentication. For more information on SIGTRAN-M3UA protocol, see the “SIGTRAN-M3UA” chapter of the *Cisco Prime Access Registrar 8.0 User Guide*.

Rules

A Rule is a function that selects services based on all input information used by the function.

Fast Rules

FastRules provides a mechanism to easily choose the right authentication, authorization, accounting, and query service(s), drop, reject, or break flows, run a script, choose a session manager and/or a chain of fast rules required for processing a packet. For more information, see the “Using FastRules to Process Packet Flow” chapter of the *Cisco Prime Access Registrar 8.0 User Guide*.

Advanced

Advanced objects let you configure system-level properties and the Attribute dictionary. Under normal system operation, you should not need to change the system-level properties.

Note The notation *required* means Cisco Prime Access Registrar needs a value for this property. For most of these properties, you can use system defaults.

Table 5-45 lists the **Advanced** properties.

Table 5-45 Advanced Object Properties

Property	Description
LogServerActivity	Required; the default is FALSE, which means Cisco Prime Access Registrar logs all responses except Access-Accepts and Access-Challenges. Accepting the default reduces the load on the server by reducing that amount of information it must log. Note, the client is probably sending accounting requests to an accounting server, so the Access-Accept requests are being indirectly logged. When you set it to TRUE, Cisco Prime Access Registrar logs all responses to the server log file.
TLSv1Enabled	Applicable only for Diameter; Set to TRUE to use TLS version 1.0 and above for Diameter connection. Set to FALSE to use TLS version greater than 1.0 for Diameter connection.
MaximumNumberOfRadiusPackets	Required; the default is 8192. This is a critical property you should set high enough to allow for the maximum number of simultaneous requests. When more requests come in than there are packets allocated, Cisco Prime Access Registrar will drop those additional requests.
UDPPacketSize	Required; the default is 4096. RFC 2138 specifies the maximum packet length can be 4096 bytes. Do not change this value.

Table 5-45 Advanced Object Properties (continued)

Property	Description
NumberOfRemoteUDPServerSockets	Required; the default value for this property is 4. The NumberOfRemoteUDPServerSockets property allows you to configure the number of source ports used while proxying requests to a remote radius server. If the NumberOfRemoteUDPServerSockets property is set to a value *n*, all remote servers share and use *n* sockets. The NumberOfRemoteUDPServerSockets value comprises a number, as in *n*, where *n* should be less than or equal to the current process file descriptor limit divided by 4. **Note** By default, the RADIUS process supports up to 1024 file descriptors. To increase the file descriptors, stop the arserver; in the arserver script, specify the required value to "NUMBER_OF_FILE_DESCRIPTORS" and restart the server. The value for "NUMBER_OF_FILE_DESCRIPTORS" should be in the range between 1024 to 65535.
NumberofRadiusIdentifiersPerSocket	This represents the number of RADIUS Identifiers that Prime Access Registrar can use per source port, while proxying requests to remote servers. To use a different source port for every request that is proxied, you need to set the value of this property to one.
MemoryLimitForRadiusProcess	This property is used to avoid crashing of the radius process. The default value is 3500 Megabytes. This property is under **/radius/advanced**. When the radius process uses memory more than the configured limit, further sessions are not created and Prime Access Registrar rejects further incoming requests.
MemorySizeCheckInterval	This property is used to avoid crashing of the radius process. This is used in conjunction with **MemoryLimitForRadiusProcess**. The default value is 5 minutes. **MemorySizeCheckInterval** is a hidden parameter in mcd database. To modify the default value, you need to export the mcd database. Typically, a separate thread is created to monitor the radius process memory usage for every 5 minutes.

Table 5-45 Advanced Object Properties (continued)

Property	Description
PerPacketHeapSize	Required; the default is 6500. This property sets the size of the initial heap for each packet. The heap is the dynamic memory a request can use during its lifetime. By preallocating the heap size at the beginning of request processing, we can minimize the cost of memory allocations. If PerPacketHeapSize is too low, Prime Access Registrar will ask the system for memory more often. If PerPacketHeapSize is too high, Prime Access Registrar will allocate too much memory for the request causing the system to use more memory than required.
RequireNASsBehindProxyBeInClientList	Required; the default is FALSE. If you accept the default, Cisco Prime Access Registrar only uses the source IP address to identify the immediate client that sent the request. Leaving it FALSE is useful when this RADIUS Server should only know about the proxy server and should treat requests as if they came from the proxy server. This might be the case with some environments that buy bulk dial service from a third party and thus do not need to, or are unable to, list all of the NASs behind the third party's proxy server. When you set it to TRUE, you must list all of the NASs behind the Proxy in the Clients list. For more information about this property, see Using the RequireNASsBehindProxyBeInClientList Property, page 5-82.
AAAFileServiceSyncInterval	Required; specified in milliseconds, the default is 75. This property governs how often the file AAA service processes accounting requests and writes the accounting records to the file. You can lower the number to reduce the delay in acknowledging the **Account-Request** at the expense of more frequent flushing of the accounting file to disk. You can raise the number to reduce the cost of flushing to disk, at the expense of increasing the delays in acknowledging the **Accounting-Request**s. The default value was determined to provide a reasonable compromise between the two alternatives.
SessionBackingStoreSyncInterval	Required; specified in milliseconds, the default is 100. If you change this value it must be a number greater than zero. This property governs how often the Session Manager backing store writes updated session information to disk. You can lower the number to reduce the delay in acknowledging requests at the expense of more frequent flushing of the file containing the session data to disk. You can raise the number to reduce the cost of flushing to disk at the expense of increasing delays in acknowledging requests. The default value was determined to provide a reasonable compromise between the two alternatives.

Table 5-45 Advanced Object Properties (continued)

Property	Description
BackingStoreDiscThreshold	Required; the default is 10 gigabytes. The value of BackingStoreDisc- Threshold is made up of a number of units which can be K, kilobyte, or kilobytes, M, megabyte, or megabytes, or G, gigabyte, or gigabytes. BackingStoreDiscThreshold is used with session management and ODBC accounting and ensures that any data log files generated will not cross the BackingStoreDiscThreshold.
SessionBackingStorePruneInterval	Required; specifies the sleep time interval of the session backing store pruning thread. The recommended and default value is 6 hours, but you can modify this based on the traffic patterns you experience. With SessionBackingStorePruneInterval set to 6 hours, pruning will occur 6 hours after you restart or reload the Prime Access Registrar server and recur every 6 hours. You can set a very low value for this property to make pruning continuous, but there might not be enough data accumulated for the pruning to occur and pruning might be less effective compared to the default setting.
PacketBackingStorePruneInterval	Required; specifies the sleep time interval of the packet backing store pruning thread. The recommended value is 6 hours, but you can modify this based on the traffic patterns you experience. When PacketBackingStorePruneInterval is set to 6 hours, pruning will occur 6 hours after you restart or reload the Prime Access Registrar server and recur every 6 hours. You can set a very low value for this property to make pruning continuous, but there might not be enough data accumulated for the pruning to occur and pruning might be less effective compared to the default setting.
RemoteLDAPServerThreadTimerInterval	Required; specified in milliseconds, the default is 10. This property governs how often the **ldap** RemoteServer thread checks to see if any results have arrived from the remote LDAP server. You can modify it to improve the throughput of the server when it proxies requests to a remote LDAP server.
RemoteSigtranServerThreadTimerInterval	Required; specified in milliseconds, the default is 10. This property governs how often the sigtran RemoteServer thread checks to see if any results have arrived from the remote HLR/AuC server. You can modify it to improve the throughput of the server when it proxies requests to a remote sigtran server.

Table 5-45 Advanced Object Properties (continued)

Property	Description
InitialBackgroundTimerSleepTime	Required; the default is 5. This property specifies the amount of time the time queue should initially sleep before beginning processing. This property is only used for initial synchronization and should not be changed.
MinimumSocketBufferSize	Required; the default is 65536 (64 K). This property governs how deep the system's buffer size is for queueing UDP datagrams until Cisco Prime Access Registrar can read and process them. The default is probably sufficient for most sites. You can, however, raise or lower it as necessary.
CertificateDBPath	Required if you are using an LDAP RemoteServer and you want Prime Access Registrar to use SSL when communicating with that LDAP RemoteServer. This property specifies the path to the directory containing the client certificates to be used when establishing an SSL connection to an LDAP RemoteServer. This directory must contain the **cert7.db** and **cert5.db** certificates and the **key3.db** and **key.db** files database used by Netscape Navigator 3.x (and above) or the **ServerCert.db** certificate database used by Netscape 2.x servers.
LogFileSize	Required; the default is 1 Megabyte. This property specifies the maximum size of the RADIUS server log file. The value for the **LogFileSize** field is a string composed of two parts; a number, and a units indicator (<n> <units>) in which the unit is one of: K, kilobyte, kilobytes, M, megabyte, megabytes, G, gigabyte, or gigabytes. The **LogFileSize** property does not apply to the **config_mcd_1_log** or **agent_server_1_log** files. See Cisco Prime Access Registrar 8.0 Reference Guide to configure these files. **Note** This does not apply to the trace log.
LogFileCount	Required; the default is 2. This property specifies the number of log files to be kept on the system. A new log file is created when the log file size reaches **LogFileCount**. The **LogFileCount** property does not apply to the **config_mcd_1_log** or **agent_server_1_log** files. See Cisco Prime Access Registrar 8.0 Reference Guide to configure these files.
TraceFileSize	Required; the default is 1 GB. This property specifies the size of the trace files to be kept on the system. A new trace file is created when the trace file size reaches **TraceFileSize**. The value for the **TraceFileSize** field is a string composed of two parts; a number, and a units indicator (<n> <units>) in which the unit is one of: K, kilobyte, kilobytes, M, megabyte, megabytes, G, gigabyte, or gigabytes.

Table 5-45 Advanced Object Properties (continued)

Property	Description
TraceFileCount	Required; this value can be set from 1-100, and the default is 2. This property specifies the number of trace files to maintain. A value of 1 indicates that no file rolling occurs.
UseAdvancedDuplicateDetection	Required; the default is FALSE. Set this property to TRUE when you want Cisco Prime Access Registrar to use a more robust duplicate request filtering algorithm. For more information on this property, see Advance Duplicate Detection Feature, page 5-83.
AdvancedDuplicateDetectionMemoryInterval	Required when the Advanced Duplicate Detection feature is enabled. This property specifies how long (in milliseconds) Cisco Prime Access Registrar should remember a request. You must specify a number greater than zero. The default is 10,000.
DetectOutOfOrderAccountingPackets	Optional; used to detect accounting packets that arrive out of sequential order. The default is FALSE. This property is useful when using accounting and session management in a RADIUS proxy service. When the DetectOutOfOrderAccountingPacket property is enabled (set to TRUE), a new *Class* attribute is included in all outgoing Accept packets. The value for this Class attribute will contain the session magic number. The client will echo this value in the accounting packets, and this will be used for comparison. The session magic number is a unique number created for all sessions when the session is created or reused and the DetectOutOfOrderAccountingPacket property is set to TRUE. The DetectOutOfOrderAccountingPacket property is used to detect out-of-order Accounting-Stop packets in roaming scenarios by comparing the session magic number value in the session with the session magic number value contained in the Accounting packet. The value of 0xffffffff is considered by the Prime Access Registrar server to be a wild card magic number. If any accounting stop packets contain the value of 0xffffffff, it will pass the session magic validation even if the session's magic number is some thing else. The format of the class attribute is as follows: <4-byte Magic Prefix><4-byte server IP address><4-byte Magic value>
DefaultReturnedSubnetSizeIfNoMatch	Optional; used with the ODAP feature and reflects the returned size of the subnet if no matched subnet is found. There are three options to select if an exactly matched subnet does not exist: Bigger, Smaller, and Exact. The default is Bigger.

Table 5-45 Advanced Object Properties (continued)

Property	Description
ClasspathForJavaExtensions	A string which is the classpath to be used to locate Java classes and jar files containing the classes required for loading the Java extensions, either Java extension points or services. **Note** The classpath will always contain the directory **$INSTALLDIR/scripts/radius/java** and all of the jar files in that directory.
JavaVMOptions	A string that can contain options to be passed to the JRE upon startup. JavaVMOptions should be used only when requested by Cisco TAC.
MaximumODBCResultSize	Specifies maximum size in bytes for an ODBC mapping. This parameter affects both ODBC result sizes and the trace log buffer for tracing script calls that access any of the dictionaries. (Default value is 256.)
ARIsCaseInsensitive	When set to FALSE, requires that you provide exact pathnames with regard to upper and lower case for all objects, subobjects, and properties. The default setting, TRUE, allows you to enter paths such as **/rad/serv** instead of **/Rad/Serv**. **Note** Prime Access Registrar always authenticates the RADIUS attribute User-Name with regard to upper and lower case, regardless of the setting of this flag.
RemoteRadiusServerInterface	When set, specifies the local interface to bind to when creating the RemoteRadiusServer socket. If not set, the Prime Access Registrar binds to IPADDR_ANY.
ODBCEnvironmentMultiValueDelimiter	Optional; allows you to specify a character that separates multivalued attributes in the marker list when using ODBC accounting
PacketBackingStoreSyncInterval	The minimum value is 1 and the maximum is a 32-bit unsigned integer. The default is 75.
ListenForDynamicAuthorizationRequests	Must be set to TRUE when using the Change of Authorization (CoA) feature or Packet of Disconnect (POD) feature. Default is FALSE.
MaximumNumberOfXMLPackets	Required when using identity caching. Indicates the maximum number of XML packets to be sent or received. The minimum value is 1 and the maximum is a 32-bit unsigned integer. The default is 1024.
XMLUDPPacketSize	Required when using identity caching. Indicates the maximum size of XML packets to be sent or received. The minimum value is 1 and the maximum is a 32-bit unsigned integer. The default is 4096.

Table 5-45 Advanced Object Properties (continued)

Property	Description
RollingEncryptionKeyChangePeriod	Used in conjunction with the session-cache ResourceManager, this property specifies the length of time a given EncryptionKey will be used before a new one is created. When the session-cache ResourceManager caches User-Password attributes, Prime Access Registrar encrypts the User-Password so it is not stored in memory or persisted on disk in clear text. Prime Access Registrar uses up to 255 encryption keys, using a new one after each RollingEncryptionKeyChangePeriod expires. If RollingEncryptionKeyChangePeriod is set to *2 days*, Prime Access Registrar will create and begin using a new EncryptionKey every two days. The oldest key will be retired, and Prime Access Registrar will re-encrypt any User-Passwords that used the old key with the new key. This way, if the RollingEncryptionKeyChangePeriod is set to *1 day*, no key will be older than 255 days.
SessionPurgeInterval	Optional; the SessionPurgeInterval property determines the time interval at which to check for timed-out sessions. If no value is set, the session timeout feature is disabled. The checks are performed in the background when system resources are available, so checks might not always occur at the exact time set. The minimum recommended value for SessionPurgeInterval is 60 minutes. The SessionPurgeInterval value is comprised of a number and a units indicator, as in n units, where a unit is one of minutes, hours, days, or weeks.
EapBadMessagePolicy	Set to one of two values: SilentDiscard (the default) or RejectFailure. When set to SilentDiscard, the Prime Access Registrar server silently discards and ignores bad EAP messages unless the protocol specification explicitly requires a failure message. When set to RejectFailure, the Prime Access Registrar server sends RADIUS Access-Rejects messages with embedded EAP-Failure in response to bad EAP messages as described in Internet RFC 3579.

Table 5-45 Advanced Object Properties (continued)

Property	Description
StaleSessionTimeout	Required; the default value is "1 hour." Specifies the time interval to maintain a session when a client does not respond to Accounting-Stop notification. When the Prime Access Registrar server does not receive an Accounting-Response from a client after sending an Accounting-Stop packet, Prime Access Registrar maintains the session for the time interval configured in this property before releasing the session. This property is stored as a string composed of two parts: a number and a unit indicator (<n> <units>) similar to the MaxFileAge property where the unit is one of: M, Minute, Minutes, H, Hour, Hours, D, Day, Days, W, Week, or Weeks.
Ports	Optional; allows you to use ports other than the default, 1812 and 1813. You can use this option to configure Prime Access Registrar to use other ports,. If you add additional ports, however, Prime Access Registrar will use the added ports and no longer use ports 1812 and 1813. These ports can still be used by adding them to the list of ports to use. For more information, see Ports, page 5-84.
Interfaces	Optional; see Interfaces, page 5-84
ReplyMessages	Optional; see Reply Messages, page 5-84.
AttributeDictionary	Optional; see Attribute Dictionary, page 5-86.
SNMP	Optional; see SNMP, page 5-87.
RFC Compliance	Optional; enables you to modify the Prime Access Registrar server to behave in a way that might deviate from RFC compliance in a special use case scenario. When AllowRejectAttrs is set to FALSE, Reply-Message attributes will not be passed in an Access Reject packet. When AllowRejectAttrs is set to TRUE, attributes will be allowed to pass in an Access Reject packet. When AllowEAPRejectAttrs is set to FALSE, Reply-Message attributes will not be passed in an Access Reject packet if the packet contains EAP-Message attribute. When AllowEAPRejectAttrs is set to TRUE, attributes will be allowed to pass in an Access Reject packet even if the packet contains EAP-Message attribute. **Note** Changing the state of either of these properties requires you to **reload** the Prime Access Registrar server.
DDNS	This subdirectory holds the SynthesizeReverseZone property and a list of Transaction Signatures (TSIG) keys.

Table 5-45 Advanced Object Properties (continued)

Property	Description
SynthesizeReverseZone	This property exists under DDNS and controls whether Prime Access Registrar automatically generates the name of the reverse zone (**in-addr.arpa**) that is updated with PTR records. If this attribute is enabled and the resource manager does not have an explicit ReverseZoneName property configured, the server uses the IP address and DNSHostBytes property to generate the reverse zone name. The default value is TRUE.
ODBCDataSources	A list of ODBC data sets and their associated environments including operating system, DBMS, and network platform used to access the DBMS an application wants to access. Required when using or ODBC accounting.
AttributeGroups	Includes a Default subdirectory with an Attributes subdirectory that contains commonly-used attributes for Change of Authorization (CoA) and Packet of Disconnect (POD). You can add new attributes to the default group or create a new group as necessary.
KeyStores	Used to protect the security and integrity of the PACs it issues. • NumberOfKeys—Number (from 1-1024) that specifies the maximum number of keys stored for EAP-FAST. • RolloverPeriod—Specifies the amount of time between key updates.
DefaultRadiusSharedSecret	Enter the default shared secret for RADIUS server.
MaximumIncomingRequestRate	Optional; the default value for this property is 0. The MaximumIncomingRequestRate property is used to limit the incoming traffic in terms of “allowed requests per second”. Serves as a soft limit. The MaximumIncomingRequestRate property comprises a number *n*, where *n* can be any nonzero value.

Table 5-45 Advanced Object Properties (continued)

Property	Description
HideSharedSecretAndPrivateKeys	Required; the default value is TRUE. The HideSharedSecretAndPrivateKeys property hides: • The secret that is shared between a RADIUS Client and a RADIUS Server or between two radius servers in a radius proxy scenario. • The PrivateKeyPassword under the certificate-based EAP services. When this property is set to TRUE, the following properties are displayed as <encrypted>: • PrivateKeyPasswords in: – peap-v0 service – peap-v1 service – eap-tls service – eap-ttls service – eap-fast service • SharedSecret in: – RemoteServers of type radius – RemoteServers of type map-gateway – Clients object – Resource Manager of type usr-vpn under Gateway subobject • PseudonymSecret in eap-sim service • DynamicAuthSecret under DynamicAuthorizationServer subbject in Clients object • RepSecret under Replication • Secret in /radius/advanced/DDNS/TSIGKeys When the value for this property is set to FALSE, all the above properties are displayed in clear text.
MaximumOutstandingRequests	Optional; the default value for this property is 0. The MaximumOutstandingRequests property is used to limit the incoming traffic in terms of “requests processed”. Serves as a hard limit. The MaximumOutstandingRequests property comprises a number *n*, where *n* can be any nonzero value.
Diameter	Required; See Diameter, page 5-88

Table 5-45 Advanced Object Properties (continued)

Property	Description
TPSSamplingPeriodInSecs	This represents the sampling period in seconds. The minimum sampling period is set to 5. The default is 30.
LogTPSActivity	When set to true this property enables to log in the TPS usage in a CSV file.The TPS is logged in the following format: <*mm-dd-yyyy*>, <*hh:mm:ss*>, <*tps-value*>, <*sigtran-m3ua traffic value*> For example, `04-24-2014,18:36:30,2998,1000` The default is False.
TPSLogFilenamePrefix	This represents the prefix of the CSV file which will be available in the logs directory of Prime Access Registrar. The following represents the CSV filename format: <*user-prefix*>-<*mm-dd-yyyy*>.csv tps-04-01-2013.csv
TPSLogFileCount	Configures the number of TPS Sampling log files to be maintained in the repository. The default value is 2.
LogSessionActivity	When set to TRUE, this property enables Prime Access Registrar to log the session count in the server.
EnableLengthFlag	Set to TRUE to enable the length flag.
SessionLogFileCount	Required only if you set LogSessionActivity to TRUE; the number of session log files to maintain in the repository. The default value is 2.
SessionLogFileNamePrefix	Required only if you set LogSessionActivity to TRUE; this represents the prefix of the session log file which will be available in the logs directory of Prime Access Registrar.
SessionSamplingPeriodInSecs	Required only if you set LogSessionActivity to TRUE; this represents the session sampling period in seconds. The minimum sampling period is set to 5. The default is 30.
FlushDiskInBackground	Set to TRUE to allow Prime Access Registrar to flush the accounting record to disk before it acknowledges the request packets.

