



## **Cisco Access Registrar User's Guide, 3.0**

September 2003

### **Corporate Headquarters**

Cisco Systems, Inc.  
170 West Tasman Drive  
San Jose, CA 95134-1706  
USA  
<http://www.cisco.com>  
Tel: 408 526-4000  
800 553-NETS (6387)  
Fax: 408 526-4100

Customer Order Number: N/A  
Text Part Number: OL-2681-03

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 Access Registrar User's Guide, 3.0*  
Copyright © 2002, Cisco Systems, Inc.  
All rights reserved.



## **About This Guide** xv

- How This Book Is Organized xv
- Obtaining Documentation xvi
  - World Wide Web xvi
  - Documentation CD-ROM xvi
  - Ordering Documentation xvi
  - Documentation Feedback xvii
- Obtaining Technical Assistance xvii
  - Cisco.com xvii
  - Technical Assistance Center xvii
    - Contacting TAC by Using the Cisco TAC Website xviii
    - Contacting TAC by Telephone xviii

## **Overview** 1-1

- RADIUS Protocol 1-1
  - Steps to Connection 1-2
  - Types of RADIUS Messages 1-3
    - Packet Contents 1-3
    - The Attribute Dictionary 1-4
  - Proxy Servers 1-4
- Cross Server Session and Resource Management 1-5
  - Overview 1-5
  - Session-Service Service Step and Radius-session Service 1-6
  - Configure Front Line Access Registrar 1-6
  - Configure Central AR 1-7

## **Using the aregcmd Commands** 2-1

- General Command Syntax 2-1
  - Configuration Objects 2-2
  - View-Only Administrator Mode 2-2
    - ViewOnly Property 2-2
- aregcmd Commands 2-3
  - add 2-3
  - cd 2-3
  - delete 2-4

- exit 2-4
- filter 2-4
- find 2-5
- help 2-5
- insert 2-5
- login 2-5
- logout 2-6
- ls 2-6
- next 2-6
- prev 2-7
- pwd 2-7
- query-sessions 2-7
- quit 2-7
- release-sessions 2-7
- reload 2-8
- reset-stats 2-8
- save 2-8
- set 2-9
- start 2-10
- stats 2-10
- status 2-11
- stop 2-11
- trace 2-12
- unset 2-13
- validate 2-13
- aregcmd Command Logging 2-13
- aregcmd Command Line Editing 2-14
- aregcmd Error Codes 2-15

**Access Registrar Server Objects 3-1**

- Radius 3-1
- UserLists 3-3
  - Users 3-4
    - HiddenAttributes Property 3-4
- UserGroups 3-5
- Policies 3-5
- Clients 3-5
- Vendors 3-6
- Scripts 3-7

Services	3-8
Types of Services	3-9
local	3-9
EAP-LEAP	3-9
EAP-MD5	3-10
radius, ldap, or tacacs-udp	3-10
file	3-10
ODBC	3-11
rex	3-12
Session Managers	3-13
Session Creation	3-13
Session Notes	3-13
Soft Group Session Limit	3-15
Session Correlation Based on User-Defined Attributes	3-15
Resource Managers	3-15
Types of Resource Managers	3-16
IP-Dynamic	3-16
IP-Per-NAS-Port	3-16
IPX-Dynamic	3-17
Subnet-Dynamic	3-17
Group-Session-Limit	3-18
User-Session-Limit	3-18
Home-Agent	3-18
USR-VPN	3-19
Gateway Subobject	3-19
Profiles	3-19
Attributes	3-20
Translations	3-20
TranslationGroups	3-21
Remote Servers	3-22
Types of Protocols	3-22
radius	3-22
ldap	3-23
odbc	3-25
tacacs-udp	3-26
Rules	3-27
Advanced	3-27
Using the RequireNASsBehindProxyBeInClientList Property	3-30
Advance Duplicate Detection Feature	3-30

- Ports **3-31**
- Interfaces **3-31**
- Reply Messages **3-31**
- Attribute Dictionary **3-33**
  - Types **3-33**
  - Vendor Attributes **3-34**
- SNMP **3-34**

**Using the radclient Command 4-1**

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

**Configuring Local Authentication and Authorization 5-1**

- Configuring a Local Service and UserList **5-1**
  - Configuring a Local Service **5-2**
  - Configuring a Userlist **5-2**
  - Configuring Cisco AR to Use the Local Service For AA **5-3**
  - Activating the Configuration **5-3**

Troubleshooting the Local Service and UserList Configuration	5-4
Verifying the Configuration	5-4
Configuring Return Attributes and Check-Items	5-6
Configuring Per User Return Attributes	5-6
Configuring Per User Check-Items	5-6
Verifying the Per User Return Attributes and Check-Items Configuration	5-7
Configuring Profiles to Group Attributes	5-7
Configuring Return Attributes and Check-Items Using UserGroup	5-8
Return Attribute Precedence	5-9
aregcmd Command Performance	5-9
UserDefined1 Property	5-10
Access-Request Logging	5-10

### Using Extension Points 6-1

Determining the Goal of the Script	6-1
Writing the Script	6-2
Choosing the Type of Script	6-2
Request Dictionary Script	6-3
Response Dictionary Script	6-3
Environment Dictionary Script	6-4
Adding the Script Definition	6-4
Adding the Example Script Definition	6-5
Choosing the Scripting Point	6-5
Testing the Script	6-5
About the Tcl/Tk 8.3 Engine	6-6

### Using Replication 7-1

Setting Up Replication	7-1
Configuring the Master	7-2
Configuring The Member	7-3
Verifying the Configuration	7-4
Replication Example	7-4
Adding a User	7-4
Master Server's Log	7-4
Member Server's Log	7-5
Verifying Replication	7-5
Master Server's Log	7-5
Member Server's Log	7-5
Using aregcmd -pf Option	7-6
Master Server's Log	7-6

- Member Server's Log 7-6
- An Automatic Resynchronization Example 7-7
  - Master Server's Log 7-7
  - Member Server's Log 7-7
- Full Resynchronization 7-8
- Replication Log Messages 7-9
  - Information Log Messages 7-10
  - Warning Log Messages 7-11
  - Error Log Messages 7-13
  - Log Messages You Should Never See 7-14
- Using On-Demand Address Pools 8-1**
  - Cisco-Incoming Script 8-3
    - How the Script Works 8-3
    - CiscoWithODAPIncomingScript 8-3
  - Vendor Type CiscoWithODAP 8-4
  - Configuring Cisco Access Registrar to Work with ODAP 8-4
    - Configuration Summary 8-4
    - Detailed Configuration 8-5
      - Setting Up an ODAP UserList 8-5
      - Adding ODAP Users 8-5
      - Setting Up an ODAP-Users Service 8-6
      - Setting Up an ODAP Accounting Service 8-7
      - Adding Session Managers 8-8
      - Setting Up Resource Managers 8-9
      - Configuring Session Managers 8-13
      - Configure Clients 8-15
      - Save Your Configuration 8-16
- Using Cisco Access Registrar Server Features 9-1**
  - Command Completion 9-1
  - Service Grouping Feature 9-2
    - Configuration Example - AccountingGroupService 9-3
      - Summary of Events 9-5
    - Configuration Example 2 - AuthenticationGroupService 9-6
      - Summary of Events 9-9
  - SHA-1 Support for LDAP-Based Authentication 9-9
    - Remote LDAP Server Password Encryption 9-10
    - Dynamic Password Encryption 9-10
    - Logs 9-11

LEAP Support	9-11
LEAP Features	9-11
Required Attributes	9-12
User Interface	9-12
Dynamic Attributes	9-12
Object Properties with Dynamic Support	9-12
Dynamic Attribute Format	9-14
Tunneling Support Feature	9-14
Configuration	9-15
Example	9-15
Notes	9-15
Validation	9-16
xDSL VPI/VCI Support for Cisco 6400	9-16
Using User-Name/User-Password for Each Cisco 6400 Device	9-16
Format of the New User-Name Attribute	9-16
Apply Profile in Cisco AR Database to Directory Users	9-17
User-Profile	9-17
User-Group	9-18
Example User-Profile and User-Group Attributes in Directory User Record	9-19
Directory Multi-Value Attributes Support	9-19
MultiLink-PPP (ML-PPP)	9-19
Dynamic Updates Feature	9-20
NAS Monitor	9-22
Automatic Customer Information Collection	9-22
Simultaneous Terminals for Remote Demonstration	9-22
Support for RADIUS Check Item Attributes	9-22
Configuring Check Items	9-23
Configuring User Check Items	9-23
Configuring Usergroup Check Items	9-23
User-Specific Attributes	9-24
<b>Using the Policy Engine</b>	<b>10-1</b>
Policies and Rules	10-1
Script and Attribute Requirements	10-2
Validation	10-3
Known Limitations	10-3
Service Determination	10-3
Inserting, Deleting and Substituting Attributes	10-4

- Wildcard Support 10-5
- Time of Day Rules 10-6
- Configuration 10-7
- Notes 10-8
- Validation 10-8
- Known Anomalies 10-8
- Prefix Feature 10-8
  - New Attributes 10-8
    - Example Configuration 10-8

**Wireless Support 11-1**

- 3GPP2 Home Agent Support 11-1
  - Home-Agent Resource Manager 11-1
    - Load Balancing 11-2
    - Configuring the Home Agent Resource Manager 11-2
  - Querying and Releasing Sessions 11-3
  - Access Request Requirements 11-3
- Session Correlation Based on User-Defined Attributes 11-3
- Managing Multiple Accounting Start/Stop Messages 11-4
- NULL Password Support 11-4
- New 3GPP2 VSAs in the CAR Dictionary 11-4

**Using LDAP 12-1**

- Configuring LDAP 12-1
  - Configuring the LDAP Service 12-1
    - MultipleServersPolicy 12-2
    - RemoteServers 12-2
  - Configuring an LDAP RemoteServer 12-2
    - DNS Look Up and LDAP Rebind Interval 12-4
- LDAP Rebind Failures 12-5
  - LDAPToRadiusMappings 12-5
  - LDAPToEnvironmentMappings 12-6
  - LDAPToCheckItemMappings 12-6
  - Setting LDAP As Authentication and Authorization Service 12-6
  - Saving Your Configuration 12-6
    - CHAP Interoperability with LDAP 12-7
    - Allowing Special Characters in LDAP Usernames 12-7
    - Dynamic LDAP Search Base 12-7
- Analyzing LDAP Trace Logs 12-7
  - Successful Bind Message 12-8

Bind Failure Messages	12-8
Login Failure Messages	12-8
<b>Using Open Database Connectivity</b>	<b>13-1</b>
Oracle Software Requirements	13-1
Configuring ODBC	13-2
Configuring the ODBC Service	13-2
Configuring an ODBC RemoteServer	13-3
ODBC Data Source	13-4
SQL Definitions	13-4
SQL Syntax Restrictions	13-4
Specifying More Than One Search Key	13-5
ODBCToRadiusMappings	13-5
ODBCToEnvironmentMappings	13-6
Configuring an ODBC DataSource	13-6
Setting ODBC As Authentication and Authorization Service	13-6
Saving Your Configuration	13-7
<b>Backing Up the Database</b>	<b>14-1</b>
Configuration	14-1
Command Line Utility	14-1
Recovery	14-1
mcdshadow Command Files	14-2
<b>Using the REX Accounting Script</b>	<b>15-1</b>
Building and Installing the REX Accounting Script	15-1
Configuring the Rex Accounting Script	15-2
Specifying REX Accounting Script Options	15-3
Example Script Object	15-4
<b>Logging Syslog Messages</b>	<b>16-1</b>
syslog Messages	16-1
Example 1	16-2
Example 2	16-2
Configuring Message Logging	16-2
Changing Log Directory	16-3
Configuring syslog Daemon (syslogd)	16-3
Creating a Log File	16-3
Restarting syslogd	16-4
Managing the Syslog File	16-4

Server Up/Down Status Change Logging 16-5  
 Header Formats 16-5  
 Example Log Messages 16-5

**Troubleshooting Cisco Access Registrar 17-1**

Gathering Basic Information 17-1  
 Troubleshooting Quick Checks 17-2  
     Disk Space 17-2  
     Resource Conflicts 17-2  
         No Co-Existence With Cisco Network Registrar 17-2  
         Port Conflicts 17-2  
         Server Running Sun SNMP Agent 17-3  
     CAR Log Files 17-3  
         Using xtail to Monitor Log File Activity 17-3  
     Modifying the Trace Level 17-3  
     Installation and Server Process Start-up 17-4  
 aregcmd and CAR Configuration 17-4  
     Running and Stopped States 17-5  
 RADIUS Request Processing 17-6  
 Other Troubleshooting Techniques and Resources 17-6  
     aregcmd Stats Command 17-6  
     Core Files 17-7  
     radclient 17-7  
     CAR Replication 17-7

**Cisco Access Registrar Tcl and REX Dictionaries A-1**

Tcl Attribute Dictionaries A-1  
     Attribute Dictionary Methods A-1  
     Tcl Environment Dictionary A-3  
 REX Attribute Dictionary A-4  
     Attribute Dictionary Methods A-4  
     REX Environment Dictionary A-9  
         REX Environment Dictionary Methods A-10

**Environment Dictionary B-1**

Cisco Access Registrar Environment Dictionary Variables B-1  
     Accepted-Profiles B-1  
     Accounting-Service B-1  
     Acquire-Group-Session-Limit B-2  
     Acquire-IP-Dynamic B-2

Acquire-IPX-Dynamic	B-2
Acquire-IP-Per-NAS-Port	B-2
Acquire-Subnet-Dynamic	B-2
Acquire-User-Session-Limit	B-2
Acquire-USR-VPN	B-2
Allow-Null-Password	B-2
Authentication-Service	B-3
Authorization-Service	B-3
Current-Group-Count	B-3
Dynamic-Search-Path	B-3
Group-Session-Limit	B-3
Ignore-Accounting-Signature	B-3
Incoming-Translation-Groups	B-3
Misc-Log-Msg-Info	B-4
PAGER Environment Variable	B-4
Reject-Reason	B-4
Remote-Server	B-4
<b>Request-Authenticator</b>	<b>B-4</b>
Request-Type	B-4
Require-User-To-Be-In-Authorization-List	B-5
Response-Type	B-5
Retrace-Packet	B-6
Skip-Session-Management	B-6
Session-Key	B-6
Session-Manager	B-6
Session-Notes	B-6
Session-Service	B-6
Source-IP-Address	B-7
Source-Port	B-7
Subnet-Size-If-No-Match	B-7
Trace-Level	B-7
Unavailable-Resource	B-7
Unavailable-Resource-Type	B-7
UserDefined1	B-7
User-Authorization-Script	B-8
User-Group	B-8
User-Group-Session-Limit	B-8
User-Name	B-8
User-Profile	B-8
User-Session-Limit	B-8

- RADIUS Attributes C-1**
  - RADIUS Attributes C-1
    - Cisco Access Registrar 3.0 Attributes C-2
    - RADIUS Attributes Numeric List C-4
  - Vendor Specific Attributes C-13
    - 3GPP VSAs C-13
    - 3GPP2 VSAs C-13
    - ACC VSAs C-16
    - Altiga VSAs C-21
    - Ascend VSAs C-23
    - Bay Networks VSAs C-32
    - Cabletron VSAs C-32
    - Cisco VSAs C-33
    - Compatible VSAs C-35
    - Microsoft VSAs C-35
    - Nomadix VSAs C-37
    - RedBack VSAs C-37
    - RedCreek VSAs C-39
    - Telebit VSAs C-40
    - Unisphere VSAs C-40
    - USR VSAs C-42



## About This Guide

---

The Cisco Access Registrar User's Guide provides information about how to use Cisco Access Registrar 3.0.

## How This Book Is Organized

The Cisco Access Registrar User's Guide is organized as follows:

Chapter 1, "Overview," provides an overview of Cisco Access Registrar.

Chapter 2, "Using the aregcmd Commands," provides information about using **aregcmd** commands.

Chapter 3, "Access Registrar Server Objects," provides information about Cisco Access Registrar server objects.

Chapter 4, "Using the radclient Command," provides information about using **radclient** commands to test Cisco Access Registrar.

Chapter 5, "Configuring Local Authentication and Authorization," provides information about how to configure local authentication and authorization and helpful examples.

Chapter 6, "Using Extension Points," provides information about how to use Cisco Access Registrar scripting to customize your RADIUS server.

Chapter 7, "Using Replication," provides information about how to use the replication feature.

Chapter 8, "Using On-Demand Address Pools," provides information about using On-Demand Address Pools.

Chapter 9, "Using Cisco Access Registrar Server Features," provides information about using Cisco Access Registrar features.

Chapter 10, "Using the Policy Engine," provides information about using the Cisco Access Registrar Policy Engine.

Chapter 11, "Wireless Support," provides information about Cisco Access Registrar support for wireless features.

"Using LDAP" provides information about using an LDAP remote server with Cisco Access Registrar.

Chapter 13, "Using Open Database Connectivity," provides information about a new type of RemoteServer object and a new service to support ODBC.

Chapter 14, "Backing Up the Database," describes the Cisco Access Registrar shadow backup facility, which ensures a consistent snapshot of Cisco Access Registrar's database for backup purposes.

Chapter 15, "Using the REX Accounting Script," describes how to use the REX Accounting scripts.

Chapter 16, “Logging Syslog Messages,” provides information about logging messages via syslog and centralized error reporting for Cisco Access Registrar.

Chapter 17, “Troubleshooting Cisco Access Registrar,” provides information about techniques used when troubleshooting Cisco Access Registrar and highlights common problems.

Appendix A, “Cisco Access Registrar Tcl and REX Dictionaries,” describes the Tcl and REX dictionaries that are used when writing Incoming or Outgoing scripts for use with Cisco Access Registrar.

Appendix B, “Environment Dictionary,” describes the environment variables the scripts use to communicate with Cisco Access Registrar or to communicate with other scripts.

Appendix C, “RADIUS Attributes,” lists the RFC 2865 RADIUS attributes with their names and values.

An index is also provided.

## Obtaining Documentation

The following sections provide sources for obtaining documentation from Cisco Systems.

### World Wide Web

You can access the most current Cisco documentation on the World Wide Web at the following sites:

- <http://www.cisco.com>
- <http://www-china.cisco.com>
- <http://www-europe.cisco.com>

### Documentation CD-ROM

Cisco documentation and additional literature are available in a CD-ROM package, which ships with your product. The Documentation CD-ROM is updated monthly and may be more current than printed documentation. The CD-ROM package is available as a single unit or as an annual subscription.

### Ordering Documentation

Cisco documentation is available in the following ways:

- Registered Cisco Direct Customers can order Cisco Product documentation from the Networking Products MarketPlace:  
[http://www.cisco.com/cgi-bin/order/order\\_root.pl](http://www.cisco.com/cgi-bin/order/order_root.pl)
- Registered Cisco.com users can order the Documentation CD-ROM through the online Subscription Store:  
<http://www.cisco.com/go/subscription>
- Nonregistered Cisco.com users can order documentation through a local account representative by calling Cisco corporate headquarters (California, USA) at 408 526-7208 or, in North America, by calling 800 553-NETS(6387).

## Documentation Feedback

If you are reading Cisco product documentation on the World Wide Web, you can submit technical comments electronically. Click **Feedback** in the toolbar and select **Documentation**. After you complete the form, click **Submit** to send it to Cisco.

You can e-mail your comments to [bug-doc@cisco.com](mailto:bug-doc@cisco.com).

To submit your comments by mail, use the response card behind the front cover of your document, or write to the following address:

Attn: Document Resource Connection  
Cisco Systems, Inc.  
170 West Tasman Drive  
San Jose, CA 95134-9883

We appreciate your comments.

## Obtaining Technical Assistance

Cisco provides Cisco.com as a starting point for all technical assistance. Customers and partners can obtain documentation, troubleshooting tips, and sample configurations from online tools. For Cisco.com registered users, additional troubleshooting tools are available from the TAC website.

### Cisco.com

Cisco.com is the foundation of a suite of interactive, networked services that provides immediate, open access to Cisco information and resources at anytime, from anywhere in the world. This highly integrated Internet application is a powerful, easy-to-use tool for doing business with Cisco.

Cisco.com provides a broad range of features and services to help customers and partners streamline business processes and improve productivity. Through Cisco.com, you can find information about Cisco and our networking solutions, services, and programs. In addition, you can resolve technical issues with online technical support, download and test software packages, and order Cisco learning materials and merchandise. Valuable online skill assessment, training, and certification programs are also available.

Customers and partners can self-register on Cisco.com to obtain additional personalized information and services. Registered users can order products, check on the status of an order, access technical support, and view benefits specific to their relationships with Cisco.

To access Cisco.com, go to the following website:

<http://www.cisco.com>

## Technical Assistance Center

The Cisco TAC website is available to all customers who need technical assistance with a Cisco product or technology that is under warranty or covered by a maintenance contract.

## Contacting TAC by Using the Cisco TAC Website

If you have a priority level 3 (P3) or priority level 4 (P4) problem, contact TAC by going to the TAC website:

<http://www.cisco.com/tac>

P3 and P4 level problems are defined as follows:

- P3—Your network performance is degraded. Network functionality is noticeably impaired, but most business operations continue.
- P4—You need information or assistance on Cisco product capabilities, product installation, or basic product configuration.

In each of the above cases, use the Cisco TAC website to quickly find answers to your questions.

To register for Cisco.com, go to the following website:

<http://www.cisco.com/register/>

If you cannot resolve your technical issue by using the TAC online resources, Cisco.com registered users can open a case online by using the TAC Case Open tool at the following website:

<http://www.cisco.com/tac/caseopen>

## Contacting TAC by Telephone

If you have a priority level 1 (P1) or priority level 2 (P2) problem, contact TAC by telephone and immediately open a case. To obtain a directory of toll-free numbers for your country, go to the following website:

<http://www.cisco.com/warp/public/687/Directory/DirTAC.shtml>

P1 and P2 level problems are defined as follows:

- P1—Your production network is down, causing a critical impact to business operations if service is not restored quickly. No workaround is available.
- P2—Your production network is severely degraded, affecting significant aspects of your business operations. No workaround is available.



## Overview

---

The chapter provides an overview of the RADIUS server, including connection steps, RADIUS message types, and using Cisco Access Registrar as a proxy server.

Cisco Access Registrar is a RADIUS (Remote Authentication Dial-In User Service) server that allows multiple dial-in Network Access Server (NAS) devices to share a common authentication, authorization, and accounting database.

Cisco Access Registrar handles the following tasks:

- Authentication—determines the identity of users and whether they may 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

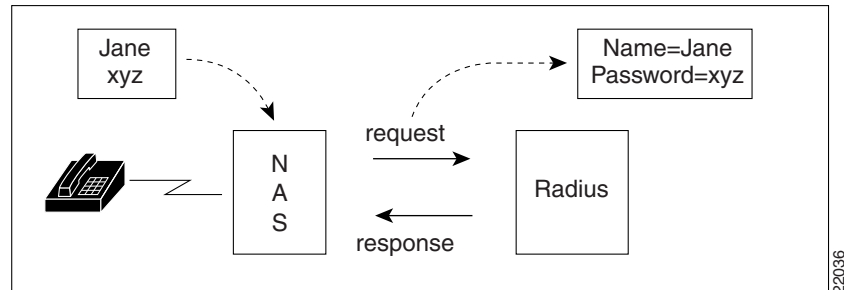
Using a RADIUS server allows you to better manage the access to your network, as it allows you to store all security information in a single, centralized database instead of distributing the information around the network in many different devices. You can make changes to that single database instead of making changes to every network access server in your network.

## RADIUS Protocol

Cisco 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 Cisco 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 Cisco Access Registrar with a name and a password. Cisco 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 1-1).

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



Cisco Access Registrar can also reject the packet if it needs to deny network access to the user. Or, Cisco Access Registrar may 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 Cisco 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.

## Steps to Connection

Three participants exist in this interaction: the user, the NAS, and the RADIUS server. The following steps 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 Cisco Access Registrar server.
- Step 3** The Cisco Access Registrar server determines what hardware sent the request (NAS) and parses the packet.
  - d. It sets up the Request dictionary based on the packet information.
  - e. It runs any incoming scripts, which are user-written extensions to Cisco Access Registrar. An incoming script can examine and change the attributes of the request packet or the environment variables, which can affect subsequent processing.
  - f. Based on the scripts or the defaults, it chooses a service to authenticate and/or authorize the user.
- Step 4** Cisco Access Registrar's authentication service verifies the username and password is in its database. Or, Cisco Access Registrar delegates the authentication (as a proxy) to another RADIUS server, an LDAP, or TACACS server.
- Step 5** Cisco 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 Cisco Access Registrar session management at your site, the Session Manager calls the appropriate Resource Managers that allocate dynamic resources for this session.

- Step 7** Cisco Access Registrar runs any outgoing scripts to change the attributes of the response packet.
- Step 8** Cisco 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 may include sending the user an IP address, to indicate the connection has been successfully established.

## Types of RADIUS Messages

The client/server packet exchange consists primarily of the following types of RADIUS messages:

- Access-Request—sent by the client (NAS) requesting access
- Access-Reject—sent by the RADIUS server rejecting access
- Access-Accept—sent by the RADIUS server allowing access
- Access-Challenge—sent by the RADIUS server requesting more information in order to allow access. The NAS, after communicating with the user, responds with another Access-Request.

When you use RADIUS accounting, the client and server can also exchange the following two types of messages:

- Accounting-Request—sent by the client (NAS) requesting accounting
- Accounting-Response—sent by the RADIUS server acknowledging accounting

## Packet Contents

The information in each RADIUS message is encapsulated in a UDP (User Datagram Protocol) data packet. A packet is a block of data in a standard format for transmission. It is accompanied by other information, such as the origin and destination of the data.

Table 1-1 lists each message packet which contains the following five fields:

**Table 1-1 RADIUS Packet Fields**

Fields	Description
Code	Indicates what type of message it is: Access-Request, Access-Accept, Access-Reject, Access-Challenge, Accounting-Request, or Accounting-Response.
Identifier	Contains a value that is copied into the server's response so the client can correctly associate its requests and the server's responses when multiple users are being authenticated simultaneously.
Length	Provides a simple error-checking device. The server silently drops a packet if it is shorter than the value specified in the length field, and ignores the octets beyond the value of the length field.

Fields	Description
Authenticator	Contains a value for a Request Authenticator or a Response Authenticator. The Request Authenticator is included in a client's Access-Request. The value is unpredictable and unique, and is added to the client/server shared secret so the combination can be run through a one-way algorithm. The NAS then uses the result in conjunction with the shared secret to encrypt the user's password.
Attribute(s)	Depends on the type of message being sent. The number of attribute/value pairs included in the packet's attribute field is variable, including those required or optional for the type of service requested.

## The Attribute Dictionary

The Attribute dictionary contains a list of preconfigured authentication, authorization, and accounting attributes that can be part of a client's or user's configuration. The dictionary entries translate an attribute into a value Cisco Access Registrar uses to parse incoming requests and generate responses. Attributes have a human-readable name and an enumerated equivalent from 1-255.

Sixty three standard attributes exist, which are defined in RFC 2138 and 2139. There also are additional vendor-specific attributes that depend on the particular NAS you are using. For a complete list of attributes, see Chapter 5 of the Cisco Access Registrar Concepts and Reference Guide.

Some sample attributes include:

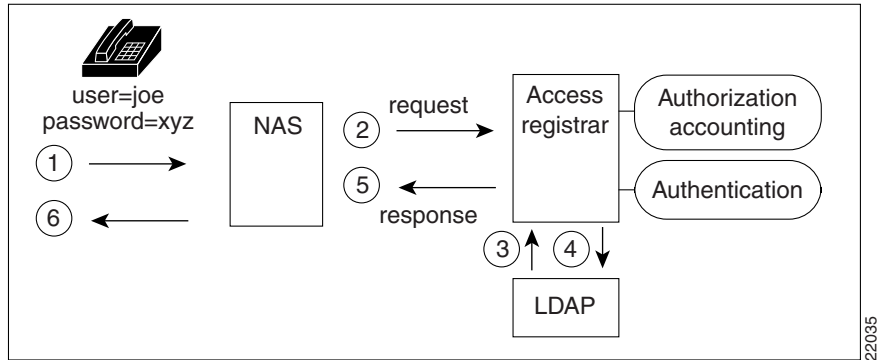
- User-Name—the name of the user
- User-Password—the user's password
- NAS-IP-Address—the IP address of the NAS
- NAS-Port—the NAS port the user is dialed in to
- Framed Protocol—such as SLIP or PPP
- Framed-IP-Address—the IP address the client uses for the session
- Filter-ID—vendor-specific; identifies a set of filters configured in the NAS
- Callback-Number—the actual callback number.

## Proxy Servers

Any one or all of the RADIUS server's three functions: authentication, authorization, or accounting can be subcontracted to another RADIUS server. Cisco Access Registrar then becomes a *proxy server*. Proxying to other servers enables you to delegate some of the RADIUS server's functions to other servers.

You could use Cisco Access Registrar to “proxy” to an LDAP server for access to directory information about users in order to authenticate them. Figure 1-2 shows user `joe` initiating a request, the Cisco Access Registrar server proxying the authentication to the LDAP server, and then performing the authorization and accounting processing in order to enable `joe` to log in.

Figure 1-2 Proxying to an LDAP Server for Authentication



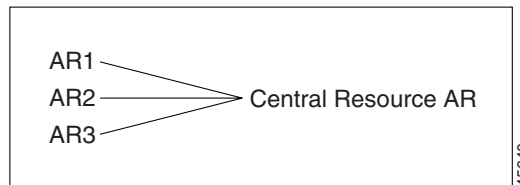
## Cross Server Session and Resource Management

Prior to AR 1.6, sessions and resources were managed locally, meaning that in a multi-AR server environment, resources such as IP addresses, user-based session limits, and group-based session limits were divided between all the AR servers. It also meant that, to ensure accurate session tracking, all packets relating to one user session were required to go to the same AR server.

### Overview

Access Registrar 1.6 and above can manage sessions and resources across AAA server boundaries. A session can be created by an Access-Request sent to AR1, and it can be removed by an Accounting-Stop request sent to AR2, as shown in Figure 1-3. 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 1-3 Multiple AR Servers



All resources that must be shared cross multiple front line ARs are configured in the Central Resource AR. Resources that are not shared can still be configured at each front line AR as done prior to the AR 1.6 release.

When the front line AR receives the access-request, it does the regular AA processing. If the packet is not rejected and a Central Resource AR is also configured, the front line AR will proxy the packet<sup>1</sup> to the configured Central Resource AR. If the Central Resource AR returns the requested resources, the

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

process continues to the local session management (if local session manager is configured) for allocating any local resources. If the Central Resource AR cannot allocate the requested resource, the packet is rejected.

When the Accounting-Stop packet arrives at the frontline AR, AR does the regular accounting processing. Then, if the front line AR is configured to use Central Resource AR, a proxy packet will be sent to Central Resource AR 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 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 = 1.6R0
  IncomingScript =
  OutgoingScript =
  DefaultAuthenticationService = local-users
  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 = 1645
    ReactivateTimerInterval = 300000
    SharedSecret = secret
    Vendor =
    IncomingScript =
    OutgoingScript =
    MaxTries = 3
    InitialTimeout = 2000
    AccountingPort = 1646
```

## Configure Central AR

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.





## Using the aregcmd Commands

---

This chapter describes how to use each of the **aregcmd** commands.

The Cisco Access Registrar **aregcmd** command is a command-line based configuration tool. It allows you to set any Cisco Access Registrar configurable option, as well as, start and stop the server and check statistics.

### General Command Syntax

Cisco 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 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.

**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 typing **cd Administrators**, you can type **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>] [-n] [<command> [<args>] [-p] [-q] [-s] [-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 administrator mode
- **-f**—Specifies a file that can contain a series of commands
- **-n**—Turns off prefix mode

- **-p**—Specifies prefix mode
- **-q**—Turns off verbose mode
- **-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.
- **-v**—Specifies verbose mode

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

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

## Configuration Objects

The Cisco Access Registrar **aregcmd** command lets you manipulate configuration objects, which define properties or the behavior of the RADIUS server, such as valid administrators and types of services. For descriptions of the those objects, see Chapter 3, “Access Registrar Server Objects.”

## View-Only Administrator Mode

Previous releases of Cisco AR provided only *super-user* administrative access. If you were able to log in to **aregcmd**, you could do anything to the system, including starting and stopping the system and changing the configuration. Cisco 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
```

## 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.

### 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, Cisco 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. Cisco 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 path name.

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, Cisco 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, Cisco 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, Cisco 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, Cisco Access Registrar displays the cluster server's health. Note, to log in to a cluster, the AR 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 in to make any modifications to the configuration hierarchy, or to manage the server(s). When you have any unsaved modifications, Cisco 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, Cisco Access Registrar displays only the first page.



### Note

---

**ls** pages 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, Cisco Access Registrar uses the current context. When you do not specify a count, Cisco Access Registrar uses the last count value you used with the **next** or **prev** command. If you never specify a count, Cisco 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>
```

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**.

## 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've made changes, the Cisco 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 off-line, however, the server believes user sessions for that NAS are still active.

The syntax is:

```
release-sessions <path> all
```

or

```
release-sessions <path> with-<type> <value>
```

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**.

## 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, Cisco Access Registrar asks you if you want to save your changes before restarting the server. Note, 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 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 2-1 lists the RADIUS server objects and the effect of Dynamic Updates upon them.

**Table 2-1** *Dynamic Updates Effect on Radius Server Objects*

Object	Add	Modify or Delete
Radius	Yes	Yes
UserLists	Yes	Yes

**Table 2-1** Dynamic Updates Effect on Radius Server Objects (continued)

Object	Add	Modify or Delete
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, Cisco 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 after you issue the **stats** command:


```
RemoteServer statistics for:ServerA, 209.165.201.1, port 1645
  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 2002
  lastAcceptTime = Mon Feb 18 17:18:11 2002
```

Table 2-2 lists the statistics displayed by the stats command and the meaning of the values.

**Table 2-2 aregcmd stats Information**

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)

Table 2-2 aregcmd stats Information (continued)

stats Value	Meaning
maxTries	Number of retry attempts to be made by the RemoteServer Object based on the RemoteServer's <i>maxTries</i> 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
	 <b>Note</b> 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.
lastRequestTime	Date and time of last proxy request
lastAcceptTime	Date and time of last ACCEPT response to a client

## status

Use the **aregcmd** command **status** to learn whether or not the specified server has been started. When the server is running, Cisco 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**

## 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 CAR server is trace level 5.



### Note

Although the highest **trace** level supported by the CAR 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.

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, Cisco 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, Cisco 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 2-3 lists the different **trace** levels and the information returned.

**Table 2-3 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: <ul style="list-style-type: none"> <li>• Which services and session managers are used to process a packet</li> <li>• Which client and vendor objects are used to process a packet</li> <li>• Detailed remote server information for LDAP and RADIUS, such as sending a packet and timing out</li> <li>• Details about poorly formed packets</li> <li>• Details included in trace level 1</li> </ul>
3	Indicates the following: <ul style="list-style-type: none"> <li>• Error traces in TCL scripts when referencing invalid RADIUS attributes.</li> <li>• Which scripts have been executed</li> <li>• Details about local UserList processing</li> <li>• Details included in trace levels 1 and 2</li> </ul>

**Table 2-3 Trace Levels and Information Returned (continued)**

Trace Level	Information Returned by Trace Command
4	Indicates the following: <ul style="list-style-type: none"> <li>• Information about advanced duplication detection processing</li> <li>• Details about creating, updating, and deleting sessions</li> <li>• Trace details about all scripting APIs called</li> <li>• Details included in trace levels 1, 2, and 3</li> </ul>
5	Indicates the following: <ul style="list-style-type: none"> <li>• Details about use of the policy engine including:               <ul style="list-style-type: none"> <li>– Which rules were run</li> <li>– What the rules did</li> <li>– If the rule passed or failed</li> <li>– Detailed information about which policies were called</li> </ul> </li> <li>• Details included in trace levels 1, 2, 3, and 4</li> </ul>

## 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, Cisco 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 Cisco Access Registrar discovers any errors, it displays them.

The syntax is:

```
validate
```

## 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 Cisco 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 Cisco 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:

```
09/23/1999 16:18:56 aregcmd Info Configuration 0 [localhost admin] --> quit
09/23/1999 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 may 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 2-4.

**Table 2-4 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 2-5 lists the error codes used in **aregcmd**.

**Table 2-5** *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 <b>aregcmd</b> and attempted another <b>aregcmd</b> 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 <b>aregcmd</b> session; occurs only when using multiple <b>aregcmd</b> instances
314	Failed to open the pager; the PAGER environment variable is set to something that does not exist and the <b>stats</b> 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 <b>aregcmd</b> log in
401	Unable to access server; <b>aregcmd</b> was unable to communicate with the <b>radius</b> 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 <b>aregcmd</b> has tried to connect with an older version of Cisco AR
500	Internal error; a generic <b>aregcmd</b> error

**Note**

---

Error codes 316 and 317 are valid for Cisco Access Registrar 3.0 version R6 and above.

---



## Access Registrar Server Objects

---

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

Cisco 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 Cisco 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
- **Translations**—contains individual Translations
- **TranslationGroup**—contains individual Translation Groups
- **RemoteServers**—contains individual RemoteServers
- **Advanced**—contains advanced properties, Ports, Interfaces, Reply Messages, and the Attribute dictionary.
- **Replication**—contains information about Replication

### Radius

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

Following is a listing of the RADIUS server object:

```
[ //localhost/Radius ]
  Name = Radius
  Description =
  Version = 1.7R0
  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/
  Advanced/
  Replication/
```

Table 3-1 lists the **Radius** properties. You can set or change Radius properties using the Cisco 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 3-1 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 Cisco Access Registrar
IncomingScript	Optional; if there is a script, it is the first script Cisco Access Registrar runs when it receives a request from any client and/or for any service
OutgoingScript	Optional; if there is a script, it is the last script Cisco Access Registrar runs before it sends a response to any client
DefaultAuthenticationService	Optional; Cisco Access Registrar uses this property when none of the incoming scripts sets the environment dictionary variable <b>Authentication-Service</b>
DefaultAuthorizationService	Optional; Cisco Access Registrar uses this property when none of the incoming scripts sets the environment dictionary variable <b>Authorization-Service</b>

**Table 3-1** *Radius Properties (continued)*

Property	Description
DefaultAccountingService	Optional; Cisco Access Registrar uses this property when none of the incoming scripts sets the environment dictionary variable <b>Accounting-Service</b> .
DefaultSessionService	Optional; Cisco Access Registrar uses this property when none of the incoming scripts sets the environment dictionary variable <b>Session-Service</b> .
DefaultSessionManager	Optional; Cisco Access Registrar uses this property if none of the incoming scripts sets the environment dictionary variable <b>Session-Manager</b> .

The remaining Cisco Access Registrar objects are subobjects of the **Radius** object.

## UserLists

The **UserLists** object contains all of the individual UserLists, which in turn, contain the specific users stored within Cisco Access Registrar. Cisco Access Registrar references each specific UserList by **name** from a Service whose type is set to **local**. When Cisco 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

User names may not include the forward slash (/) character. If the Cisco Access Registrar server receives an access request packet with a User-Name attribute containing a forward slash character and the Cisco AR server uses an internal UserList to look up users, the server produces an error (AX\_EINVAL) and may fail. If user names 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 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 3-2 lists the **UserLists** object properties.

**Table 3-2** *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 3-3 lists the **Users** object properties.

**Table 3-3 Users Properties**

Property	Description
Name	Required. Must be unique in the specific UserList.
Description	Optional description of the user.
Password	Required. The length must be between 0-253 characters.
Enabled	Required. The default is TRUE, which means the user is allowed access. Set to FALSE to cause Cisco Access Registrar to deny the user access.
Group (Overridden by User-Group)	Optional. When you set this to the name of a UserGroup, Cisco 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 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

In Cisco AR 3.0, a new property in the user object, HiddenAttributes, provides a concatenation of all user-level reply attributes. The Cisco AR 3.0 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 3-4 lists the **UserGroups** properties.

**Table 3-4 UserGroups Properties**

Property	Description
Name	Required. Must be unique in the <b>UserGroup</b> list.
Description	Optional description of the group.
BaseProfile	Optional. When you set this to the name of a Profile, Cisco 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 3-5 lists the properties required for a given **Policy**.

**Table 3-5 Policies Properties**

Property	Description
Name	Required; must be unique in the <b>Policies</b> list
Description	Optional description of the Policy
Grouping	Optional grouping of rules

## Clients

All NASs and proxy clients that communicate directly with Cisco 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 3-6 lists the **Client** object properties.

**Table 3-6 Client Properties**

Property	Description
Name	Required and should match the Client identifier specified in the standard RADIUS attribute, <b>NAS-Identifier</b> . The name must be unique within the Clients list. For more information about standard attributes, see XREF - List of Attributes.
Description	Optional description of the client.
IPAddress	Required. Must be a valid IP address and unique in the Clients list. Cisco Access Registrar uses this property to identify the Client that sent the request, either using the source IP address to identify the immediate sender and/or using the <b>NAS-IP-Address</b> attribute in the Request dictionary to identify the NAS sending the request through a proxy.
SharedSecret	Required. Must match the secret configured in the Client.
Type	Required. Accept the default, which is NAS, or set it to 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 <b>Vendor</b> object and include a Script. Cisco 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.
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.

## 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 may 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 XREF - List of Attributes.



### Note

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

Table 3-7 lists the **Vendor** object properties.

**Table 3-7 Vendor Properties**

Property	Description
Name	Required. Must be unique in the Vendors list.
Description	Optional description of the vendor.

Property	Description
IncomingScript	Optional. When you specify an IncomingScript, Cisco Access Registrar runs the script on all requests from clients that specify that vendor.
OutgoingScript	Optional. When you specify an OutgoingScript, Cisco Access Registrar runs the script on all responses to the Client.

## Scripts

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

You can write two 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.



### Note

For more information about how to write scripts and how to incorporate them into Cisco Access Registrar, see Chapter 5, “Using Extension Points.”

Table 3-8 lists the **Script** object properties.

**Table 3-8 Script Properties**

Property	Description
Name	Required. Must be unique in the Scripts list.
Description	Optional description of the script.
Language	Required. You must specify either <b>REX</b> or <b>Tcl</b> .
Filename	Required. You can specify either a relative or absolute path. When you specify a relative path, the path must be relative to the <b>\$INSTALL/scripts/radius/\$Language</b> directory. When you specify an absolute path, the server must be able to reach it.
EntryPoint	Optional. When you do not set this property, Cisco Access Registrar uses the value specified in the <b>Name</b> property.
InitEntryPoint	Optional. When you set it, it must be the name of the global symbol Cisco 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 you set it, it must be the arguments to be passed to the <b>InitEntryPoint</b> in the environmental variable <b>Arguments</b> .

The **InitEntryPoint** properties allow you to perform initialization before processing and then cleanup before stopping the server. For example, when Cisco 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 Cisco Access Registrar's file service, Cisco Access Registrar automatically closes any opened files; however, if you write scripts that manipulate files, you are responsible for closing them.

## Services

Cisco 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 Access Registrar also enables you to add new AAA services through custom shared libraries.

Table 3-9 lists the **Services** properties.

**Table 3-9 Services Properties**

Property	Description
Name	Required; must be unique in the Services list.
Description	Optional description of the service.
Type	Required, must set it to one of the following: <b>eap-leap</b> , <b>eap-md5</b> , <b>eap-sim</b> , <b>file</b> , <b>group</b> , <b>ldap</b> , <b>local</b> , <b>odbc</b> , <b>radius</b> , <b>radius-session</b> , <b>rex</b> , or <b>tacacs-udp</b> .
OutagePolicy	Required. The default is <b>RejectAll</b> . This property defines how Cisco Access Registrar handles requests if all servers listed in the <b>RemoteServers</b> properties are unavailable (that is, all remote RADIUS servers are not available). You must set it to one of the following: <b>AcceptAll</b> , <b>DropPacket</b> , or <b>RejectAll</b> .
OutageScript	Optional. If you set this property to the name of a script, Cisco 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 3-10 apply.

**Table 3-10 OutagePolicy Request Packets**

Value	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 packet will continue to flow through the server, including Session Management, if so configured, but no response will be sent. This allows you to configure the server so resources allocated by a SessionManager can be released as soon as possible, while still indicating to the client that it should keep retrying the request (with the hope the Service will be available).

## Types of Services

This section lists the types of services available with their required and optional fields. The service you specify determines what additional information you must supply. The following are the types

### local

Specify **local** when you want Cisco Access Registrar's RADIUS server to perform the authentication and/or authorization using a specific UserList. For more information, see the "UserLists" section on page 3-3.

### EAP-LEAP

Cisco Access Registrar 3.0 supports the new AAA Cisco-proprietary protocol called Light Extensible Authentication Protocol (LEAP).

The Cisco AR server supports Extensible Authentication Protocol (EAP) 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. (This type of EAP authentication mechanism is called an authentication exchange.)



#### Note

Cisco Access Registrar supports a subset of EAP to support LEAP. This is not a general implementation of EAP for Cisco Access Registrar.

The Cisco-Wireless or Lightweight Extensible Authentication Protocol (EAP-LEAP) is an EAP authentication mechanism where the user password is hashed based on an MD4 algorithm and verified by a challenge from both client and server.

Specify type **eap-leap** when you create an EAP-LEAP service. When you create an EAP-LEAP service type, you must also specify a `UserService` to perform AAA service. The `UserService` can be any configured authenticating service.

## EAP-MD5

Cisco Access Registrar 3.0 supports EAP-MD5, or MD5-Challenge, another EAP authentication exchange. In EAP-MD5 there is a CHAP-like exchange, and the password is hashed by a challenge from both client and server to verify the password is correct. Once verified correct, the connection may proceed, although the connection is periodically re-challenged (per RFC 1994).

Specify type **eap-md5** when you create an EAP-MD5 service. When you create an EAP-MD5 service type, you must also specify a `UserService` to perform AAA service. The `UserService` can be any configured authentication service.

## radius, ldap, or tacacs-udp

Specify one of the following Services when you want to use a particular remote server for:

- **radius**—authentication and/or authorization
- **ldap**—authentication and/or authorization



**Note** 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.

- **tacacs-udp**—authentication.

Configure the properties listed in Table 3-11 to use a remote server and a RADIUS, LDAP or tacacs-udp service.

**Table 3-11** *radius, ldap, or tacacs-udp Properties*

Property	Description
MultipleServersPolicy	Required. Must be set to either <b>Failover</b> or <b>RoundRobin</b> .  When you set it to <b>Failover</b> , Cisco Access Registrar directs requests to the first server in the list until it determines the server is off-line. At which time, Cisco Access Registrar redirects all requests to the next server in the list until it finds a server that is on-line.  When you set it to <b>RoundRobin</b> , Cisco Access Registrar directs each request to the next server in the RemoteServers list in order 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.

## file

You specify the **file** Service when you want Cisco 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.

When you specify a **file** Service, you must provide the information listed in Table 3-12.

**Table 3-12 File Properties**

Property	Description
FilenamePrefix	Required; a string that specifies where Cisco 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 <b>\$INSTALL/logs</b> directory. When you specify an absolute path, the server must be able to reach it. The default is <b>Accounting</b> .
MaxFileSize	Optional; stored as a string, but is composed of two parts, a number and a units indicator ( <i>&lt;n&gt; &lt;units&gt;</i> ) 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 ( <i>&lt;n&gt; &lt;units&gt;</i> ) in which the unit is one of: H, Hour, Hours, D, Day, Days, W, Week, Weeks. The default is one day.

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

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

- If the file was created and modified on the same date, the file name 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 file name is **FileNamePrefix-*<yyyymmdd>*-*<yyyymmdd>*-*<n>*.log**. The dates are creation, modification, and number.

## ODBC

To configure the ODBC Service, complete the following steps:

**Step 1** Using **aregcmd**, navigate to **//localhost /Radius/Services**.

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

  local-file/
  local-users/
```

**Step 2** Add a new Service.

```
--> add odbc

--> cd /Radius/Service/odbc-service
```

```
[ /Radius/Services/odbc-service ]
  Name = odbc-service
  Description =
  Type = odbc
  IncomingScript~ =
  OutgoingScript~ =
  OutagePolicy~ = RejectAll
  OutageScript~ =
  MultipleServersPolicy = Failover
  RemoteServers/
```

**Step 3** Change directory to RemoteServers and associate the ODBC server with the RemoteServers property.

--> **cd RemoteServers**

```
[ /Radius/Services/odbc-service/RemoteServers ]
```

--> **add odbc**

## rex

Specify the **rex** service type when you want to create a custom service by using a script for authentication, authorization, or accounting.

You must supply the information listed in Table 3-13.

**Table 3-13** *rex Properties*

Property	Description
Filename	Required. Must be either a relative or an absolute path to the shared library containing the Service. When the path name is relative, it must be relative to <b>\$INSTALL/Scripts/Radius/rex</b> .
EntryPoint	Required. Must be set to the function's global symbol.
InitEntryPoint	Required. Must be the name of the global symbol Cisco Access Registrar should call when it initializes the shared library and just before it unloads the shared library.  <b>Note</b> A rex service must have an InitEntryPoint even if the service only returns REX_OK.
InitEntryPointArgs	Optional. When you set it, it must be the arguments to be passed to the <b>InitEntryPoint</b> in the environmental variable <b>Arguments</b> .

For more information about scripting, see Chapter 5, "Using Extension Points." For more information about using the REX Attribute dictionary, see Appendix A, "Cisco Access Registrar Tcl and REX Dictionaries,".

# Session Managers

You can use session management 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.

Session Managers use Resource Managers, which in turn, manage a pool of resources of a particular type.

Table 3-14 lists the Session Manager properties.

**Table 3-14 Session Manager Properties**

Property	Description
Name	Required. Must be unique in the Session Managers list.
Description	Optional. Description of the Session Manager.
Resource Managers	Ordered list of Resource Managers.

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” section on page 2-6 and the “release-sessions” section on page 2-6.

## Session Creation

Cisco 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 Access Registrar to monitor Sessions even when it is not allocating resources. For example, when Cisco 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 Access Registrar will not create a new one.

When you do not want Cisco 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 Access Registrar server. When more than one Session Note is added, the **Session-Notes** entry should be a comma-separated list of entry names.

For 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:

```

$enviro put "Date" "Today is 12/15/98"
$enviro put "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 entries created. For example:

```

$enviro put "Session-Notes" "Date, Requested_IP_Address"

```

---

For 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:

```

pEnviro-->put(pEnviro, Date, "Today is 12/15/98.");
pEnviro-->put(pEnviro, 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:

```

pEnviro-->put(pEnviro, "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 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 12/14/98.", "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.

## 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/OutgoingScript):

```
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 once 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 once 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.

Each Resource Manager is responsible for examining the request and deciding whether to allocate a resource for the user, do nothing, or cause Cisco Access Registrar to reject the request.

Table 3-15 lists the Resource Manager properties.

**Table 3-15 Resource Manager Properties**

Property	Description
Name	Required. Must be unique in the Resource Managers list.
Description	Optional. Description of the Resource Manger.
Type	Required. Must be either <b>IP-Dynamic</b> , <b>IP-Per-NAS-Port</b> , <b>IPX-Dynamic</b> , <b>Group-Session-Limit</b> , <b>Home-Agent</b> , <b>User-Session-Limit</b> , or <b>USR-VPN</b> .

## 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 Appendix A, “Cisco Access Registrar Tcl and REX Dictionaries” for information on how to override these individual Resource Managers.

### IP-Dynamic

**IP-Dynamic** allows you to manage a pool of IP addresses from which you dynamically allocate IP addresses.

When you use this Resource Manager, supply the information listed in Table 3-16.

**Table 3-16 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.

### 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, supply the information listed in Table 3-17.

**Note**

You must have the same number of IP addresses and ports.

**Table 3-17 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. A 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, supply the information listed in Table 3-18.

**Table 3-18 IPX-Dynamic Property**

Property	Description
Networks	Required. Must be a valid set of numbers which correspond to your networks.

**Note**

You may not use IPX network number 0x0. If you attempt to configure a Resource Manager with an IPX network number of 0x0, validation will fail.

## Subnet-Dynamic

A **subnet-dynamic** Resource Manager was created to support the On Demand Address Pool feature. Subnet-dynamic resource managers are used 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 this Resource Manager, supply the information listed in Table 3-19.

**Table 3-19 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

## 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 once the configured limit has been reached.

When you use this Resource Manager, supply the information listed in Table 3-20.

**Table 3-20 Group-Session-Limit Property**

Property	Description
GroupSessionLimit	Required. Must be set to the maximum number of concurrent sessions for all users.

## 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 once the configured limit has been reached.

When you use this Resource Manager, supply the information listed in Table 3-21.

**Table 3-21 User-Session-Limit Property**

Property	Description
UserSessionLimit	Required. Must be set to the maximum number of concurrent sessions for a particular user.

## Home-Agent

**Home-Agent** is a new resource manager that supports dynamic HA assignment. You configure the home-agent resource manager with a list of IP addresses. The AR 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 Chapter 10, “Wireless Support”. When you use this Resource Manager, supply the information listed in Table 3-22.

**Table 3-22 Home-Agent Subdirectory**

Subdirectory	Description
Home-Agent-IPAddresses	A single IP address or a range of IP addresses

## 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, supply the information listed in Table 3-23.

**Table 3-23 USR-VPN Properties**

Property	Description
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 <b>RIP V2 Off</b> or <b>RIP V2 On</b> if the USR NAS is to run RIP Version 2 for the user.
Gateways	Required to set up a tunnel between the NAS and the Gateways.

## Gateway Subobject

The **Gateway** subobject includes a list of names of the Frame Relay Gateways for which to encrypt the session key.

If you use this Resource Manager, supply the information listed in Table 3-24.

**Table 3-24 Gateway Properties**

Property	Description
Name	Required. Must be unique in the Gateways list.
Description	Optional. Description of the gateway.
IPAddress	Required. The 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.

## 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 3-25 lists the **Profile** properties.

**Table 3-25 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 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 Appendix C, “RADIUS Attributes.” Table 3-26 lists the **Attribute** properties.

**Table 3-26 Attribute Properties**

Property	Description
Name=value	The attribute name is one of the attributes defined in the Attribute dictionaries. The value is appropriate for the type of attribute.

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 CAR server modifies the Request dictionary and inserts, filters and substitutes the attributes accordingly. You can set many translations for one packet and the CAR server applies these translations sequentially.

**Note**

Later translations can overwrite previous translations.

Table 3-27 lists the Translation properties.

**Table 3-27 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 CAR Rule engine.

The CAR 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 Cisco 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 3-28 lists the Translation Group properties.

**Table 3-28 TranslationGroups Properties**

Property	Description
Name	Required; must be unique in the Translations list.

**Table 3-28 TranslationGroups Properties (continued)**

Property	Description
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.

Table 3-29 lists the **RemoteServers** properties.

**Table 3-29 RemoteServers 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 <b>radius</b> , <b>ldap</b> , or <b>tacacs-udp</b> .
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.
Port	Required. The port to which Cisco Access Registrar sends proxy requests. You must specify a number greater than zero. There is no default port number, you must supply the correct port number for your remote server.
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 protocol you specify determines what additional information you must supply. The following are all of the protocols with their required and optional fields.

### radius

**radius** specifies a RADIUS server.

When you specify the **radius** protocol, supply the information in Table 3-30.

**Table 3-30 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 Access Registrar runs the IncomingScript after it receives the response.
OutgoingScript	Optional. When set, must be the name of a known outgoing script. Cisco Access Registrar runs the OutgoingScript just before it sends the proxy request to the remote server.
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 off-line. 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 default), the Cisco AR 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 Cisco AR server does not wait for the Accounting-Response and immediately returns an Accounting-Response to the client.

## Idap

**Idap** specifies an LDAP server. When you specify the **Idap** protocol, provide the information listed in Table 3-31.

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.

**Table 3-31 Idap 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.  <b>Note</b> Use InitialTimeout from above as a template, except this is timeout is specified in seconds.
HostName	Required. The LDAP server's host name or IP address.
BindName	Optional. The distinguished name (dn) to use when establishing a connection between the LDAP and RADIUS servers.

Table 3-31 *Idap Properties (continued)*

Property	Description
BindPassword	Optional. The password associated with the <b>BindName</b> .
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 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 <code>joe</code> , use the filter <code>uid=joe</code> .
UserPasswordAttribute	Required. This specifies which LDAP field the RADIUS server should check for the user’s password.
LimitOutstandingRequests	Required. The default is <b>FALSE</b> . Cisco Access Registrar uses this property in conjunction with the <b>MaxOutstandingRequests</b> property to tune the RADIUS server’s use of the LDAP server.  When you set this property to <b>TRUE</b> , the number of outstanding requests for this RemoteServer is limited to the value you specified in <b>MaxOutstandingRequests</b> . When the number of requests exceeds this number, Cisco 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 <b>LimitOutstandingRequests</b> to <b>TRUE</b> . The number you specify, which must be greater than zero, determines the maximum number of outstanding requests allowed for this remote server.
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 Access Registrar manages referrals by allowing the RADIUS server’s administrator to indicate an LDAP “referral attribute,” which may or may not appear in the user information returned from an LDAP query. When this information is returned from a query, Cisco Access Registrar assumes it is a referral and initiates another query based on the referral. Referrals can also contain referrals.  <b>Note</b> This is an LDAP v2 referral property.
ReferralAttribute	Required when you have specified a <b>MaxReferrals</b> value. This property specifies which LDAP attribute, returned from an LDAP search, to check for referral information.  <b>Note</b> This is an LDAP v2 referral property.
ReferralFilter	Required when you have specified a <b>MaxReferral</b> value. This is the filter Cisco Access Registrar uses when processing referrals. When checking referrals, the information Cisco 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 <b>Filter</b> property.  <b>Note</b> This is an LDAP v2 referral property.
PasswordEncryptionStyle	The default is <b>None</b> . You can also specify <b>crypt</b> , <b>dynamic</b> , <b>SHA-1</b> , and <b>SSHA-1</b> .

Table 3-31 *Idap Properties (continued)*

Property	Description
LDAPToRadiusMappings	<p>A list of name/value pairs in which the name is the name of the <b>ldap</b> attribute to retrieve from the user record, and the value is the name of the RADIUS attribute to set to the value of the <b>ldap</b> attribute retrieved.</p> <p>For example, when the <b>LDAPToRadiusMappings</b> has the entry: <b>FramedIPAddress = Framed-IP-Address</b>, the RemoteServer retrieves the <b>FramedIPAddress</b> attribute from the <b>ldap</b> user entry for the specified user, uses the value returned, and sets the Response variable <b>Framed-IP-Address</b> to that value.</p>
LDAPToEnvironmentMappings	<p>A list of name/value pairs in which the name is the name of the <b>ldap</b> attribute to retrieve from the user record, and the value is the name of the Environment variable to set to the value of the <b>ldap</b> attribute retrieved.</p> <p>For example, when the <b>LDAPToEnvironmentMappings</b> has the entry: <b>group = User-Group</b>, the RemoteServer retrieves the <b>group</b> attribute from the <b>ldap</b> user entry for the specified user, uses the value returned, and sets the Environment variable <b>User-Group</b> to that value.</p>
LDAPToCheckItemMappings	<p>A list of LDAP <i>attribute/value</i> pairs which must be present in the RADIUS access request and must match, both name and value, for the check to pass.</p> <p>For example, when the <b>LDAPToCheckItemMappings</b> has the entry: <b>group = User-Group</b>, the Access Request must contain the attribute <b>group</b>, and it must be set to <b>User-Group</b>.</p>
UseSSL	<p>A boolean field indicating whether you want Cisco Access Registrar to use SSL (Secure Socket Layer) when communicating with this RemoteServer. When you set it to TRUE, be sure to specify the <b>CertificateDBPath</b> field in the <b>Advanced</b> section, and be sure the port you specified for this RemoteServer is the SSL port used by the LDAP server.</p>

## odbc

**odbc** specifies an ODBC server. Cisco 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 3-32 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 3-32 *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. <b>Note</b> Use InitialTimeout from above as a template, except this is timeout is specified in seconds.
Protocol	Must be set to <b>odbc</b> .
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.
Data Source Connections	Required; default is 8. This represents the total number of connections Cisco AR can open with the ODBC server; total number of threads Cisco AR can create for the ODBC server.
ODBCDataSource	Required; defines all items required for the <b>odbc.ini</b> file. The Cisco AR server automatically creates the <b>odbc.ini</b> file based on these settings.
SQLDefinition	SQLDefinition properties define the SQL you want to execute. Type— <b>query</b> (Cisco AR supports only type <b>query</b> ). 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	A list of name and value pairs in which the name is the name of the data store attribute to retrieve from the user record, and the value is the name of the RADIUS attribute to set to the value of the data store attribute retrieved. The data store attributes must match those defined in the external SQL file.
ODBCToEnvironmentMappings	A list of name/value pairs in which the name is the name of the data store 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.

## tacacs-udp

**tacacs-udp** specifies a TACACS server. When you specify the **tacacs-udp** protocol, supply the information listed in Table 3-33.

Table 3-33 *tacacs-udp Properties*

Property	Description
MaxTries	Required. The number of times to send a proxy request to a remote server before deciding the server is off-line. You must specify a number greater than zero. The default is 3.
InitialTimeout	Required. The amount of time (in milliseconds) to wait for a response from the first proxy request. You must specify a number greater than zero. The default is 4000.

# Rules

A Rule is a function that selects services based on all input information used by the function.

## 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 Access Registrar needs a value for this property. For most of these properties, system defaults exist that you can safely use.

Table 3-34 lists the **Advanced** properties.

**Table 3-34 Advanced Object Properties**

Property	Description
LogServerActivity	Required. The default is FALSE, which means Cisco 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 Access Registrar logs all responses to the server log file.
MaximumNumberOfRadiusPackets	Required. The default is 1024. This is a <i>critical property</i> 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 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.
RequireNASsBehindProxyBeInClientList	Required. The default is FALSE. If you accept the default, Cisco 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 may 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" section on page 3-30.

Table 3-34 Advanced Object Properties

Property	Description
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 <b>Account-Request</b> 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 <b>Accounting-Requests</b> . The default value was determined to provide a reasonable compromise between the two alternatives.
SessionBackingStoreSynchronizationInterval	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.
RemoteLDAPServiceThreadTimerInterval	Required. Specified in milliseconds, the default is 10. This property governs how often the <b>ldap</b> 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.
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.
MaximumNumberOfUDPTacacsPackets	Required. The default is 100. This is a critical property you should set high enough to allow for the maximum number of simultaneous proxied requests to the remote TACACS server. If more requests come in than there are packets allocated, Cisco Access Registrar will drop those additional requests.
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 Access Registrar can read and process them. The default is probably sufficient for most sites. You can, however, raise or lower it as necessary.

Table 3-34 Advanced Object Properties

Property	Description
CertificateDBPath	Required if you are using an LDAP RemoteServer, and you want Cisco Access Registrar to use SSL when communicating with that LDAP RemoteServer. This property specifies the name of the file containing the client certificates to be used when establishing an SSL connection to an LDAP RemoteServer. It must be either the <b>cert5.db</b> certificate database used by Netscape Navigator 3.x (and above), or the <b>ServerCert.db</b> 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 <b>LogFileSize</b> 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, Gigabytes). <b>Note</b> 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 <b>LogFileSize</b> .
UseAdvancedDuplicateDetection	Required. The default is FALSE. Set this property to TRUE when you want Cisco Access Registrar to use a more robust duplicate request filtering algorithm. For more information on this property, see the “Advance Duplicate Detection Feature” section on page 3-30.
AdvancedDuplicateDetectionMemoryInterval	Required when the Advanced Duplicate Detection feature is enabled. This property specifies how long (in milliseconds) Cisco Access Registrar should remember a request. You must specify a number greater than zero. The default is 10,000.
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.
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. <b>Note</b> The classpath will always contain the directory <b>\$INSTALLDIR/scripts/radius/java</b> and all of the jar files in that directory.
JavaVMOptions	A String that can contain options that will 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.)

Table 3-34 Advanced Object Properties

Property	Description
ARIsCaseInsensitive	When set to FALSE, requires that you provide exact path names with regard to upper and lower case for all objects, subobjects, and properties. The default setting, TRUE, allows you to enter paths such as <code>/rad/serv</code> instead of <code>/Rad/Serv</code> .  <b>Note</b> Cisco AR 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 Cisco AR binds to IPADDR_ANY.
Ports/	Optional; allows you to use ports other than the default, 1645 and 1646. You can use this option to configure Cisco Access Registrar to use other ports,. If you add additional ports, however, Access Registrar will use the added ports and no longer use ports 1645 and 1646. These ports can still be used by adding them to the list of ports to use. For more information, refer to “Ports” section on page 3-31.
Interfaces	Optional; refer to “Interfaces” section on page 3-31
ReplyMessages	Optional; refer to “Reply Messages” section on page 3-31.
AttributeDictionary	Optional; refer to “Attribute Dictionary” section on page 3-33.
SNMP	Optional; refer to “SNMP” section on page 3-34.

## 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 Access Registrar will use the proxy’s IP address to identify the origin of the request.

## Advance Duplicate Detection Feature

Cisco 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 may 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) may cause Session Management problems.

One solution is the Advanced Duplicate Detection feature which causes Cisco 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) Cisco Access Registrar should remember a request.

**Note**

Enabling this feature causes Cisco 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 may need to be increased (which will increase the amount of memory used by Cisco Access Registrar) when the Advanced Duplicate Detection feature is enabled.

## Ports

The Ports list specifies which ports to listen to for requests. When you specify a port, Cisco 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 1645 and Accounting-Requests to 1646, however, Cisco Access Registrar does not check.

When you do not specify any ports, Cisco Access Registrar does the following:

- Reads the **/etc/services** file for the ports to use for access and accounting requests.
- Otherwise, uses the standard ports (1645 and 1646).

## 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 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).

## 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 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 3-35 lists the **Reply Message** properties.

Table 3-35 Reply Message Properties

Property	Description
Default	Optional. When you set this property, Cisco Access Registrar sends this value when the property corresponding to the reject reason is not set.
UnknownUser	Optional. When you set this property, Cisco Access Registrar sends back this value in the <b>Reply-Message</b> attribute whenever Cisco Access Registrar cannot find the user specified by <b>User-Name</b> .
UserNotEnabled	Optional. When you set this property, Cisco Access Registrar sends back this value in the <b>Reply-Message</b> attribute whenever the user account is disabled.
UserPasswordInvalid	Optional. When you set this property, Cisco Access Registrar sends back this value in the <b>Reply-Message</b> 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 Access Registrar sends back this value in the <b>Reply-Message</b> attribute whenever one of the Resource Managers was unable to allocate the resource for this request.
ServiceUnavailable	Optional. When you set this property, Cisco Access Registrar sends back this value in the <b>Reply-Message</b> attribute whenever a service the request needs (such as a RemoteServer) is unavailable.
InternalError	Optional. When you set this property, Cisco Access Registrar sends back this value in the <b>Reply-Message</b> attribute whenever an internal error caused the request to be rejected.
MalformedRequest	Optional. When you set this property, Cisco Access Registrar sends back this value in the <b>Reply-Message</b> attribute whenever a required attribute (such as <b>User-Name</b> ) is missing from the request.
ConfigurationError	Optional. When you set this property, Cisco Access Registrar sends back this value in the <b>Reply-Message</b> 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 <b>Authentication-Service</b> , and that object does not exist in the configuration, the reason reported is ConfigurationError.
IncomingScriptFailed	Optional. When you set this property, Cisco Access Registrar sends back this value in the <b>Reply-Message</b> attribute whenever one of the <b>IncomingScripts</b> fails to execute.
OutgoingScriptFailed	Optional. When you set this property, Cisco Access Registrar sends back this value in the <b>Reply-Message</b> attribute whenever one of the <b>OutgoingScripts</b> fails to execute.
IncomingScriptRejectedRequest	Optional. When you set this property, Cisco Access Registrar sends back this value in the <b>Reply-Message</b> attribute whenever one of the <b>IncomingScripts</b> rejects the Access-Request.
OutgoingScriptRejectedRequest	Optional. When you set this property, Cisco Access Registrar sends back this value in the <b>Reply-Message</b> attribute whenever one of the <b>OutgoingScripts</b> rejects the Access-Request.
TerminationAction	Optional. When you set this property, Cisco Access Registrar sends back this value in the <b>Reply-Message</b> attribute whenever Cisco 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 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 Appendix C, “RADIUS Attributes.”

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 3-36 lists the Attribute dictionary properties.

**Table 3-36 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 3-37. The type governs how the value is interpreted and printed.

## Types

**Types** are required and must be one of the following listed in Table 3-37.

**Table 3-37 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.

Property	Description
ENUM	Enums allow you to specify the mapping between the value and the strings. Once you have established this mapping, Cisco 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 3-38 lists the **Vendor** properties.

**Table 3-38 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 3-39 lists the five properties of the SNMP directory.

**Table 3-39 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
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 AR server is not using SNMP or if your site uses a different master agent, **arservagt** will not start your master agent.



## Using the radclient Command

This chapter describes how to use **radclient**, a RADIUS server test tool you run from the command line to test your Cisco 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, type:

```
radclient
```

After you enter the **radclient** command, you must log in to the RADIUS server and provide an administrator's username, such as admin, and the administrator's password.

## radclient Command Syntax

The **radclient** command syntax is:

```
radclient [-C <clustername>] [-N <adminname>] [-P <adminpassword>] [-i] [-n]  
[-p <load_path>] [-v] [-z debug_flags]
```

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, 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
- **-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 1 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.

### 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**, type:

```
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>`, type:

```
simple_chap bob bigDog 1
```

```
p002
```

### Viewing Packets

To view a packet or any other object, type the object identifier at the **radclient** prompt. For example, to display packet `p001`, type:

```
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 **1645**.

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 1812, type:

```
p001 send amazon 1812
```

```
p002
```

When Cisco 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 create empty packets, then 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, type:

```
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, type:

```
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, type:

```
<packet-identifier> delete
```

To delete all resources referred to by the packet p001, type:

```
p001 delete
```

# Attributes

Using the **radclient** command you can create specific RFC-defined attributes of requests and responses.

## 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`, type:

```
attrib User-Name bob
```

```
a001
```

**Note**

**a001** is the object identifier for the newly created attribute.

## 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, type:

```
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`, type:

**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.

### 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 4-1 lists the **radclient** variables used in **timetest** and **callsPerSecond** and their description.