Table 5-45 Advanced Object Properties (continued)

Property	Description
AdditionalNativeOracleErrors	Optional; used to disconnect ODBC Remote Servers when configured native Oracle Error has occurred (which are not considered as connection errors). You must specify Native Errors as comma (,) separated integer values. For example, 04/14/2013 11:06:43.692: Log: ODBC client (DataSource 'CVOracleAcctDb', Connection 6): SQLExecute failed: SQLState:HY000 NativeError:12152 ErrorText:[Easysoft][Oracle]ORA-12152: TNS:unable to send break message 04/14/2013 10:44:59.388: Log: ODBC client (DataSource 'CVOracleAcctDb', Connection 3): SQLExecute failed: SQLState:HY000 NativeError:3114 ErrorText:[Easysoft][Oracle]ORA-03114: not connected to ORACLE For the above examples, the Native Errors need to be configured as follows: **--> set AdditionalNativeOracleErrors 12152,3114** When any one of the Native Errors 12152 or 3114 occurs, Prime Access Registrar disconnects the ODBC Remote Server.
SendOpCodeInISDResponse	Set to TRUE to send operator code in the ISD response.
EnableRoutingContextInM3UA	Set to TRUE to enable routing context in M3UA.
EnableSIGTRANStackLogs	When set to TRUE, this property enables to log the SIGTRAN stack logs in stack.log file.
SIGTRANStackLogFileSize	Required if you set EnableSIGTRANStackLogs to TRUE. This property specifies the maximum size (in megabyte) of the SIGTRAN stack log file.
SIGTRANLogFileCount	Required if you set EnableSIGTRANStackLogs to TRUE. This value can be set from 1–100, and the default is 10. This property specifies the number of SIGTRAN log files to maintain in the repository.
EnableStickySessionCount	Required; either True or False and the default value is True. When set to True, Prime Access Registrar displays the peer specific stats showing the number of sticky sessions associated with a peer for Diameter proxy service in name_radius_log file.
StickySessionCountInterval	Required; specified in milliseconds and the default is 60000. When the EnableStickySessionCount is set to True, this field specifies how often the Diameter proxy service will display the number of sticky sessions associated with a peer.

Table 5-45 Advanced Object Properties (continued)

Property	Description
StickySessionSyncInterval	Required; specified in milliseconds and the default value is 500. Specifies how often the Diameter proxy service will write the sticky sessions to a file located in /cis-co-ar/temp/__sticky_sessions_store location.
ReserveRADIUSPacketPool	Percentage of the RADIUS packet pool to reserve for the RADIUS remote server responses.
EnableLocationCapability	Check the box to enable location-based attributes within RADIUS and Diameter that can be used to convey loca-tion-related information for authentication and accounting exchanges. If this parameter is set to TRUE, Prime Access Registrar retrieves the location information from the client and processes the incoming packet for AA services. For more information on location information delivery flows, refer to RFC 5580. For information on location-based attributes in Prime Access Registrar, see the "Environment Dictionary" chapter of the *Cisco Prime Access Registrar 8.0 Reference Guide.*
DiameterSessionRestorationPurgeTime	The time at which Prime Access Registrar must run the Diameter session restoration process. Format is HH:MM:SS (24 hrs format) and default value is 02:00:00. Recommended time is when the incoming traffic is minimal. **Note** This time should always be two hours behind the Diameter stale session purge time.
DiameterStaleSessionPurgeTime	The time at which Prime Access Registrar must check for Diameter stale sessions. Format is HH:MM:SS (24 hrs format) and default value is 00:00:00. Recommended time is when the incoming traffic is minimal.

The following CLI shows an example configuration of Advanced properties:

```
[ //localhost/Radius/Advanced ]
    LogServerActivity = FALSE
    TLSv1Enabled = TRUE
    MaximumNumberOfRadiusPackets = 8192
    UDPPacketSize = 4096
    NumberOfRemoteUDPServerSockets = 4
    NumberOfRadiusIdentifiersPerSocket = 256
    PerPacketHeapSize = 6500
    RequireNASsBehindProxyBeInClientList = FALSE
    AAAFileServiceSyncInterval = 75
    SessionBackingStoreSyncInterval = 100
    BackingStoreDiscThreshold = "5 Gigabyte"
    SessionBackingStorePruneInterval = "6 Hours"
    PacketBackingStorePruneInterval = "6 Hours"
    RemoteLDAPServerThreadTimerInterval = 10
    RemoteSigtranServerThreadTimerInterval = 10
```

```
 InitialBackgroundTimerSleepTime = 5
 MinimumSocketBufferSize = 65536
 CertificateDBPath =
 LogFileSize = "1 Megabyte"
 LogFileCount = 2
 TraceFileSize = "1 Gigabyte"
 TraceFileCount = 2
 MemoryLimitForRadiusProcess = "3584 Megabyte"
 UseAdvancedDuplicateDetection = FALSE
 AdvancedDuplicateDetectionMemoryInterval = 10000
 DetectOutOfOrderAccountingPackets = FALSE
 DefaultReturnedSubnetSizeIfNoMatch = BIGGER
 ClasspathForJavaExtensions =
 JavaVMOptions =
 MaximumODBCResultSize = 256
 ARIsCaseInsensitive = TRUE
 RemoteRadiusServerInterface =
 ODBCEnvironmentMultiValueDelimiter =
 PacketBackingStoreSyncInterval = 75
 ListenForDynamicAuthorizationRequests = FALSE
 MaximumNumberOfXMLPackets = 1024
 XMLUDPPacketSize = 4096
 RollingEncryptionKeyChangePeriod = "1 week"
 SessionPurgeInterval =
 EapBadMessagePolicy = SilentDiscard
 StaleSessionTimeout = "1 Hour"
 MaximumOutstandingRequests = 0
 MaximumIncomingRequestRate = 0
 HideSharedSecretAndPrivateKeys = TRUE
 DefaultRadiusSharedSecret =
 EnableLocationCapability = FALSE
 LogTPSActivity = TRUE
 TPSLogFileCount = 15
 TPSLogFilenamePrefix = tps
 TPSSamplingPeriodInSecs = 15
LogSessionActivity = TRUE
 EnableLengthFlag = FALSE
 SessionLogFileCount = 15
 SessionLogFilenamePrefix = sm
 SessionSamplingPeriodInSecs = 30
 FlushDiskInBackground = FALSE
 AdditionalNativeOracleErrors =
 SendOpCodeInISDResponse = FALSE
 EnableRoutingContextInM3UA = FALSE
 EnableStickySessionCount = TRUE
 EnableSIGTRANStackLogs = TRUE
 SIGTRANStackLogFileSize = "100 Megabyte"
 SIGTRANLogFileCount = 10
 StickySessionCountInterval = 60000
 StickySessionSyncInterval = 500
 ReserveRADIUSPacketPool = 0
 DiameterStaleSessionPurgeTime = 00:00:00
 DiameterSessionRestorationPurgeTime = 02:00:00
 Ports/
 Interfaces/
 ReplyMessages/
 Attribute Dictionary/
 SNMP/
 ServerMonitor/
 RemoteSessionServer/
 RFCCompliance/
 DDNS/
 ODBCDataSources/
 AttributeGroups/
```

```
KeyStores/
Diameter/
DiameterDictionary/
```

This section contains the following topics:

- RemoteODBCSessionServer
- Using the RequireNASsBehindProxyBeInClientList Property
- Advance Duplicate Detection Feature
- Invalid EAP Packet Processing
- Ports
- Interfaces
- Reply Messages
- Attribute Dictionary
- SNMP
- Diameter

RemoteODBCSessionServer

The following is an example of theRemoteODBCSessionServer configuration:

--> cd /Radius/Advanced/RemoteODBCSessionServer/

```
[ //localhost/Radius/Advanced/RemoteODBCSessionServer ]
    ReactivateTimerInterval = 300000
    Timeout = 15
    DataSourceConnections = 8
    ODBCDataSource =
    KeepAliveTimerInterval = 0
    BufferAccountingPackets = TRUE
    MaximumBufferFileSize = "10 Megabytes"
    CacheLimit = 250000
    UseCacheIndex = 0
```

Table 5-46 lists and defines the RemoteODBCSessionServer properties.

Table 5-46 RemoteODBCSesionServer Properties

Property	Description
ReactivateTimerInterval	Mandatory time interval (in milliseconds) to activate an inactive server; defaults to 300000 ms
Timeout	Mandatory time interval (in seconds) to wait for SQL operation to complete; defaults to 15 seconds
DataSourceConnections	Mandatory number of connections to be established; defaults to 8
ODBCDataSource	Name of the ODBCDataSource to use and must refer to one entry in the list of ODBC datasources configured under **/Radius/Advanced/ODBCDataSources**. Mandatory; no default

Table 5-46 RemoteODBCSesionServer Properties (continued)

Property	Description
KeepAliveTimerInterval	Mandatory time interval (in milliseconds) to send a keepalive to keep the idle connection active; defaults to zero (0) meaning the option is disabled
BufferAccountingPackets	Mandatory, TRUE or FALSE, determines whether to buffer the accounting packets to local file, defaults to TRUE which means that packet buffering is enabled. **Note** When set to TRUE, a constant flow of incoming accounting packets can fill the buffer backing store files in **/cisco-ar/data/odbc** beyond the size configured in MaximumBufferFileSize. Configure BackingStoreDiscThreshold in **/Radius/Advanced** when using ODBC accounting. See Advanced, page 5-65 for information about how to configure BackingStoreDiscThreshold.
MaximumBufferFileSize	Mandatory if BufferAccountingPackets is set to TRUE, determines the maximum buffer file size, defaults to 10 Megabyte)
CacheLimit	Default is 250000; This represents the overall limit on cache of all 'remote' session managers. This value is interpreted as the maximum number of packets that can be present in cache. When the number of sessions hits this limit, sessions will be 'cached out'. This cache out operation will continue, until the cache is at least 20% free.
UseCacheIndex	If set to 1, it enables a fast cache based lookup index for the items in the database. This optimizes the number of queries to the database hence will improve performance, but limits the number of sessions that can be scaled. If set to 0, it disables fast cache based lookup index.

Note Remote session manager will work only with Oracle database.

Using the RequireNASsBehindProxyBeInClientList Property

You can use the property **RequireNASsBehindProxyBeInClientList** to require NASs that send requests indirectly through a proxy to be listed in the Clients list or to allow the proxy to represent them all.

- When you want to ensure the proxy is only sending requests from NASs known to this server, set the property to TRUE, and list all of the NASs using this proxy. This increases memory usage.
- When it is impossible to know all of the NASs using this proxy or when you do not care, set the property to FALSE. Cisco Prime Access Registrar will use the proxy's IP address to identify the origin of the request.

Advance Duplicate Detection Feature

Prime Access Registrar automatically detects and handles duplicate requests it is currently working on. It also provides an optional, more complex mechanism to handle duplicate requests that can be received by the server after it has completed processing the original request. These duplicate requests can consume extra processing power, and, if received out of order (as RADIUS is a UDP-based protocol) might cause Session Management problems.

One solution is the Advanced Duplicate Detection feature which causes Prime Access Registrar to *remember* requests it has seen, as well as the response sent to that request, for a configurable amount of time.

To enable this feature, perform the following:

- Set the **UseAdvancedDuplicateDetection** property in the **/Radius/Advanced** section of the configuration to **TRUE**.
- Set the **AdvancedDuplicateDetectionMemoryInterval** in the **/Radius/Advanced** section to specify how long (in milliseconds) Prime Access Registrar should remember a request.

Note Enabling this feature causes Cisco Prime Access Registrar to keep more of its preallocated packet buffers in use for a longer period of time. The number of preallocated buffers is controlled by the **MaximumNumberOfRadiusPackets** property in the **/Radius/Advanced** section of the configuration. This property might need to be increased (which will increase the amount of memory used by Cisco Prime Access Registrar) when the Advanced Duplicate Detection feature is enabled.

Invalid EAP Packet Processing

Prime Access Registrar has been enhanced to implement *fatal error* packet handling for Extensible Authentication Protocol (EAP) messages as described in section 2.2 of Internet RFC 3579 which states the following:

> A RADIUS server determining that a fatal error has occurred must send an Access-Reject containing an EAP-Message attribute encapsulating EAP-Failure.

Because this enhancement is a deviation from various EAP specifications, you must explicitly enable this feature through a new configuration property in **/Radius/Advanced** named *EapBadMessagePolicy*.

You can set the EapBadMessagePolicy property to one of two values: SilentDiscard (the default) or RejectFailure. When set to SilentDiscard, the Prime Access Registrar server silently discards and ignores bad EAP messages unless the protocol specification explicitly requires a failure message. When set to RejectFailure, the Prime Access Registrar server sends RADIUS Access-Rejects messages with embedded EAP-Failure in response to bad EAP messages as described in Internet RFC 3579.

The implementation of EAP authentication methods in Prime Access Registrar behaves as described in Internet RFC 2284 (EAP) and related EAP method specifications. These specify *silent discard* as the standard way to handle all EAP error conditions. Any EAP response message from the client that contains an error or is received in an invalid authenticator state is discarded and there is no error response.

In a configuration where EAP requests are proxied between RADIUS servers using RADIUS messages (EAP over RADIUS), the silent discard of an EAP message means that no RADIUS response message is sent back to the originating RADIUS server. Because of this, the RADIUS server originating the request eventually declares the destination RADIUS server *dead* and fails over to a backup server (if so configured).

Ports

The Ports list specifies which ports to listen to for requests. When you specify a port, Cisco Prime Access Registrar makes no distinction between the port used to receive Access-Requests and the port used to receive Accounting-Requests. Either request can come in on either port.

Most NASs send Access-Requests to port 1812 and Accounting-Requests to 1813, however, Cisco Prime Access Registrar does not check.

When you do not specify any ports, Cisco Prime Access Registrar reads the **/etc/services** file for the ports to use for access and accounting requests. If none are defined, Prime Access Registrar uses the standard ports (1812 and 1813).

Interfaces

The Interfaces list specifies the interfaces on which the RADIUS server receives and sends requests. You specify an interface by its IP address.

- When you list an IP address, Cisco Prime Access Registrar uses that interface to send and receive Access-Requests.
- When no interfaces are listed, the server performs an interface discover and uses all interfaces of the server, physical and logical (virtual).

Note The IP address format is enhanced to support both IPv4 and IPv6.

Reply Messages

The Reply Messages list allows you to choose the reply message based on the reason the request was rejected. Each of the following properties (except **Default**) corresponds to a reason why the packet was rejected. The Reply Message properties allows you to substitute your own text string for the defined errors. After you set the property (with the **set** command) and the reason occurs, Cisco Prime Access Registrar sends the NAS that message in the Access-Reject packet as a **Reply-Message** attribute.

You might want to substitute your own messages to prevent users from getting too much information about why their requests failed. For example, you might not want users to know the password was invalid to prevent hackers from accessing your system. In such a case, you might specify the text string "`unauthorized access`" for the property **UserPasswordInvalid**.

Table 5-47 lists the **Reply Message** properties.

Table 5-47 Reply Message Properties

Property	Description
Default	Optional; when you set this property, Cisco Prime Access Registrar sends this value when the property corresponding to the reject reason is not set.
UnknownUser	Optional; when you set this property, Cisco Prime Access Registrar sends back this value in the **Reply-Message** attribute whenever Cisco Prime Access Registrar cannot find the user specified by **User-Name**.
UserNotEnabled	Optional; when you set this property, Cisco Prime Access Registrar sends back this value in the **Reply-Message** attribute whenever the user account is disabled.
UserPasswordInvalid	Optional; when you set this property, Cisco Prime Access Registrar sends back this value in the **Reply-Message** attribute whenever the password in the Access-Request packet did not match the password in the database.
UnableToAcquireResource	Optional; when you set this property, Cisco Prime Access Registrar sends back this value in the **Reply-Message** attribute whenever one of the Resource Managers was unable to allocate the resource for this request.
ServiceUnavailable	Optional; when you set this property, Cisco Prime Access Registrar sends back this value in the **Reply-Message** attribute whenever a service the request needs (such as a RemoteServer) is unavailable.
InternalError	Optional; when you set this property, Cisco Prime Access Registrar sends back this value in the **Reply-Message** attribute whenever an internal error caused the request to be rejected.
MalformedRequest	Optional; when you set this property, Cisco Prime Access Registrar sends back this value in the **Reply-Message** attribute whenever a required attribute (such as **User-Name**) is missing from the request.
ConfigurationError	Optional; when you set this property, Cisco Prime Access Registrar sends back this value in the **Reply-Message** attribute whenever the request is rejected due to a configuration error. For example, if a script sets an environment variable to the name of an object such as **Authentication-Service**, and that object does not exist in the configuration, the reason reported is ConfigurationError.
IncomingScriptFailed	Optional; when you set this property, Cisco Prime Access Registrar sends back this value in the **Reply-Message** attribute whenever one of the **IncomingScripts** fails to execute.
OutgoingScriptFailed	Optional; when you set this property, Cisco Prime Access Registrar sends back this value in the **Reply-Message** attribute whenever one of the **s** fails to execute.
IncomingScriptRejectedRequest	Optional; when you set this property, Cisco Prime Access Registrar sends back this value in the **Reply-Message** attribute whenever one of the **IncomingScripts** rejects the Access-Request.

Table 5-47 Reply Message Properties (continued)

Property	Description
OutgoingScriptRejectedRequest	Optional; when you set this property, Cisco Prime Access Registrar sends back this value in the **Reply-Message** attribute whenever one of the **s** rejects the Access-Request.
TerminationAction	Optional; when you set this property, Cisco Prime Access Registrar sends back this value in the **Reply-Message** attribute whenever Cisco Prime Access Registrar processes the Access-Request as a Termination-Action and is being rejected as a safety precaution.

Attribute Dictionary

The Attribute dictionary allows you to specify the attributes to the RADIUS server.
Cisco Prime Access Registrar comes with the standard RADIUS attributes (as defined by the RFC 2865) as well as the attributes required to support the major NASs. For more information about the standard attributes, see "RADIUS Attributes" chapter of the *Cisco Prime Access Registrar 8.0 Reference Guide.*

All RADIUS requests and responses consist of one or more *attributes*, such as the user's name, the user's password, the type of service the NAS should provide to the user, or the IP address the user should use for the session.

In the request and response packets, an attribute is composed of a number (between 1-255) that specifies the type of attribute to use, a length that specifies the entire attribute length, and a value. How the value is interpreted depends on its type. When it is a username, the value is a string. When it is the NAS's IP address, the value is an IP address, and so on.

Table 5-48 lists the Attribute dictionary properties.

Table 5-48 Attribute Dictionary Properties

Property	Description
Name	Required; must be unique in the Attribute dictionary list within the same context. Although it should be an attribute defined in the RFC, the name can be any attribute defined by your client. The NAS typically comes with a list of attributes it uses. Attributes are referenced in the Profile and by Scripts by this name. The accounting file service also uses this name when printing the attribute.
Description	Optional description of the attribute.
Attribute	Required; must be a number between 1-255. It must be unique within the Attribute dictionary list.
Type	Required; must be set to one of the types listed in Table 5-49. The type governs how the value is interpreted and printed.

Types

Types are required and must be one of the following listed in Table 5-49.

Table 5-49 Types Attributes

Property	Description
UNDEFINED	Treated as a sting of binary bytes.
UINT32	Unsigned 32-bit integer.
STRING	Character string.
IPADDR	A valid IP address in dotted-decimal format.
CHAP_PASSWORD	17-byte value representing the password.
ENUM	Enums allow you to specify the mapping between the value and the strings. After you have established this mapping, Cisco Prime Access Registrar then replaces the number with the appropriate string. The min/max properties represent the lowest to highest values of the enumeration.
VENDOR_SPECIFIC	Vendor Specific Attribute (VSAs) are a special class of attribute. VSAs were created to extend the standard 256 attributes to include attributes required by specific manufacturers. VSAs add new capabilities for the value field in an attribute. Rather than being a simple integer string, or IP address, the value of a VSA can be one or more subattributes whose meaning depends on the vendor's definition. The Vendors list allows you to add, delete, or modify the definitions of the vendors and the subattributes they specify.

Vendor Attributes

Table 5-50 lists the **Vendor** properties.

Table 5-50 Vendor Properties

Property	Description
Name	Required; must be unique in the Vendors attribute list.
Description	Optional; description of the subattribute list.
VendorID	Required; must be a valid number and unique within the entire attribute dictionary.
Type	Required; must be one of the following: UNDEFINED, UINT32, STRING, IPADDR, CHAP_PASSWORD, ENUM, or SUB_ATTRIBUTES.

SNMP

Table 5-51 lists the five properties of the SNMP directory.

Table 5-51 SNMP Properties

Property	Description
Enabled	Either TRUE or FALSE; default is FALSE
TracingEnabled	Either TRUE or FALSE; default is FALSE
InputQueueHighThreshold	An integer; default is 90

Table 5-51 SNMP Properties (continued)

Property	Description
InputQueueLowThreshold	An integer; default is 60
MasterAgentEnabled	Either TRUE or FALSE; default is TRUE

If Enabled and MasterAgentEnabled are both TRUE, **arservagt** will start and stop the SNMP daemon (**snmpd**). If either of these properties is FALSE, if the Prime Access Registrar server is not using SNMP or if your site uses a different master agent, **arservagt** will not start your master agent.

Diameter

This section explains how to configure Diameter general configuration and transport management.