**Table 4-1 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 <b>radclient</b> sends access-request packets (default is 1645). 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)

**Table 4-1** radclient Variables (continued)

Variable	Description
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 4-2 lists the applicable test types.

**Table 4-2** 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. Once 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 4-3 lists additional **radclient** variables and their description.

**Table 4-3 Additional radclient Variables**

Variable	Description
accounting_port	Port where the RADIUS server sends accounting packets (default is 1646). <b>Note</b> Changing accounting_port value does not affect the authentication port.
host	Name of host where Cisco AR 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 <b>radclient</b> sends access-request packets (default is 1645). 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

**Table 4-3 Additional radclient Variables (continued)**

<b>Variable</b>	<b>Description</b>
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 user names (default is user%d%%PPP)
UserPasswordPattern	Pattern of the user passwords (default is user%d)
verbose	Verbose flag for Tclsh





## Configuring Local Authentication and Authorization

---

Cisco CNS Access Registrar (AR) 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 Cisco AR to perform authentication and authorization using Cisco AR's 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. Cisco AR performs AA and returns the appropriate RADIUS attributes in an Access Accept packet.

### Configuring a Local Service and UserList

Cisco AR 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 AR userlist.

The following sections show the commands you enter and the expected responses from the Cisco AR server to do the following:

- Configuring a Local Service
- Configuring a Userlist
- Configuring Cisco AR 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

Cisco AR maintains **Services** under **/Radius**. To configure a local service, complete the following steps:

- 
- 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

Cisco AR maintains **UserLists** under **/Radius**. To configure a userlist, complete the following steps:

- 
- 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 AR to Use the Local Service For AA

To configure Cisco AR to use the local service for authentication and authorization, enter commands to set the DefaultAuthenticationService and DefaultAuthorizationService 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, Cisco AR attempts to validate the configuration, checks for all required properties, and ensures there are no logic errors. If the validation is successful, Cisco AR 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

This section lists steps you can take to verify the configuration changes you have made.

---

**Step 1** Check to see that the UserList exists under the service.

**Is /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.

**Is /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**.

```
04/23/2003 18:34:35: P1144: Packet received from 127.0.0.1
04/23/2003 18:34:35: P1144: Trace of Access-Request packet
04/23/2003 18:34:35: P1144:     identifier = 4
04/23/2003 18:34:35: P1144:     length = 62
04/23/2003 18:34:35: P1144:     reqauth = f5:37:f7:04:99:85:c7:63:8f:bc:f4:44:ab:03:4e:1a
04/23/2003 18:34:35: P1144:     User-Name = user1
04/23/2003 18:34:35: P1144:     User-Password = 59:fb:2e:a9:34:de:0e:15:60:8d:4b:64:77:6a:57:d8
04/23/2003 18:34:35: P1144:     NAS-Port = 2
04/23/2003 18:34:35: P1144:     NAS-Identifier = localhost
04/23/2003 18:34:35: P1144: Using Client: localhost (127.0.0.1)
04/23/2003 18:34:35: P1144: Using NAS: localhost (127.0.0.1)
04/23/2003 18:34:35: P1144: Request is directly from a NAS: TRUE
04/23/2003 18:34:35: P1144: Authenticating and Authorizing with Service SouthBay
04/23/2003 18:34:35: P1144: Getting User user1's UserRecord from UserList SouthUsers
04/23/2003 18:34:35: P1144: User user1's password matches
04/23/2003 18:34:35: P1144: No default Remote Session Service defined.
04/23/2003 18:34:35: P1144: Trace of Access-Accept packet
04/23/2003 18:34:35: P1144:     identifier = 4
04/23/2003 18:34:35: P1144:     length = 20
04/23/2003 18:34:35: P1144:     reqauth = 36:88:34:0c:cc:ea:9e:d8:6d:f5:14:f7:ab:26:e7:f6
04/23/2003 18:34:35: P1144: Sending response to 127.0.0.1
04/23/2003 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**.

```
04/23/2003 19:05:13: P1527: Packet received from 127.0.0.1
04/23/2003 19:05:13: P1527: Trace of Access-Request packet
04/23/2003 19:05:13: P1527:04/23/2003 19:05:13: P1527:04/23/2003 19:05:13: P1527:
04/23/2003 19:05:13: P1527:04/23/2003 19:05:13: P1527:
04/23/2003 19:05:13: P1527:04/23/2003 19:05:13: P1527:04/23/2003 19:05:13: P1527: Using Client: localhost
(127.0.0.1)
04/23/2003 19:05:13: P1527: Using NAS: localhost (127.0.0.1)
04/23/2003 19:05:13: P1527: Request is directly from a NAS: TRUE
04/23/2003 19:05:13: P1527: Authenticating and Authorizing with Service SouthBay
04/23/2003 19:05:13: P1527: Getting User user1's UserRecord from UserList SouthUsers
04/23/2003 19:05:13: P1527: User user1's password does not match
04/23/2003 19:05:13: P1527: Rejecting request
04/23/2003 19:05:13: P1527: Rejecting request
04/23/2003 19:05:13: P1527: Trace of Access-Reject packet
04/23/2003 19:05:13: P1527:04/23/2003 19:05:13: P1527:04/23/2003 19:05:13: P1527:
04/23/2003 19:05:13: P1527:04/23/2003 19:05:13: P1527: Sending response to 127.0.0.1
04/23/2003 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 Cisco AR server match.

## Configuring Return Attributes and Check-Items

Cisco AR supports RADIUS check item attributes at the user and group levels. You can configure Cisco AR 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 Cisco AR, refer to Appendix C, “RADIUS Attributes”.

When using check item attributes, Cisco AR 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

### 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 5551234
```

```
Set Callback-Number 5551234
```

### 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 Cisco AR 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:

```

04/24/2003 14:08:07: P1539: Packet received from 127.0.0.1
04/24/2003 14:08:07: P1539: Trace of Access-Request packet
04/24/2003 14:08:07: P1539:   identifier = 1
04/24/2003 14:08:07: P1539:   length = 71
04/24/2003 14:08:07: P1539:   reqauth = d6:86:c5:1e:0e:a0:20:4f:9a:1a:2c:35:27:16:23:36
04/24/2003 14:08:07: P1539:   User-Name = user1
04/24/2003 14:08:07: P1539:   User-Password = 99:dc:4a:22:ef:f6:8b:90:a2:3a:50:f0:a6:03:6e:b3
04/24/2003 14:08:07: P1539:   NAS-Port = 1
04/24/2003 14:08:07: P1539:   Calling-Station-Id = 5555678
04/24/2003 14:08:07: P1539:   NAS-Identifier = localhost
04/24/2003 14:08:07: P1539: Using Client: localhost (127.0.0.1)
04/24/2003 14:08:07: P1539: Using NAS: localhost (127.0.0.1)
04/24/2003 14:08:07: P1539: Request is directly from a NAS: TRUE
04/24/2003 14:08:07: P1539: Authenticating and Authorizing with Service SouthBay
04/24/2003 14:08:07: P1539: Getting User user1's UserRecord from UserList SouthUsers
04/24/2003 14:08:07: P1539: User user1's password matches
04/24/2003 14:08:07: P1539: Processing User user1's check items
04/24/2003 14:08:07: P1539: Merging User user1's Attributes into response Dictionary
04/24/2003 14:08:07: P1539: Merging attributes into the Response Dictionary:
04/24/2003 14:08:07: P1539:   Adding attribute Callback-Number, value = 5551234
04/24/2003 14:08:07: P1539:   Adding attribute Session-Timeout, value = 60
04/24/2003 14:08:07: P1539: No default Remote Session Service defined.
04/24/2003 14:08:07: P1539: Trace of Access-Accept packet
04/24/2003 14:08:07: P1539:   identifier = 1
04/24/2003 14:08:07: P1539:   length = 35
04/24/2003 14:08:07: P1539:   reqauth = cc:2d:51:71:b5:49:0e:e6:f1:eb:1c:61:51:7a:f1:cb
04/24/2003 14:08:07: P1539:   Callback-Number = 5551234
04/24/2003 14:08:07: P1539:   Session-Timeout = 60
04/24/2003 14:08:07: P1539: Sending response to 127.0.0.1
04/24/2003 14:08:07: Log: Request from localhost (127.0.0.1): User user1 accepted

```

## Configuring Profiles to Group Attributes

You can use the Cisco AR 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.

---

**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 may also be assigned to a UserGroup. You assign a profile to a group by setting the group's BaseProfile attribute to the desired profile.

**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 Cisco AR userlists that contain more than 10,000 users. Cisco AR userlists were not designed to contain 10,000 users in any one list.

If you must provide service for groups greater than 10000 users, Cisco recommends 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, Cisco AR 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.

```
04/25/2003 17:38:11 name/radius/1 Warning Protocol 0 Request from localhost (127.0.0.1):
User user1 rejected (UserPasswordInvalid)
```

```
04/25/2003 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:

```
04/25/2003 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 Access Registrar scripting to customize your RADIUS server. You can write scripts to affect the way Cisco Access Registrar handles and responds to requests, and to change the behavior of Cisco 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 Cisco Access Registrar or to communicate with other scripts.

The process for creating and implementing a script involves:

- Determining the goal of the script
- Writing the script
- Adding the new script definition to Cisco Access Registrar
- Choosing a scripting point from within Cisco Access Registrar
- Testing the script using the **radclient** command.

## 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 Cisco 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 Cisco Access Registrar processes client requests. Cisco 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 either Tcl or as shared libraries using C or C++. In this section, the scripts are shown in Tcl.

To write a script, do the following:

- 
- Step 1** Using an editor, create the Tcl source file.
  - 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 Cisco Access Registrar.

## 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.

- 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 examples show scripts using all three dictionaries.

## 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 Appendix A, "Cisco Access Registrar Tcl and REX Dictionaries." 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 Cisco 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 (only type rex)
  - 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.

---

## 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 Cisco Access Registrar server, type 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, type:

```
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 2, “Using the aregcmd Commands.”

## About the Tcl/Tk 8.3 Engine

Cisco Access Registrar 1.6 and above 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.

**Note**

---

In this release, scripts that use Tcl global variables will not work across AR extension points. A future release will address script compatibility issues.

---

Tcl/Tk8.3 also performs *byte-compile*, so no run-time interpretation is required.



## Using Replication

---

This chapter provides information about how to use the replication feature. When you set up replication, all configuration is replicated from the master server to members servers except for Administrators, Interfaces, Replication, and Advanced machine-specific settings.

### Setting Up Replication

This section provides step-by-step instructions about how to configure replication on both the master and member servers. The following section, “Replication Example” section on page 7-4, shows an example of replication configuration.

Two properties in **/Radius/Replication** control how long an out-of-sync condition lasts. The `RepTransactionSyncInterval` property controls the length of time between resynchronization processing. The `RepTransactionArchiveLimit` property controls the number of transactions to be archived before resynchronization processing takes place.

The `RepTransactionSyncInterval` property has a default setting of 60000 milliseconds (or one minute). If you set `RepTransactionSyncInterval` to a smaller value, you decrease the amount of time between resynchronization processing. Doing so reduces the amount of time that member servers are out of sync with the master server, but also results in additional network traffic, additional processing by the Cisco AR servers, and unnecessary resynchronization processing if the value is set too small. (Values as small as 10000 have been successfully tested.)

When you set the `RepTransactionArchiveLimit` property, you should consider the following:

- How much hard disk space can you devote to transaction archive storage?
- How often do you change your configuration (issue a **save** through **aregcmd**)?

If you have limited hard disk space, the default value of 100 transactions might be appropriate. If you have sufficient hard disk space, you can increase the `RepTransactionArchiveLimit` property value to a value of 10,000 or greater.

Your goal is to limit the need to perform a full-resynchronization. A full-resynchronization is required when the member has missed so many transactions that the master no longer contains all the transactions required to resynchronize the member.

Even by modifying these two properties, you might not be able to prevent the need for a full resynchronization, but you can limit the possibility.

## Configuring the Master

Use the following steps to configure the master server for replication. In a given configuration, you can have only one master server.

**Step 1** Launch **aregcmd** on the machine which is the master.

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

```
cd /Radius/Replication
```

**Step 3** Set the RepType to SMDBR.

```
set RepType SMDBR
```

**Step 4** Set the RepIPAddress to the IP address of the master.

```
set RepIPAddress 209.165.202.128
```

**Step 5** Set the RepSecret to MySecret.

```
set RepSecret MySecret
```

**Step 6** Set RepIsMaster to TRUE.

```
set RepIsMaster TRUE
```

**Step 7** Set RepMasterIPAddress to the same value used in step 3.

```
set RepIPMaster 209.165.202.128
```

**Step 8** Change directory to **/Radius/Advanced** and set the **MaximumNumberOfRadiusPackets** property to 8192.

```
cd /Radius/Advanced
```

```
set MaximumNumberOfRadiusPackets 8192
```

**Step 9** Change directory to **Rep Members**.

```
cd "rep members"
```



**Note** You must enclose Rep Members in quotes due to the space in the name.

**Step 10** Add **member1**.

```
add member1
```

**Step 11** Change directory to **member1**.

```
cd member1
```

**Step 12** Set the IPAddress to the IP Address of the machine to be the member.

```
set IPAddress 209.166.202.129
```



**Note** The RepPort and RepMasterPort properties on the Master must correspond to one of the ports configured in **/Radius/Advanced/Ports**, if one is configured. Otherwise, the default values for the RepPort and RepMasterPort properties are sufficient.

**Step 13** Save the configuration.

```
save
```

**Step 14** Reload the configuration.

```
reload
```

---

## Configuring The Member

Use the following steps to configure the member server for replication.

**Step 1** On the machine which is to be the member, using **aregcmd**, navigate to **//localhost/Radius/Replication**.

**Step 2** Set the RepType to SMDBR.

```
set RepType SMDBR
```

**Step 3** Set the RepIPAddress to the IP address of the member.

```
set RepIPAddress 209.166.202.129
```

**Step 4** Set the RepSecret to MySecret.

```
set RepSecret MySecret
```

**Step 5** Set RepMasterIPAddress to IP Address of the master (the same value used in Step 3 on page 1-1).

```
set RepMasterIPAddress 209.165.202.128
```

**Step 6** Change directory to **/Radius/Advanced** and set the **MaximumNumberOfRadiusPackets** property to 8192.

```
cd /Radius/Advanced
```

```
set MaximumNumberOfRadiusPackets 8192
```

**Step 7** If the Master has been configured to use a port other than the well-known (and default) RADIUS ports, configure each Member to use the same port.



**Note** The RepPort and RepMasterPort properties on the Master must correspond to one of the ports configured in **/Radius/Advanced/Ports**, if one is configured. Otherwise, the default values for the RepPort and RepMasterPort properties are sufficient.

---

**Step 8** Save the configuration.

```
save
```

**Step 9** Reload the configuration.

```
reload
```

---

## Verifying the Configuration

After both servers have successfully started, use **aregcmd** to make a small change to be replicated to the member server which you can easily verify. Cisco recommends setting the description in **//localhost/Radius** to something like *Test1*. After you issue an **aregcmd save** and the prompt returns, run **aregcmd** on the member server and change directory to **//localhost/Radius**. Ensure that the description is set to *Test1*. If this was successful, then replication is properly configured and functional.



### Note

Before the Cisco AR server can replicate changes, you must successfully **save** the replication configuration.

If possible, open an **xterm** window on both the master and member. In each of these windows, change directory to **\$INSTALL/logs** and run **xtail** to watch the logs. This allows you to watch replication log messages as they occur. If you are using a system which had a previous installation of Cisco Access Registrar, delete all files located in the **\$INSTALL/data/archive** directory if it is present on either the master or member systems.

## Replication Example

This section provides an example of replication and shows the actions that occur.

### Adding a User

On the master server, use **aregcmd** to add a new user to the default user list. To add a new user, perform the following steps.

- 
- Step 1** Change directory to **//localhost/Radius/UserLists/Default**.
  - Step 2** Enter the following.  
**add testuser**
  - Step 3** Change directory to **testuser**.  
**cd testuser**
  - Step 4** Set the password for **testuser**.  
**set password testuser**
  - Step 5** Confirm the password by entering **testuser** again.
  - Step 6** Enter **save** to save the configuration.
- 

### Master Server's Log

The log on the master shows the following:

```
*** ./name_radius_1_log ***
04/02/2001 23:17:07 name/radius/1 Info Server 0 Initiating Replication of Transaction
1 with 2 Elements.
```

```
04/02/2001 23:17:07 name/radius/1 Info Server 0 Replication Transaction #1 With 2
Elements Initiated
```

## Member Server's Log

The log on the member shows the following:

```
*** ./name_radius_1_log ***
4/02/2001 23:15:18 name/radius/1 Info Server 0 Radius Server is On-Line
04/02/2001 23:17:12 name/radius/1 Info Server 0 Committing Replication of Transaction
1 with 2 Elements.
04/02/2001 23:17:16 name/radius/1 Info Server 0 Replication Transaction #1 With 2
Elements Committed.
```

## Verifying Replication

You can use one of two methods to verify that the new user *testuser* was properly replicated to the member.

- Run **aregcmd** on the member and look at the default userlist to see if it is there.
- Run **radclient** on the member and enter **simple testuser testuser** to create a simple access request packet (p001).

Enter **p001 send** to send it. When it returns with p002, enter **p002** to see if it is an Access Accept packet or an Access Reject packet. If it is an Access Accept, the user was properly replicated to the member. Using **radclient** is the recommended method to validate that a user was properly replicated.

On the Master, use **aregcmd** to delete the user from the default user list and save the user list.

## Master Server's Log

The log on the master shows the following:

```
*** ./name_radius_1_log ***
04/02/2001 23:20:48 name/radius/1 Info Server 0 Initiating Replication of Transaction
2 with 1 Elements.
04/02/2001 23:20:48 name/radius/1 Info Server 0 Replication Transaction #2 With 1
Elements Initiated
```

## Member Server's Log

The log on the member shows the following:

```
*** ./name_radius_1_log ***
04/02/2001 23:20:53 name/radius/1 Info Server 0 Committing Replication of Transaction
2 with 1 Elements.
04/02/2001 23:20:57 name/radius/1 Info Server 0 Replication Transaction #2 With 1
Elements Committed.
```

Repeat the validation procedure above to ensure the user *testuser* is no longer present on the member.

## Using aregcmd -pf Option

Cisco Access Registrar's replication feature works well using **aregcmd** input files. An **aregcmd** input file contains a list of **aregcmd** commands. For example, if the initial configuration of Cisco Access Registrar were constructed in an input file, the master and member could be configured for replication first, then the input file applied to the master will be automatically replicated to the member.

To illustrate replication using an **aregcmd** input file, do the following:

---

**Step 1** Create a text file called **add5users** with the following contents:

```
add /Radius/UserLists/Default/testuser1
cd /Radius/UserLists/Default/testuser1
set password testuser1
add /Radius/UserLists/Default/testuser2
cd /Radius/UserLists/Default/testuser2
set password testuser2
add /Radius/UserLists/Default/testuser3
cd /Radius/UserLists/Default/testuser3
set password testuser3
add /Radius/UserLists/Default/testuser4
cd /Radius/UserLists/Default/testuser4
set password testuser4
add /Radius/UserLists/Default/testuser5
cd /Radius/UserLists/Default/testuser5
set password testuser5
save
```

**Step 2** On the master server, run the following command:

```
aregcmd -pf add5users
```

---

## Master Server's Log

The log on the master shows the following:

```
*** ./name_radius_1_log ***
04/02/2001 23:27:08 name/radius/1 Info Server 0 Initiating Replication of Transaction
3 with 10 Elements.
04/02/2001 23:27:08 name/radius/1 Info Server 0 Replication Transaction #3 With 10
Elements Initiated
```

## Member Server's Log

The log on the member shows the following:

```

*** ./name_radius_1_log ***
04/02/2001 23:27:12 name/radius/1 Info Server 0 Committing Replication of Transaction
3 with 10 Elements.
04/02/2001 23:27:17 name/radius/1 Info Server 0 Replication Transaction #3 With 10
Elements Committed.

```

When the prompt returns, go to the member and use **aregcmd** to view the **/radius/defaults/userlist**. There should be five users there named *testuser1* through *testuser5*.

## An Automatic Resynchronization Example

This example will illustrate resynchronization of the member. This will be accomplished by stopping the member, making changes on the master, then restarting the member forcing a resynchronization.

- 
- Step 1** At the member, stop the AR server:
- ```

/etc/init.d/arservagt stop

```
- At the master, run **aregcmd** and change directory to **/radius/userlist/default**.
- ```

cd /radius/userlist/default

```
- Step 2** Enter the following:
- ```

add fouser

```
- Step 3** Change directory to **fouser**.
- ```

cd fouser

```
- Step 4** Set the password for **fouser**.
- ```

set password fouser

```
- Step 5** Confirm the password by entering *fouser* again.
- Step 6** Save the configuration:
- ```

save

```

## Master Server's Log

The log on the master shows the following:

```

*** ./name_radius_1_log ***
04/02/2001 23:31:02 name/radius/1 Info Server 0 Initiating Replication of Transaction
5 with 2 Elements.
04/02/2001 23:31:02 name/radius/1 Info Server 0 Replication Transaction #5 With2
Elements Initiated

```

On the member, run **/etc/init.d/arservagt start**. Notice the following log messages in the Master's log:

```

*** ./name_radius_1_log ***
04/02/2001 23:33:19 name/radius/1 Info Server 0 Resynchronizing member1.

```

## Member Server's Log

The log on the member shows the following:

```

*** ./name_radius_1_log ***
04/02/2001 23:33:14 name/radius/1 Info Server 0 Radius Server is Off-Line

```

```

04/02/2001 23:33:14 name/radius/1 Info Server 0 Starting Replication Manager
04/02/2001 23:33:24 name/radius/1 Info Server 0 Master Selected As Partner (DEFAULT)
04/02/2001 23:33:24 name/radius/1 Info Server 0 Radius Server is Off-Line
04/02/2001 23:33:24 name/radius/1 Warning Server 0 Requesting resynchronization from
Master: Last Txn#3
04/02/2001 23:33:24 name/radius/1 Info Server 0 Resynchronization from Master in
progress.
04/02/2001 23:33:24 name/radius/1 Info Server 0 Committing Replication of Transaction
4 with 2 Elements.
04/02/2001 23:33:28 name/radius/1 Info Server 0 Replication Transaction #4 With 2
Elements Committed.
04/02/2001 23:33:28 name/radius/1 Info Server 0 Radius Server is On-Line

```

As the log above shows, when the member started up, it validated its last received transaction number (#3) with the master's last replicated transaction number (#4). They did not match because a replication was initiated by the master which was not received by the member (because the member was stopped). When the member detected this discrepancy, the member made a resynchronization request to the master. The master responded by transmitting the missed transaction (#4) to the member. After it received and processed the retransmitted transaction, the member determined that it was then synchronized with the master and placed itself in an on-line status.

## Full Resynchronization

Full Resynchronization means that the member has missed more transactions than are stored in the master's replication archive and can not be resynchronized automatically. There is no automatic full-resynchronization mechanism in Cisco Access Registrar's configuration replication feature. If a full resynchronization is required, you must export the master server's database and update the member configuration.



### Note

---

Before beginning, ensure there are no **aregcmd** sessions logged into the master server.

---

To perform a manual full-resynchronization, perform the following steps:

- 
- Step 1** On the master server, change directory to **\$INSTALL/data/db**.
  - Step 2** Create a tarfile made up of the three database files, **mcddb.d01**, **mcddb.d02**, and **mcddb.d03**.  

```
tar cvf /tmp/db.tar mcddb.d0*
```
  - Step 3** Create a tarfile of the archive.  

```
tar cvf /tmp/archive.tar $INSTALL/data/archive
```
  - Step 4** Copy the tarfiles (**db.tar** and **archive.tar**) to **/tmp** on each member server that requires full resynchronization.
  - Step 5** On a member server requiring resynchronization, change directory to **\$INSTALL/data/db**, then untar the compressed database files.  

```
cd $INSTALL/data/db
tar xvf /tmp/db.tar
```
  - Step 6** Rebuild the key files by typing the command:  

```
$INSTALL/bin/keybuild mcddb
```

This can take several minutes.

**Step 7** Untar the archive.

```
cd $INSTALL/data
tar xvf /tmp/archive.tar
```

**Step 8** As a safety check, run `$INSTALL1/bin/dbcheck mcddb` (UNIX) to verify the integrity of the database.




---

**Note** You must be user **root** to run **dbcheck**.

---

No errors should be detected.

**Step 9** Change directory to `//localhost` and re-enter the member's LicenseKey.

The current LicenseKey is that of the replication master.

```
cd //localhost

[ //RadiusServer ]
LicenseKey = WXYZ-WXYZ-WXYZ-WXYZ (expires in 30 days)
Radius/
Administrators/

set LicenseKey <ABCD>-<ABCD>-<ABCD>-<ABCD>
```

**Step 10** Change directory to `//radius/replication` and change the following attributes:

- Change the RepIPAddress to that of the member.
- Change RepIsMaster to FALSE.
- Remove any entries under Rep Members.

```
save

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

```
reload
```

The member will start up and show on-line status in the log after it has verified it is synchronized with the master.

---

## Replication Log Messages

This section contains typical replication log messages and explains what each means. This section include the following subsections:

- Information Log Messages
- Warning Log Messages
- Error Log Messages
- Log Messages You Should Never Receive

## Information Log Messages

### Starting Replication Manager

Displayed at start-up and indicates the Replication Manager is configured and enabled. (RepType=SMDBR)

### Replication Disabled

Displayed at start-up and indicates that Replication is not enabled. (RepType=NONE)

### Radius Server is On-Line

Displayed by the member at start-up to indicate the member is synchronized with the master and processing RADIUS requests. It is also displayed after a successfully completed resynchronization. This message is never displayed on the master.

### Radius Server is Off-Line

Displayed by the member at start-up to indicate the radius server is not processing RADIUS requests until it can ensure synchronization with the master. When this is displayed after startup, it indicates the member is no longer synchronized with the master and is directly associated with a resynchronization request to the master. This message is never displayed on the master.

### Resynchronizing <member name>

Displayed by the master to indicate that it is resynchronizing the specified member (member).

### Resynchronization from Master in progress.

Displayed by the member to indicate the master is in the process of resynchronizing it.

### Resynchronization complete.

Displayed by the member to indicate the resynchronization has completed successfully.

### Resynchronization did not complete before timeout. Retrying.

Indicates the master did not complete the resynchronization before the member expected it to complete and that the member is re-requesting resynchronization from the master for the remaining missed transactions.

### Master Selected As Partner (DEFAULT)

Displayed by the member to indicate it has successfully connected with the master.

Initiating Replication of Transaction <transaction #> with <# of elements> Elements.

Displayed by the master to indicate that it is beginning replication of a transaction to the member.

Replication Transaction #<transaction #> With <# of elements> Elements Initiated

Displayed by the master to indicate that it has completed sending the transaction to the member.

Committing Replication of Transaction <transaction #> with <# of elements> Elements.

Displayed by the member to indicate that it has received a transaction and is processing it.

Replication Transaction #<transaction#> With <# of element> Elements Committed

Displayed by the member to indicate that the transaction has been successfully processed.

Stopping Replication Manager

Displayed at shutdown by both the master and member to indicate the replication manager is being shut down.

Stopping Replication Manager - waiting for replication to complete...

Displayed by the member when a shutdown is attempted while received replications are being processed. Once the replications are complete, the shutdown will complete.

Replication in progress. Please wait...

Periodically displayed while a shutdown is pending and replications are being completed.

Replication Manager Stopped

Displayed by both the master and member to indicate the replication manager has been successfully shutdown.

## Warning Log Messages

Transaction Sync not received within configured TransactionSyncInterval.  
Communication with the Master may not be possible.

The member displays this log messages to indicate that it has not received a TransactionSync message from the master within its configured TransactionSync interval.

TXN\_SYNC Received by Master from unknown member <ip address>. Validation Failed

Displayed by the master when a TransactionSync message is received by the master. Since there can be only one configured master in a replication network, and the master is the only server who can send a TransactionSync message, this indicates there is another configured master in the replication network.

TXN\_SYNC Received from unknown Master <ip address>. Validation Failed

Displayed by the member to indicate that a TransactionSync message was received from a server not configured as its master.

Requesting resynchronization from Master: Last Txn#<transaction#>

Displayed by the member to indicate that it is requesting resynchronization from the master. The LastTxn# is the last transaction number the member received and processed successfully.

Resynchronization Request received from unknown member.

Displayed by the master when a resynchronization request is received by a member who is not listed in its **/radius/replication/rep** members configuration.

Resynchronization of <member name> requires Full Resynchronization.

Displayed by the master to indicate that the member cannot be automatically resynchronized because its last transaction number is not within the configured history length of the archive (TransactionArchiveLimit). A manual resynchronization of the member is required to put the member back in-sync.

MEMBER\_SYNC Received from unknown Master at <ip address>. Validation Failed

Displayed by a member indicating that a master, other than its configured master, is requesting partnership.

MEMBER\_SYNC Received by Master from unknown member <ip address>. Validation Failed

Displayed by the master to indicate a member not listed in its **/radius/replication/rep** members configuration has requested partnership.

TXN\_EXPECT Received by Master from unknown <ip address>.

Displayed by the master to indicate it has received a transaction which originated from another illegal master.

TXN\_EXPECT Received from unknown Master <ip address>.

Displayed by the member to indicate it has received a transaction which originated from a master other than its configured master.

TXN\_EXPECT Broadcast failed.

Indicates that the master could not initiate a replication.

DATA\_SYNC Received by Master from unknown <ip address>

Displayed by the master to indicate that it received a replication transaction from another illegal master.

DATA\_SYNC Received from unknown <ip address>

Displayed by the member to indicate that a transaction was received from a server external to the replication network.

## Error Log Messages

DATA\_SYNC Validation failed - CRC Mismatch

Displayed by the member to indicate a received transaction element is invalid.

TXN\_SYNC: Failed To Get Member Socket Handle.

TXN\_SYNC: Failed to get master's socket handle.

MEMBER\_SYNC could not get socket handle

TXN\_EXPECT: Failed to get socket handle.

DATA\_SYNC could not get socket handle.

These messages indicate an invalid interface configuration in Cisco Access Registrar.

They could also be the result of specifying an invalid RepPort setting.

Failed To Create TXN\_SYNC packet. (out of packets?)

Failed To Create TXN\_SYNC packet.

MEMBER\_SYNC Failed to create packet.(out of packets?)

MEMBER\_SYNC Failed to create packet.

TXN\_EXPECT Failed to create packet.(out of packets?)

TXN\_EXPECT Failed to create packet.

DATA\_SYNC Create packet failed.(out of packets?)

DATA\_SYNC Create packet failed.

These message indicate that a packet could not be created. This could be the result of a low memory condition or the result of the /Radius/Advanced/ MaximumNumberOfRadiusPackets setting being set too low

TXN\_SYNC validation failed - Internal error (pTxnSync=NULL).

MEMBER\_SYNC validate failed - Internal Error (pMemberSync=NULL)

DATA\_SYNC Validation Failed - Internal (pDataSync = NULL).

TXN\_EXPECT Could not add new datablock to pending transaction queue.

Replication Member could not be added to member list.

Replication Member could not be added to member list.

These messages are the result of a failed memory allocation possibly due to an out of memory condition.

```
DATA_SYNC Packet creation failed - Invalid ordinal.
Attempt To Replicate Transaction With Zero Elements.
Internal Error - Selected member not valid
Internal Replication Error ChangeType <change type> For <element path>
Internal error - Replication manager is invalid
```

These messages indicate an internal application failure.

```
Cannot archive transaction datablock
Could not archive transaction
```

These messages are the result of a failed archive attempt. This could be the result of a low disk space condition.

```
Could not commit transaction to MCD
Cannot Get Value For Unsupported DataType <data type id>
MCD Replication Cannot Delete Value <element path>
MCD Replication Cannot Delete Directory <element path>
MCD Replication Cannot Delete Value For <element path> With Unsupported DataType
<data type id>
MCD Replication Cannot Create Dir For <element path>
MCD Replication Cannot Set Value For <element path>
MCD Replication Cannot Set Value For <element path>
MCD Replication Cannot Set Value For <element path>
MCD Replication Cannot Set Value For <element path>
MCD Replication Cannot Set Value For <element path> With Unsupported DataType
<data type id>
MCD Replication Cannot Set Value For <element path> With UNKNOWN DataType <data
type id>
```

These messages are the result of a failed replication commit attempt.

## Log Messages You Should Never See

The following list contains log messages which you should never see displayed in a log. If any of these messages are displayed in the log, contact Cisco Access Registrar technical support for assistance.