Change the directory to **/Radius/Advanced/Diameter**.

```
//localhost/Radius/Advanced/Diameter
    General/
    TransportManagement/
```

The following configuration is used to configure Diameter general configuration like Product name and Version.

```
[ //localhost/Radius/Advanced/Diameter/General ]
    Product = Cisco Prime Access Registrar
    Version = 7.2.0.0
    AuthApplicationIdList = 1
    AcctApplicationIdList = 3
```

Table 5-52 describes the Diameter general properties.

Table 5-52 Diameter General Properties

Property	Description
Product	Optional; name of the product.
Version	Optional; version number.

Table 5-52 Diameter General Properties (continued)

Property	Description
AuthApplicationIdList	Specifies the list of AuthApplications that the Prime Access Registrar server registers to Diameter Base stack during start up. It is a combination of Auth ApplicationId's separated by a colon. For example: For Registering NASREQApplication, **--> set AuthApplicationIdList 1** For Registering applications with id's 1 and 5, **--> set AuthApplicationIdList 1:5** **Note** TheAuthApplicationIdsthatareconfiguredshouldbepresentin **/Radius/Advanced/Diameter/Applications** section.
AcctApplicationIdList	Specifies the list of AcctApplications that the Prime Access Registrar server registers to Diameter Base stack during start up. It is a combination of Acct ApplicationId's separated by a colon. For example: For Registering BaseAccountingApplication, **--> set AcctApplicationIdList 3** **Note** TheAcctApplicationId'sthatareconfiguredshouldbepresentin /Radius/Advanced/Diameter/Applications section.

Configuring Diameter Transport Management Properties

The following example shows the Diameter transport management configuration:

```
[ //localhost/Radius/Advanced/Diameter/TransportManagement ]
    Identity = toby-ar1.cisco.com
    Realm = cisco.com
    TCPListenPort = 3868
    SCTPListenPort = 3868
    EnableIPV6 = FALSE
    WatchdogTimeout = 500
    ReconnectInterval = 500
    MaxReconnections = 3
    RequestRetransmissionInterval = 100
    MaxRequestRetransmissionCount = 3
    ReceiveBufferSize = 2048
    AdvertisedHostNames = toby-ar1.cisco.com
```

Table 5-53 describes the Diameter transport management properties.

Table 5-53 Diameter Transport Management Properties

Property	Description
Identity	Required; identity of the system on which Diameter application is running. Must be set to a valid resolvable string.
Realm	Required; must be set to a valid Realm in the domain.
TCPListenPort	Required; port number on which Prime Access Registrar server listens for TCP peer connections.
SCTPListenPort	Required; port number on which Prime Access Registrar server listens for SCTP peer connections.
EnableIPv6	Required; if set to TRUE it enables IPV6 for the Diameter application.
WatchdogTimeout	Required; specifies the time interval between watch dog messages.
ReconnectInterval	Required; specifies the time interval between which Prime Access Registrar server attempts to connect to a disconnected peer. If set to 0, then no attempt will be made to connect to a disconnected peer.
MaxReconnections	Required; specifies the number of times Prime Access Registrar server tries to Make a reconnection attempt. If set to 0, then no attempt will be made to reconnect.
RequestRetransmissionInterval	Required; the time for which retransmission of pending requests will be done. If set to 0, then no attempt will be made to retransmit.
MaxRequestRetransmissionCount	Required, maximum number of times Prime Access Registrar server tries to retransmit a pending request. If set to 0, then no attempt will be made to retransmit.
ReceiveBufferSize	Required; initial size of buffer that is preallocated for message reception.
AdvertisedHostNames	Optional; specifies the local hostname address that will be advertised by the Prime Access Registrar server to other peers during CER/CEA exchange. For example: AdvertisedHostNames = toby-ar1.cisco.com

Configuring Diameter Session Management

The following example shows the Diameter session management configuration:

```
//localhost/Radius/Advanced/Diameter/SessionManagement ]
    MaxSessions = 10000
    AuthSessions/
    AcctSessions/
    AuthSessions/
    EnableStatefulSessions = TRUE
    AuthSessionTimeout = 5
    LifeTimeTimeout = 360
    GracePeriodTimeout = 30
    AbortRetryTimeout = 20
    AcctSessions/
    AcctSessionTimeOut = 30
    InterimInterval = 5
    RealTime = 0
```

Table 5-54 describes the Diameter Session Management properties.

Table 5-54 Diameter Session Management Properties

Property	Description
MaxSessions	Required; specifies the maximum number of concurrent Diameter sessions Prime Access Registrar server will maintain. These sessions include both Auth and Acct sessions.
AuthSessions/EnableStatefulSessions	If set to TRUE, the server will enforce stateful sessions and the client will hint for stateful sessions. Default Value is TRUE. Set the property to FALSE to disable stateful sessions.
AuthSessionTimeout	Required; specifies the timeout in seconds before a session requires reauthentication.
LifeTimeTimeout	Required; specifies the timeout in seconds before a session is terminated regardless of whether the session has been reauthenticated.
GracePeriodTimeout	Required; specifies the grace period after the life timeout and before the full termination of the session.
AbortRetryTimeout	Required; specifies the timeout between the subsequent Abort Session Request (ASR) messages if the initial attempt fails.
AcctSessions/AcctSessionTimeOut	Required; specifies the timeout in seconds before a session requires reauthentication.
InterimInterval	Required; specifies the interim interval dictated to the client if the entity is a server or hint to the server if the entity is a client.
RealTime	Required; RealTime value dictated to the client.

Configuring Diameter Application

Table 5-55 describes the Diameter Application properties.

Table 5-55 Diameter Application Properties

Property	Description
Name	Required; name of the application.
Description	Optional; description of the application.
IsVendorSpecific	Required; the default is FALSE. If set to FALSE, the application is ordinary application. If set to TRUE, the application is a VendorSpecific Application.
IsAuthApplication	Required; if set to TRUE the application represents AuthApplication else it represents Accounting Application.
Application ID	Required; specifies the unique integer value for the application. The following are examples of Diameter application: NASREQ 1 Mobile-IP 2 Diameter Base Accounting 3 **Note** ApplicationId property must be set to 0 for Base Protocol.
VendorSpecificApplicationID	Required; specifies the integer value for the vendor specific application.
VendorID	Required; specifies the VendorID for the application. Example: DIAMETER 3GPP Cx APPLICATION VendorSpecificApplicationID 16777216 VendorID 10415
ApplicationURI	Optional; specifies the URI of the Application. Eg: "ftp://ftp.ietf.org/internet-drafts/draft-ietf-aaa-diameter-nasreq-12.txt"
Commands	Required; an indexed list from 1 to <n>. Each entry in the list is the name of the command. It specifies the list of commands associated with the application.

Configuring the Diameter Application

To configure the Diameter application:

Step 1 Move to **//localhost/Radius/Advanced/Diameter/Applications** directory:

Step 2 Add the application you want to add (eg: NASREQ).

```
add NASREQ

Added NASREQ
```

cd NASREQ

```
[ //localhost/Radius/Advanced/Diameter/Applications/NASREQ ]
        Name = NASREQ
        Description =
        IsAuthApplication = TRUE
        IsVendorSpecific = FALSE
        ApplicationID =
        ApplicationURI =
        Commands/
```

Step 3 Set the ApplicationId and ApplicationURI .

set ApplicationId 1

```
Set ApplicationId 1
```

set ApplicationURL "ftp://ftp.ietf.org/internet-drafts/draft-ietf-aaa-diameter-nasreq-12.txt"

Step 4 Add the list of commands for this application.

cd commands/

```
Set 1  AA
```

Configuring Diameter Commands

Table 5-56 describes the Diameter command properties.

Table 5-56 Diameter Command Properties

Property	Description
Name	Required; name of the command.
CommandCode	Required; specifies the integer code of the command.
EnableProxyBit	Required; default is TRUE. When enabled it represents the message is proxiable.

Table 5-56 Diameter Command Properties (continued)

Property	Description

RequestMsgAVPs /	The RequestMsgAVPs define the placement of AVPs within the request command. This contains three sub directories: Fixed, Required and Optional.
	Fixed - Defines the fixed position of AVP in a request message
	Required - The AVP must be present and can appear anywhere in the request message.
	Optional - The AVP name in optional cannot evaluate to any avp name which is included in a fixed or required directory. The avp can appear anywhere in the request message.
	For example:
	`cd Fixed/`
	`Add Session-Id`
	`cd Session-Id/`
	`Name = Session-Id`
	`Description =`
	`Min = 0`
	`Max = 1`
	where:
	Min is the minimum number of times AVP element may be present in a request. The default value is 0.
	Max is the maximum number of times the element may present in a request. A value of zero implies AVP is not present in the request.

AnswerMsgAVPs/	The AnswerMsgAVPs define the placement of AVP's within the answer command. This contains three sub directories: Fixed, Required and Optional. Fixed - Defines the fixed position of AVP in the answer message. Required - The AVP must present and can appear anywhere in the answer message. Optional - The AVP name in optional cannot evaluate to any avp name which is included in a fixed or required directory. The avp can appear anywhere in the answer message.

Configuring the Diameter Commands

To configure the Diameter commands:

Step 1 Change to **/Radius/Advanced/Diameter/Commands**.

Step 2 Add AA command.

add AA

```
[ //localhost/Radius/Advanced/Diameter/Commands ]
```

cd AA/

```
cd AA/
```

Step 3 Set the properties for AA command.

```
[ //localhost/Radius/Advanced/Diameter/Commands/AA ]
    Name = AA
    Description =
    CommandCode =
    EnableProxyBit = TRUE
    RequestMsgAVPs/
    AnswerMsgAVPs/
```

set CommandCode 265

```
Set CommandCode 265
```

set EnableProxyBit TRUE

```
Set EnableProxyBit TRUE
```

Step 4 Configure the RequestMsgAVP's for the command.

cd RequestMsgAVPs/

```
[ //localhost/Radius/Advanced/Diameter/Commands/AA/RequestMsgAVPs ]
    Fixed/
    Required/
    Optional/
```

Add Fixed AVP's for the request message.

Add Fixed AVP's

```
cd Fixed/
```

add Session-Id

```
Added Session-Id
```

cd Session-Id/

```
[ //localhost/Radius/Advanced/Diameter/Commands/AA/RequestMsgAVPs/Fixed/Session-Id ]
    Name = Session-Id
    Description =
     Min = 0
     Max =
```

Maximum and Minimum property specifies the multiplicity of the AVP Inside a request (or response). Similarly add the required and Optional AVP's.

Step 5 Configure AnswerMsgAVP's similar to step 3.

cd AnswerMsgAVPs/

```
[ //localhost/Radius/Advanced/Diameter/Commands/AA/AnswerMsgAVPs ]
    Fixed/
    Required/
    Optional/
```

The following shows an example of NASREQ application configuration:

```
[ //localhost/Radius/Advanced/Diameter/Applications/NASREQ ]
    Name = NASREQ
    Description =
    IsAuthApplication = TRUE
    IsVendorSpecific = FALSE
    ApplicationID = 1
    ApplicationURI =
    ftp://ftp.ietf.org/internet-drafts/draft-ietf-aaa-diameter-nasreq-12.txt
    Commands/
    1. AA
```

The following shows an example of the AA command configuration:

```
[ //localhost/Radius/Advanced/Diameter/Commands ]
    Entries 1 to 1 from 1 total entries
    Current filter: <all>

[ //localhost/Radius/Advanced/Diameter/Commands/AA ]
    Name = AA
    Description =
    CommandCode = 265
    EnableProxyBit = TRUE
    RequestMsgAVPs/
        Fixed/
            Entries 1 to 1 from 1 total entries
            Current filter: <all>

            Session-Id/
                Name = Session-Id
                Description =
                Min = 1
                Max = 1
        Required/
            Entries 1 to 7 from 7 total entries
            Current filter: <all>

            Auth-Application-Id/
                Name = Auth-Application-Id
                Description =
                Min = 1
                Max = 1
            Auth-Request-Type/
                Name = Auth-Request-Type
                Description =
                Min = 1
                Max = 1
            Destination-Realm/
                Name = Destination-Realm
                Description =
                Min = 1
                Max = 1
            Origin-Host/
                Name = Origin-Host
                Description =
                Min = 1
                Max = 1
            Origin-Realm/
                Name = Origin-Realm
                Description =
                Min = 1
                Max = 1
            User-Name/
                Name = User-Name
```

```
                Description =
                Min = 0
                Max = 1
            User-Password/
                Name = User-Password
                Description =
                Min = 0
                Max = 1
        Optional/
            Entries 1 to 42 from 42 total entries
            Current filter: <all>

            ARAP-Password/
                Name = ARAP-Password
                Description =
                Min = 0
                Max = 1
            ARAP-Security/
                Name = ARAP-Security
                Description =
                Min = 0
                Max = 1
            ARAP-Security-Data/
                Name = ARAP-Security-Data
                Description =
                Min = 0
                Max = 100
            Auth-Grace-Period/
                Name = Auth-Grace-Period
                Description =
                Min = 0
                Max = 1
            Auth-Session-State/
                Name = Auth-Session-State
                Description =
                Min = 0
                Max = 1
            Authorization-Lifetime/
                Name = Authorization-Lifetime
                Description =
                Min = 0
                Max = 1
AnswerMsgAVPs/
    Fixed/
        Entries 1 to 1 from 1 total entries
        Current filter: <all>

        Session-Id/
            Name = Session-Id
            Description =
            Min = 1
            Max = 1
    Required/
        Entries 1 to 5 from 5 total entries
        Current filter: <all>

        Auth-Application-Id/
            Name = Auth-Application-Id
            Description =
            Min = 1
            Max = 1
        Auth-Request-Type/
            Name = Auth-Request-Type
            Description =
```

```
            Min = 1
            Max = 1
        Origin-Host/
            Name = Origin-Host
            Description =
            Min = 1
            Max = 1
        Origin-Realm/
            Name = Origin-Realm
            Description =
            Min = 1
            Max = 1
        Result-Code/
            Name = Result-Code
            Description =
            Min = 1
            Max = 1
    Optional/
        Entries 1 to 59 from 59 total entries
        Current filter: <all>

        Acct-Interim-Interval/
            Name = Acct-Interim-Interval
            Description =
            Min = 0
            Max = 1
        ARAP-Challenge-Response/
            Name = ARAP-Challenge-Response
            Description =
            Min = 0
            Max = 1
        ARAP-Features/
            Name = ARAP-Features
            Description =
            Min = 0
            Max = 1
        ARAP-Security/
            Name = ARAP-Security
            Description =
            Min = 0
            Max = 1
        ARAP-Security-Data/
            Name = ARAP-Security-Data
            Description =
            Min = 0
            Max = 100
        ARAP-Zone-Access/
            Name = ARAP-Zone-Access
            Description =
            Min = 0
            Max = 1
        Auth-Grace-Period/
            Name = Auth-Grace-Period
            Description =
            Min = 0
            Max = 1
```

```
Auth-Session-State/
    Name = Auth-Session-State
    Description =
    Min = 0
    Max = 1
Authorization-Lifetime/
    Name = Authorization-Lifetime
    Description =
    Min = 0
    Max = 1
```

Configuring Diameter Dictionary

The Diameter dictionary contains a list of application specific AVPs.

Table 5-57 describes the Diameter BaseProtocol AVP Properties.

Table 5-57 Diameter BaseProtocol AVP Properties

Property	Description
Name	Required; name of the application specific AVPs.
Description	Optional; description of the application specific AVPs.
IsVendorSpecific	Required; default is FALSE. If set to FALSE, the application is ordinary application and user is prompted to enter ApplicationID. If set to TRUE, the application is a VendorSpecific Application. User is prompted to enter VendorSpecificApplicationID and VendorID.
ApplicationID	Required; specifies the unique integer value for the application. **Note** The Application ID must be set to 0 for BaseProtocol AVPs.
VendorSpecificApplicationID	Required, Specifies the integer value for the vendor specific application.
VendorID	Required, specifies the VendorID for the application.

AVPS/	Specifies the list of application specific avps.
	Example:
	`Accounting-Realtime-Required/`
	`Name = Accounting-Realtime-Required`
	`Description =`
	`Attribute = 483`
	`Mandatory = Must`
	`May-Encrypt = Yes`
	`Protected = MustNot`
	`Type = UINT32`
	`Min = 0`
	`Max = 253`
	Refer to Table 5-58 for the description of AVP properties.

Table 5-58 lists the application specific AVP properties.

Table 5-58 AVP Properties

Property	Description
Attribute	Specifies the integer value for the AVP.
Mandatory	Specifies whether the mandatory bit of this AVP should or should not be set.
May-Encrypt	If set to 'yes' then the AVP will be sent encrypted if the connection uses CMS security.
Protected	Specifies whether the protected bit of this AVP should or should not be set.
Type	Specifies the type of the Diameter AVP.

Configure the Diameter Dictionary

To configure the Diameter Dictionary:

Step 1 Change to **/Radius/Advanced/Diameter/Diameter Dictionary**.

Step 2 Add BaseProtocolAVPs.

add BaseProtocolAVPs

```
[ //localhost/Radius/Advanced/Diameter/Diameter Dictionary ]
```

cd BaseProtocolAVPs/

Step 3 Set the properties for BaseProtocolAVPs.

```
[ //localhost/Radius/Advanced/Diameter/Diameter Dictionary/BaseProtocolAVPs ]
    Name = BaseProtocolAVPs
    Description =
    IsVendorSpecific = FALSE
    ApplicationID = 0
    AVPs/
```

set IsVendorSpecific "FALSE"

```
set IsVendorSpecific  "FALSE"
```

set ApplicationID 0

```
set ApplicationID  0
```

Step 4 Configure the application specific AVPs.

cd AVPs/

add User-Name

Step 5 Configure User-Name AVP type and number

--> cd User-Name/

```
[ //localhost/Radius/Advanced/Diameter/Diameter Dictionary/BaseProtocolAVPs/AVPs/User-Name
]
    Name = User-Name
    Description =
    Attribute = 1
    Mandatory = MustNot
    May-Encrypt = No
    Protected = MustNot
    Type = UTF8_STRING
    Min = 0
    Max = 253
```

set Attribute 1

```
set Attribute  1
```

set Type UTF8_STRING

```
set Type UTF8_STRING
```

The following is an example of Diameter BaseProtocol AVPs:

```
[ //localhost/Radius/Advanced/Diameter/Diameter Dictionary/BaseProtocolAVPs ]
        Name = BaseProtocolAVPs
        Description =
        IsVendorSpecific = FALSE
        ApplicationID = 0
        AVPs/
            Entries 1 to 55 from 55 total entries
            Current filter: <all>

            Accounting-Realtime-Required/
                Name = Accounting-Realtime-Required
                Description =
                Attribute = 483
                Mandatory = Must
                May-Encrypt = Yes
                Protected = MustNot
                Type = UINT32
                Min = 0
                Max = 253
            Accounting-Record-Number/
                Name = Accounting-Record-Number
                Description =
                Attribute = 485
                Mandatory = Must
                May-Encrypt = Yes
                Protected = MustNot
                Type = UINT32
                Min = 0
                Max = 253
            Accounting-Record-Type/
                Name = Accounting-Record-Type
                Description =
                Attribute = 480
                Mandatory = Must
                May-Encrypt = Yes
                Protected = MustNot
                Type = ENUM
```

```
        Min = 0
        Max = 253
        Enums/
            1 = "Event Record"
            2 = "Start Record"
            3 = "Interim Record"
            4 = "Stop Record"
    Accounting-Session-Id/
        Name = Accounting-Session-Id
        Description =
        Attribute = 44
        Mandatory = Must
        May-Encrypt = Yes
        Protected = May
        Type = STRING
        Min = 0
        Max = 253
    Accounting-Sub-Session-Id/
        Name = Accounting-Sub-Session-Id
        Description =
        Attribute = 287
        Mandatory = Must
        May-Encrypt = Yes
        Protected = May
        Type = UINT64
        Min = 0
        Max = 253
```

Configuring Local Authentication and Authorization

Cisco Prime Access Registrar (Prime Access Registrar) allows user information to be stored in its own internal database or external stores such as an LDAP directory or Oracle database. This chapter describes how to configure Prime Access Registrar to perform authentication and authorization using the Prime Access Registrar internal database and how to verify and troubleshoot a local service and userlist configuration.

In RADIUS, an Access Request packet is a request for authentication and authorization (AA). Authentication checks username and password credentials, while authorization typically involves returning the correct information to allow the service a user is authorized to have.
Prime Access Registrar performs AA and returns the appropriate RADIUS attributes in an Access Accept packet.

This chapter contains the following sections:

- Configuring a Local Service and UserList
- Troubleshooting the Local Service and UserList Configuration
- aregcmd Command Performance
- UserDefined1 Property
- Access-Request Logging

Configuring a Local Service and UserList

Prime Access Registrar uses services configured under **/Radius/Services** to process RADIUS requests. To process RADIUS access requests locally, you must configure a service and set its type to **local**. A local service references an Prime Access Registrar userlist.

The following sections show the commands you enter and the expected responses from the Prime Access Registrar server to do the following:

- Configuring a Local Service
- Configuring a Userlist
- Configuring Cisco Prime Access Registrar to Use the Local Service For AA
- Activating the Configuration

Throughout this chapter, the **aregcmd** commands you enter are shown in **bold** font, and the server responses are shown in `smaller plain font` as shown in the following:

command you enter

```
server response
```

Configuring a Local Service

Prime Access Registrar maintains **Services** under **/Radius**.