```
<member name> Selected As Partner (DEFAULT)
DATA_SYNC Received from non-partner <ip address>
DATA_RE_SYNC CRC mismatch. Replying with NAK
DATA_RE_SYNC Commit Failed. Replying with NAK
EVAL_SYNC Validation failed. <ip address> is not a Master or Member of the
Replication network
EVAL_SYNC Received from unknown member.
PARTNER_SYNC Received from unknown member <ip address>.
PARTNER_SYNC Received from unknown member <ip address>.
EVAL_SYNC Cannot get socket handle.
EVAL_SYNC Failed to create packet. (out of packets?)
EVAL_SYNC Failed to create packet.
EVAL_SYNC Validation failed - Internal Error (pEvalSync=NULL).
PARTNER_SYNC Failed to get socket handle.
PARTNER_SYNC Failed to create packet. (out of packets?)
PARTNER_SYNC Failed to create packet.
```

```
DATA_RE_SYNC Can't get socket handle
DATA_RE_SYNC Failed to create packet (out of packets?)
DATA_RE_SYNC Failed to create packet
DATA_RE_SYNC Failed validation - Internal Error (pReSync = NULL)
DATA_RE_SYNC Cannot Set Value For <element path>
DATA_RE_SYNC Cannot Set Value For <element path>
DATA_RE_SYNC Cannot Set Value For <element path>
DATA_RE_SYNC Cannot Set Value For <element path>
DATA_RE_SYNC Cannot Set Value For <element path> With Unsupported DataType <data
type id>
DATA_RE_SYNC Cannot Set Value For <element path> With UNKNOWN DataType <data type
id>;
DATA_RE_SYNC Received by Master from unknown member <ip address>
DATA_RE_SYNC Received from unknown Master <ip address>DATA_RE_SYNC Reply received
by Master from unknown Member <ip address>
Could not replicate data element to partners.
Could not replicate to partners - Invalid Ordinal.
```



## Using On-Demand Address Pools

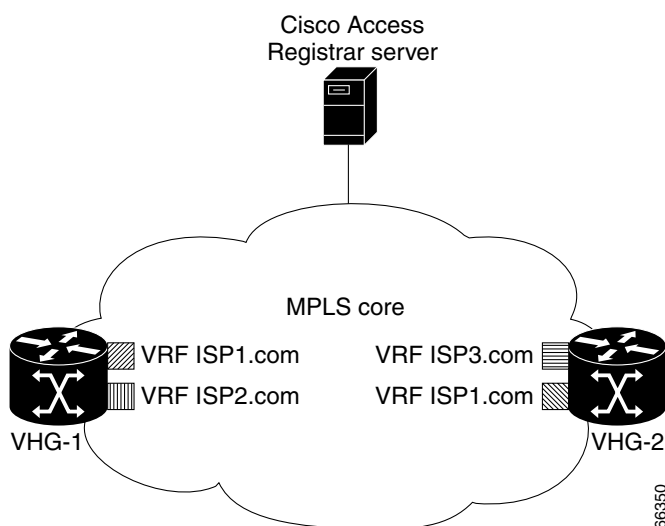
Cisco Access Registrar provides support for On-Demand Address Pools (ODAP). Using ODAP, the Cisco AR server manages pools of addresses. Each pool is divided into subnets of various sizes, and the Cisco AR server assigns the subnets to virtual home gateways (VHG) and Provider Edge (PE) routers. The VHG/PE router has one On-Demand Address Pool configured for each VPN supported by that VHG/PE.

Cisco Access Registrar has been enhanced to make ODAP functionality more accessible and to enable ODAP requests and normal user authentication to occur on the same Cisco Access Registrar server. To achieve this functionality, a new Cisco vendor script **CiscoWithODAPIncomingScript** was written to direct ODAP requests to particular services and session managers. **CiscoWithODAPIncomingScript** also provides the same functionality as the previous **CiscoIncomingScript**.

Additionally, Cisco Access Registrar has a new vendor type, **CiscoWithODAP** which references **CiscoWithODAPIncomingScript** as its IncomingScript and references the existing script, **CiscoOutgoingScript**, as its Outgoing Script.

Figure 8-1 shows a simple MPLS VPN network with two VHG/PE routers, VHG-1 and VHG-2. The Cisco AR server allocates IP subnets to the VHGs by way of VRFs which contain the subnets and addresses (address space) available.

**Figure 8-1** MPLS Core



In Cisco AR, the VRFs are configured as users in an ODAP-users list under **/Radius/UserLists**. The VRF name is set in IOS for the ODAP pool. When a VRF requests a pool of addresses, Cisco AR directs the request to a Session-Manager configured with the name **odap-<VRF name>**. Cisco AR also directs ODAP accounting requests to the service odap-accounting.

In the example network shown in Figure 8-1, the VRFs are configured with the following address spaces:

- VRF-ISP1.com—consists of the address range 10.255.0.0 - 10.255.255.255 divided among the following subnets:
  - 10.255.0.0/24
  - 10.255.1.0/24
  - ...
  - 10.255.255.0/24
- VRF-ISP2.com—consists of the address ranges 10.0.0.0 - 10.10.255.255 and 10.255.0.0 - 10.255.10.255 divided among the following subnets:
  - 10.0.0.0/16
  - 10.1.0.0/16
  - ...
  - 10.10.0.0/16

and:

- 10.255.0.0/24
- 10.255.1.0/24
- ...
- 10.255.10.0/24




---

**Note** VRF-ISP2.com requires two ResourceManagers because it has subnets of two different sizes.

---

- VRF-ISP3.com—consists of the address range 172.21.0.0 - 172.21.255.255 divided among the following subnets:
    - 172.21.0.0/18
    - 172.21.64.0/18
    - 172.21.128.0/18
- and
- 172.21.192.0/24
  - 172.21.193.0/24
  - ...
  - 172.21.255.0/24




---

**Note** VRF-ISP3.com requires two ResourceManagers because it also has subnets of two different sizes.

---

# Cisco-Incoming Script

Cisco Access Registrar 1.7R1 includes a new Cisco AR script, **CiscoWithODAPIncomingScript**, that makes ODAP functionality more accessible. The script eases the configuration required to enable ODAP requests and normal user authentication to occur on the same Cisco Access Registrar server.

**CiscoWithODAPIncomingScript** also provides the functionality of the original **CiscoIncomingScript**.

If the Cisco AR server receives an ODAP request, the server sets the Session-Key from the AcctSessionID and sets the services and session managers.

If the Cisco AR server receives a non-ODAP request, other scripts, rules or policies that you might already have in place on the Cisco AR server handle these requests.

## How the Script Works

The following describes how the script **CiscoWithODAPIncomingScript** works.

1. The script examines the incoming NAS-Identifier sent by the client (VHG). If the NAS-Identifier does not equal *odap-dhcp* then this request is not an ODAP request. Since this is not an ODAP request, the script does not do any more ODAP-specific processing and just calls **CiscoIncomingScript** to allow that script to process the request. If this is an ODAP request, this script removes the NAS-Identifier attribute because it is no longer needed.
2. The script sets the Authentication-Service and the Authorization-Service to *odap-users*, and it sets the Accounting-Service to *odap-accounting*.
3. The Cisco AR server sends the request to the appropriate Session Manager based on the username. Session Managers with *odap-<username>* must be created and configured in Cisco AR.
4. The script then uses Session IDs to identify each ODAP request. The script uses the Acct-Session-Id attribute as the Session-Key.

## CiscoWithODAPIncomingScript

The following is a Tcl script example of the script **CiscoWithODAPIncomingScript**.



### Note

**CiscoWithODAPIncomingScript** is written in C language. This example script is more easily understood in Tcl.

```
proc CiscoWithODAPIncomingScript {request response environ} {

    set RequestType [ $environ get Request-Type ]

    if { [ string compare $RequestType "Access-Request" ] == 0 ||
        [ string compare $RequestType "Accounting-Request" ] == 0 } {

        set NasID [ $request get NAS-Identifier ]

        if { [ string compare $NasID "odap-dhcp" ] == 0 } {
```

```

# Remove the NAS-Identifier - it has done it's job
$request remove NAS-Identifier

set UserName [ $environ get User-Name ]
if { [ string length $UserName ] == 0 } { set UserName [ $request get User-Name ] }

# ODAP SUBNET ASSIGNMENT
$environ put Authentication-Service "odap-users"
$environ put Authorization-Service "odap-users"
$environ put Accounting-Service "odap-accounting"
$environ put Session-Manager "odap-$UserName"

set AcctSessionId [ $request get Acct-Session-Id ]
if { [ string length $AcctSessionId ] != 0 } { $environ put Session-Key $AcctSessionId
} else {
$environ log LOG_ERROR "Missing Acct-Session-Id attribute in request-unable to set Session-Key"
}
}
}
CiscoIncomingScript $request $response $environ
}

```

**Note**


---

The final line in the example above is not how the script really works because a Tcl script can't call a C script. This is one reason why **CiscoWithODAPIncomingScript** was written in C.

---

## Vendor Type CiscoWithODAP

Cisco Access Registrar 1.7R1 includes a new vendor type, **CiscoWithODAP**. You must configure any Clients that might forward ODAP requests to the Cisco AR server as being of Vendor **CiscoWithODAP**.

This vendor type references the new script, **CiscoWithODAPIncomingScript**, as its IncomingScript and references the existing script, CiscoOutgoingScript, as its OutgoingScript.

After setting Vendor to **CiscoWithODAP**, ODAP requests are directed to the AA service, set to **odap-users**, the accounting service is set to **odap-accounting**, and the Session Manager is set to **odap-username**, where username is filled from the request. The username received in the request is a VRF name, the request is directed to the appropriate Session Manager.

## Configuring Cisco Access Registrar to Work with ODAP

This section provides information about how to configure Cisco AR to work with ODAP.

### Configuration Summary

This section provides the steps required to configure Cisco AR to work with ODAP. For detailed information about configuring Cisco AR to work with ODAP, refer to the following section, Detailed Configuration.

1. Create and configure an ODAP-users UserList  
All ODAP users are configured under this UserList.
2. Add all ODAP users to the ODAP-users UserList

- Username must be of the form `<vrf name>` with the `AllowAnonymousPassword` property set to `TRUE`.
3. Create and configure a service for ODAP-users
  4. Create and configure an ODAP accounting service  
Set the accounting service Type to *file* and FilenamePrefix *odap-accounting*.
  5. Create a Session Manager for each of the VRFs  
There must be a separate Session Manager for each VRF pool.
  6. Create and configure Resource Managers to be referenced by the Session Managers  
Subnet pools of different sizes (different subnet masks) require separate Resource Managers.
  7. Configure the Session Managers with the Resource Managers
  8. Configure any Clients that might send ODAP requests to Vendor type `CiscoWithODAP`
  9. Save your configuration

## Detailed Configuration

The following steps provide a detailed description of configuring Cisco Access Registrar to work with ODAP.

### Setting Up an ODAP UserList

**Step 1** Create a UserList for ODAP users.

```
--> cd /radius/userlists

[ //localhost/Radius/UserLists ]
  Entries 1 to 1 from 1 total entries
  Current filter: <all>

  Default/

--> add odap-users

Added odap-users
```

### Adding ODAP Users

**Step 2** Add the ODAP users to the ODAP UserList and set the `AllowAnonymousPassword` property to `TRUE`.

Each user is a VRF name set for each ODAP client.

```
[ //localhost/Radius/UserLists/odap-users ]

  Entries 0 to 0 from 0 total entries
  Current filter: <all>

  Name = odap-users
  Description =

--> add vrf-ISP1.com
```

```

Added vrf-ISP1.com

--> add vrf-ISP2.com

Added vrf-ISP2.com

--> add vrf-ISP3.com

Added vrf-ISP3.com

--> ls

[ //localhost/Radius/UserLists/odap-users ]
  Entries 1 to 3 from 3 total entries
  Current filter: <all>

  Name = odap-users
  Description =
  vrf-ISP1.com/
  vrf-ISP2.com/
  vrf-ISP3.com/

```

**Step 3** Set the AllowNullPassword property to TRUE for each ODAP user.

```

--> cd vrf-ISP2.com

[ //localhost/Radius/UserLists/odap-users/vrf-ISP2.com ]
  Name = vrf-ISP2.com
  Description =
  Password =
  Enabled = TRUE
  Group~ =
  BaseProfile~ =
  AuthenticationScript~ =
  AuthorizationScript~ =
  UserDefined1 =
  AllowNullPassword = FALSE

--> set AllowNullPassword TRUE

```

## Setting Up an ODAP-Users Service

**Step 4** Add and configure a service for ODAP Users.

```

--> cd /radius/services

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

  local-file/
  local-users/

--> add odap-users

```

```
Added odap-users

--> cd odap-users

[ //localhost/Radius/Services/odap-users ]
  Name = odap-users
  Description =
  Type =
  IncomingScript~ =
  OutgoingScript~ =

--> set type local

Set Type local

--> set userlist odap-users

Set UserList odap-users

--> ls

[ //localhost/Radius/Services/odap-users ]
  Name = odap-users
  Description =
  Type = local
  IncomingScript~ =
  OutgoingScript~ =
  OutagePolicy~ = RejectAll
  OutageScript~ =
  UserList = odap-users
```

## Setting Up an ODAP Accounting Service

**Step 5** Add and configure an ODAP accounting service.

```
--> cd /radius/services

[ //localhost/Radius/Services ]
  Entries 1 to 3 from 3 total entries
  Current filter: <all>

  local-file/
  local-users/
  odap-users/

--> add odap-accounting

Added odap-accounting

--> cd odap-accounting

[ //localhost/Radius/Services/odap-accounting ]
  Name = odap-accounting
  Description =
  Type =
  IncomingScript~ =
  OutgoingScript~ =
```

```

--> set type file

Set Type file

--> ls

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

--> set FilenamePrefix odap-accounting

Set Filenameprefix odap-accounting

```

## Adding Session Managers

**Step 6** Create one Session Manager for each of the VRF pools.

Create one Session Manager for each of the users you specify in the odap-users UserList. The Session Managers must be called odap-VRF\_name to meet the requirements of CiscoWithODAPIncomingScript.

```

--> cd /radius/sessionmanagers

[ //localhost/Radius/SessionManagers ]
  Entries 1 to 1 from 1 total entries
  Current filter: <all>

  session-mgr-1/

--> add odap-vrf-ISP1.com

Added odap-vrf-ISP1.com

--> add odap-vrf-ISP2.com

Added odap-vrf-ISP2.com

--> add odap-vrf-ISP3.com

Added odap-vrf-ISP3.com

```

## Setting Up Resource Managers

**Step 7** Set up subnet-dynamic Resource Managers that are to be referenced by the Session Managers.

Session Managers can manage multiple Resource Managers. One or more subnet pools can be set up of varying sizes to allocate the ranges of subnet addresses you have available. Subnets of different sizes require different Resource Managers.

```
--> cd /radius/resourcemanagers
```

```
[ //localhost/Radius/ResourceManagers ]
  Entries 1 to 5 from 5 total entries
  Current filter: <all>
```

```
  IPA-Pool/
  IPA-Pool-2/
  IPX-Pool/
  Per-Group/
  Per-User/
```

```
--> add odap-vrf-ISP1.com
```




---

**Note** The names of Resource Managers do not have to be related to VRFs.

---

```
Added odap-vrf-ISP1.com
```

```
--> cd odap-vrf-ISP1.com
```

```
[ //localhost/Radius/ResourceManagers/odap-vrf-ISP1.com ]
  Name = odap-vrf-ISP1.com
  Description =
  Type =
```

```
--> set type subnet-dynamic
```

```
Set Type subnet-dynamic
```

```
--> ls
```

```
[ //localhost/Radius/ResourceManagers/odap-vrf-ISP1.com ]
  Name = odap-vrf-ISP1.com
  Description =
  Type = subnet-dynamic
  NetMask =
  SubnetAddresses/
```

```
-> set netmask 255.255.255.0
```

```
Set NetMask 255.255.255.0
```

```
-> cd subnetaddresses
```

```
[ //localhost/Radius/ResourceManagers/odap-vrf-ISP1.com/SubnetAddresses ]
  Entries 0 to 0 from 0 total entries
  Current filter: <all>
```

```
--> add 10.255.0.0-10.255.255.255
```

```
Added 10.255.0.0-10.255.255.255
```

**Note**

Two Resource Managers are required for VRF-ISP3.com and VRF-ISP2.com because their address spaces are made up of subnets of the different sizes.

```
--> cd /radius/resourcemanagers
```

```
[ //localhost/Radius/ResourceManagers ]
  Entries 1 to 5 from 5 total entries
  Current filter: <all>
```

```
  IPA-Pool/
  IPA-Pool-2/
  IPX-Pool/
  odap-vrf-ISP1.com/
  Per-Group/
  Per-User/
```

```
--> add odap-vrf-ISP3-a.com
```

```
Added odap-vrf-ISP3-a.com
```

```
--> cd odap-vrf-ISP3-a.com
```

```
[ //localhost/Radius/ResourceManagers/odap-vrf-ISP3-a.com ]
  Name = odap-vrf-ISP3-a.com
  Description =
  Type =
```

```
--> set type subnet-dynamic
```

```
Set Type subnet-dynamic
```

```
--> ls
```

```
[ //localhost/Radius/ResourceManagers/odap-vrf-ISP3-a.com ]
  Name = odap-vrf-ISP3-a.com
  Description =
  Type = subnet-dynamic
  NetMask =
  SubnetAddresses/
```

```
-> set netmask 255.255.192.0
```

```
Set NetMask 255.255.192.0
```

```
-> cd subnetaddresses
```

```
[ //localhost/Radius/ResourceManagers/odap-vrf-ISP3-a.com /SubnetAddresses ]
  Entries 0 to 0 from 0 total entries
  Current filter: <all>
```

```
--> add 171.21.0.0-172.21.191.255
```

```
Added 172.21.0.0-172.21.191.255

-> cd /radius/resourcemanagers
[ //localhost/Radius/ResourceManagers ]
  Entries 1 to 10 from 10 total entries
  Current filter: <all>

  IPA-Pool/
  IPA-Pool-2/
  IPX-Pool/
  odap-vrf-ISP1.com/
  odap-vrf-ISP3-a.com /
  Per-Group/
  Per-User/

--> add odap-vrf-ISP3-b.com

Added odap-vrf-ISP3-b.com

--> cd odap-vrf-ISP3-b.com

[ //localhost/Radius/ResourceManagers/odap-vrf-ISP3-b.com ]
  Name = odap-vrf-ISP3-b.com
  Description =
  Type =

--> set type subnet-dynamic

Set Type subnet-dynamic

--> ls

[ //localhost/Radius/ResourceManagers/odap-vrf-ISP3-b.com ]
  Name = odap-vrf-ISP3-b.com
  Description =
  Type = subnet-dynamic
  NetMask =
  SubnetAddresses/

-> set netmask 255.255.255.0

Set NetMask 255.255.255.0

-> cd subnetaddresses

[ //localhost/Radius/ResourceManagers/odap-vrf-ISP3-b.com /SubnetAddresses ]
  Entries 0 to 0 from 0 total entries
  Current filter: <all>

--> add 172.21.191.0-172.21.255.255

Added 172.21.191.0-172.21.255.255

-> cd /radius/resourcemanagers
[ //localhost/Radius/ResourceManagers ]
  Entries 1 to 10 from 10 total entries
```

```

Current filter: <all>

IPA-Pool/
IPA-Pool-2/
IPX-Pool/
odap-vrf-ISP1.com/
odap-vrf-ISP3-a.com /
odap-vrf-ISP3-b.com /
Per-Group/
Per-User/

--> add odap-vrf-ISP2-a.com

Added odap-vrf-ISP2-a.com

--> cd odap-vrf-ISP2-a.com

[ //localhost/Radius/ResourceManagers/odap-vrf-ISP2-a.com ]
  Name = odap-vrf-ISP2.com
  Description =
  Type =

--> set type subnet-dynamic

Set Type subnet-dynamic

--> ls

[ //localhost/Radius/ResourceManagers/odap-vrf-ISP2-a.com ]
  Name = odap-vrf-ISP2-a.com
  Description =
  Type = subnet-dynamic
  NetMask =
  SubnetAddresses/

-> set netmask 255.255.0.0

Set NetMask 255.255.0.0

-> cd subnetaddresses

[ //localhost/Radius/ResourceManagers/odap-vrf-ISP2-a.com /SubnetAddresses ]
  Entries 0 to 0 from 0 total entries
  Current filter: <all>

--> add 10.0.0.0-10.10.255.255

Added 10.0.0.0-10.255.255.255

-> cd /radius/resourcemanagers

[ //localhost/Radius/ResourceManagers ]
  Entries 1 to 10 from 10 total entries
  Current filter: <all>

IPA-Pool/
IPA-Pool-2/
IPX-Pool/
odap-vrf-ISP1.com/

```

```

odap-vrf-ISP3-a.com /
odap-vrf-ISP3-b.com /
odap-vrf-ISP2-a.com /
Per-Group/
Per-User/

--> add odap-vrf-ISP2-b.com

Added odap-vrf-ISP2-b.com

--> cd odap-vrf-ISP2-b.com

[ //localhost/Radius/ResourceManagers/odap-vrf-ISP2-b.com ]
  Name = odap-vrf-ISP2-b.com
  Description =
  Type =

--> set type subnet-dynamic

Set Type subnet-dynamic

--> ls

[ //localhost/Radius/ResourceManagers/odap-vrf-ISP2-b.com ]
  Name = odap-vrf-ISP2-b.com
  Description =
  Type = subnet-dynamic
  NetMask =
  SubnetAddresses/

-> set netmask 255.255.255.0

Set NetMask 255.255.255.0

-> cd subnetaddresses

[ //localhost/Radius/ResourceManagers/odap-vrf-ISP2-b.com /SubnetAddresses ]
  Entries 0 to 0 from 0 total entries
  Current filter: <all>

--> add 10.255.0.0-10.255.10.255

Added 10.255.0.0-10.255.10.255

```

## Configuring Session Managers



### Note

It is not necessary to configure Session Managers in two instances. All SessionManager configuration can be done at one time before configuring the Resource Managers.

### Step 8

Configure the Session Managers to be referenced by the Resource Managers.

```
--> cd/radius/sessionmanagers
```

```

[ //localhost/Radius/SessionManagers ]
  Entries 1 to 4 from 4 total entries
  Current filter: <all>

  odap-vrf-ISP1.com/
  odap-vrf-ISP2.com/
  odap-vrf-ISP3.com/
  session-mgr-1/

--> cd odap-vrf-ISP2.com

[ //localhost/Radius/SessionManagers/odap-vrf-ISP2.com ]
  Name = odap-vrf-ISP2.com
  Description =
  AllowAccountingStartToCreateSession = FALSE
  ResourceManagers/

--> cd resourcemanagers

--> set 1 odap-vrf-ISP2-a.com

Set 1 odap-vrf-ISP2-a.com

--> set 2 odap-vrf-ISP2-b.com

Set 2 odap-vrf-ISP2-b.com

--> cd/radius/sessionmanagers

[ //localhost/Radius/SessionManagers ]
  Entries 1 to 4 from 4 total entries
  Current filter: <all>

  odap-vrf-ISP1.com/
  odap-vrf-ISP2.com/
  odap-vrf-ISP3.com /
  session-mgr-1/

--> cd odap-vrf-ISP3.com

[ //localhost/Radius/SessionManagers/odap-vrf-ISP3.com ]
  Name = odap-vrf-ISP3.com
  Description =
  AllowAccountingStartToCreateSession = FALSE
  ResourceManagers/

--> cd resourcemanagers

--> set 1 odap-vrf-ISP3-a.com

Set 1 odap-vrf-ISP3-a.com

--> set 2 odap-vrf-ISP3-b.com

Set 2 odap-vrf-ISP3-b.com

--> cd/radius/sessionmanagers

```

```

[ //localhost/Radius/SessionManagers ]
  Entries 1 to 4 from 4 total entries
  Current filter: <all>

  odap-vrf-ISP1.com/
  odap-vrf-ISP2.com/
  odap-vrf-ISP3.com/
  session-mgr-1/

--> cd odap-vrf-ISP1.com

[ //localhost/Radius/SessionManagers/odap-vrf-ISP1.com ]
  Name = odap-vrf-ISP1.com
  Description =
  AllowAccountingStartToCreateSession = FALSE
  ResourceManagers/

--> cd resourcemangers

--> set 1 odap-vrf-ISP1.com

Set 1 odap-vrf-ISP1.com

```

## Configure Clients

- Step 9** For any client that might forward ODAP requests to the Cisco AR server, set the Vendor property to CiscoWithODAP.

```

--> cd /radius/clients

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

  localhost/
  vhg-1/
  vhg-2/

--> cd vhg-1

[ //localhost/Radius/Clients/vhg-1 ]
  Name = vhg-1
  Description =
  IPAddress = 209.165.200.225
  SharedSecret = secret
  Type = NAS
  Vendor =
  IncomingScript~ =
  OutgoingScript~ =
  UseDNIS = FALSE
  DeviceName = a_name
  DevicePassword = password

--> set vendor CiscoWithODAP

Set Vendor CiscoWithODAP

```

## Save Your Configuration

**Step 10** After completing the configuration, save your changes.

```
--> save
```

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

---



## Using Cisco Access Registrar Server Features

---

This chapter provides information about how to use the following Cisco AR server features:

- “Command Completion”
- “Service Grouping Feature” section on page 2
- “SHA-1 Support for LDAP-Based Authentication” section on page 9
- “LEAP Support” section on page 11
- “Dynamic Attributes” section on page 9-12
- “Tunneling Support Feature” section on page 9-14
- “xDSL VPI/VCI Support for Cisco 6400” section on page 9-16
- “Apply Profile in Cisco AR Database to Directory Users” section on page 9-17
- “Directory Multi-Value Attributes Support” section on page 9-19
- “MultiLink-PPP (ML-PPP)” section on page 9-19
- “Dynamic Updates Feature” section on page 9-20
- “NAS Monitor” section on page 9-22
- “Automatic Customer Information Collection” section on page 9-22
- “Simultaneous Terminals for Remote Demonstration” section on page 9-22
- “Support for RADIUS Check Item Attributes” section on page 9-22
- “User-Specific Attributes” section on page 9-24

### Command Completion

Cisco Access Registrar’s command completion feature provides online help by listing possible entries to the current command line when you press the Tab key after entering a partial command. The Cisco AR 3.0 server responds based on:

- The location of the cursor including the current directory
- Any data you have entered on the command line prior to pressing the Tab key

The command completion feature emulates the behavior of Cisco IOS and Kermit. When you press the Tab key after entering part of a command, the Cisco AR 3.0 server provides any identifiable object and property names. For example, after you first issue **aregcmd** and log in to Cisco AR 3.0, enter the following:

**cd <Tab>**

```
Administrators/ Radius/
```

Pressing the Tab key consecutively displays possible context-sensitive choices.

In the following example, after changing directory to **/Radius/services/local-file** an administrator wants to see the possible types of authentication services that can set.

**cd /Radius/services/local-file**

```
//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 =
```

**set type <Tab>**

```
eap-leap      file          local          radius-session
eap-md5       group         odbc           rex
eap-sim      ldap          radius         tacacs-udp
```

Values can also be tab-completed. For example, if you decide to set the local-file service's type to file, you can do the following:

**set type f<Tab>**

and the command line completes to:

**set type file**

## Service Grouping Feature

The Service Grouping feature enables you to specify multiple services (called *subservices*) to be used with authentication, authorization, or accounting requests. The general purpose is to enable multiple Remote Servers to process requests.

Perhaps the most common use of this feature will be to send accounting requests to multiple Remote Servers thus creating multiple accounting logs. Another common use might be to authenticate from more than one Remote Server where, perhaps the first attempt is rejected, other Remote Servers may be attempted and an Access-Accept obtained.

Clearly, in the accounting request example, each request must be successfully processed by each subservice in order for the originator of the accounting request to receive a response. This is known as a **logical AND** of each of the subservice results. In the authenticate example, the first subservice which responds with an accept is returned to the client or if all subservices respond with **reject**, then a reject is returned to the client. This is known as a **logical OR** of each of the subservice results.

A Service is specified as a Group Service by setting its type to *group*, specifying the ResultRule (AND or OR) and specifying one or more subservices in the GroupServices subdirectory. The subservices are called in numbered order and as such are in an indexed list similar to Remote Server specification in a radius Service. Incoming and outgoing scripts for the Group Service may be optionally specified.

A subservice is any configured non-Group Service. When a Group Service is used, each subservice is called in exactly the same manner as when used alone (such as if specified as the DefaultAuthenticationService). Incoming and Outgoing scripts are executed if configured and Outage Policies are honored.

## Configuration Example - AccountingGroupService

The following example shows how to configure an accounting Group Service to deliver accounting requests to multiple Remote Servers.

**Step 1** The first task is to setup the subservices which are to be part of the AccountingGroupService. Since subservices are merely configured Services which have been included in a service group, you need only define two new Services.

For this example, we will define two new radius Services called *OurAccountingService* and *TheirAccountingService*. A provider might want to maintain duplicate accounting logs in parallel with their bulk customer's accounting logs.

**Step 2** Change directory to `/radius/services`. At the command line, enter the following:

```
--> cd /radius/services
[ //localhost/Radius/Services ]
  Entries 1 to 2 from 2 total entries
  Current filter: <all>
  local-file/
  local-users/
```

**Step 3** At the command line, enter the following:

```
--> add OurAccountingService
--> add TheirAccountingService
```

The configuration of these Services is very similar to stand-alone Radius accounting service. Step-by-step configuration instructions are not provided, but the complete configuration is shown below:

```
[ //localhost/Radius/Services/OurAccountingService ]
  Name = OurAccountingService
  Description =
  Type = radius
  IncomingScript = OurAccountingInScript
  OutgoingScript = OurAccountingOutScript
  OutagePolicy = RejectAll
  OutageScript =
  MultipleServersPolicy = Failover
  RemoteServers/
    1. OurPrimaryServer
```

```

2. OurSecondaryServer

[ //localhost/Radius/Services/TheirAccountingService ]
Name = TheirAccountingService
Description =
Type = radius
IncomingScript = TheirAccountingInScript
OutgoingScript = TheirAccountingOutScript
OutagePolicy = RejectAll
OutageScript =
MultipleServersPolicy = Failover
RemoteServers/
1. TheirPrimaryServer
2. TheirSecondaryServer

```

The next step is to create the new **AccountingGroupService**. The purpose of this Service is to process Accounting requests through both OurAccountingService and TheirAccountingService.

**Step 4** At the command line, enter the following:

```

--> add AccountingGroupService
Added AccountingGroupService

--> cd AccountingGroupService

[ //localhost/Radius/Services/AccountingGroupService ]
Name = AccountingGroupService
Description =
Type =
IncomingScript =
OutgoingScript =

--> set type group
Set Type group

```

**Step 5** Set the ResultRule to **AND** to ensure that both services process the accounting request successfully.