To configure a local service:

Step 1 Use the **add** command at **/Radius/Services** to create a Service.

cd /Radius/Services

```
[ //localhost/Radius/Services ]
```

add SouthBay

```
Added SouthBay
```

Step 2 Change directory to the new service and set its type to local.

cd SouthBay

```
[ //localhost/Radius/Services/SouthBay ]
```

set type local

```
Set Type local
```

Step 3 Use the **set** command to associate a userlist with the service.

set userlist SouthUsers

```
Set UserList SouthUsers
```

Configuring a Userlist

Prime Access Registrar maintains **UserLists** under **/Radius**.

To configure a userlist:

Step 1 Use the **add** command at **/Radius/UserLists** to create a userlist.

cd /Radius/UserLists

```
[ //localhost/Radius/UserLists ]
```

add SouthUsers

```
Added SouthUsers
```

Step 2 Change directory to the userlist and add users.

cd SouthUsers

```
[ //localhost/Radius/UserLists/SouthUsers ]
```

add user1

```
Added user1
```

Step 3 Change directory to each user you add and set the user's password.

cd user1

```
[ //localhost/Radius/UserLists/SouthUsers/user1 ]
```

set Password test

```
Retype password to confirm:
Set Password <encrypted>
```

Configuring Cisco Prime Access Registrar to Use the Local Service For AA

To configure Prime Access Registrar to use the local service for authentication and authorization, enter commands to set the DefaultAuthenticationService and DefaultAuthenticationService to the service you created, as shown in the following:

cd /Radius

```
[ //localhost/Radius ]
```

set DefaultAuthenticationService SouthBay

```
Set DefaultAuthenticationService SouthBay
```

set DefaultAuthorizationService SouthBay

```
Set DefaultAuthorizationService SouthBay
```

Activating the Configuration

To activate the configuration changes you have made, enter the **save** command:

save

```
Validating //localhost...

Saving //localhost...
```

After you issue the **save** command, Prime Access Registrar attempts to validate the configuration, checks for all required properties, and ensures there are no logic errors. If the validation is successful, Prime Access Registrar saves the configuration to the MCD database.

Troubleshooting the Local Service and UserList Configuration

Before you begin troubleshooting, ensure that the current configuration is valid and active. To ensure that any configuration changes you have made are valid and stored in the database, you must issue the **save** command.

save

```
Validating //localhost...
Saving //localhost...
```

To ensure that the current configuration is active, issue the **reload** command.

reload

```
Reloading Server 'Radius'...
Server 'Radius' is Running, its health is 10 out of 10
```

Verifying the Configuration

To verify the configuration changes you have made:

Step 1 Check to see that the UserList exists under the service.

ls /Radius/Services/SouthBay

```
[ /Radius/Services/SouthBay ]
    Name = SouthBay
        Description =
        Type = local
        IncomingScript~ =
        OutgoingScript~ =
        OutagePolicy~ = RejectAll
        OutageScript~ =
        UserList = SouthUsers
```

Step 2 Check to see that user **user1** exists under the SouthUsers userlist.

ls /Radius/UserLists/SouthUsers

```
[ /Radius/UserLists/SouthUsers ]
    Entries 1 to 1 from 1 total entries
    Current filter: <all>
    Name = SouthUsers
    Description =
    user1/
```

Step 3 Turn on debugging.

trace /r 5

```
Traced "/Radius: Trace level is set to 5"
```

Step 4 Use **radclient** to send an Access-Request for user **user1**.

simple user1 test

The debugging output will be sent to the file **name_radius_1_log** in the **/opt/CSCOar/ logs** directory. The following example shows items you should expect in a successful Access-Request.

Note Lines of interest are in **bold** font.

```
11/12/2013 18:34:35: P1144: Packet received from 127.0.0.1
11/12/2013 18:34:35: P1144: Trace of Access-Request packet
11/12/2013 18:34:35: P1144:    identifier = 4
11/12/2013 18:34:35: P1144:    length = 62
11/12/2013 18:34:35: P1144:    reqauth = f5:37:f7:04:99:85:c7:63:8f:bc:f4:44:ab:03:4e:1a
11/12/2013 18:34:35: P1144:    User-Name = user1
11/12/2013 18:34:35: P1144:    User-Password = 59:fb:2e:a9:34:de:0e:15:60:8d:4b:64:77:6a:57:d8
11/12/2013 18:34:35: P1144:    NAS-Port = 2
11/12/2013 18:34:35: P1144:    NAS-Identifier = localhost
11/12/2013 18:34:35: P1144: Using Client: localhost (127.0.0.1)
11/12/2013 18:34:35: P1144: Using NAS: localhost (127.0.0.1)
11/12/2013 18:34:35: P1144: Request is directly from a NAS: TRUE
11/12/2013 18:34:35: P1144: Authenticating and Authorizing with Service SouthBay
11/12/2013 18:34:35: P1144: Getting User user1's UserRecord from UserList SouthUsers
11/12/2013 18:34:35: P1144: User user1's password matches
11/12/2013 18:34:35: P1144: No default Remote Session Service defined.
11/12/2013 18:34:35: P1144: Trace of Access-Accept packet
11/12/2013 18:34:35: P1144:    identifier = 4
11/12/2013 18:34:35: P1144:    length = 20
11/12/2013 18:34:35: P1144:    reqauth = 36:88:34:0c:cc:ea:9e:d8:6d:f5:14:f7:ab:26:e7:f6
11/12/2013 18:34:35: P1144: Sending response to 127.0.0.1
11/12/2013 18:34:35: Log: Request from localhost (127.0.0.1): User user1 accepted
```

The following example shows a trace for an unsuccessful Access-Request due to an invalid password.

Note Lines of interest are in **bold** font.

```
11/12/2013 19:05:13: P1527: Packet received from 127.0.0.1
11/12/2013 19:05:13: P1527: Trace of Access-Request packet
11/12/2013 19:05:13: P1527:11/12/2013 19:05:13: P1527:11/12/2013 19:05:13: P1527:
11/12/2013 19:05:13: P1527:11/12/2013 19:05:13: P1527:
11/12/2013 19:05:13: P1527:11/12/2013 19:05:13: P1527:11/12/2013 19:05:13: P1527: Using Client: localhost
(127.0.0.1)
11/12/2013 19:05:13: P1527: Using NAS: localhost (127.0.0.1)
11/12/2013 19:05:13: P1527: Request is directly from a NAS: TRUE
11/12/2013 19:05:13: P1527: Authenticating and Authorizing with Service SouthBay
11/12/2013 19:05:13: P1527: Getting User user1's UserRecord from UserList SouthUsers
11/12/2013 19:05:13: P1527: User user1's password does not match
11/12/2013 19:05:13: P1527: Rejecting request
11/12/2013 19:05:13: P1527: Rejecting request
11/12/2013 19:05:13: P1527: Trace of Access-Reject packet
11/12/2013 19:05:13: P1527:11/12/2013 19:05:13: P1527:11/12/2013 19:05:13: P1527:
11/12/2013 19:05:13: P1527:11/12/2013 19:05:13: P1527: Sending response to 127.0.0.1
11/12/2013 19:05:13: Log: Request from localhost (127.0.0.1): User user1 rejected (UserPasswordInvalid)
```

If a user's password is invalid, reset the password to ensure it was entered correctly. Also check that the shared secret being used by the RADIUS client and the Prime Access Registrar server match.

Configuring Return Attributes and Check-Items

Prime Access Registrar supports RADIUS check item attributes at the user and group levels. You can configure Prime Access Registrar to check for attributes that must be present or attributes that must not be present in the Access-Request packet for successful authentication. For a complete list of attributes supported in Prime Access Registrar, see Cisco Prime Access Registrar 8.0 Reference Guide.

When using check item attributes, Prime Access Registrar rejects Access-Requests if either of the following conditions exist:

- Any configured check item attributes are not present in the Access-Request packet
- Any Access-Request packet's check item attribute values do not match with those configured check item attribute values

This section contains the following topics:

- Configuring Per User Return Attributes
- Configuring Per User Check-Items
- Verifying the Per User Return Attributes and Check-Items Configuration
- Configuring Return Attributes and Check-Items Using UserGroup

Configuring Per User Return Attributes

User return attributes are attributes that are specific for a given user each time they log in. To configure a user's return attributes, change directory to the user's Attributes subdirectory and configure the desired attributes.

cd /Radius/UserLists/SouthUsers/User1/Attributes

```
[ //localhost/Radius/UserLists/SouthUsers/user1/Attributes ]
```

set Session-Timeout 60

```
Set Session-Timeout  60
```

set Callback-Number 5551124

```
Set Callback-Number  5551124
```

Configuring Per User Check-Items

Check Items are a way to check that certain attribute/values exist in a user‘s access-request. If the attribute/values are not present in the access-request, the Prime Access Registrar server rejects the access-request.

To check that an access-request for `user1` has the Calling-Station-Id attribute set to 5555678, enter the following:

cd /Radius/UserLists/SouthUsers/User1/CheckItems

```
[ //localhost/Radius/UserLists/SouthUsers/user1/CheckItems ]
```

set Calling-Station-Id 5555678

```
Set Calling-Station-Id  5555678
```

Be sure to **save** your configuration to preserve your changes.

Verifying the Per User Return Attributes and Check-Items Configuration

A successful request will produce a trace similar to the following:

```
11/12/2013 14:08:07: P1539: Packet received from 127.0.0.1
11/12/2013 14:08:07: P1539: Trace of Access-Request packet
11/12/2013 14:08:07: P1539:    identifier = 1
11/12/2013 14:08:07: P1539:    length = 71
11/12/2013 14:08:07: P1539:    reqauth = d6:86:c5:1e:0e:a0:20:4f:9a:1a:2c:35:27:16:12:36
11/12/2013 14:08:07: P1539:    User-Name = user1
11/12/2013 14:08:07: P1539:    User-Password = 99:dc:4a:22:ef:f6:8b:90:a2:3a:50:f0:a6:03:6e:b3
11/12/2013 14:08:07: P1539:    NAS-Port = 1
11/12/2013 14:08:07: P1539:    Calling-Station-Id = 5555678
11/12/2013 14:08:07: P1539:    NAS-Identifier = localhost
11/12/2013 14:08:07: P1539: Using Client: localhost (127.0.0.1)
11/12/2013 14:08:07: P1539: Using NAS: localhost (127.0.0.1)
11/12/2013 14:08:07: P1539: Request is directly from a NAS: TRUE
11/12/2013 14:08:07: P1539: Authenticating and Authorizing with Service SouthBay
11/12/2013 14:08:07: P1539: Getting User user1's UserRecord from UserList SouthUsers
11/12/2013 14:08:07: P1539: User user1's password matches
11/12/2013 14:08:07: P1539: Processing User user1's check items
11/12/2013 14:08:07: P1539: Merging User user1's Attributes into response Dictionary
11/12/2013 14:08:07: P1539: Merging attributes into the Response Dictionary:
11/12/2013 14:08:07: P1539:   Adding attribute Callback-Number, value = 5551124
11/12/2013 14:08:07: P1539:   Adding attribute Session-Timeout, value = 60
```

```
11/12/2013 14:08:07: P1539: No default Remote Session Service defined.
11/12/2013 14:08:07: P1539: Trace of Access-Accept packet
11/12/2013 14:08:07: P1539:    identifier = 1
11/12/2013 14:08:07: P1539:    length = 35
11/12/2013 14:08:07: P1539:    reqauth = cc:2d:51:71:b5:49:0e:e6:f1:eb:1c:61:51:7a:f1:cb
11/12/2013 14:08:07: P1539:    Callback-Number = 5551124
11/12/2013 14:08:07: P1539:    Session-Timeout = 60
11/12/2013 14:08:07: P1539: Sending response to 127.0.0.1
11/12/2013 14:08:07: Log: Request from localhost (127.0.0.1): User user1 accepted
```

Configuring Profiles to Group Attributes

You can use thePrime Access Registrar profile object to group attributes. For example, you might want to group attributes for all PPP users. All PPP users could then be assigned the profile and the attributes contained in the profile would be returned in their access-accepts.

To configure profiles to group attributes:

Step 1 Change directory to **/Radius/Profiles** and add a profile.

cd /Radius/Profiles

```
[ //localhost/Radius/Profiles ]
```

add PPP-Profile

```
Added PPP-Profile
```

Step 2 Change directory to the new profile, then change directory to the profile's Attributes subdirectory.

cd PPP-Profile

```
[ //localhost/Radius/Profiles/PPP-Profile ]
```

cd Attributes

```
[ //localhost/Radius/Profiles/PPP-Profile/Attributes ]
```

Step 3 Configure the desired attributes for the profile.

set Service-Type Framed

```
Set Service-Type  Framed
```

set Framed-Protocol PPP

```
Set Framed-Protocol  PPP
```

Note When you need to set an attribute to a value that includes a space, you must double-quote the value, as in the following: *set Framed-Route "192.168.1.0/24 192.168.1.1"*

Step 4 Assign the profile to a user by setting the user's BaseProfile attribute to the desired profile.

cd /Radius/UserLists/SouthUsers/User1

```
[ //localhost/Radius/UserLists/SouthUsers/user1 ]
```

set BaseProfile PPP-Profile

```
Set BaseProfile PPP-Profile
```

Configuring Return Attributes and Check-Items Using UserGroup

A profile can also be assigned to a UserGroup. You assign a profile to a group by setting the group's BaseProfile attribute to the desired profile.

To configure return attributes and check-items using usergroup:

Step 1 Change directory to **/Radius/UserGroups** and add a UserGroup.

cd /Radius/UserGroups

```
[ //localhost/Radius/UserGroups ]
```

add PPP-Group

```
Added PPP-Group
```

Step 2 Change directory to the new UserGroup and add Return Attributes.

cd PPP-Group

```
[ //localhost/Radius/UserGroups/PPP-Group ]
```

cd Attributes

```
[ //localhost/Radius/UserGroups/PPP-Group/Attributes ]
```

set Service-Type Outbound

```
Set Service-Type Outbound
```

Step 3 Change directory to the UserGroups' Check-Items subdirectory and add CheckItems.

cd ../CheckItems/

```
[ //localhost/Radius/UserGroups/PPP-Group/CheckItems ]
```

set Service-Type Framed

```
Set Service-Type  Framed
```

Step 4 Assign the UserGroup to a User.

cd /Radius/UserLists/SouthUsers/User2

```
[ //localhost/Radius/UserLists/SouthUsers/user2 ]
```

set Group PPP-Group

```
Set Group PPP-Group
```

Return Attribute Precedence

Because there are multiple ways of returning attributes, you might at some time have an attribute clash. In case of an attribute clash, the attribute precedence is as follows (from highest to lowest):

1. User attribute
2. User profile
3. UserGroup attribute
4. UserGroup profile

aregcmd Command Performance

You can impact **aregcmd** command performance and server response time by having Prime Access Registrar userlists that contain more than 10,000 users. Prime Access Registrar userlists were not designed to contain 10,000 users in any one list.

If you must provide service for groups greater than 10000 users, we recommend that you use an external data store such as an LDAP directory or an Oracle database. If you are unable to use an external data store, create multiple userlists instead, keeping each userlist under 10,000 users.

Multiple userlists require multiple services (one for each userlist), because a service cannot reference more than one userlist. The multiple services can then be combined using the Service Grouping feature with ResultRule, OR, as follows:

```
[ //localhost/Radius/Services/GroupService ]
    Name = GroupService
    Description =
    Type = group
    IncomingScript~ =
    OutgoingScript~ =
    ResultRule = OR
```

```
    GroupServices/
    1. UserService1
    2. UserService2
    3. UserService3
```

UserDefined1 Property

The UserDefined1 property of a user object is a free text field. You can use the UserDefined1 property to store additional user information much like the Description property, but its most powerful use is to pass information to an extension point script. The value set in the UserDefined1 property is automatically set to the environment variable of the same name during authentication. Any extension point script that subsequently runs has access the value in that property.

```
[ //localhost/Radius/UserLists/Default/bob ]
    Name = bob
    Description =
    Password = <encrypted>
    AllowNullPassword = FALSE
    Enabled = TRUE
    Group~ =
    BaseProfile~ =
    AuthenticationScript~ =
    AuthorizationScript~ =
    UserDefined1 =
    Attributes/
    CheckItems/
```

Access-Request Logging

By default, Prime Access Registrar logs all dropped and rejected requests in the name_radius_1_log file. The following are examples of log entries for dropped or rejected requests.

```
11/12/2013 17:38:11 name/radius/1 Warning Protocol 0 Request from localhost (127.0.0.1):
User user1 rejected (UserPasswordInvalid)
```

```
11/12/2013 18:05:12 name/radius/1 Warning Protocol 0 Packet from 128.107.132.106: that
address is not in the Clients list <unknown user>
```

To log all accepted requests as well, set the LogServerActivity advanced property to TRUE:

set /Radius/Advanced/LogServerActivity TRUE

Set /Radius/Advanced/LogServerActivity TRUE

save

```
Validating //localhost...


Saving //localhost...
```

reload

```
Reloading Server 'Radius'...
Server 'Radius' is Running, its health is 10 out of 10


Access-Accept packets are now logged as well:
11/12/2013 18:22:32 name/radius/1 Activity Protocol 0 Request from localhost (127.0.0.1):
User user2 accepted
```

Using Extension Points

This chapter describes how to use Cisco Prime Access Registrar (Prime Access Registrar) scripting to customize your RADIUS server. At specific points during a Prime Access Registrar Request-Response packet flow, service providers can initiate scripts using REX (C/C++), Java, TCL, or internal script interfaces, to customize or modify the packet flow. When the script finishes, the packet flow continues with the next step.You can write scripts to affect the way Prime Access Registrar handles and responds to requests and to change the behavior of Prime Access Registrar after a script is run.

All scripts reference the three dictionaries listed below, which are data structures that contain key/value pairs.

- **Request dictionary**—contains all of the attributes from the access-request or other incoming packets, such as the username, password, and service hints.
- **Response dictionary**—contains all of the attributes in the access-accept or other response packets. As these are the attributes the server sends back to the NAS, you can use this dictionary to add or remove attributes.
- **Environment dictionary**—contains well-known keys whose values enable scripts to communicate with Prime Access Registrar or to communicate with other scripts.

This chapter contains the following sections:

- Determining the Goal of the Script, page 7-2
- Writing the Script, page 7-2
- Adding the Script Definition, page 7-5
- About the Tcl/Tk 8.3 Engine, page 7-6
- Cisco Prime Access Registrar Scripts, page 7-6
- Extension Points in Cisco Prime Access Registrar, page 7-17

Client Scripting

Cisco is not liable for scripts developed by clients.

Though Prime Access Registrar allows external code (Tcl/C/C++/Java) to be used by means of a script, custom service, policy engine, and so forth, while processing request, response, or while working with the environment dictionaries, it shall not be responsible for the scripts used and will not be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of the script.

Determining the Goal of the Script

The goal of the script and its scripting point are tied together. For example, when you want to create a script that performs some special processing of a username before it is processed by the Prime Access Registrar server, you would reference this script as an *incoming* script.

When on the other hand, you would like to affect the response, such as setting a specific timeout when there is not one, you would reference the script as an *outgoing* script.

In order to be able to create a script, you need to understand the way Prime Access Registrar processes client requests. Prime Access Registrar processes requests and responses in a hierarchical fashion; incoming requests are processed from the most general to the most specific levels, whereas, outgoing responses are processed from the most specific to the most general levels. Extension points are available at each level.

An incoming script can be referenced at each of the following extension points:

- RADIUS server
- Vendor (of the immediate client)
- Client (individual NAS)
- NAS-Vendor-Behind-the-Proxy
- Client-Behind-the-Proxy
- Remote Server (of type RADIUS)
- Service

An authentication or authorization script can be referenced at each of the following extension points:

- Group Authentication
- User Authentication
- Group Authorization
- User Authorization

The outgoing script can be referenced at each of the following extension points:

- Service
- Client-Behind-the-Proxy
- NAS-Vendor-Behind-the-Proxy
- Client (individual NAS)
- NAS Vendor
- RADIUS server

Writing the Script

You can write scripts in Tcl, REX, Java, or as shared libraries using C or C++. In this section, the scripts are shown in Tcl.

Writing the Script

To write a script:

Step 1 Create the Tcl source file using an editor.

Step 2 Give it a name.

Step 3 Define one or more procedures, using the following syntax:

```
proc name {request response environment}
{Body}
```

Step 4 Create the body of the script.

Step 5 Save the file and copy it to the **/opt/CSCOar/scripts/radius/tcl** directory, or to the location you chose when you installed Prime Access Registrar.

Note You can also use the Prime Access Registrar GUI or CLI to write internal scripts, by which you can add, modify, or delete attribute in the request, response, and environment dictionaries for RADIUS, Diameter, and TACACS+. For more information, see Internal Scripts, page 7-14.

Choosing the Type of Script

When you create a script, you can use any one or all of the three dictionaries: Request, Response, or Environment. Here is what each dictionary does it for you,

- When you use the Request dictionary, you can modify the contents of a NAS request. Scripts that use the Request dictionary are usually employed as incoming scripts.
- When you use the Response dictionary, you can modify what the server sends back to the NAS. These scripts are consequently employed as outgoing scripts.
- When you use the Environment dictionary, you can do the following:
  - Affect the behavior of the server after the script is run. For example, you can use the Environment dictionary to decide which of the multiple services to use for authorization, authentication, and accounting.
  - Communicate among scripts, as the scripts all share these three dictionaries. For example, when a script changes a value in the Environment dictionary, the updated value can be used in a subsequent script.

The following sections show scripts examples of all the three dictionaries:

- Request Dictionary Script
- Response Dictionary Script
- Environment Dictionary Script

Request Dictionary Script

The Request Dictionary script is referenced from the server's IncomingScript scripting point. It checks to see whether the request contains a **NAS-Identifier** or a **NAS-IP-Address**. When it does not, it sets the **NAS-IP-Address** from the request's source IP address.

```
proc MapSourceIPAddress {request response environment}
{
    if { ! ( [ $request containsKey NAS-Identifier ] ||
```

```
        [ $request containsKey NAS-IP-Address ] ) } {
        $request put NAS-IP-Address [ $environment get Source-IP-Address ]
    }
```

Tcl scripts interpret **$request** arguments as active commands that can interpret strings from the Request dictionary, which contains keys and values.

The **containsKey** method has the syntax: <*$dict*> **containsKey** <*attribute*>. In this example, <*$dict*> refers to the Request dictionary and the attributes **NAS-identifier** and **NAS-IP-Address**. The **containsKey** method returns **1** when the dictionary contains the attribute, and **0** when it does not. Using the **containsKey** method prevents you from overwriting an existing value.

The **put** method has the syntax: <*$dict*> **put** <*attribute value*>[<*index*>]. In this example, <*$request*> refers to the Request dictionary and the attribute is **NAS-IP-Address**. The **put** method sets the NAS's IP address attribute.

The **get** method has the syntax: <*$dict*> get <*attribute*>. In this example, <*$dict*> refers to the Environment dictionary and <*attribute*> is the **Source-IP-Address**. The **get** method returns the value of the attribute from the environment dictionary.

For a list of the methods you can use with scripts, see
Cisco Prime Access Registrar 8.0 Reference Guide. They include **get**, **put**, and others.

Response Dictionary Script

This script is referenced from either the user record for users whose sessions are always PPP, or from the script, **AuthorizeService**, which checks the request to determine which service is desired. The script merges the Profile named **default-PPP-users** into the Response dictionary.

```
proc AuthorizePPP {request response environment}
{
    $response addProfile default-PPP-users
```

The **addProfile** method has the syntax: <*$dict*> **addProfile** <*profile*>[<*mode*>]. In this example, <*$dict*> refers to the Response dictionary and the profile is **default-PPP-users**. The script copies all of the attributes of the Profile <*profile*> into the dictionary.

Environment Dictionary Script

This script is referenced from the NAS Incoming Script scripting point. It looks for a realm name on the username attribute to determine which AAA services should be used for the request. When it finds `@radius`, it selects a set of AAA services that will proxy the request to a remote RADIUS server. When it finds `@tacacs`, it selects the Authentication Service that will proxy the request to a TACACS server for authentication. For all of the remaining usernames, it uses the default Service (as specified in the configuration by the administrator).

Note the function, **regsub**, is a Tcl function.

```
proc ParseProxyHints {request response environment} {
    set userName [ $request get User-Name ]
    if { [ regsub "@radius" $userName "" newUserName ] } {
    $request put User-Name $newUserName
    $radius put Authentication-Service "radius-proxy"
    $radius put Authorization-Service "radius-proxy"
    $radius put Accounting-Service "radius-proxy"
    } else {
    if { [ regsub "@tacacs" $userName "" newUserName ] } {
    $request put User-Name
    $radius put Authentication-Service "tacacs-client"
```

Adding the Script Definition

After you have written the script, you must add the script definition to the Prime Access Registrar's script **Configuration** directory so it can be referenced. Adding the script definition involves:

- Specifying the script definition; it must include the following:
  - **Name**—used in other places in the configuration to refer to the script. It must be unique among all other scripts.
  - **Language**—can be either Tcl or REX (shared libraries)
  - **Filename**—the name you used when you created the file
  - **EntryPoint**—the function name.

  The **Name** and the **EntryPoint** can be the same name, however they do not have to be.
- Choosing a scripting point; nine exist for incoming and outgoing scripts. These include:
  - the server
  - the vendor of the immediate client
  - the immediate client
  - the vendor of the specific NAS
  - the specific NAS
  - the service (rex or tcl)
  - the group (only AA scripts
  - the user record (only AA scripts)
  - remote server (only type RADIUS)

The rule of thumb to use in determining where to add the script is the most general scripts should be on the outermost points, whereas the most specific scripts should be on the innermost points.

Note The client and the NAS are the same entity, unless the immediate client is acting as a proxy for the actual NAS.

This section contains the following topics:

- Adding the Example Script Definition
- Choosing the Scripting Point
- Testing the Script

Adding the Example Script Definition

In the server configuration a **Scripts** directory exists. You must add the script you created to this directory. To add the **ParseProxyHints** script to the Prime Access Registrar server, enter the following command and supply the following information:

Name=**ParseProxyHints**
Description=**ParseProxyHints**
Language=**tcl**

Filename=**ParseProxyHints**
Entrypoint=**ParseProxyHints**

aregcmd add /Radius/Scripts/ParseProxyHints ParseProxyHints tcl ParseProxyHints ParseProxyHints

Choosing the Scripting Point

As the example script, **ParseProxyHints**, applies to a specific NAS (NAS1), the entry point should be that NAS. To specify the script at this scripting point, enter:

aregcmd set /Radius/Clients/NAS1/IncomingScript ParseProxyHints

Testing the Script

To test the script, you can use the **radclient** command, which lets you create and send packets. For more information about the **radclient** command, see Chapter 4, "Setting the CPAR Configurable Option."

About the Tcl/Tk 8.3 Engine

Prime Access Registrar uses Tcl engine is version Tcl/Tk8.3. Since the Tcl/Tk8.3 engine supports a multi-threading application environment, it can achieve much better performance than Tcl/Tk7.6.

Tcl/Tk8.3 also performs ***byte-compile***, so no runtime interpretation is required.

Cisco Prime Access Registrar Scripts

The Prime Access Registrar scripts are stored in **/localhost/Radius/Scripts**. Most of the scripts are written in the RADIUS Extension language (REX). Some scripts are provided in both REX and Tcl. The scripts written in Tcl all begin with the letter **t** followed by their functional name. The Tcl scripts are listed below:

tACME
tAuthorizePPP
tAuthorizeService
tAuthorizeTelnet
tMapSourceIPAddress
tParseAARealm
tParseAASRealm
tParseProxyHints
tParseServiceAndAAARealmHints
tParseServiceAndAAASRealmHints
tParseServiceAndAARealmHints
tParseServiceAndAASRealmHints
tParstServiceAndProxyHints
tParseServiceHints

You can use the Prime Access Registrar GUI to write internal scripts, by which you can add, modify, or delete attribute in the request, response, and environment dictionaries for RADIUS, Diameter, and TACACS+. For more information about configuring internal scripts by using the GUI, see Cisco Prime Access Registrar 8.0 User Guide.

This section contains the following topics:

- ACME
- AltigaIncomingScript
- AltigaOutgoingScript
- ANAAAOutgoing
- AscendIncomingScript
- AscendOutgoingScript
- AuthorizePPP
- AuthorizeService
- AuthorizeSLIP
- AuthorizeTelnet
- CabletronIncoming
- CabletronOutgoing
- CiscoIncoming
- CiscoOutgoing
- CiscoWithODAPIncomingScript
- ExecCLIDRule
- ExecDNISRule
- ExecFilterRule
- ExecNASIPRule
- ExecRealmRule
- ExecTimeRule
- LDAPOutage
- MapSourceIPAddress
- ParseAAARealm
- ParseAAASRealm
- ParseAARealm
- ParseAASRealm
- ParseProxyHints
- ParseServiceAndAAARealmHints
- ParseServiceAndAAASRealmHints
- ParseServiceAndAARealmHints
- ParseServiceAndAASRealmHints
- ParseServiceAndProxyHints

- ParseServiceHints
- ParseTranslationGroupsByCLID
- ParseTranslationGroupsByDNIS
- ParseTranslationGroupsByRealm
- UseCLIDAsSessionKey
- USRIncomingScript
- USRIncomingScript-IgnoreAccountingSignature
- USROutgoingScript
- Internal Scripts, page 7-14
- Blacklisting Script, page 7-16

ACME

ACME is referenced from Vendor ACME for the outgoing script. If the Prime Access Registrar server accepts this Access-Request and the response does not yet contain a Session-Timeout, set it to 3600 seconds.

AltigaIncomingScript

AltigaIncomingScript maps Altiga-proprietary attributes to Prime Access Registrar's global attribute space.

AltigaOutgoingScript

AltigaOutgoingScript maps Altiga attributes from Prime Access Registrar's global attribute space to the appropriate Altiga-proprietary attributes.

ANAAAOutgoing

ANAAAOutgoing can be referenced from either the client or vendor outgoing scripting point to be used in HRPD/EV-DO networks where Prime Access Registrar is the Access Network (AN) AAA server. ANAAAOutgoing checks to see if the response contains the Callback-Id attribute. If the response contains the Callback-Id attribute and the value is less than 253 characters, ANAAAOutgoing prefixes a zero (0) to the value. For example, it changes "112" into "0112." The ANAAAOutgoing script always returns REX_OK.

AscendIncomingScript

AscendIncomingScript maps Ascend-proprietary attributes to Prime Access Registrar's global attribute space.

AscendOutgoingScript

AscendOutgoingScript maps Ascend attributes from Prime Access Registrar's global attribute space to the appropriate Ascend-proprietary attributes.

AuthorizePPP

AuthorizePPP is referenced from either the use record for users who's sessions are always PPP or from the from the script AuthorizeService, which checks the request to determine which service is desired. This script merges in the Profile named "default-PPP-users" into the response dictionary.

AuthorizeService

AuthorizeService is referenced from user record for users who's sessions might be PPP, SLIP or Telnet depending on how they are connecting to the NAS. This script checks the request to determine which service is desired. If it is telnet, it calls the script AuthorizeTelnet. If it is PPP, it calls the script AuthorizePPP. If it is SLIP, it calls the script AuthorizeSLIP. If it is none of these, it rejects the request.

AuthorizeSLIP

AuthorizeSLIP is referenced from either the user record for users who's sessions are always SLIP or from the from the script AuthorizeService, which checks the request to determine which service is desired. This script merges in the Profile named "default-SLIP-users" into the response dictionary.

AuthorizeTelnet

AuthorizeTelnet is referenced from either the user record for users who's sessions are always telnet or from the from the script AuthorizeService, which checks the request to determine which service is desired. This script merges in the Profile named "default-Telnet-users" into the response dictionary.

CabletronIncoming

CabletronIncoming maps Cabletron-proprietary attributes to Prime Access Registrar's global attribute space.

CabletronOutgoing

Use CabletronOutgoing to map Cisco-proprietary attributes from Prime Access Registrar's global attribute space to the appropriate Cabletron-proprietary attributes.

CiscoIncoming

Use CiscoIncoming to map Cisco-proprietary attributes to Prime Access Registrar's global attribute space.

CiscoOutgoing

Use CiscoOutgoing to map Cisco-proprietary attributes from Prime Access Registrar's global attribute space to the appropriate Cabletron-proprietary attributes.

CiscoWithODAPIncomingScript

Use CiscoWithODAPIncomingScript to map Cisco-proprietary attributes to Prime Access Registrar's global attribute space and to map ODAP requests to the appropriate services and session managers.

CiscoWithODAPIncomingScript checks the incoming NAS-Identifier sent by the client. If the NAS-Identifier does not equal odap-dhcp, the request is not an ODAP request. If the request is not an ODAP request, the script does no more ODAP-specific processing, and calls CiscoIncomingScript to allow it to process the request.

If the request is an ODAP request, CiscoWithODAPIncomingScript removes the NAS-Identifier attribute because it is no longer required. The script then sets the Authentication-Service and the Authorization-Service to odap-users and sets the Accounting-Service to odap-accounting.

ExecCLIDRule

ExecCLIDRule is referenced from the policy engine to determine the authentication and authorization service and policy based on the CLID set in the policy engine.

ExecDNISRule

ExecDNISRule is referenced from the policy engine to determine the authentication and authorization service and policy based on the DNIS set in the policy engine.

ExecFilterRule

ExecFilterRule is referenced from the policy engine to determine whether a user packet should be rejected or not based on whether a special character like "*", "/", "\" or "?" shows up in the packet.

ExecNASIPRule

ExecNASIPRule is referenced from the policy engine to enable configuration of policies based on the incoming NAS-IP-Address. You can configure two attributes, *client-ip-address* and *subnetmask*, to match the incoming NAS-IP-Address and its subnet mask. If the attributes match, ExecNASIPrule sets the environment variables (if they are configured in that rule).

ExecRealmRule

ExecRealmRule is referenced from the policy engine to determine the authentication and authorization service and policy based on the realm set in the policy engine.

ExecTimeRule

ExecTimeRule either rejects or accepts Access Request packets based on the time range specified in a user's login profile. You can configure the TimeRange and AcceptedProfile attributes.

The format for the TimeRange is to set the allowable days followed by the allowable times, as in:

TimeRange = dateRange, timeRange

The dateRange can be in the form of a date, a range of allowable dates, a day, or a range of allowable days. The timeRange should be in the form of hh:mm-hh:mm.

Here are a few examples:

mon-fri,09:00-17:00

Allows access Monday through Friday from 9 AM until 5 PM.

mon,09:00-17:00;tue-sat,12:00-13:00

Allows access on Monday from 9 AM until 5 PM and from 12 noon until 1 PM on Tuesday through Saturday

mon,09:00-24:00;tue,00:00-06:00

Allows access on Monday from 9 AM until Tuesday at 6 AM

1-13,10-17:00; 15,00:00-24:00

Allows access from the first of the month until the thirteenth of the month from 10 AM until 5 PM and all day on the fifteenth of the month

LDAPOutage

LDAPOutage is referenced from LDAP Services as OutageScript. LDAPOutage logs when the LDAP binding is lost.

MapSourceIPAddress

MapSourceIPAddress is referenced from the Prime Access Registrar server's IncomingScript scripting point. MapSourceIPAddress checks to see if the request contains either a NAS-Identifier or a NAS-IP-Address. If not, this script sets the NAS-IP-Address from the request's source IP address.

The Tcl version of this script is tMapSourceIPAddress.

ParseAAARealm

ParseAAARealm is referenced from the NAS IncomingScript scripting point. It looks for a realm name on the username attribute as a hint of which AAA service should be used for this request. If @<realm> is found, the AAA service is selected which has the same name as the realm.

ParseAAASRealm

ParseAAASRealm is referenced from the NAS incoming script extension point. ParseAAASRealm looks for a realm name on the username attribute as a hint of which AAA service and which SessionManager should be used for this request. If @<realm> is found, the AAA service and SessionManager which have the same name as the realm are selected.

ParseAARealm

ParseAARealm is referenced from the NAS IncomingScript scripting point. It looks for a realm name on the username attribute as a hint of which authentication and authorization service should be used for this request. If @<realm> is found, it selects the AA service that has the same name as the realm and the DefaultAccountingService (as specified in the configuration by the administrator).

The Tcl version of this script is named tParseAARealm.

ParseAASRealm

ParseAASRealm is referenced from the NAS IncomingScript scripting point. It looks for a realm name on the username attribute as a hint of which AA service and which SessionManager should be used for this request. If @<realm> is found, the AA service and the SessionManager which have the same name as the realm are selected. The Accounting service will be the DefaultAccountingService (as specified in the configuration by the administrator).

The Tcl version of this script is named tParseAASRealm.

ParseProxyHints

ParseProxyHints is referenced from the NAS IncomingScript scripting point. It looks for a realm name on the username attribute as a hint of which AAA services should be used for this request. If @radius is found, a set of AAA services is selected which will proxy the request to a remote radius server. If @tacacs is found, the AuthenticationService is selected that will proxy the request to a tacacs server for authentication. For any services not selected, the default service (as specified in the configuration by the administrator) will be used.

The Tcl version of this script is named tParseProxyHints.

ParseServiceAndAAARealmHints

ParseServiceAndAAARealmHints is referenced from the NAS IncomingScript scripting point. It calls both ParseServiceHints and ParseAAARealm.

The Tcl version of this script is named tParseServiceAndAAARealmHints.

ParseServiceAndAAASRealmHints

ParseServiceAndAAASRealmHints is referenced from the NAS IncomingScript scripting point. It calls both ParseServiceHints and ParseAAASRealm.

The Tcl version of this script is named tParseServiceAndAAASRealmHints.

ParseServiceAndAARealmHints

ParseServiceAndAARealmHints is referenced from the NAS IncomingScript scripting point. It calls both ParseServiceHints and ParseAARealm.

The Tcl version of this script is named tParseServiceAndAARealmHints.

ParseServiceAndAASRealmHints

ParseServiceAndAASRealmHints is referenced from the NAS IncomingScript scripting point. It calls both ParseServiceHints and ParseAASRealm.

The Tcl version of this script is named tParseServiceAndAASRealmHints.

ParseServiceAndProxyHints

ParseServiceAndProxyHints is referenced from the NAS IncomingScript scripting point. It calls both ParseServiceHints and ParseProxyHints.

The Tcl version of this script is named tParseServiceAndProxyHints.

ParseServiceHints

ParseServiceHints is referenced from the NAS IncomingScript scripting point. Check to see if we are given a hint of the service type or the realm. If so, set the appropriate attributes in the request or radius dictionary to record the hint and rewrite the username to remove the hint.

The Tcl version of this script is named tParseServiceHints.

ParseTranslationGroupsByCLID

ParseTranslationGroupsByCLID is referenced from the policy engine to determine the incoming and outgoing translation groups based on CLID set in the policy engine so that the attributes can be added and/or filtered out by the configuration data set in MCD.

ParseTranslationGroupsByDNIS

ParseTranslationGroupsByDNIS is referenced from the policy engine to determine the incoming and outgoing translation groups based on realm set in the policy engine so that the attributes can be added/filtered out by the configuration data set in MCD.

ParseTranslationGroupsByRealm

ParseTranslationGroupsByRealm is referenced from the policy engine to determine the incoming and outgoing translation groups based on the realm set in the policy engine. ParseTranslationGroupsByRealm allows the attributes to be added or filtered out by the configuration data set in MCD.

UseCLIDAsSessionKey

UseCLIDAsSessionKey is used to specify that the Calling-Station-Id attribute should be used as the session key to correlate requests for the same session. This is a typical case for 3G mobile user session correlation.

USRIncomingScript

USRIncomingScript maps USR-proprietary attributes to Prime Access Registrar's global attribute space.

USRIncomingScript-IgnoreAccountingSignature

USRIncomingScript-IgnoreAccountingSignature maps USR-proprietary attributes to Prime Access Registrar's global attribute space and sets a flag to ignore the signature on Accounting-Request packets. Earlier versions of the USR RADIUS client did not correctly sign Accounting-Request packets.

USROutgoingScript

USROutgoingScript maps USR attributes from Prime Access Registrar's global attribute space to the appropriate USR-proprietary attributes.

Internal Scripts

Prime Access Registrar allows you to write internal scripts, by which you can add, modify, or delete attributes in the request, response, and environment dictionaries for RADIUS, Diameter, and TACACS+. You can use the Prime Access Registrar GUI or CLI to configure the internal scripts.

Prime Access Registrar allows you to create script statements for the following operations:

- Simple Attribute Operation—allows you to add, modify, or delete an attribute value to the request, response, or environment dictionary
- Copy an Attribute—allows you to copy an attribute value from one dictionary to another
- Concatenate Operation—allows you to concatenate an attribute value from one dictionary to another
- Replace Operation—allows you to replace an attribute value from one dictionary to another
- Value Based Manipulations—allows you to manipulate attribute values in a dictionary based on a given text

- Log or Trace Messages—allows you to create different levels of log or trace messages
- I can do it myself—allows you to create your own script for the selected protocol

You can also use internal scripts as part of the FastRules workflow. For more information about FastRules, see "Using FastRules to Process Packet Flow" chapter of the *Cisco Prime Access Registrar 8.0 User Guide.*

To configure internal scripts using the Prime Access Registrar GUI, see the "Using the Graphical User Interface" chapter of the *Cisco Prime Access Registrar 8.0 User Guide*.

CLI to Configure Internal Scripts

A sample CLI to configure internal script statements is given below:

```
--> cd /r/scripts/test

[ //localhost/Radius/Scripts/test ]
    Name = test
    Description =
    Language = internal
    Statements/

--> cd statements/

[ //localhost/Radius/Scripts/test/Statements ]
    1. #req:User-Name=~(.*)(@[a-z]+.[a-z]+)~\1

[ //localhost/Radius/Scripts/test1 ]
    Name = test1
    Description =
    Language = internal
    Statements/

--> cd statements/

[ //localhost/Radius/Scripts/test1/Statements ]
    1. -rsp:Framed-IP-Address=1.1.1.1

-->

[ //localhost/Radius/Scripts/test3/Statements ]
    1. +rsp:Tacacs-AVpair=cmd=show running-config
    2. +rsp:Tacacs-AVPair=aaa
    3. -rsp:Tacacs-AVPair=aaa

-->

[ //localhost/Radius/Scripts/test4/Statements ]
    1. -req:Cisco-AVPair=bbb

--> cd ../../test7/statements/

[ //localhost/Radius/Scripts/test7/Statements ]
    1. #rsp:Framed-IPX-Network=2
    2. +rsp:State=Delivered
    3. -rsp:State
    4. +req:Cisco-AVPair=aaaa
    5. #req:Cisco-AVPair=5
    6. #rsp:Framed-IPX-Network=req:Cisco-AVPair
    7. -req:Cisco-AVPair
```

Blacklisting Script

Prime Access Registrar supports two types of blacklisting:

- IMSI-Based Blacklisting, page 7-16
- IP-Based Blacklisting, page 7-17

Note One instance of Prime Access Registrar can have only one type of blacklisting; either IMSI-based or IP-based.