```

--> set ResultRule AND
Set ResultRule AND

--> ls

[ //localhost/Radius/Services/AccountingGroupService ]
Name = AccountingGroupService
Description =
Type = group

```

```

IncomingScript =
OutgoingScript =
ResultRule = AND
GroupServices/

```

```
--> set IncomingScript AcctGroupSvcInScript
```

```
--> set OutgoingScript AcctGroupSvcOutScript
```

Now we must add the Services we created OurAccountingService and TheirAccountingService as subservices of the Group Service.

**Step 6** At the command line, enter the following:

```
--> cd GroupServices
```

```
[ //localhost/Radius/Services/AccountingGroupService/GroupServices ]
```

```
--> set 1 OurAccountingService
```

```
Set 1 OurAccountingService
```

```
--> Set 2 TheirAccountingService
```

```
Set 2 TheirAccountingService
```

```
--> ls
```

```
[ //localhost/Radius/Services/AccountingGroupService ]
```

```

Name = AccountingGroupService
Description =
Type = group
IncomingScript = AcctGroupSvcInScript
OutgoingScript = AcctGroupSvcOutScript
ResultRule = AND
GroupServices/
  1. OurAccountingService
  2. TheirAccountingService

```

---

This completes the setup of the AccountingGroupService. To use this Service simply set it as the DefaultAccountingService and/or configure a policy/rule set which will select this Service. Essentially, this may be used in the same manner as any other stand-alone service.

## Summary of Events

The following describes the flow of what happens when a client sends an accounting request which is processed by the AccountingGroupService:

1. ActGroupSvcInScript is executed.
2. OurAccountingService is called.

3. OurAccountingService's Incoming Script, OurAccountingInScript is called.
4. The request is sent to the Remote Server OurPrimaryServer and/or OurSecondaryServer, if necessary.
5. If a response is not received, because we used the **AND** ResultRule, the request failed and no response is sent to the client and the request is dropped. If a response is received, then the process continues.
6. OurAccountingService's Outgoing Script, OurAccountingOutScript is called.
7. TheirAccountingService is called.
8. TheirAccountingService's Incoming Script, TheirAccountingInScript is called.
9. The request is sent to the Remote Server TheirPrimaryServer and/or TheirSecondaryServer, if necessary.
10. If a response is not received, because we used the **AND** ResultRule, the request failed and no response is sent to the client and the request is dropped. If a response is received, then the process continues.
11. TheirAccountingService's Outgoing Script, TheirAccountingOutScript is called.
12. AcctGroupSvcOutScript is executed.
13. Standard processing continues.

## Configuration Example 2 - AuthenticationGroupService

In this example, we will configure a Group Service for the purposes of providing alternate Remote Servers for a single authentication. Simply put, if Service A rejects the request, try Service B.

**Step 1** The first task is to setup the subservices which are to be part of the AuthenticationGroupService. Since subservices are merely configured Services which have been included in a service group, we will simply define two new Services. For simplicity, we will define two new radius Services called AuthenticationServiceA and AuthenticationServiceB.

**Step 2** At the command line, enter the following:

```
--> cd /radius/services
[ //localhost/Radius/Services ]
  Entries 1 to 2 from 2 total entries
  Current filter: <all>
  local-file/
  local-users/

--> add AuthenticationServiceA
--> add AuthenticationServiceB
```

**Step 3** The configuration of these Services is very similar to stand-alone Radius authentication service. Step-by-step configuration instructions are not provided, but the complete configuration is shown below:

```
[ //localhost/Radius/Services/AuthenticationServiceA ]
  Name = AuthentictionServiceA
  Description =
```

```

Type = radius
IncomingScript = AuthAInScript
OutgoingScript = AuthAOutScript
OutagePolicy = RejectAll
OutageScript = AuthAOutageScript
MultipleServersPolicy = Failover
RemoteServers/
  1. PrimaryServerA
  2. SecondaryServerA

[ //localhost/Radius/Services/AuthenticationServiceB ]
Name = AuthenticationServiceB
Description =
Type = radius
IncomingScript = AuthBInScript
OutgoingScript = AuthBOutScript
OutagePolicy = RejectAll
OutageScript = AuthBOutageScript
MultipleServersPolicy = Failover
RemoteServers/
  1. PrimaryServerB
  2. SecondaryServerB

```

The next step is to create the new "AuthenticationGroupService". The purpose of this Service is to process authentication requests through both AuthenticationServiceA and AuthenticationServiceB if AuthenticationServiceA rejects the request.

**Step 4** At the command line, enter the following:

```
--> add AuthenticationGroupService
Added AuthenticationGroupService
```

```
--> cd AuthenticationGroupService
```

```
[ //localhost/Radius/Services/AuthenticationGroupService ]
Name = AuthenticationGroupService
Description =
Type =
IncomingScript =
OutgoingScript =

```

```
--> set type group
Set Type group
```

Next set the ResultRule to **OR** because we want to ensure that if the first subservice rejects the request, we then try the second subservice. If the second subservice rejects the request, then the response to the client is a reject.

**Step 5** At the command line, enter the following:

```
--> set ResultRule OR
      Set ResultRule OR

--> Set IncomingScript AuthGroupSvcInScript
      Set OutgoingScript AuthGroupSvcOutScript

--> Set IncomingScript AuthGroupSvcInScript
      Set OutgoingScript AuthGroupSvcOutScript

--> ls
      [ //localhost/Radius/Services/AuthenticationGroupService ]
      Name = AuthenticationGroupService
      Description =
      Type = group
      IncomingScript = AuthGroupSvcInScript
      OutgoingScript = AuthGroupSvcOutScript
      ResultRule = OR
      GroupServices/
```

Now we must add the services we created "AuthenticationServiceA" and "AuthenticationServiceB" as subservices of the Group Service.

**Step 6** At the command line, enter the following:

```
--> cd GroupServices
      [ //localhost/Radius/Services/AuthenticationGroupService/GroupServices ]

--> set 1 AuthenticationServiceA
      Set 1 AuthenticationServiceA

--> Set 2 AuthenticationServiceB
      Set 2 AuthenticationServiceB

--> ls
      [ //localhost/Radius/Services/AuthenticationGroupService ]
      Name = AuthenticationGroupService
      Description =
      Type = group
      IncomingScript = AuthGroupSvcInScript
      OutgoingScript = AuthGroupSvcOutScript
      ResultRule = OR
      GroupServices/
```

1. AuthenticationServiceA
  2. AuthenticationServiceB
- 

This completes the setup of the AuthenticationGroupService. To use this Service simply set it as the DefaultAuthenticationService and/or configure a policy/rule set which will select this Service. Essentially, this may be used in the same manner as any other stand-alone Service.

## Summary of Events

The following describes the flow of what happens when a client sends an Authentication request which is processed by the AuthenticationGroupService:

1. AuthGroupSvcInScript is executed.
2. AuthenticationServiceA is called.
3. AuthenticationServiceA's Incoming Script, AuthAInScript is called.
4. If the response is a reject or the request is dropped (due to an Outage Policy):
  - a. AuthenticationServiceA's Outgoing Script, AuthAOutScript is called.
  - b. Processing continues with the next service.
5. If the response is an Accept:
  - a. AuthenticationServiceA's Outgoing Script, AuthAOutScript is called.
  - b. Skip to step 9.
6. AuthenticationServiceB is called.
7. AuthenticationServiceB's Incoming Script, AuthBInScript is called.
8. Since this is the last subservice in our Group Service:
  - a. AuthenticationServiceB's Outgoing Script, AuthBOutScript is called.
  - b. Regardless of whether the request is Accepted or Rejected, processing will continue at step 9.
9. AuthGroupSvcOutScript is executed.
10. Standard processing continues.

## SHA-1 Support for LDAP-Based Authentication

The Cisco Access Registrar server supports secure hash algorithm (SHA-1) for LDAP-based authentication. This feature enables the Cisco AR server to authenticate users whose passwords are stored in LDAP servers and hashed using the SHA-1 encoding scheme.

SHA-1 support actually adds functionality for the following three features to Cisco Access Registrar:

- Authentication of PAP access requests against an LDAP user entry that uses the SHA-algorithm to the hash password attribute
- Authentication of PAP access requests against an LDAP user entry that uses the SSHA algorithm to hash the password attribute
- Configuration of the Cisco AR server to dynamically determine how password attributes retrieved from LDAP are encrypted and process them accordingly

This enhancement is 100% backwards compatible. All previously supported values for the PasswordEncryptionStyle property are still supported and still provide the same behavior. The only noticeable change is that **dynamic** is now the default value for the PasswordEncryptionStyle property.

## Remote LDAP Server Password Encryption

Prior to Cisco Access Registrar 1.7, the PasswordEncryptionStyle property on a Remote LDAP Server was limited to two values, none and crypt. SHA-1 supports adds three additional values for the PasswordEncryptionStyle property. Table 9-1 lists the valid values for this property and describes the corresponding behavior.

**Table 9-1 Remote LDAP Server Password Encryption Style Values**

PasswordEncryptionStyle	Access Registrar Behavior
none	All passwords retrieved from this LDAP server are assumed to be returned to Cisco AR as clear text. (There is no change in this functionality.)
crypt	All passwords retrieved from this LDAP server are assumed to be returned to Cisco AR as passwords encrypted using the UNIX <i>crypt</i> algorithm. (There is no change in this functionality.)  Passwords may be preceded by the {crypt} prefix which is stripped before comparing passwords.
SHA-1	All passwords retrieved from this LDAP server are assumed to be returned to Cisco AR as a Base64-encoded version of the user's password after it has been hashes using the SHA-1 mechanism (as defined by Netscape).  Passwords may be preceded by the {sha} prefix, which is stripped before comparing passwords.
SSHA-1	All passwords retrieved from this LDAP server are assumed to be encrypted/hashed using the SSHA mechanism (as defined by Netscape). Passwords may be preceded by the {ssha} prefix which is stripped before comparing passwords.  <b>Note</b> This is a Netscape/iPlanet-specific mechanism.
dynamic	The value instructs Cisco Access Registrar to choose the encryption mechanism on a case-by-case basis after it determines the presence of a known prefix, which the LDAP server prepends to the value of the password attribute.  For example, if the following was returned from an LDAP server as a password attribute: {SHA}qZk+NkcGgWq6PiVxeFDCbJzQ2J0=, the password would be processed using the SHA-1 mechanism. This value will be the new default for the PasswordEncryptionStyle property.

## Dynamic Password Encryption

When using the dynamic setting for the PasswordEncryptionStyle property on a Remote LDAP Server, the Cisco Access Registrar server looks for the prefixes listed in Table 9-2 to determine if encryption or a hash algorithm should be used during password comparison.



### Note

Password prefixes are not case sensitive.

**Table 9-2 Remote LDAP Server Password Prefix Values**

Password Prefix	Encryption/Hash Algorithm Used
none	None; when no known prefix is found, the password attribute is assumed to be in clear text.
{crypt}	UNIX crypt algorithm
{sha}	Secure Hash Algorithm, version 1 (SHA-1)
{ssha}	SSHA-1, as defined by Netscape.

In Cisco AR 1.7, the default value for the PasswordEncryptionStyle property on a Remote LDAP Server is **dynamic**.

**Note**

Using the *dynamic* setting for the PasswordEncryptionStyle property will require a bit more processing for each password comparison. When using dynamic, the Cisco AR server must examine each password for a known prefix. This should have no visible impact on performance.

## Logs

Turn on **trace** to level 4 to indicate (via the trace log) which password comparison method is being used.

## LEAP Support

The Cisco Access Registrar server supports the new AAA Cisco-proprietary protocol called Light Extensible Authentication Protocol (LEAP). LEAP was defined and implemented by Cisco's Aironet product family.

**Note**

Cisco Access Registrar supports a subset of EAP to support LEAP. This is not a general implementation of EAP for Cisco Access Registrar.

## LEAP Features

Cisco Aironet wireless LAN products defined and implemented this new authentication protocol to support mutual authentication between the client and AAA server and to support the dynamic generation of WEP keys. LEAP provides the following features:

- Authentication of the wireless client to the AAA server
- Authentication of the AAA server to the client (to allow the client to verify that it is connected to an authorized network)
- Enough information to allow the client and the AAA server to independently generate the WEP key that the client and access point will use to encrypt all of the traffic between them for this session.

Without LEAP, the client and access point use a shared secret common to all of the clients of that access point. Because the shared secret is common, it is expensive to change frequently. This makes it easier for an outside party to gather enough data to figure out what the shared secret is, thus gaining access to the network.

With LEAP, each session for each client has a unique shared secret. This makes it much more difficult for an outside party to penetrate network security.

## Required Attributes

Cisco Access Registrar recognizes access requests that contain an EAP-Message attribute and a LEAP-specific EAP-Message. The LEAP-specific attribute is used to send the Challenge request or receive the Challenge response. Other EAP-Messages are not supported.



### Note

---

A Message-Authenticator attribute must be present in any Access-Request, Access-Accept, Access-Reject or Access-Challenge that contains an EAP-Message.

---

## User Interface

Cisco Access Registrar supports LEAP without change to the user interface.

## Dynamic Attributes

Cisco Access Registrar 1.6 supports dynamic values for the configuration object properties listed below. Previous releases of Cisco Access Registrar only handles static values for all the object properties.

Dynamic attributes are similar to UNIX shell variables. With dynamic attributes, the value is evaluated at run time. All of the objects that support dynamic attributes will have validation turned off in **aregcmd**.

## Object Properties with Dynamic Support

The following object properties support dynamic values:

### Radius

DefaultAuthenticationService

DefaultAuthorizationService

DefaultAccountingService

DefaultSessionManager

IncomingScript

OutgoingScript



---

**Note** Do not use the following environment variables:  
Accounting-Service for the **/Radius/DefaultAccountingService**,  
Authentication-Service for the **/Radius/DefaultAuthenticationService**, or  
Authorization-Service for the **/Radius/DefaultAuthorizationService**  
User-Profile for the **BaseProfile**, User-Group for the **Group**, User-Authorization  
for the **AuthorizationScript**, Session-Manager for the **DefaultSessionManager**,  
or Session-Service for the **DefaultSessionService**.

---

#### **/Radius/Clients**

client1/  
    IncomingScript  
    OutgoingScript

#### **/Radius/Userlist/Default**

user1/  
    Group  
    BaseProfile  
    AuthenticationScript  
    AuthorizationScript

#### **/Radius/UserGroup**

Group1/  
    BaseProfile  
    AuthenticationScript  
    AuthorizationScript

#### **/Radius/Vendor**

Vendor1/  
    IncomingScript  
    OutgoingScript

#### **/Radius/Service**

Service1/  
    IncomingScript  
    OutgoingScript  
    OutageScript  
    OutagePolicy

#### **/Radius/RemoteServers**

remoteserver1/  
    IncomingScript  
    OutgoingScript  
Remoteldapservers1/  
    Searchpath

Filter

**Note**

To differentiate the properties that support dynamic attributes, we place a tilde (~) after each property, as in IncomingScript~. However, when the Cisco AR administrator is required to set values for those properties, continue to use the original property name, such as set IncomingScript \${elrealm}{Test}. The tilde is only for visual effect, and including the tilde will generate an error (“310 command Failed.”)

## Dynamic Attribute Format

The format of the dynamic attribute is:

```
#{eqplattribute-name}{default-name}
```

where **e** stands for environment dictionary, **q** stands for request dictionary and **p** stands for response dictionary. You can use e, q and p in any order. The attribute name is the name for the attribute from environment dictionary, request dictionary, or response dictionary.

For example,

```
/Radius
DefaultAuthenticationService = #{eq|realm}{local-users}
```

The default Authentication Service is determined at run time. Cisco Access Registrar first checks to see if there is one value of **realm** in the environment dictionary. If there is, it becomes the value of DefaultAuthenticationService. If there is not, check the value of realm in the request dictionary. If there is one value, it becomes the value of DefaultAuthenticationService. Otherwise, local-users is the DefaultAuthenticationService. If we don't set local-users as the default value, the DefaultAuthenticationService is *null*. The same concept applies to all other attribute properties.

The validation for the dynamic values of the object property will only validate the default value. In the above example, Cisco Access Registrar will do validation to check whether local-users is one of services defined in the service subdirectory.

**Note**

When setting specific property values, do not use the tilde (~) in the property name. Doing so generates a *310 Command Failed* error.

## Tunneling Support Feature

Tunneling support is strictly based upon the IETF RFC: “RADIUS Attributes for Tunnel Protocol Support” (<http://www.ietf.org/rfc/rfc2868.txt>).

Table 9-3 lists the tunneling attributes supported in this Cisco Access Registrar release.

**Table 9-3 Tunneling Attributes Supported by Cisco Access Registrar**

Attribute Number	Attribute
64	Tunnel-Type
65	Tunnel-Medium-Type
66	Tunnel-Client-Endpoint

**Table 9-3 Tunneling Attributes Supported by Cisco Access Registrar (continued)**

Attribute Number	Attribute
67	Tunnel-Server-Endpoint
69	Tunnel-Password
81	Tunnel-Private-Group-ID
82	Tunnel-Assignment-ID
83	Tunnel-Preference
90	Tunnel-Client-Auth-ID
91	Tunnel-Server-Auth-ID

The tunneling attribute has the following format:

(1 byte)	(1 byte)	(1 byte)	(variable number of bytes)
Type	Length	Tag	Value

## Configuration

1. Configure the tag attributes as untagged attributes under the **/Radius/Advanced/Attribute Dictionary** directory (for example, **Tunnel-Type**).
2. Attach the “\_tag” tag to these attributes when configuring the attributes under all of the other directories as tagged attributes (for example, **Tunnel-Type\_tag10** under the **/Radius/Profiles/test** directory). Without the tag number, the default value is (**\_tag = \_tag0**).

## Example

```

/Radius/Advanced/Attribute Dictionary
  /Tunnel-Client-ID
    Name = Tunnel-Client-Endpoint
    Description =
    Attribute = 66
    Type = STRING
    Min = 0
    Max = 253

/Radius/Profiles/test
  Name = test
  Description =
  /Attributes
    Tunnel-Client-Endpoint_tag3 = "129.56.123.1"

```

## Notes

1. “\_tag” is reserved for the tunneling attributes. No other attributes should include this suffix.
2. The tag number value can range from 0 through 31.

## Validation

The Cisco Access Registrar server checks whether the tag attributes are defined under the **/Radius/Advanced/Attribute Dictionary** directory. The server also checks whether the tag number falls within the range (0-31).

## xDSL VPI/VCI Support for Cisco 6400

To provide this support, a distinction must be made between device authentication packets and regular user authentication packets.

## Using User-Name/User-Password for Each Cisco 6400 Device

This approach assumes that for every 6400 NAS, a device-name/device-password is created for each. Following are the required changes:

For each NAS in Cisco Access Registrar:

```
Name = test6400-1
Description =
IPAddress = 209.165.200.224
SharedSecret = secret
Type = NAS
Vendor =
IncomingScript =
OutgoingScript =
Device-Name = theDevice
Device-Password = thePassword
```

When the 6400 sends out the device authentication packet, it may have different **User-Name/User-Password** attributes for each 6400 NAS. When Cisco Access Registrar receives the packet, it tries to obtain the **Device-Name/Device-Password** attributes from the NAS entry in the Cisco Access Registrar configuration database. When the **User-Name/User-Password** in the packet match the configured **Device-Name/Device-Password** attribute values, Cisco Access Registrar assumes that it must get the device. The next step is to replace the **User-Name** attribute with the concatenated `<module>/<slot>/<port>` string. From this point, the packet is treated as a regular packet.



### Note

---

A user record with the name of the concatenated string must be created.

---

## Format of the New User-Name Attribute

After the device is identified, the **User-Name** attribute is replaced with the new value. This new value is the concatenation of 6400 `<module>/<slot>/<port>` information from the NAS-Port attribute and the packet is treated as a regular user authentication from this point on.



### Note

---

This format only supports NAS Port Format D. Refer to Cisco IOS documentation for more information about NAS port formats.

---

The format of the new **User-Name** attribute is the **printf** of “%s-%d-%d-%d-%d” for the following values:

**NAS-IP**—in dot format of the **NAS-IP-Address** attribute. For example, 10.10.10.10.

**slot**—apply mask 0xF0000000 on **NAS-Port** attribute and shift right 28 bits. For example, **NAS-Port** is 0x10000000, the slot value is 1.

**module**—apply mask 0x08000000 on **NAS-Port** attribute and shift right 27 bits. For example, **NAS-Port** is 0x08000000, the module value is 1.

**port**—apply mask 0x07000000 on **NAS-Port** attribute and shift right 24 bits. For example, **NAS-Port** is 0x06000000, the port value is 6.

**VPI**—apply mask 0x00FF0000 on **NAS-Port** attribute and shift right 16 bits. For example, **NAS-Port** is 0x00110000, the VPI value is 3.

**VCI**—apply mask 0x0000FFFF on **NAS-Port** attribute. For example, **NAS-Port** is 0x00001001, the VCI value is 9.

## Apply Profile in Cisco AR Database to Directory Users

You can define the **User-Profile** and **User-Group** environment variables in the directory mapping and Cisco Access Registrar will apply the profiles defined in the Cisco Access Registrar database to each directory user having any of these two variables set.

### User-Profile

This attribute is of type string with the format:

<Value1>::<>Value2> ...

The **User-Profile** attribute is intended to hold a list of profile names. <Value1> and <Value2> represent the names of the profiles. They are separated by the “::” character, therefore, the “::” can not be part of the profile name. The order of values in the string has significance, as the profiles are evaluated from left to right. In this example, profile <Value2> is applied after profile <Value1>.

Assume the user record has a field called `UserProfile` that holds the name of the profile that applies to this user. This field is mapped to the environment attribute **User-Profile**. Following is how the mapping is done with **aregcmd**:

```
QuickExample/
  Name = QuickExample
  Description =
  Protocol = ldap
  IPAddress = 209.165.200.224
  Port = 389
  ReactivateTimerInterval = 300000
  Timeout = 15
  HostName = QuickExample.company.com
  BindName =
  BindPassword =
  UseSSL = FALSE
  SearchPath = "o=Ace Industry, c=US"
  Filter = (uid=%s)
  UserPasswordAttribute = password
  LimitOutstandingRequests = FALSE
  MaxOutstandingRequests = 0
  MaxReferrals = 0
  ReferralAttribute =
  ReferralFilter =
  PasswordEncryptionStyle = None
  LDAPToEnvironmentMappings/
    UserProfile = User-Profile
  LDAPToRadiusMappings/
```

After Cisco Access Registrar authenticates the user, it checks whether **User-Profile** exists in the environment dictionary. If it finds **User-Profile**, for each value in **User-Profile**, Cisco Access Registrar looks up the profile object defined in the configuration database and adds all of the attributes in the profile object to the response dictionary. If any attribute is included in more than one profile, the newly applied profile overrides the attribute in the previous profile.

## User-Group

You can use the **User-Group** environment variable to apply the user profile as well. In Cisco Access Registrar, a user can belong to a user group, and that user group can have a pointer to a user profile. When Cisco Access Registrar finds that a packet has **User-Group** set, it obtains the value of the **User-Profile** within the user group, and if the **User-Profile** exists, it applies the attributes defined in the user profile to that user.

Note that in Cisco Access Registrar, every user can also directly have a pointer to a user profile. Cisco Access Registrar applies profiles in the following order:

1. If the user profile defined in the user group exists, apply it.
2. If the user profile defined in the user record exists, apply it.

The profile in **User-Group** is more generic than in **User-Profile**. Therefore, Cisco Access Registrar applies the profile from generic to more specific.

## Example User-Profile and User-Group Attributes in Directory User Record

You can use an existing user attribute in the user record to store profile info. When this is a new attribute, we suggest you create a new auxiliary class **AR\_UserRecord** for whichever user class is used.

**AR\_User\_Profile** and **AR\_User\_Group** are two optional members in this class. They are of type string. The mapping is as follows:

```
LDAPToEnvironmentMappings/
  AR_User_Profile = User-Profile
  AR_User_Group = User-Group
```

## Directory Multi-Value Attributes Support

If any attributes mapped from the LDAP directory to the Cisco Access Registrar response dictionary are multi-valued, the attributes are mapped to multiple RADIUS attributes in the packet.

## MultiLink-PPP (ML-PPP)

Cisco Access Registrar supports MultiLink-PPP (ML-PPP). ML-PPP is an IETF standard, specified by RFC 1717. It describes a Layer 2 software implementation that opens multiple, simultaneous channels between systems, providing additional bandwidth-on-demand, for additional cost. The ML-PPP standard describes how to split, recombine, and sequence datagrams across multiple B channels to create a single logical connection. The multiple channels are the ports being used by the Network Access Server (NAS).

During the AA process, Cisco Access Registrar authenticates the user connection for each of its channels, even though they belong to the same logical connection. The Authentication process treats the multilink connection as if it is multiple, single link connections. For each connection, Cisco Access Registrar creates a session dedicated for management purposes. The session stays active until you logout, which subsequently frees up all of the ports in the NAS assigned to each individual session, or until the traffic is lower than a certain threshold so that the secondary B channels are destroyed thereafter. Cisco Access Registrar has the responsibility of maintaining the active session list and discards any session that is no longer valid in the system, by using the accounting stop packet issued from NAS. The multiple sessions that were established for a single logical connection must be destroyed upon the user logging out.

In addition, the accounting information that was gathered for the sessions must be aggregated for the corresponding logical connection by the accounting software. Cisco Access Registrar is only responsible for logging the accounting start and accounting stop times for each session. As those sessions belong to the same bundle, IETF provides two standard RADIUS attributes to identify the related multilink sessions. The attributes are **Acct-Multi-Session-Id** (attribute **50**) and **Acct-Link-Count** (attribute **51**), where **Acct-Multi-Session-Id** is a unique Accounting identifier used to link multiple related sessions in a log file, and **Acct-Link-Count** provides the number of links known to have existed in a given multilink session at the time the Accounting record was generated. The Accounting software is responsible for calculating the amount of the secondary B channel's connection time.

The secondary B channel can go up and down frequently, based upon traffic. The Ascend NAS supports the **Target-Util** attribute, which sets up the threshold for the secondary channel. When the traffic is above that threshold the secondary channel is up, and when the traffic is below that threshold, the secondary B channel is brought down by issuing an Accounting stop packet to Cisco Access Registrar. On the other hand, if you bring down the primary channel (that is, log out), the secondary B channel is also destroyed by issuing another Accounting stop packet to Cisco Access Registrar.

Table 9-4 lists ML-PPP related attributes.

**Table 9-4 ML-PPP Attributes**

Number	Attribute	Cisco NAS (IOS 11.3 Release)	Ascend NAS
44	Acct-Session-Id	Supported	Supported
50	Acct-Multi-Session-Id	Supported	Supported
51	Acct-Link-Count	Supported	Supported
62	Port-Limit	Supported	Supported
234	Target-Util	Not Supported	Supported
235	Maximum-Channels	Supported	Supported

Following are sample configurations for ML-PPP:

```

/Radius
  /Profile
    /Default-ISDN-Users
      Name = Default-ISDN-Users
      Description =
      Attributes/
        Port-Limit = 2
        Target-Util = 70
        Session-Timeout = 70

/Radius
  /UserGroups
    /ISDN-Users
      Name = ISDN-Users
      Description = " Users who always use ISDN"
      BaseProfile = Default-ISDN-Users
      Authentication-Script =
      Authorization-Script =

```

The **Port-Limit** attribute controls the number of concurrent sessions a user can have. The **Target-Util** attribute controls the threshold level at which the second B channel should be brought up.

## Dynamic Updates Feature

The Dynamic Updates feature enables changes to server configurations made using **aregcmd** to take effect in the Cisco Access Registrar server after issuing the **save** command, eliminating the need for a server **reload** after making changes.

Table 9-5 lists the Radius object and its child objects. For each object listed, the **Add** and **Modify or Delete** columns indicate whether a dynamic update occurs after adding, modifying, or deleting an object or attribute. Entries in the **Add** and **Modify or Delete** columns also apply to child objects and child attributes of the objects listed, unless the child object is explicitly listed below the object, such as **/Radius/Advanced/Ports** or **/Radius/Advanced/Interfaces**.

**Table 9-5 Dynamic Updates Effect on Radius Server Objects**

Object	Add	Modify or Delete
Radius	Yes	Yes
UserLists	Yes	Yes

**Table 9-5 Dynamic Updates Effect on Radius Server Objects (continued)**

Object	Add	Modify or Delete
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	No	No
Advanced	Yes	Yes
SNMP	No	No
Ports	No	No
Interfaces	No	No

The Dynamic Updates feature is subject to the following limitations:

- Changes to the Ports or Interfaces objects are not dynamically updated. An **aregcmd reload** command must be issued for these changes to be propagated to the Cisco Access Registrar server.
- Changes (modifications and deletions) to existing Session Manager and Resource Manager objects are not dynamically updated. An **aregcmd reload** command must be issued for these changes to be propagated to the Cisco Access Registrar server. However, additions of new Session Manager and Resource Manager objects are dynamically updated. Active sessions and allocated resources are preserved in this case.
- Changes to the Cisco Access Registrar configuration may not be immediately propagated to the server. Dynamic updates are only carried out in a *safe* environment (that is, when packets are not being processed and when packet processing can be delayed until the changes can be made on the server safely). Dynamic updates will yield to packet processing when appropriate, thus not significantly impacting server performance.
- Changes to SNMP require the Cisco Access Registrar server to be restarted (**/etc/init.d/arservagt restart**)

## NAS Monitor

The ability to monitor when a NAS is *down* (really only unreachable from AR) is provided by **nasmonitor**. This program will repeatedly query a TCP port at the specified IP address until the device (NAS) is reachable. If the NAS is not reachable after a period of time, a warning E-mail is sent; if the NAS is still not reachable after another period of time, a message is sent to Cisco AR to release all sessions associated with that NAS. The port to query, the query frequency, the first time interval, the backoff time interval, and the E-mail address to send to are all configurable (with defaults); the only required parameter is the NAS IP address. This program will work for any device that has a TCP port open; it can either be run by hand, when desired, or put in a **cron** job. See **nasmonitor -h** for details.



### Note

You must have **tcsh** installed in **/usr/local/bin** to use **nasmonitor**. **tcsh** is part of the standard Tcl installation that can be downloaded from <http://www.scripts.com>.

## Automatic Customer Information Collection

The program **arbug** can be used to collect information about the Cisco AR server from a customer. There are several levels of detail that can be specified, from level 1 being the least to level 3 being the most. The results are collected into a tarball that can be E-mailed or **ftped** to Cisco as requested. See **arbug -h** for details.

## Simultaneous Terminals for Remote Demonstration

Multiple people can view and interact in a single demonstration by using the **screen** program, a standard GNU release with a special configuration for use with Cisco AR. To run **screen**, a technical support specialist (CSE/DE) will **telnet** to your server and log in as **cisco**. While you run **/opt/AICar1/usrbin/screen** (assuming **/opt/AICar1** is the Cisco AR path) as **root**, the CSE/DE runs **/opt/AICar1/usrbin/screen -r root**. Now both people (or more) can see what the other types, as well as the results of the commands entered. The special Cisco AR configuration only allows **root** and **cisco** to run **screen**.

## Support for RADIUS Check Item Attributes

Cisco Access Registrar supports RADIUS check item attributes configuration at the user and group levels. You can configure the Cisco AR server to check for attributes that must be present or attributes that must not be present in the Access-Request packet for successful authentication.

When using check item attributes, the Cisco AR server will reject Access-Requests if:

- Any of the configured check item attributes are not present in the Access-Request packet
- Any of the Access-Request packet's check item attribute values do not match with those configured check item attribute values

For remote servers using either LDAP or ODBC, Cisco AR allows for mapping of certain LDAP or ODBC fields to check item attributes. The mapped attributes can be used as check item attributes while processing the Access-Request packets.

When you configure check item attributes at both the user and group levels, the Cisco AR 3.0 server first checks the attributes of the user level before those of the group level. The Cisco AR 3.0 server must first authenticate the user's password in the Access-Request before validating the check item attributes.

The Cisco AR 3.0 server logs details about any rejected Access-Requests as a result of check items processing.

## Configuring Check Items

You use **aregcmd** to configure check item attributes.

### Configuring User Check Items

The follow example shows how to configure UserList check item attributes.

- Step 1** Log in to the Cisco AR 3.0 server, and use **aregcmd** to navigate to **//localhost/Radius/UserLists/default/bob**.

```
[ //localhost/Radius/UserLists/Default/bob ]
Name = bob
Description =
Password = <encrypted>
AllowAnonymousPassword = FALSE
Enabled = TRUE
Group~ = PPP-users
BaseProfile~ =
AuthenticationScript~ =
AuthorizationScript~ =
UserDefined1 =
Attributes/
CheckItems/
```

- Step 2** Change directory to CheckItems.

**cd CheckItems**

```
[ //localhost/Radius/UserLists/Default/bob/CheckItems ]
```

- Step 3** Use **set** to add any attributes to be used as check items.

**set calling-Station-Id 4085551212**

**save**

### Configuring Usergroup Check Items

The follow example shows how to configure UserGroups check item attributes.

- Step 1** Log in to the Cisco AR 3.0 server, and use **aregcmd** to navigate to **//localhost/Radius/UserGroups/Default**.

**cd /Radius/UserGroups/Default**

```
[ //localhost/Radius/UserGroups/Default ]
  Name = Default
  Description = "Users who sometimes connect using PPP and sometimes connect "
  BaseProfile~ =
  AuthenticationScript~ =
  AuthorizationScript~ = AuthorizeService
  Attributes/
  CheckItems/
```

**Step 2** Change directory to CheckItems.

```
cd CheckItems
```

```
[ //localhost/Radius/UserGroups/Default/CheckItems ]
```

**Step 3** Use set to add any attributes to be used as check items.

```
set NAS-IP-Address 10.10.10.10
```

```
save
```

## User-Specific Attributes

The Cisco AR 3.0 server supports user-specific attributes which enables the Cisco AR server to return attributes on a per-user or per-group basis without having to use profiles.

The Cisco AR 3.0 server adds a new property called HiddenAttributes to the User and UserGroup object. The HiddenAttributes property contains a concatenation of all user-level reply attributes. The HiddenAttributes property is not displayed, nor can the value be set or unset using the command-line interface.

The order of application of attributes is as follows:

1. UserGroup Base Profile
2. UserGroup Attributes
3. User Base Profile
4. User Attributes

The value of the HiddenAttributes property is used dynamically to construct and populate a virtual *attributes* directory in the User object. All values from the Attributes directory will go into the HiddenAttributes property. This occurs transparently when the administrator issues a save command.



## Using the Policy Engine

---

The Cisco Access Registrar (AR) Rule Policy provides an interface to define and configure a policy and to apply the policy to the corresponding access-request packets. Authentication service is based on Realm, Dialed Number Information String (DNIS), and Calling Line ID (CLID).

This chapter contains the following sections:

- Policies and Rules, page 10-1
- Script and Attribute Requirements, page 10-2
- Validation, page 10-3
- Known Limitations, page 10-3
- Service Determination, page 10-3
- Prefix Feature, page 10-8

## Policies and Rules

The following is an example set of policies and rules:

```
/Policies
  /SelectPolicy
    Name = SelectPolicy
    Description =
    Grouping = CiscoRealmRule|CiscoCLIDRule

  /CiscoSelectPolicy
    Name = CiscoSelectPolicy
    Description =
    Grouping = CiscoGroupRule

  /CiscoDefaultPolicy
    ...

  /CiscoExecPolicy
    Name = CiscoExecPolicy
    Description =
    Grouping = CiscoExecTimeRule1&CiscoExecSecurityRule|CiscoExecTimeRule2
    ....

/Rules
  /CiscoRealmRule
    Name = CiscoRealmRule
```

```

Description =
Script = ExecRealmRule
/Attributes
    Realm = @cisco.com
    AuthenticationService = jen1-ultra
    Policy = CiscoSelectPolicy

/CiscoCLIDRule
Name = CiscoCLIDRule
Description =
Script = ExecCLIDRule
/Attributes
    ...

/CiscoGroupRule
Name = CiscoGroupRule
Description =
Script = ExecGroupRule
/Attributes
    Group = CiscoExec
    Policy = CiscoExecPolicy
    DefaultPolicy = CiscoDefaultPolicy

/CiscoExecTimeRule1
Name = CiscoExecTimeRule1
Description =
Script = ExecTimeRule
/Attributes
    TimeRange = "mon-fri,06:00-18:00";
    AcceptedProfiles = PPP-USER

/CiscoExecTimeRule2
Name = CiscoExecTimeRule2
Description =
Script = ExecTimeRule
/Attributes
    TimeRange = "sun,18:00-24:00;sat,18:00-24:00";
    AcceptedProfiles = TELNET-USER

/CiscoExecSecurityRule
Name = SecurityRule
Description =
Script = ExecSecurityRule
/Attributes
    TunnelEnforcement = TRUE
    AuthenticationProtocol = CHAP

```

## Script and Attribute Requirements

The following requirements exist:

- **/Radius/Policies/SelectPolicy** is the first policy Cisco Access Registrar applies.
- The characters “|” and “&” are reserved as logical operands in a **Grouping** definition; they cannot be included in a **/Radius/Rules** name.



**Note** The “&” characters has a higher precedence than the “|” in policy grouping expressions.

- A space is not permitted between the logical operands and the rules in a **Grouping** definition.
- The scripts included in the rules must be defined under the **/Radius/Scripts** directory.
- The attributes included in the rules must be defined under the **/Radius/Advanced/Attribute Dictionary** directory.
- The rules included in the policies must be defined under the **/Radius/Rules** directory.

## Validation

When policies are configured, the following validations are performed by Cisco Access Registrar:

1. A check is performed to ensure the scripts included in the rules are defined under the **/Radius/Scripts** directory.
2. A check is performed to ensure the attributes included in the rules are defined under the **/Radius/Advanced/Attribute Dictionary** directory.
3. A check is performed to ensure the rules included in the policies are defined under the **/Radius/Rule** directory.

## Known Limitations

The following limitations currently exist:

1. **Grouping** expressions are not checked for validity.
2. The use of parentheses to denote precedence is not supported in a **Grouping** definition.
3. A check is not performed to determine whether a policy that is included within another policy is defined under the **/Radius/Policies** directory.

## Service Determination

The Cisco Access Registrar server must determine whether to provide local or proxy service by performing Service Determination based on DNIS, Realm, CLID, or a Pattern Match of these parameters

Service determination is set through the Cisco Access Registrar Policy Engine. To make all scripts ready to run, all scripts must be configured and set up through the **aregcmd** command. The following scripts are used for service determination:

- ExecRealmRule
- ExecDNISRule
- ExecCLIDRule

These scripts extract the domain of the username or called-station-id or calling-station-id from the access request packet and compares it with the Realm, DNIS, or CLID set within the rule. Upon finding a match, the scripts will set the services (**Authentication-Service**, **Authorization-Service**, or **Accounting-Service**) into the Environment Dictionary.

In order to invoke this service enhancement, you must add rules and policies. Under the **/Radius/Rules** directory, you set the script that is going to be executed. Under the **/Radius/Policies** directory, you set the combination of rules.

Following is an example for adding a new Realm rule:

```
cd /Radius/Rules
Add RealmRule
cd RealmRule
Set Script ExecRealmRule
cd Attributes
Set Realm @cisco.com
Set Authentication-Service local-users
Set Authorization-Service local-users
```

where **Realm** is the domain filter for user name. If the user-name contains @cisco.com, the **ExecRealmRule** script sets **Authentication-Service** and **Authorization-Service** to local-users. The current release also supports the #cisco.com format.

Besides setting up the rules, you must also set up one or more policies. Policies can be any combination of rules using the *and* (&) and *or* (|) operators. Using the above example, a policy is setup as follows:

```
cd /Radius/Policies
Add SelectPolicy
cd SelectPolicy
Set Grouping RealmRule
```

## Inserting, Deleting and Substituting Attributes

This feature supports the RADIUS proxy with the ability to customize attribute filters so that RADIUS attribute value pairs can be inserted/deleted/substituted.

For example, when roaming a packet from ISP A to ISP B, some attribute value (AV) pairs may be substituted (such as IP address) as they may not be valid on B's network. Additionally, B may return some vendor-specific attributes (VSAs) that are not applicable to A's network. Therefore, A will substitute B's VSA value pairs for A's VSAs.

Two configuration points under the **/Radius** directory support this feature: **Translations** and **TranslationGroups**. 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.

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.

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 Cisco 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`).

The following is an example of setting up a rule for a translation group.

```
cd /Radius/Rules
Add CiscoTranslationRule
cd CiscoTranslationRule
cd Attributes
Set Realm @cisco.com
Set Incoming-Translation-Groups CiscoIncoming
Set Outgoing-Translation-Groups CiscoOutgoing
```

The `CiscoTranslationRule` rule must be referred to in the **/Radius/Policies** directory so the Cisco Access Registrar Policy Engine can invoke this rule. If the pattern of **Realm**, **DNIS**, or **CLID** matches the one defined in the rule, the Cisco Access Registrar Policy Engine sets **Incoming-Translation-Groups** to `CiscoIncoming` in the Environment Dictionary. By looking up the definition of `CiscoIncoming`, Cisco Access Registrar applies all of the translations to the incoming packet (before it is proxied to the other server).

When the proxied packet comes back to the RADIUS server, Cisco Access Registrar does a similar process to the outgoing packet.

**DNIS**, **CLID**, and **Realm** are supported for filtering packets in the current release.

**Note**

---

**Realm** in the above example is a filter for packets whose user-name contains `@cisco.com`.

---

## Wildcard Support

Cisco Access Registrar supports limited wildcard functionality. Cisco Access Registrar supports the “\*” and “?” wildcard characters. “\*” matches any number of characters, including the Null character, and “?” matches any single character, not including the Null character. Currently, wildcards apply to **Realm**, **DNIS**, and **CLID** attributes.

**Note**

---

All Realms should start with the “@” character. For example, `Realm=@cisco.com`.

---

The following is an example using the “\*” wildcard character:

```

/Radius
  /Rules
    /Rule1
      Name=rule1
      Description =
      ScriptName = ExecRealmRule
      Attributes/
        Authentication-Service = Local-Users
        Authorization-Service = Local-Users
        Realm = @*cisco.com

```

In the above example, when the domain of the user name in an access request matches the `@*cisco.com` pattern (for example, `@us.cisco.com`, `@eng.cisco.com`, and `@cisco.com` are all good matches), the **ExecRealmRule** script sets **Authentication-Service** and **Authorization-Service** to `Local-Users` in the environment dictionary.

The following is an example using the “?” wildcard character:

```

/Radius
  /Rules
    /Rule2
      Name = rule2
      Description =
      ScriptName = ExecDNISRule
      Attributes/
        Authentication-Service = Translation-Service
        Authorization-Service = Translation-Service
        DNIS = 1800345987?

```

In the above example, if the **Called-Station-Id** attribute in the packet matches `1800345987?` (for example, `18003459876` and `18003459870` are good matches, while `1800345987` is not), the **ExecDNISRule** script sets **Authentication-Service** and **Authorization-Service** to `Translation-Service` in the environment dictionary.

Cisco Access Registrar also supports both wildcard characters in one pattern. For example, `CLID = 180098?87*` is valid.

## Time of Day Rules

The **ExecTimeRule** script, invoked by the Cisco Access Registrar Policy Engine, applies the **TimeOfDay** rule to the access request packet during the RADIUS server’s incoming packet processing. The **ExecTimeRule** script either rejects or accepts the access request packets based upon the allowable user’s login profiles within a certain time range.

## Configuration

The format of the **TimeRange** internal attribute is as follows:

```
TimeRange = dateRange, timeRange [; dateRange, TimeRange]
dateRange = mdayRange | weekdayRange
mdayRange = number [-number]
number = 1 | 2 | 3 | ... | 31
weekdayRange = weekday [-weekday]
weekday = sun | mon | tue | wed | thu | fri | sat
timeRange = hh:mm - hh:mm
hh = 00 | 01 | ... | 23
mm = 00 | 01 | ... | 59
```

For example:

```
mon-fri,09:00-17:00
mon,09:00-17:00; tue-sat,12:00-13:00
mon,09:00-24:00;tue,00:00-06:00
1-13,10-17:00; 15,00:00-24:00
```

The format of the **AcceptedProfiles** internal attribute is as follows:

```
AcceptedProfiles = userProfile[; userProfile]
```

The following is an example:

```
/Policies
...

/MarketingTODPolicy
Name = MarketingTODPolicy
Description =
Grouping = MarketingTODRule

/EngineeringTODPolicy
Name = EngineeringTODPolicy
Description =
Grouping = EngineeringTODRule

...

/Rules
...

/MarketingTODRule
Name = MarketingTODRule
Description =
Script = ExecTimeRule
/Attributes
TimeRange = "mon-fri,8:00-17:00;sat,20:00-23:59;sun,00:00-7:00"
AcceptedProfile = PPP-users

/EngineeringTODRule
Name = EngineeringTODRule
Description =
Script = ExecTimeRule
/Attributes
TimeRange = "sun-sat,00:00-23:59;"
AcceptedProfiles = PPP-users;Telnet-users

...
```

## Notes

Spaces cannot be used in the **TimeRange** or **AcceptedProfiles** attributes.

The lower limit must be less than the upper limit within any specified range.

## Validation

Cisco Access Registrar validates the above configuration as follows:

1. Checks whether the **ExecTimeRule** script is defined under the **/Radius/Scripts** directory.
2. Checks whether the **TimeRange** and **AcceptedProfiles** attributes are defined under the **/Radius/Advanced/Attributes** Dictionary.
3. Checks whether the profiles included in the **AcceptedProfiles** attribute are defined under the **/Radius/Profiles** directory.

## Known Anomalies

1. Cisco Access Registrar does not check the format of the **TimeRange** attribute.
2. Cisco Access Registrar does not validate the lower limit and the upper limit with the **TimeRange** attribute.

## Prefix Feature

The Cisco Access Registrar prefix feature enables you to select a service based on the prefix in the User-Name attribute. Three new attributes and a new script, ExecPrefixRule, have been added to Cisco AR.

The prefix feature supports wildcard matching and multi-valued attributes by which multiple prefixes can be configured under the same rule.

## New Attributes

Three new attributes have been added to Cisco AR to support the prefix feature:

Prefix—List of valid prefixes

Delimiters—List of valid delimiters

StripPrefix—Option to strip or not to strip the prefix from the User-Name. If you do not configure this attribute, the default (Yes) is to strip the prefix from the User-Name. Allowed values are Yes and No.

## Example Configuration

The following example configuration is based on a new rule created to support the prefix feature. The new rule should be set under a Policy.

In the example configuration, if **cisco/bob@cisco.com** is the User-Name attribute, the ExecPrefixRule script sets the Authentication-Service to **cisco** and User-Name to **cisco/bob@cisco.com** because the StripPrefix attribute is configured **No**.

If the StripPrefix attribute is not configured, or if it is configured to **Yes**, the prefix will be stripped from the UserName, and UserName will become **bob@cisco.com**. The following steps provide an example configuration.

---

**Step 1** Using **aregcmd**, navigate to **//localhost /Radius/Rules**.

```
[ //localhost/Radius/Rules ]
  Entries 0 to 0 from 0 total entries
  Current filter: <all>
```

**Step 2** Add a new rule.

```
-->add rule1
```

```
Added rule1
```

**Step 3** Change directory to that rule, and set the rule's script to ExecPrefixRule.

```
--> cd rule1
```

```
[ //localhost/Radius/Rules/rule1 ]
  Name = rule1
  Description =
  Script~ =
  Attributes/
```

```
--> set Script ExecPrefixRule
```

```
Set Script ExecPrefixRule
```

**Step 4** Change directory to **Attributes**, and set the attributes to the following:

```
--> cd Attributes
```

```
[ //localhost/Radius/Rules/rule1/Attributes ]
```

```
--> set Authentication-Service cisco
```

```
Set Authentication-Service cisco
```

```
--> set Prefix "cisco" "ibm"
```

```
Set Prefix cisco ibm
```

```
--> set Delimiters @#%&/
```

```
Set Delimiters @#%&/
```

```
--> set StripPrefix No
```

```
Set StripPrefix No
```

```
--> ls
```

```
[ //localhost/Radius/Rules/rule1/Attributes ]
```

```
Authentication-Service = cisco
Delimiters = @#%&/
Prefix = cisco
Prefix = ibm
StripPrefix = No
```

**Step 5** Save the configuration changes.

```
-->save
```

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

```
->
```



## Wireless Support

---

This chapter provides information about using Cisco Access Registrar for wireless support. The following topics are included in this chapter:

- 3GPP2 Home Agent Support, page 11-1
- Session Correlation Based on User-Defined Attributes, page 11-3
- Managing Multiple Accounting Start/Stop Messages, page 11-4
- NULL Password Support, page 11-4
- New 3GPP2 VSAs in the CAR Dictionary, page 11-4

### 3GPP2 Home Agent Support

The Cisco Access Registrar server supports 3GPP2 home agents. This support enables mobile IP clients that authenticate through a Cisco Access Registrar RADIUS server to be told which home agent they should use.

Every Mobile IP client has a home domain that is served by a group of Home Agents (HA). The Mobile IP client sets up a tunnel to one (and only one) HA during a session while it roams. Typically, the domain can be determined by the Mobile IP client's network access identifier (NAI).



**Note**

---

The NAI is the userID submitted by the client during PPP authentication. In roaming, the purpose of the NAI is to identify the user as well as to assist in the routing of the authentication request.

---

During the authentication and authorization phase for each Mobile IP client, the RADIUS server must decide which HA from a group of HAs should be chosen to serve the client. This is called dynamic HA assignment.

### Home-Agent Resource Manager

Cisco Access Registrar 1.7 supports dynamic HA assignment with a new resource manager type called home-agent. You configure the home-agent resource manager with a list of IP addresses. The CAR 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.

## Load Balancing

The goal of dynamic HA assignment is to have load balancing among HAs. The Cisco Access Registrar server achieves this by evenly distributing mobile clients among HAs. At the same time, the CAR server ensures that the same HA is always assigned to the same Mobile IP client for the same session.

## Configuring the Home Agent Resource Manager

Use the **aregcmd** command **add** to create a new resource manager.

**Step 1** Use the **cd** command to change to the **Radius/ResourceManagers** level.

```
--> cd /Radius/ResourceManagers

[ //localhost/Radius/ResourceManagers ]
  Entries 0 to 0 from 0 total entries
  Current filter: <all>
```

**Step 2** Use the **add** command to specify the name of a resource manager to create.

```
--> add home-agent-pool
--> Added home-agent-pool
```

**Step 3** Use the **cd** command to change to the **Radius/ResourceManagers/home-agent-pool** level.

```
--> cd home-agent-pool

[ //localhost/Radius/ResourceManagers/home-agent-pool ]
  Name = home-agent-pool
  Description =
  Type =
```

**Step 4** Use the **set** command to set the resource manager type to **home-agent**.

```
--> set type home-agent
```

**Step 5** Use the **ls** command to view the subdirectories under home-agent-pool.

```
--> ls

[ //localhost/Radius/ResourceManagers/home-agent-pool ]
  Name = home-agent-pool
  Description =
  Type = home-agent
  Home-Agent-IPAddresses/
```

**Step 6** Use the **cd** command to change to the **Radius/ResourceManagers/home-agent-pool/Home-Agent-IPAddresses** level.

```
--> cd Home-Agent-IPAddresses

[ //localhost/Radius/ResourceManagers/home-agent-pool/Home-Agent-IPAddresses ]
```

**Step 7** Use the **add** command to add a single IP address or a range of IP addresses.

```
--> add 209.165.200.200-209.165.200.254
--> Added 209.165.200.200-209.165.200.254
```

## Querying and Releasing Sessions

The **aregcmd** program has been modified to support a new filter for **query-session** and **release-session**. You can use this filter to restrict a request (either query or release) to just the sessions with a given home-agent IP address. For example, consider the following command line.

```
--> query-session /radius with-home-agent 10.10.10.1
```

This command line will return all sessions that have a home-agent resource equal to the IP address 10.10.10.1.

Querying sessions using **aregcmd** displays the home-agent resource in each session as:

```
HA ddd.ddd.ddd.ddd
```

where each *ddd* is a decimal number from 0-255.

## Access Request Requirements

When the home-agent resource manager receives an Access-Request that contains a CDMA-HA-IP-Addr attribute, the home-agent resource manager checks the response dictionary to see if it already has a CDMA-HA-IP-Addr attribute. If it does, then the Mobile IP client has been assigned a HA address already and the resource manager does not need to do anything.

If the value of the CDMA-HA-IP-Addr attribute in the request dictionary is 0.0.0.0, the home-agent resource manager assigns a HA and puts a new CDMA-HA-IP-Addr attribute whose value is the IP address of the HA in the response dictionary.

If the value of the CDMA-HA-IP-Addr attribute is not 0.0.0.0, the Mobile IP client has been assigned a HA address already. The home-agent resource manager copies the attribute (with its value) from the request dictionary into the response dictionary.

The CAR server may select the session manager session manager based on the domain (using the rule engine, dynamic properties, or scripting), and it allows each session manager to have its own home-agent resource manager.

## 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 script **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** 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**.

## Managing Multiple Accounting Start/Stop Messages

Since the PDSN is aware when it sends a RADIUS stop followed by a start record, it inserts the new Session Continue attribute (3GPP2/48) into the stop record. The existence of the Session Continue attribute denotes that a start record will immediately be sent and the packet data session continues on the PDSN.

When CAR 1.7 receives an accounting stop packet, the following two conditions trigger a release of a session and its resources.

- There is no 3GPP2/48 Session Continue attribute in the stop packet and the number of accounting stops received is greater or equal to the starts received for this session
- The 3GPP2/48 Session Continue attribute is present in the stop packet, but its value is zero (0)



**Note**

---

One of the conditions above must be true to release the session and its resources.

---

## NULL Password Support

CAR 1.7 defines a new CAR environment variable, *Allow-NULL-Password*. At authentication time, if the following three conditions are met, user authentication is bypassed.

1. *Allow-NULL-Password* environment variable is set to TRUE.
2. The User-Password or CHAP-Password must be NULL in the incoming request. (If it is not NULL, normal password checking will occur.)
3. A user record exists for this user.

By default, the *Allow-NULL-Password* environment variable is not set.



**Note**

---

You should be aware of the security impact when using the NULL Password feature.

---

You can set this environment variable three different ways:

1. For the user in local database, one new field *AllowNullPassword* is added in the user record. When Cisco Access Registrar fetches a user record for authentication, if this field is set to TRUE and *Allow-NULL-Password* environment variable does not exist, it sets *Allow-NULL-Password* environment variable to TRUE.
2. If the user record is in LDAP database, then the *LDAPToEnvironmentMappings* must be defined to map an attribute in LDAP user record to *Allow-NULL-Password* environment variable.
3. Through scripting. This allows the decision to be made based on run-time condition, such as attributes in the access-request or policies.

## New 3GPP2 VSAs in the CAR Dictionary

CAR 1.7 supports 3GPP2 attributes in the vendor-specific dictionary.

**Note**

---

There is no planned support for the Accounting-Container (3GPP2/6) attribute because it has different syntax than other VSAs and requires special processing.

---





## Using LDAP

This chapter provides information about using Lightweight Directory Access Protocol (LDAP) with Cisco Access Registrar to access information directories. You can use Cisco AR to authenticate and authorize access requests by querying user information through LDAP.



**Note**

Cisco Access Registrar 3.0 requires an LDAP Version 3 directory server for any remote server used for LDAP.

## Configuring LDAP

To use LDAP in Cisco AR, use **aregcmd** to do the following:

1. Configure an LDAP Service
2. Configure an LDAP RemoteServer object
3. Set LDAP service as the default AA service
4. Save your configuration

After you issue the **save** command, Cisco AR attempts to validate the configuration, checks for all required properties, and ensures there is no logic error. If the validation is successful, Cisco AR saves the configuration to the MCD database. When Cisco AR is reloaded, it shuts down any current LDAP connections and builds new connections for the configured LDAP remote servers.

## Configuring the LDAP Service

You configure an LDAP service under **/Radius/Services**. When you define an LDAP service under **/Radius/Services**, you must set its type to LDAP.

```
[ //localhost/Radius/Services/AR-LDAP ]
  Name = AR-LDAP
  Description =
  Type = ldap
  IncomingScript~ =
  OutgoingScript~ =
  OutagePolicy~ = RejectAll
  OutageScript~ =
  MultipleServersPolicy = Failover
  RemoteServers/
```

Table 12-1 describes the LDAP service properties.

**Table 12-1 LDAP Service Properties**

Parameter	Description
Name	Mandatory; inherited from the upper directory
Description	An optional description of the service
Type	Must be set to LDAP for LDAP service
IncomingScript	Optional
OutgoingScript	Optional
OutagePolicy	Mandatory; must be set to AcceptAll or Drop Packet, or defaults to RejectAll
OutageScript	Optional
MultipleServersPolicy	Mandatory; must be set to RoundRobin or defaults to Failover.
RemoteServers	Mandatory; list of one or more remote servers defined under <b>/Radius/Services/LDAP/RemoteServers</b> . These servers must be listed in order under <b>/Radius/RemoteServers</b> .

## MultipleServersPolicy

Use the MultipleServersPolicy property to configure the LDAP remote servers in RoundRobin mode, or the default Failover mode applies. When set to Failover, Cisco AR directs requests to the first server in the **/Radius/Services/LDAP/RemoteServers** list. If that server should fail or go offline, Cisco AR redirects all requests to the next server in the list. The process continues until Cisco AR locates an on-line server.

When set to RoundRobin, Cisco AR directs each request to the next server in the RemoteServers list to share the resource load across all listed servers.

## RemoteServers

Use the RemoteServers directory to list one or more remote servers to process access requests. The servers must also be listed in order under **/Radius/RemoteServers**.

The order of the RemoteServers list determines the sequence for directing access requests when MultipleServersPolicy is set to RoundRobin mode. The first server in the list receives all access requests when MultipleServersPolicy is set to Failover mode.

## Configuring an LDAP RemoteServer

Use the **aregcmd add** command to add LDAP servers under **/Radius/RemoteServers**. You must configure an LDAP RemoteServer object for each RemoteServer object you list under **/Radius/Services/LDAP/RemoteServers**.

The following properties must be configured to use an LDAP remote server:

- Name
- Protocol
- IP Address

- Port
- HostName
- BindName
- BindPassword
- SearchPath

Table 12-2 describes the LDAP Remote Server properties.

**Table 12-2 LDAP Remote Server Properties**

Parameter	Description
Name	Mandatory name you assign
Description	Optional description of the server
Protocol	Mandatory and must be set to LDAP; no default value
IPAddress	Mandatory IP Address, but this field is ignored
Port	Mandatory; port on which LDAP server listens, default is zero
ReactivateTimerInterval	Mandatory; default is 300000 (ms)
Timeout	Mandatory; specifies length of time Cisco AR waits for a response from the LDAP server before noting the server as down; default is 15 (seconds)
HostName	Qualified or unqualified DNS name; should be resolvable to the IP address of the LDAP server
BindName	Specifies the distinguished name (DN) in the LDAP server for Cisco AR to bind with the LDAP server
BindPassword	Specifies the password for the distinguished name
UseSSL	FALSE by default
SearchPath~	Specifies search base to the organization and domain; for example: o=cisco.com
EscapeSpecialCharInUserName	FALSE by default
Filter~	(uid=%s) by default
UserPasswordAttribute	Should be set to the attribute in the directory server which stores users' passwords; default is <i>userpassword</i>
LimitOutstandingRequests	FALSE by default
MaxOutstandingRequests	Limits the number of requests to the LDAP server; used to throttle the request load when the LDAP server does not function well under high TPS rates (default is 0)
MaxReferrals	Limits the number of referrals Cisco AR allows when working with LDAPv2 (default is 0)
ReferralAttribute	LDAP attribute that contains a referral for LDAPv2
ReferralFilter	Filter used when following a referral for LDAPv2

Table 12-2 LDAP Remote Server Properties (continued)

Parameter	Description
PasswordEncryptionStyle	<p>Dynamic by default; must be set to one of the following depending on the algorithm used by the LDAP server to encrypt passwords:</p> <ul style="list-style-type: none"> <li>Dynamic</li> <li>Crypt</li> <li>None</li> <li>SHA-1</li> <li>SSHA-1</li> </ul> <p>When set to <i>Dynamic</i>, Cisco AR analyzes the password and detects the encryption algorithm used.</p> <p><i>None</i> indicates that the LDAP server stores clear text passwords.</p> <p><b>Note</b> If CHAP authentication is used with LDAP backing store, passwords in LDAP must be stored as clear text.</p>
DNSLookupAndLDAPRebindInterval	Specifies the time-out period after which the Cisco AR server will attempt to resolve the LDAP host name to IP address (DNS resolution); 0 by default
LDAPToRadiusMappings	<p>A list of name/value pairs in which the name is the name of the <b>ldap</b> attribute to retrieve from the user record, and the value is the name of the RADIUS attribute to set to the value of the <b>ldap</b> attribute retrieved.</p> <p>For example, when the <b>LDAPToRadiusMappings</b> has the entry: <b>FramedIPAddress = Framed-IP-Address</b>, the RemoteServer retrieves the <b>FramedIPAddress</b> attribute from the <b>ldap</b> user entry for the specified user, uses the value returned, and sets the Response variable <b>Framed-IP-Address</b> to that value.</p>
LDAPToEnvironmentMappings	<p>A list of name/value pairs in which the name is the name of the <b>ldap</b> attribute to retrieve from the user record, and the value is the name of the Environment variable to set to the value of the <b>ldap</b> attribute retrieved.</p> <p>For example, when the <b>LDAPToEnvironmentMappings</b> has the entry: <b>group = User-Group</b>, the RemoteServer retrieves the <b>group</b> attribute from the <b>ldap</b> user entry for the specified user, uses the value returned, and sets the Environment variable <b>User-Group</b> to that value.</p>
LDAPToRadiusMappings	Optional; list of <i>name/value</i> pairs where name is the <i>name</i> of the data store attribute to retrieve from the user record and <i>value</i> is the name of the RADIUS attribute to set to the value of the data store attribute retrieved.
LDAPToEnvironmentMappings	Optional; list of <i>name/value</i> pairs in which <i>name</i> is the name of the data store attribute to retrieve from the user record, and <i>value</i> is the name of the Environment variable to set to the value of the LDAP attribute retrieved.
LDAPToCheckItemMappings	Optional; list of LDAP <i>attribute/value</i> pairs which must be present in the RADIUS access request and must match, both name and value, for the check to pass.

## DNS Look Up and LDAP Rebind Interval

Cisco AR provides a DNS Look-up and LDAP Rebind feature that enables you to use a smart DNS server for LDAP host name resolution, allows you to query a DNS server at set intervals to resolve the LDAP host name, and optionally rebind to the LDAP server, if necessary.

When you configure Cisco AR to use an LDAP directory server, you can specify the host name of the LDAP directory server. The host name can be a qualified or an unqualified name. You can also specify a time-out period after which Cisco AR will again resolve the host name. If the IP address returned is different from the previous, Cisco AR establishes a new LDAP bind connection.

The `DNSLookupAndLDAPRebindInterval` property specifies the time-out period after which the Cisco AR server will attempt to resolve the LDAP host name to IP address (DNS resolution). When you do not modify `DNSLookupAndLDAPRebindInterval`, the default value zero indicates the server will perform normal connection and binding only at start-up time or during a reload. Unless you change the default to a value greater than zero, the server will not perform periodic DNS lookups.

Cisco AR maintains and uses the existing bind connection until a new one is established to minimize any performance impact during the transfer. Cisco AR ensures that no requests are dropped or lost during the transfer to a new LDAP binding.

Set the `DNSLookupAndLDAPRebindInterval` using a numerical value and the letter H for hours or M for minutes, such as in the following examples:

```
set DNSLookupAndLDAPRebindInterval 15M—performs DNS resolution every 15 minutes
```



**Note**

We recommend that you do not set `DNSLookupAndLDAPRebindInterval` to a value less than 15 minutes to minimize its effect on server performance.

```
set DNSLookupAndLDAPRebindInterval 1h—performs DNS resolution every hour
```

The following shows an example configuration for the DNS Look-up and LDAP Rebind feature.

**Step 1** Login to the Cisco AR server, and use `aregcmd` to navigate to `//localhost/Radius/Remoteservers`. If necessary, add the LDAP server, or change directory to it.

```
cd /Radius/RemoteServers/ldap-serv1/
```

**Step 2** Set the `DNSLookupAndLDAPRebindInterval` property to the interval time desired.

```
set DNSLookupAndLDAPRebindInterval 30 M
```

## LDAP Rebind Failures

Cisco AR records any name resolution failures, bind successes and failures, and the destination host name and IP address in the log file. At trace level 3, Cisco AR also logs the time of any new bind connections and the closing of any old bind connections.

If either the name resolution or bind attempt fail, Cisco AR continues using the existing bind connection until the time-out has expired again. If there is no existing bind connection, Cisco AR marks the remote server object as *down*.

## LDAPToRadiusMappings

Configure `LDAPToRadiusMappings` with a list of *name/value* pairs where name is the name of the data store attribute to retrieve from the user record and the value is the name of the RADIUS attribute to set to the value of the data store attribute retrieved.

Values stored in a multi-valued field in the LDAP directory are mapped to multiple RADIUS attributes. For example, if the `LDAPToRadiusMappings` has the following entry:

```
tunnel-info = Cisco-AVPair
```

The following LDAP fields in the user's record will create four Cisco-AVPair attributes in the user's Access-Accept RADIUS packet:

```
tunnel-info: vpdn:tunnel-id=ssg001
tunnel-info: vpdn:tunnel-type=12tp
tunnel-info: vpdn:ip-addresses=10.2.2.2
tunnel-info: vpdn:12tp-tunnel-password=secret
```

## LDAPToEnvironmentMappings

LDAPToEnvironmentMappings comprises a list of attribute name/value pairs or AV pairs where the name is the name of the data store 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 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

LDAPToCheckItemMappings comprises a list of LDAP AV pairs which must be present in the RADIUS access request and must match, both name and value, for the check to pass. Cisco AR will first authenticate the user's password in the Access-Request before validating the check item attributes.

## Setting LDAP As Authentication and Authorization Service

Use **aregcmd** to configure the LDAP Service as the default authentication and authorization service under **/Radius** as in the following:

```
set DefaultAuthenticationService AR-LDAP
```

```
set DefaultAuthorizationService AR-LDAP
```

## Saving Your Configuration

When you use **aregcmd** to **save** your configuration, Cisco AR does the following:

- Attempts to validate the configuration
- Checks for all required parameters
- Ensures there are no logic errors

If the validation is successful, Cisco AR saves the configuration to the MCD database. When you **reload**, Cisco AR shuts down any current LDAP connections and builds new connections for the configured LDAP servers.

## CHAP Interoperability with LDAP

If the you plan to use CHAP authentication with an LDAP backing store, the password in LDAP must be stored as clear text. This is due to the one-way hash used by the CHAP, crypt, SHA-1, and SSHA encryption algorithms.

## Allowing Special Characters in LDAP Usernames

This feature allows you to use special characters in LDAP usernames. The allowable special characters are \*, (, and \. These special characters can be included in the string passed to LDAP as the LDAP username value (usually the RADIUS username attribute).

The default of `EscapeSpecialCharInUserName` is `FALSE`. To enable this feature, use `aregcmd` to set the `EscapeSpecialCharInUserName` attribute in `/Radius/RemoteServers/ldap-server` to `TRUE`, as shown in the following example.

```
cd /Radius/RemoteServers/ldap-server

set EscapeSpecialCharInUserName TRUE

/Radius/RemoteServers/Ldap-Server
EscapeSpecialCharInUserName = TRUE
```



**Note**

This feature supports the LDAP V3 library.

## Dynamic LDAP Search Base

A new environment variable, `Dynamic-Search-Path` (see `rex.h`), can be used to set the dynamic LDAP search base. If this environment variable is defined for an LDAP service, it will override the default LDAP search base defined in the LDAP Remote Server configuration. This allows the LDAP search base to be configured on a per-user basis.

For example, you could match the search base to the organization and domain (in a Tcl script called from `/Radius/IncomingScript`):

```
set user [ $request get User-Name ]
if { [ regexp {^[^@]+@[^\.\.]+\.(.+)$} $user m org domain ] } {
$environ put Dynamic-Search-Path "ou=$org,ou=people,o=$domain"
```

## Analyzing LDAP Trace Logs

Cisco AR records in the log files any name resolution failures, bind successes and failures, and the destination host name and IP address. At trace level 3, Cisco AR logs the time of any new bind connections and the closure of any old bind connections and also information about user login requests and reply messages.

## Successful Bind Message

The following message is logged in the **name\_radius\_1\_trace** file, when the Cisco AR server successfully binds to the LDAP server. In this case, spatula-u5 is the LDAP server listening on port number 389.

```
04/23/2003 11:02:57: Log: Successfully bind to LDAP Server ldapserver (spatula-u5:389)
```

## Bind Failure Messages

The following messages are logged in the **name\_radius\_1\_trace** file, when AR server fails to bind to the LDAP server.

```
04/23/2003 11:10:50: Log: Write in LDAPClient returned an error (32)
```

```
04/23/2003 11:10:50: Log: Remote LDAP Server ldapserver (spatula-u5:387): Unable to bind to LDAP Server: Can't contact LDAP server
```

```
04/23/2003 11:10:50: Log: Remote LDAP Server ldapserver (spatula-u5:387): Failed to open the connection to the LDAP server
```

Messages like those above could indicate that the hostname specified does not resolve to the correct IP address of the LDAP server or the configured port number might not be the port on which the LDAP server listens.

The following messages are logged in the **name\_radius\_1\_trace** file, when AR server fails to bind to the LDAP server.

```
04/23/2003 11:45:14: Log: Remote LDAP Server ldapserver (spatula-u5:389): Unable to bind to LDAP Server: No such object ()
```

```
04/23/2003 11:45:14: Log: Remote LDAP Server ldapserver (spatula-u5:389): Failed to open the connection to the LDAP server
```

The Distinguished Name (DN) provided in the BindName property was invalid. The DN provided in the BindName property should contain the exact string used in the directory server to define the object.

The following messages are logged in the **name\_radius\_1\_trace** file, when AR server fails to bind to the LDAP server.

```
04/23/2003 11:51:55: Log: Remote LDAP Server ldapserver (spatula-u5:389): Unable to bind to LDAP Server: Invalid credentials
```

```
04/23/2003 11:51:55: Log: Remote LDAP Server ldapserver (spatula-u5:389): Failed to open the connection to the LDAP server
```

The messages above indicate that the password provided in the BindPassword property was incorrect.

## Login Failure Messages

The following messages are logged in the **name\_radius\_1\_trace** file, when user *jane* tries to login. These messages indicate that user *jane* does not have a record in the directory server or the SearchPath property has an incorrect value. The SearchPath property should have the directory where the user record is stored in the directory server.

Notice how the messages specify the service, remote LDAP server, user name, and contents of the Access-Reject packet.

```

04/23/2003 11:24:17: P8457: Authenticating and Authorizing with Service AR-LDAP
04/23/2003 11:24:17: id = 5
04/23/2003 11:24:17: P8457: Remote LDAP Server ldapserver (spatula-u5: 389): Querying
LDAP server, id = 5.
04/23/2003 11:24:17: P8457: Remote LDAP Server ldapserver (spatula-u5: 389): GotLDAP
response, id = 5.
04/23/2003 11:24:17: P8457: Remote LDAP Server ldapserver (spatula-u5: 389): No
matching entries returned from LDAP query.
04/23/2003 11:24:17: P8457: User jane was not found in the LDAP store
04/23/2003 11:24:17: P8457: Rejecting request
04/23/2003 11:24:17: P8457: Rejecting request
04/23/2003 11:24:17: P8457: Trace of Access-Reject packet
04/23/2003 11:24:17: P8457: identifier = 4
04/23/2003 11:24:17: P8457: length = 35
04/23/2003 11:24:17: P8457: reqauth = 01:ad:cf:c7:4f:8e:a4:38:b0:d8:0a:e5:3d:9f:64:16
04/23/2003 11:24:17: P8457: Reply-Message = Access Denied

```

The following messages are logged in the **name\_radius\_1\_trace** file, when user **bob** tries to login. These messages indicate that user **bob** tried to login with an incorrect password.

```

04/23/2003 11:36:59: P8461: Authenticating and Authorizing with Service AR-LDAP
04/23/2003 11:36:59: id = 7
04/23/2003 11:36:59: P8461: Remote LDAP Server ldapserver (spatula-u5: 389): Querying
LDAP server, id = 7.
04/23/2003 11:36:59: P8461: Remote LDAP Server ldapserver (spatula-u5: 389): Got LDAP
response, id = 7.
04/23/2003 11:36:59: P8461: Remote Server ldapserver (spatula-u5:389): User bob's
password does not match
04/23/2003 11:36:59: P8461: User bob's password does not match
04/23/2003 11:36:59: P8461: Rejecting request
04/23/2003 11:36:59: P8461: Rejecting request
04/23/2003 11:36:59: P8461: Trace of Access-Reject packet
04/23/2003 11:36:59: P8461: identifier = 6
04/23/2003 11:36:59: P8461: length = 35
04/23/2003 11:36:59: P8461: reqauth = de:8d:4b:c4:f9:c0:06:a6:98:2d:8c:e9:f3:a9:a3:c2
04/23/2003 11:36:59: P8461: Reply-Message = Access Denied

```

The following messages are logged in the **name\_radius\_1\_trace** file, when user **bob** tries to login. These messages indicate the user record for user **bob** does not contain an attribute called **pass**. The UserPasswordAttribute property has an incorrect value called **pass**. The UserPasswordAttribute property should have the attribute name in the directory records where the user password is stored.