Blacklisting support is available for the following:

- Diameter remote server—You can choose to configure blacklisting as part of the outgoing script of the remote server
- SIGTRAN-M3UA remote server—You can choose to configure blacklisting as part of the global title translation script of the remote server.

IMSI-Based Blacklisting

Prime Access Registrar allows you to blacklist one or more IMSI values available in the incoming EAP-SIM or EAP-AKA requests. A scripting point option is provided such that you can set an environment dictionary variable Blacklisted-IMSI to ***TRUE*** or ***FALSE*** to blacklist or whitelist IMSI values respectively. An IMSI value marked as blacklisted is rejected and will not be processed further.

The IMSI-based blacklisting script is shown below:

```
proc CheckBlackList {request response environ}
{
    set imsi [ $environ get IMSI ]
    if { [ string compare $imsi 984579621012345 ] == 0 }
    {
        $environ put Blacklisted-IMSI TRUE
        $environ put Notification-Code 19384
    }
}
```

Where, ***CheckBlackList*** is the entrypoint variable of the global title translation script *checklist*, as shown in the example below:

```
[ //localhost/Radius/Scripts/checklist ]
    Name = checklist
    Description =
    Language = tcl
    Filename = tclscript.tcl
    EntryPoint = CheckBlackList
    InitEntryPoint =
    InitEntryPointArgs =
```

If the environment variable *Blacklisted-IMSI* is set as ***TRUE*** and if the IMSI value available in the incoming script matches the given string, then that IMSI is blacklisted and will not be processed. You can configure a notification code to represent failure. If no notification code is set, 16384 representing *General Failure* is sent upon rejection of an IMSI value. For more information about the Notification-Code variable, see Cisco Prime Access Registrar 8.0 Reference Guide

IP-Based Blacklisting

Prime Access Registrar allows you to blacklist one or more IP addresses available in the incoming EAP-SIM or EAP-AKA requests. A scripting point option is provided such that you can set an environment dictionary variable Blacklisted-Key to the IP address that you want to check. An IP addressed mapped to the Blacklisted-Key variable is rejected and will not be processed.

Run the following script:

```
proc CheckIPRange {request response environ}

{
$environ put "Blacklisted-Key" [ $request get Framed-IP-Address ]
}
```

Where:

- **Blacklisted-Key** is the environment variable that is mapped with the IP address to be checked
- ***CheckIPRange*** is the entrypoint variable of the global title translation script *checkIPList*, as shown in the example below:

```
[ //localhost/Radius/Scripts/CheckIPRange ]
    Name = CheckIPRange
    Description =
    Language = rex
    Filename = librexblacklist.so
    EntryPoint = CheckIPList
    InitEntryPoint = InitIPList
    InitEntryPointArgs =
/opt/CSCOar/logs/Whitelist.txt,Blacklisted-Key,Whitelist,10
```

If the IP address available in the incoming script is mapped as Blacklisted-Key, then that IP address is blacklisted and will not be processed further. You can configure a notification code to represent failure. If no notification code is set, 16384 representing *General Failure* is sent upon rejection of an IP address. For more information about the Notification-Code variable, see
Cisco Prime Access Registrar 8.0 Reference Guide

Extension Points in Cisco Prime Access Registrar

Table 7-1 lists the scripting points used in Prime Access Registrar. You can find the scripting point information in the file /cisco-ar/examples/rexscript/rex.h.

Note the following:

- All the scripting points can be used for processing packets of type RADIUS, Diameter, or TACACS+.
- The packet type—RADIUS, Diameter, or TACACS+—can be obtained from the ‘Request-Type’ or ‘Response-Type’ attribute of the environment dictionary. For more details, see
  Cisco Prime Access Registrar 8.0 Reference Guide

Table 7-1 Scripting Points in Prime Access Registrar

Service	Scripting Point Number	Scripting Point	Description	Sample Use Case (if applicable)
—	1	Server incoming Command-line interface (CLI) configuration path: /Radius/IncomingScript	Script runs for every request packet.	To inspect each packet and determine the kind of service (authentication, authorization, or accounting) to be provided.
		Remote Server incoming CLI configuration path: /Radius/RemoteServers/<remote server name>	Script runs when a packet is received from a remote server.	To add/modify/delete attributes coming from the remote server response.
—	2	Vendor incoming CLI configuration path: /Radius/Vendors/<vendorname>/IncomingScript	Script runs only for requests from all the clients mapped to a particular vendor.	To translate vendor proprietary attributes to RADIUS/Diameter vendor-specific attributes.
—	3	Client incoming CLI configuration path: /Radius/Clients/<clientname>/IncomingScript	Script runs only for requests from the specified client (router, NAS, and so on).	To choose AAA service based on an NAS IP address.
—	5	Service incoming CLI configuration path: /Radius/Services/<servicename>/IncomingScript	Script runs when a service is called.	To add/modify/delete attributes before calling a particular service.
Local	6	User group authentication CLI configuration path: /Radius/UserGroups/<name>/AuthenticationScript	Script runs for authentication requests for a user belonging to a particular user group.	—
	7	User record authentication CLI configuration path: /Radius/UserLists/<Userlistname>/<username>/AuthenticationScript	Script runs for authentication requests for a particular user.	—
	8	User group authorization CLI configuration path: /Radius/UserGroups/<name>/AuthorizationScript	Script runs for authorization requests for a user belonging to a particular user group.	—
	9	User record authorization CLI configuration path: /Radius/UserLists/<Userlistname>/<username>/AuthorizationScript	Script runs for authorization requests for a particular user.	—

Table 7-1 Scripting Points in Prime Access Registrar (continued)

Service	Scripting Point Number	Scripting Point	Description	Sample Use Case (if applicable)
—	10	Service outgoing CLI configuration path: /Radius/Services/<servicename>/OutgoingScript	Script runs after a service is executed.	To add/modify/delete attributes before sending the response.
—	12	Client outgoing CLI configuration path: /Radius/Clients/<clientname>/OutgoingScript	Script runs when sending a response against the request received from a specific client (router, NAS, and so on.)	To add/modify/delete attributes before sending a response to the particular client.
—	13	Vendor outgoing CLI configuration path: /Radius/Vendors/<vendorname>/OutgoingScript	Script runs when sending a response against the requests received from all the clients mapped to a particular vendor.	To translate RADIUS/Diameter vendor-specific attributes to the appropriate vendor proprietary attributes.
—	14	Server outgoing CLI configuration path: /Radius/OutgoingScript	Script runs for every response packet.	To add/modify/delete attributes before sending the response.
		Remote Server outgoing CLI configuration path: /Radius/RemoteServers/<remote server name>	Script runs when a packet is sent out to a remote server.	To add/modify/delete attributes in a packet going to a remote server.
Remote	16	Remote server outage CLI configuration path: /Radius/Services/<service name>	Script runs when the remote server is down.	To add/modify/delete attributes when the remote servers defined in the service are down.
Session Manager	19	Session manager incoming CLI configuration path: /Radius/SessionManagers/<session manager name>	Script runs to direct requests to be processed by a specific session manager after authentication and authorization.	To add attributes to the cache for a user before the session manager is called.
	20	Session manager outgoing CLI configuration path: /Radius/SessionManagers/<session manager name>	Script runs to direct responses from a specific session manager after authentication and authorization.	To translate the session cache attribute before sending the response.

Table 7-1 Scripting Points in Prime Access Registrar (continued)

Service	Scripting Point Number	Scripting Point	Description	Sample Use Case (if applicable)
—	21	Virtual server outgoing	Script set any time during packet run time and runs after Server Outgoing scripting point. Can be triggered from any other scripting point by setting the environment variable **Virtual-Server-Outgoing-Script**. To learn more about environment variables, see Cisco Prime Access Registrar 8.0 R eference Guide. This scripting point is applicable only for RADIUS packets.	—
SIGTRAN/ M3UA	22	Global title translation CLI configuration path: /Radius/RemoteServers/<re mote server name>	Script runs before sending out request to Home Registration Register (HLR) to perform Global Title Translation (GTT).	To define GTT based on incoming International Mobile Subscriber Identity (IMSI) ranges.
EAP-AKA	23	Quintet generation CLI configuration path: /Radius/Services/<EAP-AK A service name>	Script runs to generate quintets by using a simulator for EAP-AKA service.	To fetch quintet information from a configured file based on the incoming IMSI.
Translation	24	IMSI translation CLI configuration path: /Radius/RemoteServers/<re mote server name>	Script runs to change the incoming IMSI.	To modify the incoming IMSI ranges before sending a request to a Signal Transfer Point (STP)/HLR.
FastRules	25	FastRule	Script is configured within a fast rule execution path.	To add an attribute in the same flow after authentication and authorization of a packet.
Request Translation	26	Prerequest translation CLI configuration path: /Radius/Services/<service name>	To add/modify/delete incoming RADIUS/Diameter attribute values in the request before translation.	To add/modify/delete incoming RADIUS/Diameter attribute values in the request before translation.
	27	Postrequest translation CLI configuration path: /Radius/Services/<service name>	To add/modify/delete translated Diameter/RADIUS attributes in the request after translation.	To add/modify/delete translated Diameter/RADIUS attributes in the request after translation.

Table 7-1 Scripting Points in Prime Access Registrar (continued)

Service	Scripting Point Number	Scripting Point	Description	Sample Use Case (if applicable)
Response Translation	28	Preresponse translation CLI configuration path: /Radius/Services/<service name>	To add/modify/delete Diameter/RADIUS attribute values in the response before translation.	To add/modify/delete Diameter/RADIUS attribute values in the response before translation.
	29	Postresponse translation CLI configuration path: /Radius/Services/<service name>	To add/modify/delete RADIUS/Diameter attributes in the response after translation.	To add/modify/delete RADIUS/Diameter attributes in the response after translation.

Testing the RADIUS Server

This chapter describes how to use **radclient**, a RADIUS server test tool you run from the command line to test your Cisco Prime Access Registrar RADIUS server. You can use **radclient** to create packets, send them to a specific server, and examine the response.

Because the **radclient** command is Tcl-based, you can use it interactively or you can execute it as a Tcl script file.

To run the **radclient** command, enter:

radclient

After you enter the **radclient** command, you must log into the RADIUS server and provide an administrator's username, such as admin, and the administrator's password.

This chapter contains the following sections:

- radclient Command Syntax
- Working with Packets
- Attributes
- Using radclient Test Commands

radclient Command Syntax

The **radclient** command syntax is:

radclient [**-C** <*clustername*>] [**-N** <*adminname*>] [**-P** <*adminpassword*>] [**-i**] [**-n**] [**-p** <*load_path*>] **[-v] [-z debug_flags]** [**-I** flag]

Valid flags are:

- **-C** <*clustername*>
- **-N** <*adminname*>
- **-P** <*adminpassword*>
- **-i**—Forces interactive mode
- **-n**—Skips loading **radclient.tcl**
- **-p <path>**—Specifies the load_path
- **-s**—Uses default cluster, admin user, and password

  If you delete the admin user or modify the admin user's password, this option will no longer work.

- **-S <file>**—Sources specified file
- -**v**—Prints version and exits
- -**I** *<0 or 1>*—Enables to set as IPv4 or IPv6 client. *0* specifies IPv4 client and *1* specifies IPv6 client

-z debug_flags—Specify debug levels. Debug flags must be of the format *X*=*n*, where *X* is the letter corresponding to the type of debug information you want to see, and *n* is the value. The higher the value, the more output. *X* can also be a string or a range of letters.

For example, the following command line sets the debug levels for A, B, and C to 3:

radclient -z ABC=3

The following example command line sets the debug levels for everything between A and Z inclusive and l to 5:

radclient -z A-Zl=5

Working with Packets

Using the **radclient** command, you can create packets (default or specific packets), view packets, send packets, read the value of packets, and delete packets.

This section contains the following topics:

- Creating Packets
- Creating CHAP Access-Request Packets
- Viewing Packets
- Sending Packets
- Creating Empty Packets
- Setting Packet Fields
- Reading Packet Fields
- Deleting Packets

Creating Packets

To create a basic RADIUS Access-Request packet, use the **radclient** command **simple**. This function creates a packet and fills in basic attributes. The syntax of the **simple** command is:

simple *<user_name> <user_password>*

For example, to create an Access-Request packet for user **bob** whose password is **bigDog**, enter:

simple bob bigDog

```
p001
```

The **radclient** command responds with `p001`, which is the identifier (name) of the newly created packet.

Creating CHAP Access-Request Packets

To create a CHAP Access-Request packet, use the **radclient** command **simple_chap**. The syntax of the **simple_chap** command is:

simple_chap <*user_name*> <*user_password*> <*use_challenge*>

<*use_challenge*> is a boolean that indicates whether to use the **CHAP-Challenge** attribute.

For example, to create a CHAP packet and use a <*use_challenge*>, enter:

simple_chap bob bigDog 1

```
p002
```

Viewing Packets

To view a packet or any other object, enter the object identifier at the **radclient** prompt. For example, to display packet `p001`, enter:

p001

```
Packet: code=Access-Request,id=0,length=0, attributes =
User-Name = bob
User-Password = bigDog
NAS-Identifier = localhost
NAS-Port = 0
```

Sending Packets

To send a packet, specify the packet identifier and enter the word **send**.

p001 send

You can optionally specify the host and port to which to send the packet. The default host is **localhost**, and the default port is **1812**.

When you want to send a packet to a different host and different port, you must specify them on the command line. For example, to send a packet to the RADIUS server `amazon`, at port number `1813`, enter:

p001 send amazon 1813

```
p002
```

When Prime Access Registrar receives a response to the packet you sent, it prints the response packet's object identifier before the **radclient** prompt returns.

The TCL variable *tries* determines how many times **radclient** retries to send the packet.

Creating Empty Packets

You can use **radclient** to **c**reate empty packets, them modify the packets to contain the appropriate fields. To create an empty packet, the syntax is:

packet <*packet-type*>

The optional <*packet-type*> argument can be the numerical RADIUS packet type identifier, such as `2`, or the string representation, such as `Access-Accept`:

packet 2

```
p00d
```

p00d

```
Packet: code = Access-Accept, id = 0, length = 0, attributes =
```

Setting Packet Fields

You can modify the value of a packet field using the following syntax:

<*packet-identifier*> **set** <*field*> <*value*>

<*packet-identifier*> is the packet number, such as `p001`.

<*field* > is the packet field you want to modify and can be one of the following:

- attrib—Set attributes in the packet; <*value*> is the attribute identifier.
- code— The packet type (such as Access-Request); <*value*> is either a numeric packet-type or the string representation (for example, 1 or Access Request).
- identifier— Set the packet ID; <*value*> is the numeric ID.
- length—Set the packet length; <*value*> is the numeric length.
- requestAuthenticator—Set the request authenticator; <*value*> is a hex string with a colon separating each byte.

<*value*> is either a numeric packet-type, the string representation, or the hex string with a colon separating each byte.

For example, to set the identifier field to `99`, enter:

p001 set identifier 99

```
99
```

p001

```
Packet: code = Access-Request, id = 99, length = 0, attributes =
        User-Name = bob
        User-Password = bigDog
        NAS-Identifier = localhost
        NAS-Port = 0
```

Reading Packet Fields

You can read (**get**) the value of any of the packet fields by using the syntax:

<*packet-identifier*> **get** <*attrib*>

For example, to **get** the **identifier** field, enter:

p001 get identifier

```
99
```

Deleting Packets

When you are writing long-running or iterating scripts, you might want to conserve memory by deleting packets when you are finished with them.

To delete a packet, enter:

<*packet-identifier*> **delete**

To delete all resources referred to by the packet `p001`, enter:

p001 delete

Attributes

Using the **radclient** command you can create specific RFC-defined attributes of requests and responses.

This section contains the following topics:

- Creating Attributes
- Setting Multivalued Attributes
- Viewing Attributes
- Getting Attribute Information
- Deleting Attributes
- Using the radclient Command

Creating Attributes

To create an attribute object, the syntax is:

<*attrib*> **name** <*value*>

<*attrib*> is a recognized RADIUS attribute name. <*value*> is the value of the attribute.

For example, to create the attribute **User-Name** and set its value to `bob`, enter:

attrib User-Name bob

```
a001
```

Note **a001** is the object identifier for the newly created attribute.

Setting Multivalued Attributes

Prime Access Registrar supports setting multivalued attributes (MVAs) in **radclient**. Use the set **mattrib** command to set multivalued attributes, as shown in the following example:

simple bob bob

```
p001
```

attrib cisco-avpair blah

```
a005
```

attrib cisco-avpair boo

```
a006
```

p001 set mattrib a005

p001

```
Packet: code = Access-Request, id = 0, length = 0, attributes =
User-Name = bob
User-Password = bob
NAS-Identifier = localhost
NAS-Port = 1
Cisco-AVPair = blah
```

p001 set mattrib a006

p001

```
Packet: code = Access-Request, id = 0, length = 0, attributes =
User-Name = bob
User-Password = bob
NAS-Identifier = localhost
NAS-Port = 1
Cisco-AVPair = blah
Cisco-AVPair = boo
```

Viewing Attributes

To view an attribute, or any other object, type the object identifier at the **radclient** prompt. For example, to display attribute `a001` created in the example above, enter:

a001

```
User-Name = bob
```

Getting Attribute Information

You can get the name and value of an attribute in various formats:

- get name—gets the name as a string
- get value—gets the value as a string
- get type—gets the name as an integer
- get valueAsInt—gets the value as an integer
- get valueAsIPAddress—gets the value as an IP address.

The following examples show how to get an attribute's name, type, value, and value as integer:

a001 get name

```
User-Name
```

a001 get type

```
1
```

a001 get value

```
bob
```

a001 get valueAsInt

```
a001: the value is not an int
```

Deleting Attributes

When you are writing long running or iterating scripts, you might want to conserve memory by deleting attributes when you are finished with them (be sure not to delete attributes being referred to by other objects, like packets.)

To delete all resources referred to by the attribute `a001`, enter:

a001 delete

Using the radclient Command

The following three examples show how to use **radclient** to create, send, and modify packets.

Example 1

This example creates an Access-Request packet for user `jane` with password `jane`, and sends it to the default RADIUS server (**localhost**).

simple jane jane

```
p001
```

The command **simple jane jane** creates the packet; the packet object identifier is **p001**. When you enter the packet object identifier, **radclient** displays the contents of the packet.

p001

```
Packet: code = Access-Request, id = 0, length = 0, attributes =
    User-Name = jane
    User-Password = jane
    NAS-Identifier = localhost
    NAS-Port = 0
```

When you enter the packet identifier and the command **send**, **radclient** sends the packet to the RADIUS server and prints the response packet object identifier.

p001 send

```
p002
```

When you enter the packet object identifier of the response, **radclient** displays the contents of the response packet.

p002

```
Packet: code = Access-Accept, id = 1, length = 38, attributes =
Login-IP-Host = 204.253.96.3
Login-Service = Telnet
Login-TCP-Port = 541
```

Example 2

The following example creates a simple Access-Request packet, then adds other attributes to it.

simple jane jane

```
p003
```

The command line above shows creation of the packet `p003` using user-ID `jane` and password `jane`.

attrib Service-Type Framed

```
a00c
```

The line above shows creating the **Service-Type** attribute (with the object identifier `a00c`).

a00c

```
Service-Type = Framed
```

Entering the attribute object identifier `a00c` displays the attribute object.

p003 set attrib a00c

The line above adds the newly set attribute to the packet. The following line creates another attribute.

attrib NAS-Port 99

```
a00d
```

a00d

```
NAS-Port = 99
```

p003 set attrib a00d

The same steps add the **NAS-Port** attribute to the packet, and finally, the packet contents are displayed.

p003

```
Packet: code = Access-Request, id = 0, length = 0, attributes =

User-Name = jane
User-Password = jane
NAS-Identifier = localhost
Service-Type = Framed
NAS-Port = 99
```

Example 3

Example 3 performs the same tasks as Example 2 using the command substitution feature of Tcl which allows you to use the results of one command as an argument to another command. Square brackets invoke command substitution. The statement inside the brackets is evaluated, and the result is used in place of the bracketed command.

simple jane jane

```
p004
```

p004 set attrib [attrib Service-Type Framed]

p004 set attrib [attrib NAS-Port 99]

p004

```
Packet: code = Access-Request, id = 0, length = 0, attributes =
        User-Name = jane
        User-Password = jane
        NAS-Identifier = localhost
        Service-Type = Framed
        NAS-Port = 99
```

Using radclient Test Commands

You can use the **radclient** commands **timetest** and **callsPerSecond** to test the RADIUS server.

This section contains the following topics:

- radclient Variables
- Using timetest
- Using callsPerSecond
- Additional radclient Variables

radclient Variables

You control how **timetest** and **callsPerSecond** work using **radclient** variables. To set a **radclient** variable, use the **set** command as follows:

set ***variable value***

Table 8-1 lists the **radclient** variables used in **timetest** and **callsPerSecond** and their description.

Table 8-1 radclient Variables

Variable	Description
host	Destination host to send the packets (default is localhost)
num_packets	Number of packets to send at once (default is 256)
num_users	Modulus for the username pattern (default is 10000)
port	Port where **radclient** sends access-request packets (default is 1812). Changing this port does not affect the accounting_port.
retry_timeout	Length of time to wait after a timeout occurs before retrying
secret	Shared secret configured on the RADIUS server for the client (default is secret)
timeout	Length of time to wait before a timeout occurs
tries	Number of times to attempt to send
UserNamePattern	Pattern of the usernames (default is user%d%%PPP)
UserPasswordPattern	Pattern of the user passwords (default is user%d)

Using timetest

The **timetest** command sends a number of requests to the RADIUS server then waits for a response. When a response arrives, **timetest** immediately sends another request. **timetest** can keep up to 256 requests outstanding all the time.

The syntax of the **timetest** command is:

timetest <*testtype*> [<*cycles*> [<*repetitions*> [<*starting user number*> [<*increment user number*>]]]]

Table 8-2 lists the applicable test types.

Table 8-2 Test Types

Test Type	Description
1	Access-Request
2	Access-Request + Accounting-Start + Accounting-Stop
3	Accounting-Start + Accounting-Stop
4	Ascend-IPA-Allocate + Ascend-IPA-Release
5	Access-Request + Ascend-IPA-Allocate + Ascend-IPA-Release
6	Access-Request + Ascend-IPA-Allocate + Accounting-Start + Ascend-IPA-Release + Accounting-Stop
7	Access-Request + USR-Resource-Free-Request
8	LEAP Identity + LEAP-Challenge Response + LEAP Challenge
9	LEAP Identity + LEAP-Challenge Response + LEAP Challenge + Accounting-Start + Accounting-Stop
10	Access-Request + Accounting-Start + Accounting-Stop with Home-Agent request
11	Access-Request + Accounting-Start + Accounting-Stop with ODAP request

Consider this **timetest** example with **radclient** variables set to the following:

host—1.1.1.2

port—1812

secret—cisco

UserNamePattern—user%d

UserPasswordPattern—puser%d

num_users—100,000

num_packets—128

In this example, **timetest** sends packets directly to the host at IP address 1.1.1.2 on port 1812 with a shared secret `cisco`. There are 100,000 users in the server's user database with the name pattern *user#* and password pattern *puser#*, where # ranges from 0-99,999, inclusive. The number of outstanding requests are limited to 128.

Before starting the timing test, **timetest** sends an Accounting-On packet to the AAA Server and waits for a response to make sure that any session management being performed on the AAA Server is reset before running the test. After a response is received, the **timetest** can begin.

Using callsPerSecond

The **callsPerSecond** command is a smart throttle that sends packets at a rate you set. If you set **callsPerSecond** to two transactions per second (TPS), **callsPerSecond** sends a packet every 0.5 seconds.

The syntax of the **callsPerSecond** command is:

callsPerSecond <*callsPerSecond*> <*testtype*> [<*cycles*> [<*repetitions*> [<*starting user number*> [<*increment user number*>]]]]

Additional radclient Variables

Table 8-3 lists additional **radclient** variables and their description.

Table 8-3 Additional radclient Variables

Variable	Description
accounting_port	Port where the RADIUS server sends accounting packets (default is 1813). **Note** Changing accounting_port value does not affect the authentication port.
host	Name of host where Prime Access Registrar is installed
ignore_signature_errs	Causes server to ignore signature in the response
load_path	Search path to load source files with user processes
NASIdentifier	Value to set NAS-Identifier attribute
NASIPAddress	Value to set NAS-IP-Address attribute
NASPort	Value to set NAS-Port attribute
num_packets	Number of packets to send at once (default is 256)
num_users	Modulus for the username pattern (default is 10000)
port	Port where **radclient** sends access-request packets (default is 1812). Changing this port does not affect the accounting_port.
retry_timeout	Length of time to wait before attempting a retry
secret	Shared secret configured on the RADIUS server for the client (default is secret)
tclDefaultLibrary	Tclsh default library
tcl_patchLevel	Tclsh version with patch level
tcl_pkgPath	Tclsh install path
tcl_traceExec	Tclsh boolean to activate tracing
tcl_platform	Tclsh platform array
tcl_version	Tclsh version
tries	Number of retry attempts
UserNamePattern	Pattern of the usernames (default is user%d%%PPP)
UserPasswordPattern	Pattern of the user passwords (default is user%d)
verbose	Verbose flag for Tclsh

Using Trusted ID Authorization with SESM

Cisco Prime Access Registrar (Prime Access Registrar) can be used in a Service Selection Gateway (SSG) - Cisco Subscriber Edge Services Manager (SESM) deployment to enable the Trusted Identity (Trusted ID) Authorization feature. This chapter describes how to use Prime Access Registrar with SESM, and how to configure Prime Access Registrar to use the Trusted ID feature.

The Trusted ID feature provides transparent login capabilities for users based on a trusted ID instead of the user's name, enabling end users of an SSG to maintain an always-on connection without the need to authenticate on each connect. Using SSG's Transparent Auto-Login (TAL) feature, a TAL access-request packet contains a Trusted ID, such as a MAC address, that identifies the user without the user's real username and password.The *SESM Profile Management Guide* provides detailed information about Trusted ID authorization in SESM.

If Prime Access Registrar knows the user associated with the Trusted ID, Prime Access Registrar uses the Trusted ID to authenticate and authorize the user. If the authentication and authorization succeeds, Prime Access Registrar returns the user's username in the Access-Accept so the SSG can include the user's identity in subsequent Accounting-Requests.

If Prime Access Registrar does not know the user associated with the Trusted ID,
Prime Access Registrar returns an Access-Reject. The Access-Reject causes the SSG to redirect the user to a SESM web portal login page. When the user explicitly authenticates, Prime Access Registrar captures the Trusted ID and maps it to a user association so subsequent attempts to authenticate with the Trusted ID succeed.

This chapter contains the following sections:

- Trusted ID Operational Overview
- Software Requirements
- Configuring Cisco Prime Access Registrar for Trusted Identity with SESM
- Configuration Imported by TrustedIdInstall Program
- Configuring EAP-MD5 Authentication

Trusted ID Operational Overview

This section describes the following operations of the Trusted ID Authentication feature:

- Configuration Overview
- Request Processing
- Session Cache Life Cycle

- Configuration Restrictions

Configuration Overview

The Trusted ID features require two objects in Prime Access Registrar, a UserService, a SessionManager, and a ResourceManager. The UserService references another service called to perform the authentication and authorization (AA). The SessionManager references a SessionManager that contains a reference to a session-cache Resource Manager. These objects are imported into the Prime Access Registrar server configuration when you run the **TrustedIdInstall.bin** program. Configuration Imported by TrustedIdInstall Program, page 9-13 lists the configuration imported into the Prime Access Registrar server by the **TrustedIdInstall.bin** program.

The Resource Manager is configured with the QueryKey property set to a RADIUS attribute that contains the Trusted ID such as the Calling-Station ID. The Query Key should be set to an attribute present in all appropriate AA requests that uniquely identifies the user such as Calling-Station ID. The Query Key can be set to only one RADIUS attribute.

The Resource Manager is also configured to cache the attributes required to identify the user, username, and the user's credentials, password or CHAP-Password and CHAP-Challenge. The attributes User-Name, User-Password, NAS-Identifier, NAS-Port, or NAS-Port-Type are not appropriate choices for Query Key because they do not uniquely identify users.

The RollingEncryptionKeyChangePeriod specifies the length of time a given EncryptionKey will be used before a new one is created. When the session-cache ResourceManager caches User-Password attributes, Prime Access Registrar encrypts the User-Password so it is not stored in memory or persisted on disk in clear text. Prime Access Registrar uses up to 255 encryption keys, using a new one after each RollingEncryptionKeyChangePeriod expires. If RollingEncryptionKeyChangePeriod is set to *2 days*, Prime Access Registrar will create and begin using a new EncryptionKey every two days. The oldest key will be retired, and Prime Access Registrar will re-encrypt any User-Passwords that used the old key with the new key. This way, if the RollingEncryptionKeyChangePeriod is set to *1 day*, no key will be older than 255 days.

The encryption keys are indirectly connected to Trusted ID. Since User-Passwords might be stored for a long time in memory and on disk, Prime Access Registrar uses the RollingEncryptionKey to encrypt the User-Passwords. The RollingEncryptionKey makes it more difficult for someone to crack or decode the User-Passwords because the key used changes frequently. If someone were to break one key, that would only give them the ability to decrypt those User-Passwords that had been encrypted with that key. All others, including those yet to be encrypted after the key change period expires would not be vulnerable.

Request Processing

When the Trusted ID service processes Access-Requests, it queries the session-cache Resource Manager for a cache entry associated with the Trusted ID. If found, the Resource Manager returns the cached attributes. The Trusted ID service replaces the request's existing attributes with the cached attributes.

After the Resource Manager is queried (and the request's existing attributes are replaced with the cached attributes if the cache entry exists), the Trusted ID's UserService authenticates and authorizes the request. The UserService is always called whether the cache entry exists or not. The only attributes cached in the Resource Manager are the ones listed in AttributesToBeCached. The user profile is usually not cached and is retrieved each time by the UserService.

Whether the request succeeds or not, the request is passed on to the service referenced by the UserService property. When that service completes authentication and authorization, control returns to the Trusted ID service. The session-cache might be updated if AA is successful.

Session Cache Life Cycle

Session cache management comprises adding and deleting Trusted ID to user mapping to and from the cache and is initiated from the Trusted ID service. The mapping is one-to-one mapping. For each Trusted ID, there can be only one cache entry, and conversely for each cache entry, there can be only one Trusted ID.

If a user is not presently in the session cache (the query failed), the AA done by the UserService succeeded and the internal attribute (Implicit-Auth-Enabled) was returned with a value of *true*, Prime Access Registrar adds the user to the cache. Since the AA succeeded, Prime Access Registrar assumes this is an explicit authentication by the user and the attributes required by the session-cache are present in the Access-Request.

If the user is already in the session cache (the query succeeded) and the AA done by the UserService failed, the internal attributes Implicit-Auth-Enabled was not returned, or was returned with a value other than true, Prime Access Registrar removes the user from the session cache.

If the user has enabled implicit authentication (and if that results in Implicit-Auth-Enabled being returned as *true*), after the first Explicit Auth (from the login page), the user will be in the cache and will always be implicitly authenticated and authorized. In this case, you can get them out of the cache three ways:

- Have the user disable implicit authentication, then reconnect
- Have the system administrator release the session using **aregcmd** commands
- Use the SessionTimeout property in the Session Manager

If the user's account becomes orphaned (the user no longer exists), the cache entry will persist until it is removed using **aregcmd**.

If you have disabled implicit authentication, you are forced to authenticate each time and the cache is not updated. If you subsequently enable implicit authentication, you must explicitly authenticate one more time to create the user's cache entry. After creating the user's cache entry, they will not need to explicitly authenticate again (with this instance of Prime Access Registrar) as long as implicit authentication is enabled.

Configuration Restrictions

The Session Manager referenced by the TrustedID Service should not be used for general session management. The Trusted ID Session Manager should be a separate Session Manager used only for the Trusted ID session cache. The data in the session-cache must persist longer than the length of the session. If the Trusted ID Session Manager was used for general session management, the cache would be updated for the general session, overwriting the cache entry for the special session created for the Trusted ID service. When the general session ended it would delete that data and subsequent queries for implicit authentication would fail.

Software Requirements

The Trusted ID feature requires the following software to be installed:

- Cisco Subscriber Edge Services Manager (SESM) 3.3(1)
- Cisco Subscriber Policy Engine (SPE) 2.1.12
- Cisco Prime Access Registrar

In addition to the software listed above, you must run **TrustedIdInstall.rpm**, a Java application that runs on the Linux platform.

Note The disk space required to run the **TrustedIdInstall** program is about 1.3 MB.

The **TrustedIdInstall** program verifies the software prerequisites, installs the required jar files, and extends the configuration for Prime Access Registrar. The **TrustedIdInstall** program is only available on Cisco.com under the Prime Access Registrar download area at the following URL:

http://www.cisco.com/cgi-bin/tablebuild.pl/access-registrar-encrypted

This section contains the following topics:

- Installing Cisco Prime Access Registrar
- Running the TrustedIdInstall Program

Installing Cisco Prime Access Registrar

See the *Cisco Prime Access Registrar 8.0 Installation Guide*. for detailed information about how to install Prime Access Registrar software.

Note You must specify a Java Runtime Environment (JRE) when you install Prime Access Registrar software.

Running the TrustedIdInstall Program

Cisco provides a Java-based program called **TrustedIdInstall** that installs required jar files, the configuration for Subscriber Policy Engine (SPE), and Prime Access Registrar. The **TrustedIdInstall** program can be run as an InstallShield wizard using the graphical user interface (GUI) or from the command line.

Note Before running the TrustedIDInstall program, ensure that the SPE 2.1.12 software has been installed with SESM 3.3(1) (in SPE mode).

Using the TrustedIdInstall.bin GUI

You must run the **TrustedIdInstall** program on the workstation where Prime Access Registrar is installed with a Java Runtime Environment (JRE) up to and including 1.4.2 in the path.

Installing the TrustedIdInstall

To install TrustedIdInstall:

Step 1 Log in as a user with root privileges.

Step 2 Enter the following from the Prime Access Registrar server's command line:

TrustedIdInstall.rpm

The following message appears after you enter the command line above:

```
# TrustedIdInstall.rpm
InstallShield Wizard

Initializing InstallShield Wizard...

Searching for Java(tm) Virtual Machine...
.............running under 1.2
```

Figure 9-1 shows the welcome window of the Trusted ID Azn AR SESM Integration 1.0 Installer.

Figure 9-1 Trusted ID Azn AR SESM Integration 1.0 Installer Welcome

Step 3 Click **Next** to continue.

The **InstallIdInstall.rpm** wizard displays the Prerequisites window.

Step 4 Check to ensure that Cisco SESM 3.3(1) is installed and available on the network, then click **Next**.

The **InstallIdInstall.rpm** wizard checks for Prime Access Registrar 5.1 software. You will need the SESM 3.3(1) configuration parameters later in this procedure.

Step 5 Select the vendor name of the LDAP data store you are using for SPE, then click **Next**.

The **InstallIdInstall.rpm** wizard displays the Password Encryption Panel. This panel prompts you for a master password (entered twice to ensure accuracy) and a Password Encryption Algorithm which can be None, SHA, or SHA-1.

Note If you plan to use EAP-MD5 authentication, choose **None**. See Configuring EAP-MD5 Authentication, page 9-15 for information about configuring EAP-MD5 authentication.

Step 6 Enter the password in field provided, and select the password encryption type, then click **Next**.

Step 7 If in **Step 5** you selected iPlanet as the Data Store Type, continue with **Step 8**. If you chose any other Data Store Type, proceed to **Step 9**.

The iPlanet Data Store Type requires that you set the value for the naming variable in **ACNSchema.xml** and **DESSSchema.xml**, either for Uid or Cn as shown in Figure 9-2. You can set the naming variable to either Uid or Cn.

Figure 9-2 Selecting iPlanet Naming Variable

Step 8 Select either **Uid** or **Cn** as the inetOrgPerson naming variable, then click **Next**.

The **InstallIdInstall.rpm** wizard displays the Service Type Selection panel.

Step 9 Accept the default Trusted ID Service Enable True or click to select False, then click **Next**.

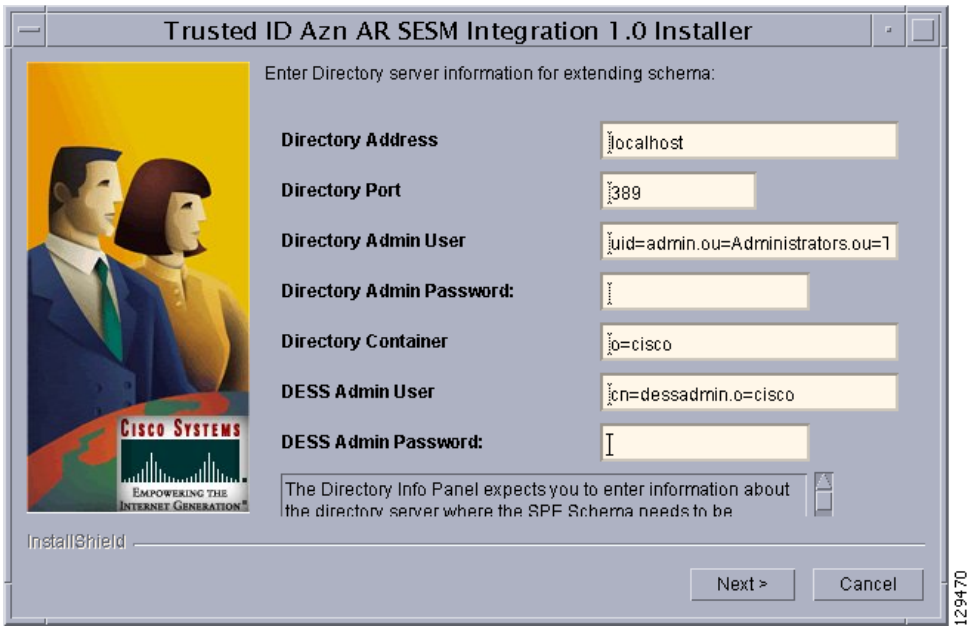
The TrustedIdInstall program displays a panel that indicates the following:

- Location where the Trusted ID Authorization SESM Integration files will be stored (/cisco-ar)
- Features to be stored (Admin Tool)
- Amount of space required (about 1.3 MB)

The **InstallIdInstall.rpm** wizard displays the Directory Information panel, requesting information about the directory server required to extend the schema.

Step 10 Provide the requested Directory Server information as shown in Figure 9-3.

Figure 9-3 Directory Server Information



Contact the directory administrator if you are unsure about the information required.

a. Enter a **Directory Address**.

   The Directory Address field requires the directory server IP address or DNS hostname.

b. Enter a **Directory Port** number.

   Provide the TCP/IP port on which your directory server listens. (This is usually port 389.)

c. Enter a **Directory Admin User**.

   Provide the User ID of the directory server administrator with permissions to extend the schema in the form:

   `cn=admin`

d. Enter a **Directory Admin Password**.

   Provide the password for the directory administrator user.

e. Enter a **Directory Container**.

   Provide the container in which the default RBAC objects should be created in the form:

   `ou=sesm,o=cisco`

f. Enter a **DESS Admin User**.

   Provide the User ID of the DESS administrator in the form:

   `uid=admin,ou=sesm,o=cisco`

g. Enter a **DESS Admin Password**.

   Provide the password for the DESS administrator.

Step 11 Click **Next** to continue.

The **InstallIdInstall.rpm** wizard begins the installation and displays a progress bar. When the installation completes, the wizard displays any warnings or errors it might have detected. Both boxes being empty indications a successful install.

Step 12 Click **Next** to continue.

A final window indicates a successful installation of the Trusted ID Authorization AR SESM Integration software.

Step 13 Click **Finish**.

Using the TrustedIdInstall Command Line

You can run the **TrustedIdInstall** program using the command line option on a workstation where Prime Access Registrar is installed with a JRE up to and including 1.4.2 in the path. The command line interface requires the same information as the GUI method.

Note You must be a root user to run the **TrustedIdInstall** program

Installing the TrustedIdInstall using Command Line

To install TrustedIdInstall using command line:

Step 1 To run the **TrustedIdInstall** program using the command line interface, enter the following from the Prime Access Registrar server's command line:

TrustedIdInstall.rpm -console

```
InstallShield Wizard

Initializing InstallShield Wizard...

Searching for Java (tm) Virtual Machine...
.............
-------------------------------------------------------------------------------

Welcome to the InstallShield Wizard for Trusted ID Azn AR SESM Integration.
The InstallShield Wizard will install Trusted ID Azn AR SESM Integration
 on your computer.
To continue, choose Next.
Trusted ID Azn AR SESM Integration1.0
Cisco Systems, Inc.
http://www.cisco.com

   Press 1 for Next panel, 3 to Cancel or 4 to Redisplay [1] 1
```

The line above provides a way for you to enter your selection. You can press **Enter** to go to the next panel. Enter 3 to cancel the installation, or enter 4 to redisplay the current panel.

Step 2 Press **Enter** to go to the next panel.

```
-------------------------------------------------------------------------------

Please read the information below.

Cisco Systems
 Prerequisites
Please ensure that minimally the following products are installed.
1 Check to ensure that Cisco SESM 3.3(1) is installed and available on the
network
2 Checking for Prime AR 6.0 or later
```

```
 Please ensure the configuration parameter supplied during SESM installation
is used in this integration.

   Press 1 for Next panel, 2 for Previous panel, 3 to Cancel or 4 to
   Redisplay [1] 1
```

This panel lists prerequisites required for successful installation. Before continuing to the next panel, ensure that SESM 3.3(1) is installed and available or the network. The program checks for Prime Access Registrar 3.5.3 (or later).

Step 3 After insuring that SESM 3.3(1) is installed and available on the network, press **Enter**.

```
[X]  1   -  Novell Directory Server
    [ ]      -  iPlanet
    [ ]      -  Data Communications Directory
    [ ]      -  IBM Directory Server
    [ ]      -  Open LDAP

   Choose the Vendor for Directory ,Select 0 to exit [0]

   Press 1 for Next panel, 2 for Previous panel, 3 to Cancel or 4 to
   Redisplay [1]
```

This panel requests the data store type selection and indicates the Novell Directory Server is the default selection.

Step 4 Press **Enter** to select the Novell Directory Server.

You can press **2** to select iPlanet, **3** to select Data Communications Directory, **4** to select IBM Directory Server, or **6** to select Open LDAP.

```
--------------------------------------------------------------------------------

Enter the master password for SPE

   Master Password []
```

This panel requests a master password for SPE.

Step 5 Enter a password to be used as the master password for SPE and press **Enter**.

You are asked to re-enter the master password. The following panel requests an encryption algorithm and generates a secret key using the master password and selected algorithm.

```
[X]  1   -  NONE
 [ ]     -  SHA
 [ ]     -  SSHA

   Choose the installation type for SPE ,Select 0 to exit [0]

   Press 1 for Next panel, 2 for Previous panel, 3 to Cancel or 4 to
   Redisplay [1] 1
```

This panel indicates the default installation type as None. Enter “2” and press **Enter** to select SHA, or enter “3” and press **Enter** to select SSHA.