```

04/23/2003 12:02:09: P9865: Authenticating and Authorizing with Service AR-LDAP
04/23/2003 12:02:09: id = 2
04/23/2003 12:02:09: P9865: Remote LDAP Server ldapserver (spatula-u5: 389): Querying
LDAP server, id = 2.
04/23/2003 12:02:09: P9865: Remote LDAP Server ldapserver (spatula-u5: 389): Got LDAP
response, id = 2.
04/23/2003 12:02:09: P9865: Remote LDAP Server ldapserver (spatula-u5: 389): LDAP
entry for user bob did not have a password (" pass") attribute
04/23/2003 12:02:09: P9865: User bob's password does not match
04/23/2003 12:02:09: P9865: Rejecting request
04/23/2003 12:02:09: P9865: Rejecting request
04/23/2003 12:02:09: P9865: Trace of Access-Reject packet
04/23/2003 12:02:09: P9865: identifier = 10
04/23/2003 12:02:09: P9865: length = 35
04/23/2003 12:02:09: P9865: reqauth = 0d:b6:83:f9:e8:3d:a4:ad:f1:c9:33:72:91:0b:29:1c
04/23/2003 12:02:09: P9865: Reply-Message = Access Denied

```

**Note**

---

Remember to **reload** the Cisco AR server after any changes to the LDAP server configuration.

---



## Using Open Database Connectivity

---

Cisco AR supports Open Database Connectivity (ODBC), an open specification that provides application developers a vendor-independent API with which to access data sources. Cisco AR provides a new type of RemoteServer object and a new service to support ODBC. You can use Cisco AR to authenticate and authorize access requests by querying user information through ODBC.

ODBC is an application program interface (API). Real data exchange between an application and data store is still carried out by SQL through ODBC. To achieve the most flexibility, you are required to define your own SQL using **aregcmd**. Cisco AR will register the SQL statements and send them to the data store through ODBC when required. Because you can define your own SQL, Cisco AR supports sites that have their own data stores.

ODBC is configured using **.ini** files, specifically **odbc.ini** and **odbcinst.ini**. However, you cannot create or modify these files directly. Cisco AR creates the **.ini** files after you use **aregcmd** to configure the ODBC connection. The SQL is stored in the local database (MCD). During execution, the Cisco AR server reads the local database, prepares the SQL statements, and sends the SQL to the data source.



### Note

---

Cisco AR uses its own ODBC driver manager and does not share existing ODBC drivers (if you already have ODBC installed). If you are already using ODBC, you will have to maintain two separate ODBC installations.

---

The ODBC memory requirement depends on your configuration. The more datasources you configure, the more memory is required. Packet processing time might increase if you configure a large number of SQL statements under SQLDefinition.

The Cisco Access Registrar 3.0 package includes some ODBC Drivers, and you should use the included driver whenever possible. If a data store's ODBC driver is not included with Cisco AR, you are required to install it. You configure the driver library using **aregcmd** to modify the associated **ini** file.

## Oracle Software Requirements

The Cisco Access Registrar 3.0 ODBC feature requires that you have Oracle 8.1.6 or 8.1.7 client software installed. All Oracle client software library files are expected under **\$ORACLE\_HOME/lib**.

When you install Cisco Access Registrar 3.0 software, the installation process prompts you for **ORACLE\_HOME** variable and sets it in the Cisco AR start-up script, **/etc/init.d/arsserver**. Two other environment variables (**ODBCINI** and **ODBCSYSINI**) are also set in the **arsserver** script. To change any of these variables, modify the **/etc/init.d/arsserver** script and restart the Cisco AR server.

# Configuring ODBC

You use **aregcmd** to define your ODBC configuration and SQL statements. The Cisco AR server automatically creates the **ODBC.ini** file for your driver manager and driver based on how you configure ODBC.

To use ODBC in Cisco AR, you must do the following:

1. Configure an ODBC Service
2. Configure an ODBC RemoteServer object
3. Configure an ODBC DataSource
4. Set ODBC service as the default AA service
5. Save your configuration

After you **save** and validate your configuration, it is saved in the MCD database. If you have configured an ODBC service, Cisco AR will query the MCD database and create or modify the **odbc.ini** file before it builds a connection to the database. When you reload your configuration, Cisco AR shuts down any existing ODBC connections, then queries the MCD database to create or modify the **odbc.ini** file and build a new connection for any configured ODBC Data Sources.

## Configuring the ODBC Service

You configure an ODBC service under **/Radius/Services**. When you define an ODBC service under **/Radius/Services**, you must set its type to ODBC and provide the following configuration options:



### Note

We will use ODBC as the ODBC service name in the following examples.

```
[ //localhost/Radius/Services/ODBC ]
  Name = ODBC
  Description =
  Type = odbc
  IncomingScript~ =
  OutgoingScript~ =
  OutagePolicy~ = RejectAll
  OutageScript~ =
  MultipleServersPolicy = Failover
  RemoteServers/
```

Table 13-1 describes the ODBC service parameters.

**Table 13-1 ODBC Service Parameters**

Parameter	Description
Name	Mandatory; inherited from the upper directory
Description	An optional description of the service
Type	Must be set to ODBC for ODBC service
IncomingScript	Optional
OutgoingScript	Optional
OutagePolicy	Mandatory; must be set to AcceptAll or Drop Packet, or defaults to RejectAll

**Table 13-1 ODBC Service Parameters**

Parameter	Description
OutageScript	Optional
MultipleServersPolicy	Mandatory; must be set to RoundRobin or defaults to Failover.  When set to Failover, Cisco AR directs requests to the first server in the list until it determines the server is off-line. If so, Cisco AR redirects all requests to the next server in the list until it finds an on-line server.  When set to RoundRobin, Cisco AR directs each request to the next server in the RemoteServers list in order to share the resource load across all servers in the RemoteServers list.
RemoteServers	Mandatory list of remote servers defined under <b>/Radius/Services/ODBC/RemoteServers</b> such as <b>ODBC-Primary</b> and <b>ODBC-Secondary</b>

## Configuring an ODBC RemoteServer

You must configure an ODBC RemoteServer object for each RemoteServer object you list under **/Radius/Services/ODBC/RemoteServers**. Use the **aregcmd** command **add** to add ODBC servers under **/Radius/RemoteServers**.

Table 13-2 describes the ODBC service parameters.

**Table 13-2 ODBC Remote Server Parameters**

Parameter	Description
Name	Mandatory; inherited from the upper directory
Description	An optional description of the server
Protocol	Mandatory and must be set to ODBC; no default value
ReactivateTimeInterval	Mandatory; default is 300000 (ms)
Timeout	Mandatory; default is 15 (seconds)
DataSourceConnections	Mandatory; number of concurrent connections to data source (default is 8)
ODBCDataSource	Mandatory; no default value
SQLDefinition	SQLDefinition/ (mandatory, no default); UserPasswordAttribute = (mandatory, no default; data store field for user password)  SQLStatements/ SQLStatement1/ SQLStatement2/
ODBCToRadiusMappings	(optional)
ODBCToEnvironmentMappings	(optional)
ODBCToCheckItemMappings	(optional)

## ODBC Data Source

ODBCDataSource is the name of the datasource to be used by the remote server. An ODBCDataSource name can be reused by multiple remote servers. You configure ODBCDataSources under **/Radius/Advanced/ODBCDataSources**. Refer to *Configuring an ODBC DataSource*, page 13-6, for more information.

## SQL Definitions

SQLDefinitions lists the UserPasswordAttribute and one or more SQL statements, listed numerically in the order to be run. The UserPasswordAttribute represents a column in the database that contains users' password information. Individual SQLStatements are numbered SQL1 through SQL*n* under SQLStatements, as shown in the following example:

```
SQLDefinition/
  UserPasswordAttribute = asdfjkl
  SQLStatements/
    SQL1/
    SQL2/
    SQL3
    ...
```

The following example is an SQL statement used for Authentication and Authorization:

```
SQLStatements/
  SQL1
    Name = SQL1
    Type = query (mandatory, no default; must be query)
    SQL = SQL statement (mandatory, no default)
    ExecutionSequenceNumber = Sequence number for SQLStatement execution. (mandatory,
    no default and must be greater than zero).
    MarkerList = UserName/SQL_DATA_TYPE ..... (mandatory, UserName must be defined)
```

Table 13-3 describes the SQL Statement parameters.

**Table 13-3 SQL Statement Parameters**

Parameter	Description
Name	Name/number of SQL statement
Type	Query (mandatory, no default value)
SQL	SQL query statement
ExecutionSequenceNumber	Sequence number for SQLStatement execution, must be greater than zero (mandatory, no default)
MarkerList	Defines all markers for the query. MarkerList uses the format <i>UserName/SQL_DATA_TYPE</i> .

## SQL Syntax Restrictions

You must observe the following SQL syntax restrictions in SQL queries for Cisco Access Registrar 3.0.

1. The SQL statement must be in the format of SELECT ... FROM ... WHERE ... (Statements might be in lower-case.)



**Note** 'WHERE' is compulsory in the SQL statement.

- Any arguments to Oracle functions like *distinct*, *count* must be given within braces, as shown in the following example:

```
select distinct(attribute),password from profiles where username=?
```

The resulted column from *distinct(attribute)* will be put into *attribute* which can be used for ODBC Mappings. The actual result set from Oracle for this column would be named *distinct(attribute)*.

- The column list in the SQL statement must be delimited with a comma (,) and any extra spaces between statements are ignored. Aliasing for column names in SQL is not allowed. SQLDefinition properties define the SQL you want to execute, as shown in the following example.

## Specifying More Than One Search Key

You can specify more than one search key for a table in the SQL SELECT. To do so, add another search criteria to the SQL statement and add the environment variable name to the MarkerList. For example, the following query and MarkerList can be used to look up a username and CLID match.

```
select password from user_table where username = ? and clid = ?
```

In this case, the marker list would look like this:

```
UserName/SQL_CHAR clid/SQL_CHAR
```

To configure the multiple entries in the MarkerList list, surround the entire string in double quotes like the following:

```
set MarkerList "UserName/SQL_CHAR CLID/SQL_CHAR"
```

To make this work, a variable called CLID must be in the environment dictionary. You can use a script to copy the appropriate value into the variable.

## ODBCToRadiusMappings

You configure ODBCToRadiusMappings with a list of *name/value* pairs where name is the name of the data store attribute to retrieve from the user record and the value is the name of the RADIUS attribute to set to the value of the data store attribute retrieved.

For example, when the ODBCToRadiusMappings has the following entry, the RemoteServer retrieves the attribute from the data store user entry for the specified user, uses the value returned, and sets the response variable *Framed-IP-Address* to that value:

```
FramedIPAddress = Framed-IP-Address
```

When an SQL select statement returns more than one row for a column mapped under ODBCToRadiusMappings, multiple RADIUS attributes will be created. As of the Cisco AR 3.0R6 release, the table.column syntax requires an SQL alias for the mapping to work, as shown in the following example:

```
SQLStatements/
SQL1/
select table1.abc as t1abc, password from table2 where username = ?
Mapping: t1abc = my_mapping
```

If two rows are returned for ciscoavpair column, two Cisco-AVPair attributes will be created.

## ODBCToEnvironmentMappings

Under ODBCToEnvironmentMappings there is a list of name/value pairs in which the name is the name of the data store 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 attribute from the ODBC user entry for the specified user, uses the value returned, and sets the Environment variable User-Group to that value. When an SQL select statement returns more than one row for a column mapped under ODBCToEnvironmentMappings, the first row's value will be taken.



### Note

As of the Cisco AR 3.0R6 release, the `table.column` syntax requires an SQL alias for the mapping to work.

## Configuring an ODBC DataSource

ODBCDataSource is the name of the datasource to be used by the remote server. You configure ODBCDataSources under **/Radius/Advanced/ODBCDataSources**. Multiple remote servers can use the same ODBCDataSource.

Under the ODBCDataSource object definition, a list defines **ODBC.ini** file name/value pairs for a connection. The list includes a Type field and a Driver field, different for each Driver and Data Source, to indicate its Driver and Data Source. Cisco Access Registrar 3.0 currently supports only the Easysoft Open Source Oracle Driver.

Table 13-4 describes the Easysoft Open Source Oracle Driver options.

**Table 13-4 Easysoft Open Source Oracle Driver Options**

Parameter	Description
Name	Name of the ODBCDataSource
Type	Mandatory; must be Oracle_es
Driver	Mandatory; <b>liboarodbc.so</b> (default value)
Database	Mandatory; Oracle Client configuration database name (no default value)
UserID	Mandatory; database user name (no default value)
Password	Optional user password; shown encrypted

## Setting ODBC As Authentication and Authorization Service

Use **aregcmd** to configure the ODBC Service as the default authentication and authorization service under **//localhost /Radius** as in the following:

```
set DefaultAuthenticationService odbc-service
```

```
set DefaultAuthorizationService odbc-service
```

## Saving Your Configuration

When you use **aregcmd** to **save** your configuration, Cisco AR attempts to validate the configuration, checks for all required parameters, and ensures there is no logic error. If the validation is successful, the configuration is saved to the MCD database. When you **reload**, Cisco AR shuts down any current ODBC connections and builds new connections for the configured ODBC Data Sources.





## Backing Up the Database

---

This chapter describes the Cisco Access Registrar shadow backup facility, which ensures a consistent snapshot of Cisco Access Registrar's database for backup purposes.

Because the Cisco Access Registrar's database (called MCD) does a variety of memory caching, and may be active at any time, you cannot simply rely on doing system backups to protect the data in the database. At the time you run a system backup, there may be Cisco Access Registrar operations in progress that cause the data copied to the system backup tape to be inconsistent and unusable as a replacement database.

To ensure a consistent backup, Cisco Access Registrar uses a shadow backup facility. Once a day, at a configurable time, Cisco Access Registrar suspends all activity to the database and takes a snapshot of the critical files. This snapshot is guaranteed to be a consistent view of the database, and it is preserved correctly on a system backup tape.

## Configuration

The only configuration for this facility is through a single entry in the system Registry at `$INSTALL/conf/car.conf` is the registry path to this item.

This entry is a string that represents the time-of-day at which the shadow backup is scheduled to occur (in 24 hour HH:MM format). The default is 23:45.

When you remove this entry or set it to an illegal value (for example, anything that does not begin with a digit), backups are suppressed. The server is otherwise unaffected.

## Command Line Utility

In addition to being available at a scheduled time of day, you can also force a shadow backup by using the `mcshadow` utility located in the `$INSTALL/bin` directory. There are no command-line arguments.

This may take a few minutes to complete as a full copy of the database is created.

## Recovery

When it is necessary to use the shadow backup to recover data, either because the regular working database has been corrupted by a system crash, or because the disk on which it resides has become corrupted, perform the following:

- Step 1** Stop all Cisco Access Registrar servers.
- Step 2** Make sure three files (**mcddb.d01**, **mcddb.d02**, and **mcddb.d03**) exist in the **\$INSTALL/data/db.bak** directory.
- Step 3** Copy the files into the **\$INSTALL/data/db** directory. Do not move them because they might be needed again.
- Step 4** Change directory to the **\$INSTALL/data/db** directory.
- ```
cd $INSTALL/data/db
```
- Step 5** Rebuild the key files by typing the command:
- ```
$INSTALL/bin/keybuild mcddb
```
- This might take several minutes.
- Step 6** As a safety check, run **\$INSTALL/bin/dbcheck mcddb** (UNIX) to verify the integrity of the database. Note, you must be user **root** to run **dbcheck**.
- No errors should be detected.

## mcdshadow Command Files

The **mcdshadow** command uses the files listed in Table 14-1.

**Table 14-1** *mcdshadow Files*

File	Description
<b>mcddb.dbd</b>	Template file that describes the low-level data schema for the Raima run-time library.
<b>mcddb.k01</b> <b>mcddb.k02</b> <b>mcddb.k03</b>	Key files that contain the data that is redundant with the data files. Cisco Access Registrar does not back up these files because they can be completely rebuilt with the <b>keybuild</b> command.
<b>mcddcd.d01</b> <b>mcddcd.d02</b> <b>mcddcd.d03</b>	Data files that contain the backup.
<b>mcdConfig.txt</b>	Text file from which Cisco Access Registrar configures the initial at-install-time database.
<b>mcdschema.txt</b>	Text file that contains a version number denoting the level of the schema contained in the dbd file. Cisco Access Registrar will not attempt to open the database unless the number in this file matches a constant that is hard-coded in the libraries. If the result of the <b>mcdshadow</b> command (which uses copies of the data files) is divorced from its original <b>mcdschema.txt</b> , you will not be able to run Cisco Access Registrar.
<b>vista.taf</b> <b>vista.tcf</b> <b>vista.tjf</b>	Working files used by the Raima run-time library to ensure transactional integrity.



## Using the REX Accounting Script

---

This chapter describes how to use the REX Accounting script. The REX Accounting Script (RAS) writes RADIUS Accounting requests to a local, flat file and is included as an option for Cisco Access Registrar. It is designed to be attached to a Cisco Access Registrar IncomingScript or OutgoingScript point. When used in conjunction with the Cisco 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, Cisco recommends that you use service grouping to log and proxy accounting packets.

---

RAS can be attached to more than one Cisco Access Registrar extension point. For example, in a dial-up resale scenario, you might configure Cisco Access Registrar to proxy Accounting requests to many different Remote Servers (by realm). For some subset of those, you may 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 Cisco Access Registrar to drop all Accounting-Requests with an **Acct-Status-Type** of **Accounting-On** or **Accounting-Off**.

---

## Building and Installing the REX Accounting Script

To build and install RAS you must do the following:

- 
- 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 Cisco Access Registrar software was installed, and then choose a compiler (`gcc` or `SUNPro CC`).
  - Step 3** From the command line prompt, type:  
`host% make`
  - Step 4** Login as user `root`.
  - Step 5** From the command line prompt, type:  
`host# make install`

# Configuring the Rex Accounting Script

To configure RAS, do the following:

**Step 1** Start the Cisco Access Registrar **aregcmd** configuration utility and login:

```
> $INSTALL/usrbin/aregcmd -C localhost -N admin -P aicuser
Access Registrar Configuration Utility Version 1.3

Copyright (C) 1995-1998 by American Internet Corporation, and 1998-1999 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 Cisco Access Registrar Script object:

```
--> cd /Radius/Scripts
[ //localhost/Radius/Scripts ]
    Entries 1 to 20 from 39 total entries
    Current filter: <all>
    ACMEOutgoingScript/
    AscendIncomingScript/

<... other output deleted...>

--> add LocalAccounting
Added LocalAccounting
```

**Step 3** Using **aregcmd**, fill in the details of the new Cisco Access Registrar Script object. See Chapter 3, “Access Registrar Server Objects,” 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 initemptypoint InitRexAccountingScript
Set InitEntryPoint InitRexAccountingScript

--> set initemptypointargs "-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 Cisco Access Registrar Script object to the appropriate Cisco Access Registrar Scripting point. See Chapter 3, "Access Registrar Server Objects," 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 15-1.

**Table 15-1 REX Accounting Script Supported Options**

Option	Description
<b>-f</b> <filename>	Required. Specify the name of the output file.
<b>-t</b> <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 <b>-i</b> option.
<b>-i</b> <seconds>	Specify the number of seconds between rolling the output file, beginning at start-up. This option can not be used with the <b>-t</b> option.

Table 15-1 REX Accounting Script Supported Options (continued)

Option	Description
<b>-s</b> <size>[k mlg]	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> may 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.
<b>-g</b>	Use GMT when writing the date/time in the Accounting output file for each record (default is local time).
<b>-G</b>	Use GMT when naming rolled output files (default is local time).
<b>-A</b>	Process all packets, not just Accounting-Requests.
<b>-I</b>	Ignore errors when processing packets, always return successfully.
<b>-a</b> <buffer-count>	Pre-allocate this many Accounting buffers to improve performance.
<b>-T</b> <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.)
<b>-O</b> <script-description>	Call another REX extension before calling the <b>RexAcctScript</b> .
<b>-o</b> <script-description>	Call another REX extension after calling the <b>RexAcctScript</b> .

## Example Script Object

This is an example of what a Cisco Access Registrar Script object using RAS might look like when viewed in the Cisco 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.



## Logging Syslog Messages

---

Logging messages via syslog provides centralized error reporting for Cisco Access Registrar. Local logging and syslog logging can be turned on or off at any time by modifying the control flags in the `$INSTALLPATH/conf/aic.conf` file.

Logging syslog messages requires a UNIX host running a *syslog daemon* as a receiver for Cisco Access Registrar messages. Cisco Access Registrar and the syslog daemon can be running on the same host or different hosts.

### syslog Messages

Messages that are sent to the following logs will be forwarded to syslog server in a slightly different format. The logs are:

- `aregcmd_log`
- `config_mcd_[1..n]_log`
- `name_radius_[1..n]_log`
- `agent_server_[1..n]_log`

The messages are in the following format:

```
MMM DD hh:mm:ss hostname %CAR-[severity]-[mnemonic]: [#n], [System|Server]:  
message_description
```

Where:

**MMM DD** is the month and date that the message is received by the syslog server

**hh:mm:ss** is the arrival time of the message

**hostname** is the name of the syslog server

**severity** is one of the following levels:

- 0 - emergency
- 1 - alert
- 2 - critical
- 3 - error
- 4 - warning
- 5 - notification
- 6 - informational

7 - debugging

**mnemonic** can be *aregcmd*, *name\_radius*, *agent\_server* and *config\_mcd* for the identification of AR subsystems that the message relates to.

**#n** is the id for the components: *name\_radius*, *agent\_server*, and *config\_mcd*

**message\_description** provides detailed information of the message

## Example 1

```
May 19 14:28:44 dwlau-ultra2.cisco.com
%CAR-3-name_radius: #1, System: Remote LDAP Server.Unable to bind.
```

## Example 2

```
May 19 14:28:45 dwlau-ultra2.cisco.com
%CAR-6-name_radius: #1, Server: Stopping server
```

# Configuring Message Logging

Message logging is on by default, and all logs are stored in the **\$INSTALL/logs** directory. To turn logging off, or to change the location where logs are stored, you must modify the **\$INSTALLPATH/conf/aic.conf** file.

In **\$INSTALLPATH/conf/aic.conf** file, the following lines control logging.

```
LOCAL_LOGGING [ON|OFF]
LOGDIR full_path
DATADIR full_path
SYSLOG_LOGGING [ON|OFF]
SERVER_IP_ADDRESS [ip_address]
FACILITY_LOCAL_NUMBER [0..7]
```

Where:

**LOCAL\_LOGGING** enables (ON) or disables (OFF) the local logging function. (Local logging is on by default.)

**LOGDIR** specifies a full pathname to a different local log directory.

**DATADIR** specifies a full pathname to a different data directory.

**SYSLOG\_LOGGING** enables (ON) or disables (OFF) the syslog logging function. (syslog logging is on by default.)

**SERVER\_IP\_ADDRESS** specifies the IP address of the host to which AR will send syslog messages.

**FACILITY\_LOCAL\_NUMBER** specifies the facility being used by the syslogd.

The following is an example

```
LOCAL_LOGGING OFF
SYSLOG_LOGGING ON
```

```
SERVER_IP_ADDRESS 209.165.200.224
FACILITY_LOCAL_NUMBER 7
```

**Note**

You must first stop the CAR server prior to changing the **aic.conf** file, then restart the server. If you change the directory location where logs or database data are stored, you should also copy all log files or data files to that same directory before restarting the CAR server.

## Changing Log Directory

You can change the directory where local log messages are stored by adding the following line in the **\$INSTALLPATH/conf/aic.conf** file.

```
LOGDIR full_path
```

Where *full\_path* is a full path to the directory where you want to store the log messages. For example, to store all system logs in **/var/log/AICar1**, add the following line in the **\$INSTALLPATH/conf/aic.conf** file:

```
LOGDIR /var/log/AICar1
```

You must first stop the CAR server prior to changing the **aic.conf** file. After changing the **aic.conf** file, copy all existing log files to the new directory, then restart the server.

**Note**

Specifying a path for local logging does not affect the storage location of syslog messages.

## Configuring syslog Daemon (syslogd)

You must specify the facility from which *syslogd* will receive messages and the file into which the messages will be deposited.

In the syslog server's **/etc/syslog.conf** file, the following line may be needed.

```
localn.info <tab> <tab> <tab> /var/log/filename.log
```

**Note**

Use at least one <tab> as a field separator.

Where:

**localn**—is the facility being used for *syslogd*; **n** must be a value from 0-7 and match the **FACILITY\_LOCAL\_NUMBER** used in AR's **aic.conf** file.

**/var/log/**—is the path to the file that stores **syslogd** messages.

**filename.log**—is the file that stores **syslogd** messages. You may give this file a name of your choice.

### Creating a Log File

To create a syslog log file, complete the following steps:

- 
- Step 1** Log in as user *root*.
- Step 2** Enter the following command, where *filename.log* is a name you choose.
- ```
touch filename.log
```
- Step 3** Change permissions on the syslog log file by entering the following:
- ```
chmod 664 filename.log
```
- 

### Restarting syslogd

To restart the **syslog** daemon, log in as user *root* and enter the following commands:

```
/etc/init.d/syslog stop
/etc/init.d/syslog start
```

## Managing the Syslog File

Left unmanaged, the **syslog** file will grow in size over time and eventually fill all available disk space in its partition. Cisco recommends that you use the **cron** program to manage the syslog files.

The following example **crontab** file performs a weekly archival of the existing **syslog** file (named **ar\_syslog.log** in this example). This scheme keeps the previous two week's worth of **syslog** files.

```
#
# At 02:01am on Sundays:
# Move a weeks worth of 'ar_syslog.log' log messages to 'ar_syslog.log.1'.
# If there was a 'ar_syslog.log.1' move it to 'ar_syslog.log.2'.
# If there was a 'ar_syslog.log.2' then it is lost.
01 02 * * 0 cd /var/log;
if [ -f ar_syslog.log ];
then if [ -f ar_syslog.log.1 ];
then /bin/mv ar_syslog.log.1 ar_syslog.log.2;
fi;
/usr/bin/cp ar_syslog.log ar_syslog.log.1;
>ar_syslog.log;
fi
```




---

**Note** Consider using move (**mv**) or copy (**cp**) commands to store the previous week's syslog files in a different disk partition to reserve space for the current syslog file.

---

To add this **crontab** segment to the existing **cron** facility in **/usr/spool/cron/crontabs** directory, complete the following steps at the syslog server console.

- 
- Step 1** Log in as user *root*.
- Step 2** Enter the following commands:

```
crontab -e
```

## Server Up/Down Status Change Logging

Cisco Access Registrar supports RADIUS server up/down detection and logging. The information messages are saved in the `$INSTALL/logs/name_radius_1_log` file where `$INSTALL` is the Cisco Access Registrar installation directory. Each message consists of a header and a message description.

### Header Formats

The format of a header entry is:

```
mm/dd/yyyy HH:MM:SS name/radius/n Error Server 0
```

### Example Log Messages

Following are the descriptions and types of messages that can be found within the `<AR_install_dir>/logs/name_radius_1_log` file.

1. Cisco Access Registrar detects a Remote Server when it responds for the first time or after it is reentered into Cisco Access Registrar's server pool for retry. The format of the message is:

```
Remote Server <hostname> (<ipaddress>:<port>) is UP!
```

The following is an example header and message:

```
09/14/1999 17:56:32 name/radius/1 Error Server 0  
Remote Server dave-ultra (171.69.237.99:1645) is UP!
```

Cisco Access Registrar detects the Remote Server is not responding to its request. The format of the message is:

```
Remote Server <hostname> (<ipaddress>:<port>) is DOWN!
```

The following is an example header and message:

```
09/14/1999 17:57:12 name/radius/1 Error Server 0 Remote  
server dave-ultra (171.69.237.99:1645) is DOWN!
```

2. Cisco Access Registrar receives no response from the Remote Server after the server is reentered into Cisco Access Registrar's server pool for retry. The format of the message is:

```
Remote Server <hostname> (<ipaddress>:<port>) remains DOWN!
```

The following is an example header and message:

```
09/14/1999 17:56:32 name/radius/1 Error Server 0 Remote  
server dave-ultra (171.69.237.99:1645) remains DOWN!
```

3. The Remote Server is responding to the first retry but not the initial request. The format of the message is:

```
Remote Server <hostname> (<ipaddress>:<port>) is UP but slow!
```

The following is an example header and message:

```
09/14/1999 17:56:32 name/radius/1 Error Server 0 Remote
server dave-ultra (171.69.237.99:1645) is UP but slow!
```

4. The Remote Server is responding to the second retry request but not the initial request or the first retry request. The format of the message is:

Remote Server <hostname> (<ipaddress>:<port>) is UP but very slow!

The following is an example header and message:

```
09/14/1999 17:56:32 name/radius/1 Error Server 0 Remote
server dave-ultra (171.69.237.99:1645) is UP but very slow!
```

5. The Remote Server has been marked inactive and is being put back into Cisco Access Registrar's server pool for later use. The format of the message is:

Remote Server <hostname> (<ipaddress>:<port>) is being reactivated for later use.

The following is an example header and message:

```
09/14/1999 17:56:32 name/radius/1 Error Server 0 Remote
server dave-ultra (209.165.200.224:1645) is being reactivated for later use.
```



## Troubleshooting Cisco Access Registrar

This chapter provides information about techniques used when troubleshooting Cisco Access Registrar and highlights common problems.

### Gathering Basic Information

Table 17-1 lists UNIX commands that provide basic and essential information to help you understand the CAR installation environment.

**Table 17-1 UNIX Commands to Gather Information**

UNIX Command	Information Returned
<code>/usr/bin/uname -r</code>	Solaris release level
<code>/usr/bin/uname -i</code>	Machine hardware name
<code>/usr/bin/uname -v</code>	Solaris version
<code>/usr/bin/uname -a</code>	All system information including hostname, operating system type and release, machine model and type
<code>/usr/sbin/prtconf</code>	System configuration information including memory capacity, machine type, and peripheral equipment
<code>/usr/sbin/df -k</code>	File system disk space usage including partitions, capacity, and space used
<code>/usr/bin/ps -ef</code>	Currently running processes
<code>/usr/sbin/psinfo -v</code>	Information about processors
<code>/usr/bin/pkginfo -l AICar1</code>	Software package information about CAR version number and installation directory



**Note**

More information about these commands and their options is available using the **man** command in a terminal window on the Sun workstation.

# Troubleshooting Quick Checks

Many of the most common problems can be diagnosed by doing the following:

- Check disk space
- Check for resource conflicts
- Check the CAR log files

## Disk Space

Running out of disk space can cause a number of problems including:

- Failure to process RADIUS requests
- Parts of the CAR configuration *disappearing* in **aregcmd**
- Failure to log into **aregcmd**

Check that the CAR installation partition (**\$INSTALL**) and **/tmp** are not at capacity.

## Resource Conflicts

Resource conflicts are a common reason for the Cisco Access Registrar server failing to start. The most common resource conflicts are the following:

- Cisco Network Registrar is running on the CAR server
- Another application is also using ports 1645 and 1646
- A network management application is using the Sun SNMP Agent

## No Co-Existence With Cisco Network Registrar

Cisco Network Registrar cannot coexist on a machine running CAR for this reason. You can determine if CNR is running by entering the following command line in a terminal window:

```
--> pkginfo | grep -i "network registrar"
```

## Port Conflicts

The default ports used by the CAR server are ports 1645 and 1646. You should check to determine that no other applications are listening on the same ports as CAR.

You can check to see which TCP ports are in use by entering the following command line:

```
--> netstat -aP tcp
```

You can check to see which UDP ports are in use by entering the following command line:

```
--> netstat -aP udp
```

**Note**

---

If you configure the CAR server to use ports other than the default, you will have to specifically add ports 1645 and 1646 if you want to also use those ports.

---

## Server Running Sun SNMP Agent

If you plan to use the CAR server's SNMP agent, you cannot use the Sun Microsystems SNMP agent that comes with the Solaris operating system.

## CAR Log Files

Examining the CAR log files can help you diagnose most CAR issues. By default, the CAR log files are located in `/opt/AICar1/logs`. Table 17-2 lists the CAR log files and the information stored in each log.

**Table 17-2 CAR Log Files**

Log File	Information Recorded
<code>agent_server_1_log</code>	Log of the server agent process
<code>ar-status</code>	Log of CAR stop and start using the <code>arservagt</code> utility
<code>aregcmd_log</code>	Log of commands executed in <code>aregcmd</code> (very useful for tracing the steps that took place before a problem occurred)
<code>config_mcd_1_log</code>	Log of the mcd internal database
<code>name_radius_1_log</code>	Log of the radius server process
<code>name_radius_1_trace</code>	Debugging output of RADIUS request processing (only generated when the trace level, set in <code>aregcmd</code> , is greater than zero)

## Using xtail to Monitor Log File Activity

A useful way of monitoring all of the log files is to run `xtail`, a utility provided with Cisco Access Registrar. The `xtail` program monitors one or more files and displays all data written to a file since command invocation.

Run `xtail` in a dedicated terminal window. It is very useful for monitoring multiple logfiles simultaneously, such as with a command line like the following:

```
> xtail $INSTALL/logs/*
```

## Modifying the Trace Level

By modifying the trace level, you can gather more detailed information in the log files about what is happening in the Cisco Access Registrar server. There are five different trace levels. Each higher trace level also includes the information logged using lower trace levels. The different trace levels provide the following information:

- Level 0—No tracing occurs
- Level 1—Indicates when a packet is sent or received and when a status change occurs in a remote server (RADIUS Proxy and LDAP)
- Level 2—Information includes the following:
  - Which services and session managers are used to process

- Which client and vendor objects are being used to process a packet
- More details about remote servers (RADIUS Proxy and LDAP), packet transmission, and time-outs
- Details about poorly-formed packets.
- Level 3—Information includes the following:
  - Tracing of errors in Tel scripts when referencing invalid RADIUS attributes
  - Which scripts have been run
  - Details about local userlist processing
- Level 4—Information includes the following:
  - Advanced duplication detection processing
  - Details about creating, updating, and deleting sessions
  - Tracing of all APIs called during the running of a script
- Level 5—Provides information about policy engine operations

## Installation and Server Process Start-up

The installation process installs the CAR software to the specified installation directory and then starts the server processes. This process rarely fails but the following checks should always be performed:

- Ensure that there is an **installation success message** at the end of the **pkgadd** dialog, otherwise check the dialog for the problem
- Follow the installation instructions carefully especially when performing an upgrade. For example, when upgrading to 1.6R1, 1.6R2, or 1.6R3, a post-installation upgrade script needs to be run
- Pay attention to the information included in README files

At the end of a successful installation, **arstatus** should show the following four server processes:

```
> $INSTALL/usrbin/arstatus
AR RADIUS server running      (pid: 6285)
AR MCD lock manager running   (pid: 6284)
AR MCD server running         (pid: 6283)
AR Server Agent running       (pid: 6277)
```

If any of the above processes are not displayed, check the log file of the failed process to determine the reason. The MCD processes may fail to start if Cisco Network Registrar is installed on the same machine.