Note If you plan to use EAP-MD5 authentication, choose **None**. See Configuring EAP-MD5 Authentication, page 9-15 for information about configuring EAP-MD5 authentication.

Step 6 If in **Step 4** you selected iPlanet as the Data Store Type, continue with **Step 7**. If you chose any other Data Store Type, proceed to **Step 8**.

```
--------------------------------------------------------------------------------

 [X]  1   -  Uid
 [ ]      -  Cn
--------------------------------------------------------------------------------
```

The iPlanet Data Store Type requires that you set the value for the naming variable in **ACNSchema.xml** and **DESSSchema.xml**, either for Uid or Cn as shown above.

Step 7 Press **Enter** to use the naming variable to Uid, or press **2** to select Cn.

```
Service Type Selection panel

Trusted ID Service Enable
 [X]  1  -  True
 [ ]  2  -  False

   To select a choice enter its number, or 0 when you are finished [0]:

   Press 1 for Next panel, 2 for Previous panel, 3 to Cancel or 4 to
   Redisplay [1] 1
```

The Service Type Selection panel asks if you want to enable the Trusted ID service. Enter 2 to choose to not enable the Trusted ID service.

Step 8 Press **Enter** to enable the Trusted ID service.

```
Trusted ID Azn AR SESM Integration will be installed in the following
location:
/cisco-ar
with the following features:
Admin tool
for a total size:
 1.3 MB

   Press 1 for Next panel, 2 for Previous panel, 3 to Cancel or 4 to
   Redisplay [1] 1
```

This panel indicates the location where the TrustedIdInstall program will write data and the amount of storage required.

Step 9 Press **Enter** to begin writing data.

```
--------------------------------------------------------------------------------

Enter the IP Address (or) hostname of the system where the directory server is
running.
Please contact your directory administrator if you are not sure about this
information.

   Please enter the host address  [localhost]:
```

Step 10 Press **Enter** to use the current system as the directory server, or enter another directory server name or IP address.

```
Enter the TCP/IP Port on which your directory server listens. Usually, the
port is 389.
Please contact your directory administrator if you are not sure about this
information.

   Please enter the Port number  [389]:
```

Step 11 Press **Enter** to use the default port, 389, or enter a different port number.

Note Contact your directory server administrator if you are not sure about which port to use or other information required in the following steps.

```
Enter the User Id of the directory server with permissions to extend schema.
Please contact your directory administrator if you are not sure about this
information.

   Please enter directory user
   [uid=admin,ou=Administrators,ou=TopologyManagement,o=NetscapeRoot]:
```

Step 12 Enter the User ID of the directory server administrator with the necessary permissions to extend the schema.

```
Enter the password for the above user.
Please contact your directory administrator if you are not sure about this
information.

   Please enter the password  []: cisco
```

Step 13 Enter the password for the user provided in the previous step.

```
Enter the container in which the default RBAC objects should be created.
Please contact your directory administrator if you are not sure about this
information.

   Please enter the container  [o=cisco]:
```

Step 14 Press **Enter** to use the default container, or enter a different container and press **Enter**.

```
Enter the User Id of the DESS user.

   Please enter Dess user  [cn=dessadmin,o=cisco]:
```

Step 15 Press **Enter** to use the default DESS user, or enter a different user ID and press **Enter**.

```
Enter the password of the DESS user.
Please contact your directory administrator if you are not sure about this
information.

   Please enter the Dess user password  []: cisco
```

Step 16 Enter the DESS user password, then press **Enter**.

```
Press 1 for Next panel, 3 to Cancel or 4 to Redisplay [1] 1
```

At this point, the software installation is ready to begin.

Step 17 Press **Enter** to begin the software installation and extend the schema.

As the installation proceeds, status messages will be displayed.

When the installation completes successfully, the following message displays:

```
Trusted ID Azn AR SESM Integration 1.0 installation completed

The InstallShield Wizard has successfully installed Trusted ID Azn AR SESM
Integration. Choose Finish to exit the wizard.

   Press 3 to Finish or 4 to Redisplay [3] 3
```

Step 18 Press **Enter** to end the program.

Configuring Cisco Prime Access Registrar for Trusted Identity with SESM

Use the command line interface **aregcmd** to configure Prime Access Registrar to use Trusted ID authorization in SSG-SESM deployments.

This section contains the following topics:

- Configuring the RADIUS Ports
- Configuring NAS Clients
- Configuring AAA and SPE Services

Configuring the RADIUS Ports

By default, Prime Access Registrar listens on ports 1812 and 1813 for any type of RADIUS request. It might be necessary to change the port assignments in the case of a resource collision. For example, if the RADIUS Directory Enabled Service Selection (DESS) Proxy (RDP) component of SPE is using ports 1812 and 1813, a port assignment change would be required.

The following command sequence causes Prime Access Registrar to listen on the explicitly defined ports, 1812 and 1813, for all types of RADIUS requests.

cd /Radius/Advanced/Ports

add 1812 ""radius

```
Added 1812
```

add 1813 ""radius

```
Added 1813
```

After changing the port assignments, Prime Access Registrar no longer listens on the default ports. It might be necessary to add ports 1812 and 1813 if you are also using Prime Access Registrar for other AAA functionality.

Note By default, Prime Access Registrar listens on ports 1812 and 1813 for the Linux platform.

Configuring NAS Clients

Change directory to **/Radius/Clients**, then add and configure the NAS clients required by SESM deployments:

cd /Radius/Clients

add SESM1 "" 10.3.3.2 cisco

```
Added SESM1
```

add SESM2 "" 10.3.3.101 cisco

```
Added SESM2
```

add SESM3 "" 10.3.3.102 cisco

```
Added SESM3
```

Configuring AAA and SPE Services

To configure AAA and SPE services:

Step 1 Change directory to **/Radius/Services**, then add and configure an accounting service.

cd /Radius/Services

add SESMaccounting "" file

```
Added SESMaccounting
```

Step 2 Change directory to **/Radius**, then configure a DefaultAccountingService.

cd /Radius

set DefaultAccountingService SESMaccounting

```
Set DefaultAccountingService SESMaccounting
```

Configuration Imported by TrustedIdInstall Program

The following is a listing of the configuration imported into the Prime Access Registrar server when you run the TrustedIdInstall program:

- /Radius
- /radius/services/spe
- /radius/services/trusted-id
- /Radius/SessionManagers/session-cache/
- /radius/ResourceManagers/session-cache
- /radius/advanced/
- /Radius/Scripts/ChangeServiceType

/Radius

```
DefaultAuthenticationService trusted-id
DefaultAuthorizationService trusted-id
```

/radius/services/spe

```
type java
ClassName com.cisco.cns.security.arspe.SPEExtension
```

/radius/services/trusted-id

```
type trusted-id
UserService spe
SessionManager session-cache
```

/Radius/SessionManagers/session-cache/

```
AllowAccountingStartToCreateSession FALSE
ResourceManagers/1 session-cache
```

/radius/ResourceManagers/session-cache

```
type session-cache
OverwriteAttributes TRUE
PendingRemovalDelay 10
QueryKey Calling-Station-ID
AttributesToBeCached/1 User-Name
AttributesToBeCached/2 User-Password
AttributesToBeCached/3 Calling-Station-ID
```

/radius/advanced/

```
ClasspathForJavaExtensions /cisco-ar/conf
```

/Radius/Scripts/ChangeServiceType

```
Language TCL
Filename ChangeServiceType.tcl
EntryPoint ChangeServiceType
IncomingScript ChangeServiceType
```

Configuring EAP-MD5 Authentication

EAP-MD5 authentication is an optional authentication configuration. The following configuration changes are required to support EAP-MD5 authentication:

- Creating the CheckEap.tcl Script
- Adding the CheckEap.tcl Script
- Using the CheckEap.tcl Script
- Adding the EAP-MD5 Authentication Service
- Adding an LDAP Remote Server
- Adding an LDAP Service
- Saving the Configuration and Reloading the Server
- Cisco SSG VSAs in Cisco Prime Access Registrar Dictionary

Note If you configure Prime Access Registrar to use EAP-MD5 authentication with the Trusted ID feature, you will not be able to use the Transparent Auto Login feature.

Creating the CheckEap.tcl Script

The **CheckEap.tcl** script must be created and stored in the file called **/cisco-ar/scripts/radius/tcl/CheckEap.tcl**. Use a text editor and copy the following lines into the **CheckEap.tcl** file:

```
proc CheckEap { request response environment } {
    if { [ $request containsKey EAP-Message ] } {
        $environ put Authentication-Service "EAP-MD5"
        $environ put Authorization-Service "spe"
    }
}
```

Adding the CheckEap.tcl Script

To add the CheckEap.tcl script:

Step 1 Start **aregcmd**, then change directory to **/Radius/Scripts** and add the CheckEap script.

cd /Radius/Scripts

add EapCheck

Step 2 Change directory to **EapCheck**.

cd EapCheck

```
[ //localhost/Radius/Scripts/EapCheck ]
        Name = EapCheck
        Description =
        Language =
```

Step 3 Set the Language property to TCL.

set Language TCL

```
Set Language TCL
```

Step 4 Set the filename property to CheckEap.tcl.

set Filename CheckEap.tcl

```
Set Filename CheckEap.tcl
```

Step 5 Set the EntryPoint property to CheckEap.

set EntryPoint CheckEap

```
Set EntryPoint CheckEap
```

Note The following sections also require you to use **aregcmd**, the command line interface.

Using the CheckEap.tcl Script

This section describes how to configure Prime Access Registrar to use the CheckEap script by setting the **/Radius/IncomingScript** property to CheckEap.

cd /Radius

set IncomingScript EapCheck

Adding the EAP-MD5 Authentication Service

To add and configure the EAP-MD5 service:

Step 1 Change directory to **/Radius/Services** and add an EAP-MD5 service.

cd /Radius/Services

add EAP-MD5

Step 2 Change directory to the EAP-MD5 service and set the Type and UserService properties as shown below:

cd EAP-MD5

Step 3 Change directory to the EAP-MD5 service.

cd EAP-MD5

Step 4 Set the service Type property to eap-md5 and the UserService property to LDAP.

set Type eap-md5

set UserService LDAP

The following example shows the configuration of the EAP-MD5 service:

```
[ //localhost/Radius/Services/EAP-MD5 ]
    Name = EAP-MD5
    Description =
    Type = eap-md5
    IncomingScript~ =
    OutgoingScript~ =
    AuthenticationTimeout = 120
    UserService = LDAP
```

Adding an LDAP Remote Server

Prime Access Registrar adds a new type of service and remote server called ldap-accounting that enables inserting accounting records into LDAP. You can write accounting records into LDAP by referring this service in /Radius/DefaultAccountingService or in the Accounting-Service environment variable.

Adding and Configuring an LDAP Remote Server

To add and configure an LDAP remote server:

Step 1 Change directory to **/Radius/RemoteServers** and add a RemoteServer object.

cd /Radius/RemoteServers

add LDAP

Step 2 Change directory to the LDAP RemoteServer.

cd LDAP

```
[ //localhost/Radius/RemoteServers/LDAP ]
    Name = LDAP
    Description =
    Protocol =
```

Step 3 Set the RemoteServer protocol property to ldap.

set Protocol ldap

The following example shows the default configuration of an LDAP remote server:

```
[ //localhost/Radius/RemoteServers/LDAP ]
    Name = LDAP
    Description =
    Protocol = ldap
    Port = 389
    ReactivateTimerInterval = 300000
    Timeout = 15
    HostName =
    BindName =
    BindPassword =
    UseSSL = FALSE
    SearchPath~ =
    Filter~ = (uid=%s)
    UserPasswordAttribute = userpassword
    LimitOutstandingRequests = FALSE
    MaxOutstandingRequests = 0
    MaxReferrals = 0
    ReferralAttribute =
    ReferralFilter =
    PasswordEncryptionStyle = Dynamic
    EscapeSpecialCharInUserName = FALSE
    DNSLookupAndLDAPRebindInterval =
    LDAPToRadiusMappings/
    LDAPToEnvironmentMappings/
    LDAPToCheckItemMappings/
```

Step 4 Set the HostName property to the SPE/DESS directory IP address or hostname.

Step 5 Set the BindName property to the SPE/DESS administrator name.

Step 6 Set the BindPassword property to the SPE/DESS administrator password.

Step 7 Set the SearchPath property to the SPE/DESS directory container.

Step 8 Set the UserPasswordAttribute property type to clearpassword.

Adding an LDAP Service

You must configure a service of type ldap-accounting under /Radius/Services using the ldap accounting feature.

Adding and Configuring an LDAP Service

To add and configure an LDAP service:

Step 1 Change directory to **/Radius/Service** and add LDAP.

cd /Radius/Service

add LDAP

Step 2 Change directory to LDAP and set the type property to ldap.

cd LDAP

set Type ldap

The following shows the default configuration for an LDAP service:

```
[ //localhost/Radius/Services/LDAP ]
    Name = LDAP
    Description =
    Type = ldap
    IncomingScript~ =
    OutgoingScript~ =
    OutagePolicy~ = RejectAll
    OutageScript~ =
    MultipleServersPolicy = Failover
    RemoteServers/
```

Step 3 Change directory to RemoteServers and associate the LDAP RemoteServer with the LDAP service.

cd RemoteServers

set 1 LDAP

Saving the Configuration and Reloading the Server

Use the **save** command to save the configuration, then **reload** the Prime Access Registrar server.

save

```
Validating //localhost...
Saving //localhost...
```

reload

```
Reloading Server 'Radius'...
Server 'Radius' is Running, its health is 10 out of 10
```

Cisco SSG VSAs in Cisco Prime Access Registrar Dictionary

The following vendor-specific attributes (VSAs) are defined by default in the attribute dictionary after installing Prime Access Registrar software:

- Cisco-AVPair
- Cisco-SSG-Account-Info
- Cisco-SSG-Service-Info
- Cisco-SSG-Command-Code
- Cisco-SSG-Control-Info

Using the REX Accounting Script

This chapter describes how to use the REX Accounting Script (RAS). The RAS writes RADIUS Accounting requests to a local, flat file and is included as an option for Cisco Prime Access Registrar (Prime Access Registrar). It is designed to be attached to a Prime Access Registrar IncomingScript or OutgoingScript point. When used in conjunction with the Prime Access Registrar built-in proxy support, the server will concurrently store a local copy of an Accounting request and proxy another copy to another RADIUS server.

Note Unless you require log rotation at an exact time or when the accounting log reaches a specific file size, we recommend that you use service grouping to log and proxy accounting packets.

RAS can be attached to more than one Prime Access Registrar extension point. For example, in a dial-up resale scenario, you might configure Prime Access Registrar to proxy Accounting requests to many different Remote Servers (by realm). For some subset of those, you might want to keep a local copy of the Accounting requests. In this case, RAS could be installed as the IncomingScript on just the Services for which a local copy is desired.

Note Also included is the **DropAcctOnOff** Script. This script causes Prime Access Registrar to drop all Accounting-Requests with an **Acct-Status-Type** of **Accounting-On** or **Accounting-Off**.

This chapter contains the following sections:

- Building and Installing the REX Accounting Script
- Configuring the Rex Accounting Script
- Specifying REX Accounting Script Options, page 10-4

Building and Installing the REX Accounting Script

The RAS writes RADIUS Accounting requests to a local, flat file and is included as an option for Prime Access Registrar. It is designed to be attached to a Prime Access Registrar IncomingScript or OutgoingScript point.

Building and Installing the REX Accounting Script

To build and install RAS:

Step 1 Change directory to **$INSTALL/examples/rexacctscript**.

Step 2 Modify the **Makefile** to ensure the **AR_INSTALL_DIR** variable points to the directory where the Prime Access Registrar software was installed, and then choose a compiler (**gcc** or SUNPro **CC**).

Step 3 From the command line prompt, enter:

host% **make**

Step 4 Log in as user **root**.

Step 5 From the command line prompt, enter:

host# **make install**

Configuring the Rex Accounting Script

RAS can be attached to more than one Prime Access Registrar extension point. For example, in a dial-up resale scenario, you might configure Prime Access Registrar to proxy Accounting requests to many different Remote Servers (by realm).

Configuring the Rex Accounting Script

To configure RAS:

Step 1 Start the Prime Access Registrar **aregcmd** configuration utility and login:

> $INSTALL/usrbin/aregcmd -C localhost -N admin -P aicuser

```
Access Registrar Configuration Utility

Copyright (C) 1995-2016 by Cisco Systems, Inc. All rights reserved.

Logging in to localhost

[ //localhost ]

      LicenseKey = xxxx-xxxx-xxxx-xxxx
      Radius/
      Administrators/

Server 'Radius' is Running, its health is 10 out of 10

-->
```

Step 2 Using **aregcmd**, create a new Prime Access Registrar Script object:

--> cd /Radius/Scripts

```
[ //localhost/Radius/Scripts ]
      Entries 1 to 20 from 39 total entries
      Current filter: <all>
      ACME/
      AscendIncomingScript/
```

<... other output deleted...>

--> add LocalAccounting

```
Added LocalAccounting
```

Step 3 Using **aregcmd**, fill in the details of the new Prime Access Registrar Script object. See Chapter 5, "Configuring and Monitoring the RADIUS Server," for more details.

--> cd LocalAccounting

```
[ //localhost/Radius/Scripts/LocalAccounting ]
      Name = LocalAccounting
      Description =
      Language =
      Filename =
      EntryPoint =
      InitEntryPoint =
      InitEntryPointArgs =
```

--> set Desc "Log Accounting requests to local file"

```
Set Description "Log Accounting requests to local file"
```

--> set lang REX

```
Set Language REX
```

--> set filename libRexAcctScript.so

```
Set Filename libRexAcctScript.so
```

--> set entry RexAccountingScript

```
Set EntryPoint RexAccountingScript
```

--> set initentrypoint InitRexAccountingScript

```
Set InitEntryPoint InitRexAccountingScript
```

--> set initentrypointargs "-f Accounting -t 1:15"

```
Set InitEntryPointArgs "-f Accounting -t 1:15"
```

--> ls

```
[ //localhost/Radius/Scripts/LocalAccounting ]
      Name = LocalAccounting
      Description = "Log Accounting requests to local file"
      Language = REX
      Filename = libRexAcctScript.so
      EntryPoint = RexAccountingScript
      InitEntryPoint = InitRexAccountingScript
      InitEntryPointArgs = "-f Accounting -t 1:15"
```

-->

Step 4 Using **aregcmd**, attach the new Prime Access Registrar Script object to the appropriate Prime Access Registrar Scripting point. See Chapter 5, "Configuring and Monitoring the RADIUS Server," for more details.

--> set /radius/IncomingScript LocalAccounting

```
Set /Radius/IncomingScript LocalAccounting
```

Step 5 Using **aregcmd**, save the configuration modifications:

--> save

```
Validating //localhost...
Saving //localhost...
```

Step 6 Using **aregcmd**, reload the server:

```
--> reload
Reloading Server 'Radius'...
Server 'Radius' is Running, its health is 10 out of 10
```

Specifying REX Accounting Script Options

The REX Accounting Script supports the options shown in Table 10-1.

Table 10-1 REX Accounting Script Supported Options

Option	Description
-f <*filename*>	Required. Specify the name of the output file.
-t <HH:MM[:SS]>	Specify a time of day to roll the output file. Note, this is time on the 24-hour clock, for example, 00:05 = 12:05am, 13:30 = 1:30pm. This option can not be used with the **-i** option.
-i <*seconds*>	Specify the number of seconds between rolling the output file, beginning at start-up. This option can not be used with the **-t** option.
-s <*size*>[k|m|g]	Specify the maximum size for an output file. When the file reaches this size, it will be rolled. When specifying the <*size*> option, a <*unit*> can be included. When a <*unit*> is not included, the <*size*> is in bytes. Note, do not use a space character between the <*size*> and <*unit*> options. <*unit*> can be either: k = 1K, m = 1Meg, g = 1Gig.
-g	Use GMT when writing the date/time in the Accounting output file for each record (default is local time).
-G	Use GMT when naming rolled output files (default is local time).
-A	Process all packets, not just Accounting-Requests.
-I	Ignore errors when processing packets, always return successfully.
-a <*buffer-count*>	Pre-allocate this many Accounting buffers to improve performance.
-T <*trace-level*>	Set the trace level. This trace info appears in the output file (as its written by the background thread which no longer has a packet to use for logging or tracing.)
-O <*script-description*>	Call another REX extension before calling the **RexAcctScript**.
-o <*script-description*>	Call another REX extension after calling the **RexAcctScript**.

Example Script Object

This is an example of what a Prime Access Registrar Script object using RAS might look like when viewed in the Prime Access Registrar configuration utility, **aregcmd**:

```
[ //localhost/Radius/Scripts/REX-Accounting-Script ]
      Name = REX-Accounting-Script
      Description =
      Language = REX
      Filename = librexacctscript.so
      EntryPoint = RexAccountingScript
      InitEntryPoint = InitRexAccountingScript
      InitEntryPointArgs = "-f Accounting -t 16:20 -s 100k -o
         libRexAcctScript.so:DropAcctOnOff"
```

This example causes RAS to write to a file called **Accounting.log** (in the **logs** directory of the installation tree). The file rolls every day at 4:20pm (local time), as well as whenever it grows larger than 100k in size. RAS also runs the **DropAcctOnOff** script against every packet, after it has processed the packet.