The manual method of starting and stopping the CAR processes is using the **arservagt** utility.

To start CAR processes: **arservagt start**

To stop CAR processes: **arservagt stop**

To restart CAR processes: **arservagt restart**

## aregcmd and CAR Configuration

While troubleshooting, you should always use the **aregcmd** command **trace** to turn on tracing. With tracing active, CAR generates debugging output to the log file **name\_radius\_1\_trace**. The syntax is:

```
trace [<server>] [<level>]
```

When you do not specify a server, CAR sets the trace level for all servers in the current cluster. When you do not specify a trace level, the currently set level is used. The default trace level is 0.

## Running and Stopped States

CAR can be in two states, running or stopped. In either state, all four CAR processes remain running. The state of CAR will be displayed when logging into **aregcmd** or by using the **aregcmd status** command:

```
--> status
```

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

The **start** and **stop** commands allow CAR to move between states. **Reload** is equivalent to a **stop** followed by a **start** if CAR is already running, and just a **start** if it is already stopped.

```
--> stop
```

```
Stopping Server 'Radius'...
Server 'Radius' is Stopped
```

```
--> start
```

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

```
--> reload
```

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

During the transition from running to stopped, CAR stops processing new RADIUS requests and releases resources such as memory, network and database connections and open files.

During the transition from stopped to running, CAR reverses this process by opening a connection with its internal database, reading configuration data, claiming memory, establishing network connections, opening files, and initializing scripts. During this transition, problems can occur. CAR may fail to start and display the following:

```
--> reload
```

```
Reloading Server 'Radius'...
310 Command failed
```

CAR failed to move from stopped state to running:

```
--> status
```

```
Server 'Radius' is Stopped
```

This may occur for a number of reasons including the following:

- An invalid configuration
- Insufficient memory
- Listening ports already in use by another application
- Unable to open files
- Unable to initialize scripts

Check the **name\_radius\_1\_log** file for the one of these indications.

## RADIUS Request Processing

The main technique for troubleshooting RADIUS request processing in CAR is to examine the `name_radius_1_trace` log file with the trace level set to 5. Most issues are fairly self-explanatory. Some issues that can arise are:

- CAR has marked a remote server as *down*
- A resource manager has run out of resources (for example, user or group session limit has been reached or no more IP addresses are available)
- A configuration error (such as an accounting service not being set)
- A run time error in a script

Some issues are not immediately evident from the log files though, such as the following:

- Failure to save or reload CAR after a configuration change
- CAR is not listening on the correct UDP ports for RADIUS requests

## Other Troubleshooting Techniques and Resources

### aregcmd Stats Command

The `aregcmd` command `stats` provides statistics on request processing.

--> `stats`

```
Global Statistics for Radius:
serverStartTime = Tue Oct  2 10:28:02 2001
serverResetTime = Tue Oct  2 20:25:12 2001
serverState = Running
totalPacketsInPool = 1024
totalPacketsReceived = 0
totalPacketsSent = 0
totalRequests = 0
totalResponses = 0
totalAccessRequests = 0
totalAccessAccepts = 0
totalAccessChallenges = 0
totalAccessRejects = 0
totalAccessResponses = 0
totalAccountingRequests = 0
totalAccountingResponses = 0
totalStatusServerRequests = 0
totalAscendIPAAllocateRequests = 0
totalAscendIPAAllocateResponses = 0
totalAscendIPAReleaseRequests = 0
totalAscendIPAReleaseResponses = 0
totalUSRNASRebootRequests = 0
totalUSRNASRebootResponses = 0
totalUSRResourceFreeRequests = 0
totalUSRResourceFreeResponses = 0
totalUSRQueryResourceRequests = 0
totalUSRQueryResourceResponses = 0
totalUSRQueryReclaimRequests = 0
totalUSRQueryReclaimResponses = 0
totalPacketsInUse = 0
totalPacketsDrained = 0
```

```
totalPacketsDropped = 0
totalPayloadDecryptionFailures = 0
```

## Core Files

A core file in the CAR installation directory is an indication that CAR has crashed and restarted. Check that the radius server process generated the core file using the UNIX **file** command:

```
> file core
```

```
core: ELF 32-bit MSB core file SPARC Version 1, from 'radius'
```

Check the timestamp on the core file and look for corresponding log messages in the **name\_radius\_1\_log** file in **\$INSTALL/logs**. The word *assertion* commonly appears in core messages. Try to establish what caused the problem and contact Cisco TAC.

## radclient

The CAR package provides a utility called **radclient** that allows RADIUS requests to be generated. Use **radclient** to test configurations and troubleshoot problems.

## CAR Replication

For information about troubleshooting CAR replication refer to Chapter 6 of the Cisco Access Registrar Concepts and Reference Guide, Understanding Replication, ([http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/cnsar/1\\_7/referenc/repl\\_ref.htm](http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/cnsar/1_7/referenc/repl_ref.htm)).

For more information about using CAR replication, refer to Chapter 7, “Using Replication.”





## Cisco Access Registrar Tcl and REX Dictionaries

---

This appendix describes the Tcl and REX dictionaries that are used when writing Incoming or Outgoing scripts.

A dictionary is a data structure that contains key/value pairs. Two types of dictionaries exist: the Attribute dictionaries (used by the Request and Response dictionaries), and the Environment dictionary.

This section contains the dictionaries you reference when writing a Tcl script and the dictionaries you reference when you write a script using the shared libraries (REX—RADIUS EXtension).

### Tcl Attribute Dictionaries

An *Attribute dictionary* is a dictionary in which the keys are constrained to be the names of attributes as defined in the Cisco Access Registrar server configuration, and the values are the string representation of the legal values for that particular attribute. For example, IP addresses are specified by the dotted-decimal string representation of the address, and enumerated values are specified by the name of the enumeration. This means numbers are specified by the string representation of the number.

Attribute dictionaries have the unusual feature that there can be more than one instance of a particular key in the dictionary. These instances are ordered, with the first instance at index zero. Some of the methods of an Attribute dictionary allow an index to be specified to indicate a particular instance or position in the list of instances to be referenced.

### Attribute Dictionary Methods

Attribute dictionaries use active commands, called *methods*, that allow you to change and access the values in the dictionaries. Table A-1 lists all of the methods you can use with the Request and Response dictionaries.

Table A-1 Tcl Attribute Dictionary Methods

Name	Syntax	Description
<b>addProfile</b>	<b>\$dict addProfile</b> <profile> [<mode>]	Copies all of the attributes in the profile <profile> into the dictionary. Note, <profile> must be the name of one of the profiles listed in the server configuration. When <mode> is not provided or when <mode> equals the special value <b>REPLACE</b> , any duplicate instances of the attributes in the dictionary are replaced with the attribute from <profile>. When <mode> is provided and equals the special value <b>APPEND</b> , new instances of the attributes are appended to the attributes already in the dictionary. When <mode> is provided and equals the special value <b>AUGMENT</b> , only add the attribute when it does not already exist.
<b>clear</b>	<b>\$dict clear</b>	Removes all entries from the dictionary.
<b>containsKey</b>	<b>\$dict containsKey</b> <attribute>	Returns 1 when the dictionary contains the attribute <attribute>, otherwise returns 0.
<b>firstKey</b>	<b>\$dict firstKey</b>	Returns the name of the first attribute in the dictionary. Note, the attributes are not stored sorted by name.
<b>get</b>	<b>\$dict get</b> <attribute> [<index> [bMore]]	Returns the value of the <attribute> attribute from the dictionary, represented as a string. When the dictionary does not contain the <attribute>, an empty string is returned.  When <index> is provided, return the <index>'th instance of the attribute. Some attributes can appear more than once in the request (or response) packet. The <index> argument is used to select which instance to return.  When <b>bMore</b> is provided, the <b>get</b> method sets <b>bMore</b> to 1 when more attributes exist after the one returned, and to 0 otherwise. You can use this to determine whether another call to <b>get</b> should be made to retrieve other instances of the attribute.
<b>isEmpty</b>	<b>\$dict isEmpty</b>	Returns 1 when the dictionary has no entries, otherwise returns 0.
<b>log</b>	<b>\$dict log</b> <level> <message> ...	Outputs a message into the RADIUS server's logging system. The <level> should be either <b>LOG_ERROR</b> , <b>LOG_WARNING</b> , or <b>LOG_INFO</b> . The remaining arguments are concatenated together and sent to the logging system at the specified level.
<b>nextKey</b>	<b>\$dict nextKey</b>	Returns the name of the next attribute in the dictionary that follows the attribute returned in the last call to <b>firstKey</b> or <b>nextKey</b> .
<b>put</b>	<b>\$dict put</b> <attribute> <value> [<index>]	Associates <value> with the attribute <attribute> in the dictionary. When <index> is not provided or when <index> equals the special value <b>REPLACE</b> , any existing instances of <attribute> are replaced with the single value. When <index> is provided and equals the special value <b>APPEND</b> , a new instance of <attribute> is appended to the end of the list of instances of the <attribute>. When <index> is provided and is a number, a new instance of <attribute> is inserted at the position indicated. When <index> is provided and equals the special value <b>AUGMENT</b> , only put the attribute when it does not already exist.

Table A-1 Tcl Attribute Dictionary Methods (continued)

Name	Syntax	Description
remove	<code>\$dict remove &lt;attribute&gt; [&lt;index&gt;]</code>	Removes the <attribute> attribute from the dictionary. When <index> is not provided or when <index> equals the special value <b>REMOVE_ALL</b> , remove any existing instances of <attribute>. When <index> is provided and is a number, remove the instance of <attribute> at the position indicated.  Always returns 1, even when the dictionary did not contain the <attribute> at that <index>.
size	<code>\$dict size</code>	Returns the number of entries in the dictionary.
trace	<code>\$dict trace &lt;level&gt; &lt;message&gt; ...</code>	Outputs a message into the packet tracing system used by the RADIUS server. At level 0, no tracing occurs. At level 1, only an indication the server received the packet and sent a reply is output. As the number gets higher, the amount of information output increases, until at level 4, where everything is traced as output. The remaining arguments are concatenated and sent to the tracing system at the specified level.

## Tcl Environment Dictionary

A dictionary is a data structure that contains key/value pairs. An Environment dictionary is a dictionary in which the keys and values are constrained to be strings. The Tcl Environment dictionary is used to communicate information from the script to the server and from script to script within the processing of a particular request. Note, there can be only one instance of a key in the Environment dictionary.

Table A-2 lists of all the methods you can use with the Request and Response dictionaries.

Table A-2 Tcl Environment Dictionary Methods

Name	Syntax	Description
clear	<code>\$dict clear</code>	Removes all entries from the dictionary.
containsKey	<code>\$dict containsKey &lt;key&gt;</code>	Returns 1 when the dictionary contains the <key> key, otherwise returns 0.
firstKey	<code>\$dict firstKey</code>	Returns the name of the first key in the dictionary. Note, the keys are not stored sorted by name.
get	<code>\$dict get &lt;key&gt;</code>	Returns the value of <key> from the dictionary. When the dictionary does not contain the <key>, an empty string is returned.
isEmpty	<code>\$dict isEmpty</code>	Returns 1 when the dictionary has no entries, otherwise returns 0.
log	<code>\$dict log &lt;level&gt; &lt;message&gt; ...</code>	Outputs a message into the logging system used by the RADIUS server. <level> should be one of <b>LOG_ERROR</b> , <b>LOG_WARNING</b> , or <b>LOG_INFO</b> . The remaining arguments are concatenated together and sent to the logging system at the specified level.
nextKey	<code>\$dict nextKey</code>	Returns the name of the next key in the dictionary that follows the key returned in the last call to <b>firstKey</b> or <b>nextKey</b> .
put	<code>\$dict put &lt;key&gt; &lt;value&gt;</code>	Associates <value> with the <key> key in the dictionary, replacing an existing instance of <key> with the new value.
remove	<code>\$dict remove &lt;key&gt;</code>	Removes the <key> key from the dictionary. Always returns 1, even when the dictionary did not contain the <key>.

Table A-2 Tcl Environment Dictionary Methods (continued)

Name	Syntax	Description
size	<code>\$dict size</code>	Returns the number of entries in the dictionary.
trace	<code>\$dict trace &lt;level&gt; &lt;message&gt;</code> ...	Outputs a message into the packet tracing system used by the RADIUS server. At level 0, no tracing occurs. At level 1, only an indication the server received the packet and sent a reply is output. As the number gets higher, the amount of information output is greater, until at level 4, where everything the server traces is output. The remaining arguments are concatenated together and sent to the tracing system at the specified level.

## REX Attribute Dictionary

A dictionary is a data structure that contains key/value pairs. An Attribute dictionary is a dictionary in which the keys are constrained to be the attributes as defined in the RADIUS server configuration and the values are constrained to be legal values for that particular attribute. Attribute dictionaries have the unusual feature that there can be more than one instance of a particular key in the dictionary. These instances are ordered, with the first instance at index 0. Some of the methods of an Attribute dictionary allow an index to be specified to indicate a particular instance or position in the list of instances to be referenced.

When writing REX scripts, you can specify keys as the string representation of the name of the attribute or by type, which is a byte sequence defining the attribute. The values can also be specified as the string representation of the value or as the byte sequence, which is the attribute. These options mean some of these access methods have four different variations that are the combinations of string or type for the key, and string or bytes for the value.

## Attribute Dictionary Methods

Attribute dictionaries use active commands, called *methods*, that allow you to change and access the values in the dictionaries.

Table A-3 lists all of the methods you can use with the Request and Response dictionaries.

Table A-3 REX Attribute Dictionary Methods

Name	Syntax	Description
<b>addProfile</b>	<b>abool_t</b> <b>pDict-&gt;addProfile(rex_AttributeDictionary_t*</b> <b>pDict, const char* &lt;pszProfile&gt;, int &lt;iMode&gt;)</b>	Copies all of the attributes in the <pszProfile> profile into the dictionary. Note, <pszProfile> must be the name of one of the profiles listed in the server configuration. When <iMode> equals the special value <b>REX_REPLACE</b> , it replaces any duplicate instances of the attributes in the dictionary with the attribute from the profile. When <iMode> equals the special value <b>REX_APPEND</b> , it appends a new instance of the attributes to any attributes already in the dictionary. When <iMode> equals the special value <b>REX_AUGMENT</b> , it only puts the attribute when does not already exist.
<b>allocateMemory</b>	<b>void*</b> <b>pDict-&gt;allocateMemory(rex_AttributeDictionary_t*</b> <b>y_t* pDict, unsigned int &lt;iSize&gt;)</b>	Allocates memory for use in scripts that persist only for the lifetime of this request. This memory is released when processing for this request is complete.
<b>clear</b>	<b>void pDict-&gt;clear(rex_AttributeDictionary_t*</b> <b>pDict)</b>	Removes all entries from the dictionary.
<b>containsKey</b>	<b>abool_t</b> <b>pDict-&gt;containsKey(rex_AttributeDictionary_t*</b> <b>pDict, const char* &lt;pszAttribute&gt;)</b>	Returns TRUE when the dictionary contains <pszAttribute>, otherwise returns FALSE.
<b>containsKeyByType</b>	<b>abool_t</b> <b>pDict-&gt;containsKeyByType(rex_AttributeDictionary_t*</b> <b>pDict, const abytes_t* &lt;pAttribute&gt;)</b>	Returns TRUE when the dictionary contains <pAttribute>, otherwise returns FALSE.
<b>firstKey</b>	<b>const char*</b> <b>pDict-&gt;firstKey(rex_AttributeDictionary_t*</b> <b>pDict)</b>	Returns the name of the first attribute in the dictionary. Note, the attributes are not stored sorted by name.
<b>firstKeyByType</b>	<b>const abytes_t* pDict-&gt;firstKeyByType</b> <b>(rex_AttributeDictionary_t* pDict)</b>	Returns a pointer to the byte sequence defining the first attribute in the dictionary. Note, attributes are not stored sorted by name.
<b>get</b>	<b>const char*</b> <b>pDict-&gt;get(rex_AttributeDictionary_t* pDict,</b> <b>const char* pszAttribute, int &lt;iIndex&gt;, abool_t*</b> <b>&lt;pbMore&gt;)</b>	Returns the value of the <iIndex>'d instance of the attribute from the dictionary, represented as a string. When the dictionary does not contain the attribute (or that many instances of the attribute), an empty string is returned.  When <pbMore> is non-zero, the <b>get</b> method sets <pbMore> to TRUE when more instances of the attribute exist after the one returned, and to FALSE otherwise. This can be used to determine whether another call to <b>get</b> should be made to retrieve other instances of the attribute.

Table A-3 REX Attribute Dictionary Methods (continued)

Name	Syntax	Description
getBytes	<pre>const abytes_t* pDict-&gt;getBytes(rex_AttributeDictionary_t* pDict, const char* pszAttribute, int &lt;iIndex&gt;, abool_t* &lt;pbMore&gt;)</pre>	<p>Returns the value of the &lt;iIndex&gt;'d instance of the attribute from the dictionary, as a sequence of bytes. When the dictionary does not contain the attribute (or that many instances of the attribute), 0 is returned.</p> <p>When &lt;pbMore&gt; is non-zero, the <b>getBytes</b> method sets &lt;pbMore&gt; to TRUE when more instances of the attribute exist after the one returned, and to FALSE otherwise. This can be used to determine whether another call to <b>getBytes</b> should be made to retrieve other instances of the attribute.</p>
getBytesByType	<pre>const abytes_t* pDict-&gt;getBytesByType (rex_AttributeDictionary_t* pDict, const abytes_t* pAttribute, int &lt;iIndex&gt;, abool_t* &lt;pbMore&gt;)</pre>	<p>Returns the value of the &lt;iIndex&gt;'d instance of the attribute from the dictionary, as a sequence of bytes. When the dictionary does not contain the attribute (or that many instances of the attribute), 0 is returned instead.</p> <p>When &lt;pbMore&gt; is non-zero, sets the variable pointed to TRUE when more instances of the attribute exist after the one returned, and to FALSE otherwise. This can be used to determine whether another call to <b>get</b> should be made to retrieve other instances of the attribute.</p>
getByType	<pre>const char* pDict-&gt;getByType(rex_AttributeDictionary_t* pDict, const abytes_t* &lt;pszAttribute&gt;, int &lt;iIndex&gt;, abool_t* &lt;pbMore&gt;)</pre>	<p>Returns the value of the &lt;iIndex&gt;'d instance of the attribute from the dictionary, as represented as a string. When the dictionary does not contain the attribute (or that many instances of the attribute), returns an empty string.</p> <p>When &lt;pbMore&gt; is non-zero, the <b>getByType</b> method sets &lt;pbMore&gt; to TRUE when more instances of the attribute exist after the one returned, and to FALSE otherwise. This can be used to determine whether another call to <b>getByType</b> should be made to retrieve other instances of the attribute.</p>
getType	<pre>const char* pDict-&gt;getType(rex_AttributeDictionary_t* pDict, const abytes_t* &lt;pAttribute&gt;)</pre>	<p>Returns a pointer to the byte sequence defining the attribute, when the attribute name matches a configured attribute, zero otherwise.</p>
isEmpty	<pre>abool_t pDict-&gt;isEmpty(rex_AttributeDictionary_t* pDict)</pre>	<p>Returns TRUE when the dictionary has 0 entries, FALSE otherwise.</p>

Table A-3 REX Attribute Dictionary Methods (continued)

Name	Syntax	Description
log	<code>abool_t pDict-&gt;log(rex_AttributeDictionary_t* pDict, int &lt;iLevel&gt;, const char* &lt;pszFormat&gt;, ...)</code>	Outputs a message into the logging system used by the RADIUS server. <iLevel> should be one of <b>REX_LOG_ERROR</b> , <b>REX_LOG_WARNING</b> , or <b>REX_LOG_INFO</b> . The <b>pszFormat</b> argument is treated as a <b>printf</b> -style format string, and it, along with the remaining arguments, are formatted and sent to the logging system at the specified level.
nextKey	<code>const char* pDict-&gt;nextKey(rex_AttributeDictionary_t* pDict)</code>	Returns the name of the <i>next</i> attribute in the dictionary that follows the attribute returned in the last call to <b>firstKey</b> or <b>nextKey</b> .
nextKeyByType	<code>const abytes_t* pDict-&gt;nextKeyByType(rex_AttributeDictionary_t* pDict)</code>	Returns a pointer to the byte sequence defining the next attribute in the dictionary that follows the attribute returned in the last call to <b>firstKeyByType</b> or <b>nextKeyByType</b> .
put	<code>abool_t pDict-&gt;put(rex_AttributeDictionary_t* pDict, const char* &lt;pszAttribute&gt;, const char* &lt;pszValue&gt;, int &lt;iIndex&gt;)</code>	Converts <pszValue> to a sequence of bytes, according to the definition of <pszAttribute> in the server configuration. Associates that sequence of bytes with <pszAttribute> in the dictionary. When <iIndex> equals the special value <b>REX_REPLACE</b> , it replaces any existing instances of <pszAttribute> with a single value. When <iIndex> equals the special value <b>REX_APPEND</b> , it appends a new instance of <pszAttribute> to the end of the list of existing instances of <pszAttribute>. Otherwise, a new instance of <pszAttribute> is inserted at the position indicated. This method returns TRUE unless <pszAttribute> does not match any configured attributes or the value could not be converted to a legal value. When <iIndex> equals the special value <b>REX_AUGMENT</b> , only <b>put</b> <pszAttribute> when it does not already exist.

Table A-3 REX Attribute Dictionary Methods (continued)

Name	Syntax	Description
putBytes	<pre> abool_t pDict-&gt;putBytes(rex_AttributeDictionary_t* pDict, const char* &lt;pszAttribute&gt;, const abytes_t* &lt;pValue&gt;, int &lt;iIndex&gt;) </pre>	<p>Associates &lt;pValue&gt; with the attribute &lt;pszAttribute&gt; in the dictionary. When &lt;iIndex&gt; equals the special value <b>REX_REPLACE</b>, it replaces any existing instances of the &lt;pszAttribute&gt; with a single new value. When &lt;iIndex&gt; equals the special value <b>REX_APPEND</b>, it appends a new instance of &lt;pszAttribute&gt; to the end of the list of existing instances of &lt;pszAttribute&gt;. When &lt;iIndex&gt; equals the special value <b>REX_AUGMENT</b>, only put the &lt;pszAttribute&gt; when it does not already exist. Otherwise, a new instance of &lt;pszAttribute&gt; is inserted at the position indicated.</p> <p>This method returns TRUE unless the attribute name does not match any configured attributes.</p>
putBytesByType	<pre> abool_t pDict-&gt;putBytesByType(rex_AttributeDictionary_t* pDict, const abytes_t* &lt;pAttribute&gt;, const abytes_t* &lt;pValue&gt;, int &lt;iIndex&gt;) </pre>	<p>Associates &lt;pValue&gt; with the attribute &lt;pAttribute&gt; in the dictionary. When &lt;iIndex&gt; equals the special value <b>REX_REPLACE</b>, it replaces any existing instances of &lt;pAttribute&gt; with the new value. When &lt;iIndex&gt; equals the special value <b>REX_APPEND</b>, it appends a new instance of &lt;pAttribute&gt; to the end of the list of existing instances of &lt;pAttribute&gt;. When &lt;iIndex&gt; equals the special value <b>REX_AUGMENT</b>, only put &lt;pAttribute&gt; when it does not already exist. Otherwise, insert a new instance of &lt;pAttribute&gt; at the position indicated.</p> <p>This method returns TRUE unless the attribute name does not match any configured attributes.</p>
putByType	<pre> abool_t pDict-&gt;putByType(rex_AttributeDictionary_t* pDict, const abytes_t* &lt;pszAttribute&gt;, const char* &lt;pszValue&gt;, int &lt;iIndex&gt;) </pre>	<p>Converts &lt;pszValue&gt; to a sequence of bytes, according to the definition of &lt;pszAttribute&gt; in the server configuration. Associates that sequence of bytes with &lt;pszAttribute&gt; in the dictionary. When &lt;iIndex&gt; equals the special value <b>REX_REPLACE</b>, it replaces any existing instances of &lt;pszAttribute&gt; with a single new value. When &lt;iIndex&gt; equals the special value <b>REX_APPEND</b>, it appends a new instance of &lt;pszAttribute&gt; to the end of the list of existing instances of &lt;pszAttribute&gt;. Otherwise, it inserts a new instance of &lt;pszAttribute&gt; at the position indicated. This method returns TRUE unless &lt;pszAttribute&gt; does not match any configured attributes, or the value could not be converted to a legal value.</p>

Table A-3 REX Attribute Dictionary Methods (continued)

Name	Syntax	Description
remove	<pre>abool_t pDict-&gt;remove(rex_AttributeDictionary_t* pDict, const char* &lt;pszAttribute&gt;, int &lt;iIndex&gt;)</pre>	Removes the <pszAttribute> from the dictionary. When <iIndex> equals the special value <b>REX_REMOVE_ALL</b> , removes any existing instances of <pszAttribute>. Otherwise, it removes the instance of <pszAttribute> at the position indicated. Returns TRUE, even when the dictionary did not contain <pszAttribute> at the <iIndex>, unless <pszAttribute> does not match any configured attribute.
removeByType	<pre>abool_t pDict-&gt;removeByType(rex_AttributeDictionary_ t* pDict, const abytes_t* &lt;pAttribute&gt;, int &lt;iIndex&gt;)</pre>	Removes the <pAttribute> from the dictionary. When <iIndex> equals the special value <b>REX_REMOVE_ALL</b> , it removes any existing instances of <pszAttribute>. Otherwise, the instance of <pAttribute> at the position indicated is removed. Always returns TRUE, even when the dictionary did not contain <pAttribute> at the <iIndex>.
reschedule	<pre>abool_t pDict-&gt;reschedule(rex_AttributeDictionary_t* pDict)</pre>	Enables control over asynchronous activities. It enables you to collect similar activities and mark them as pending. You can then process them and reschedule them. You can only use this attribute with multithreaded services. Use caution when employing this method.
size	<pre>int pDict-&gt;size(rex_AttributeDictionary_t* pDict)</pre>	Returns the number of entries in the dictionary.
trace	<pre>abool_t pDict-&gt;trace(rex_AttributeDictionary_t* pDict, int &lt;iLevel&gt;, const char* &lt;pszFormat&gt;, ...)</pre>	Outputs a message into the packet tracing system used by the RADIUS server. At level 0, no tracing occurs. At level 1, only an indication the packet was received and a reply was sent is output. As the number gets higher, the amount of information output is greater, until at level 4, where everything traceable is output. The remaining arguments are formatted and sent to the tracing system at the specified level.

## REX Environment Dictionary

A dictionary is a data structure that contains key/value pairs. An Environment dictionary is a dictionary in which the keys and values are constrained to be strings. The REX Environment dictionary is used to communicate information from the script to the server and from script to script within the processing of a particular request. Note, there can be only one instance of a key in the Environment dictionary.

## REX Environment Dictionary Methods

The Environment dictionary uses active commands, called *methods*, to allow you to change and access the values in the dictionary. Table A-4 lists all of the methods you can use with the REX Environment dictionary.

Table A-4 REX Environment Dictionary Methods

Name	Syntax	Description
allocateMemory	void* pDict->allocateMemory(rex_EnvironmentDictionary_t* pDict, unsigned int <iSize>)	Allocate memory for use in scripts that persist only for the lifetime of this request. This memory is released when processing for this request is complete.
clear	void pDict->clear(rex_EnvironmentDictionary_t* pDict)	Removes all entries from the dictionary.
containsKey	abool_t pDict->containsKey(rex_EnvironmentDictionary_t* pDict, const char* <pszKey>)	Returns TRUE when the dictionary contains <pszKey>, otherwise returns FALSE.
firstKey	const char* pDict->firstKey(rex_EnvironmentDictionary_t* pDict)	Returns the name of the first key in the dictionary. Note, the keys are not stored sorted by name.
get	const char* pDict->get(rex_EnvironmentDictionary_t* pDict, const char* <pszKey>)	Returns the value associated with <pszKey> from the dictionary. When the dictionary does not contain <pszKey>, an empty string is returned.
isEmpty	abool_t pDict->isEmpty(rex_EnvironmentDictionary_t* pDict)	Returns TRUE when the dictionary has 0 entries, FALSE otherwise.
log	abool_t pDict->log(rex_EnvironmentDictionary_t* pDict, int <iLevel>, const char* <pszFormat>, ...)	Outputs a message into the logging system used by the RADIUS server. <iLevel> should be one of <b>REX_LOG_ERROR</b> , <b>REX_LOG_WARNING</b> , or <b>REX_LOG_INFO</b> . The <pszFormat> argument is treated as a <b>printf</b> -style format string, and it, along with the remaining arguments, are formatted and sent to the logging system at the specified level.
nextKey	const char* pDict->nextKey(rex_EnvironmentDictionary_t* pDict)	Returns the name of the next key in the dictionary that follows the key returned in the last call to <b>firstKey</b> or <b>nextKey</b> .
put	abool_t pDict->put(rex_EnvironmentDictionary_t* pDict, const char* <pszValue>, const char* <pszKey>)	Associates the value with <pszKey> in the dictionary, replacing any existing instance of <pszKey> with the new <pszValue>.
remove	abool_t pDict->remove(rex_EnvironmentDictionary_t* pDict, const char* <pszKey>)	Removes <pszKey> and the associated value from the dictionary. Always returns TRUE, even when the dictionary did not contain <pszKey>

Table A-4 REX Environment Dictionary Methods (continued)

Name	Syntax	Description
reschedule	<b>abool_t</b> <b>pDict-&gt;reschedule(rex_AttributeDictionary_t*</b> <b>pDict)</b>	Enables control over asynchronous activities. It enables you to collect similar activities and mark them as pending. You can then process them and reschedule them. You can only use this attribute with multithreaded services. Use caution when employing this method.
size	<b>int pDict-&gt;size(rex_EnvironmentDictionary_t*</b> <b>pDict)</b>	Returns the number of entries in the dictionary.
trace	<b>abool_t</b> <b>pDict-&gt;trace(rex_EnvironmentDictionary_t* pDict,</b> <b>int &lt;iLevel&gt;, const char* &lt;pszFormat&gt;, ...)</b>	Outputs a message into the packet tracing system used by the RADIUS server. At level 0, no tracing occurs. At level 1, only an indication the packet was received and a reply was sent is output. As the number gets higher, the amount of information output is greater, until at level 4, where everything traceable is output. The remaining arguments are formatted and sent to the tracing system at the specified level.





## Environment Dictionary

---

This appendix describes the environment variables the scripts use to communicate with Cisco Access Registrar or to communicate with other scripts.

Cisco Access Registrar sets the **arguments** variable in the Environment dictionary, before calling the **InitEntryPoint** of each script. The **arguments** variable is set to the value of the **InitEntryPointArgs** property corresponding to that script, and it allows the administrator to pass (possibly unique) information to each script initialization function.

Environment variables that are set and read for resource management override provide scripts further control over session management. These environment variables, including the following **Acquire-User-Session-Limit**, **Acquire-Group-Session-Limit**, **Acquire-IP-Dynamic**, **Acquire-IP-Per-NAS-Port**, **Acquire-IPX-Dynamic**, and **Acquire-USR-VPN**, can be set at any point before session management is invoked. These environment variables are read as the packet flows through each Resource Manager that the chosen Session Manager calls. The default setting for these environment variables is TRUE. See the “Resource Managers” section on page 3-15 for additional information about Resource Managers.

## Cisco Access Registrar Environment Dictionary Variables

The following variables are text strings stored in the Environment dictionary passed to each scripting point.

### Accepted-Profiles

**Accepted-Profiles** is read during authorization after calling server and client incoming scripts (not set by Cisco AR code). If set, the authorization done by local user lists checks to see if the given user's profile as specified in the user record is one of those in the separated list of profiles. If it is not in the separated list of profiles, the request is rejected.

### Accounting-Service

**Accounting-Service** is set after calling server and client incoming scripts and is used to determine which accounting service is used for this request. If set, the server directs the request to be processed by the specified accounting service.

When **Accounting-Service** is not set, the **DefaultAccountingService** (as defined in the server configuration) is used instead.

## Acquire-Group-Session-Limit

**Acquire-Group-Session-Limit** is set and read for resource management override.

**Acquire-Group-Session-Limit** is set to FALSE to override the use of group session limit resource management.

## Acquire-IP-Dynamic

**Acquire-IP-Dynamic** is set and read for resource management override. **Acquire-IP-Dynamic** is set to FALSE to override the use of a managed pool of IP addresses resource management.

## Acquire-IPX-Dynamic

**Acquire-IPX-Dynamic** is set and read for resource management override. **Acquire-IPX-Dynamic** is set to FALSE to override the use of a managed pool of IPX addresses resource management.

## Acquire-IP-Per-NAS-Port

**Acquire-IP-Per-NAS-Port** is set and read for resource management override.

**Acquire-IP-Per-NAS-Port** is set to FALSE to override the use of ports associated with specific IP addresses resource management.

## Acquire-Subnet-Dynamic

**Acquire-Subnet-Dynamic** is not always used. If set to FALSE, subnet-dynamic resource managers are skipped.

## Acquire-User-Session-Limit

**Acquire-User-Session-Limit** set and read for resource management override.

**Acquire-User-Session-Limit** is set to FALSE to override the use of user session limit resource management.

## Acquire-USR-VPN

**Acquire-USR-VPN** is set and read for resource management override. **Acquire-USR-VPN** is set to FALSE to override the use of Virtual Private Networks (VPNs) that use USR NAS Clients resource management.

## Allow-Null-Password

**Allow-Null-Password** is read during password matching and set in local userlist password matching if not set prior. If **Allow-Null-Password** is set to TRUE, the Cisco AR server accepts requests with null passwords.

## Authentication-Service

**Authentication-Service** is set and read for authentication service selection and is used to determine which service is used to authenticate the user. If set, the server directs the request to be processed by the specified authentication service. When **Authentication-Service** is not set, the **DefaultAuthenticationService** is used instead.

## Authorization-Service

**Authorization-Service** is set and read for authorization service selection and is used to determine which service to use to authorize the user. If set, the server directs the request to be processed by the specified authorization service. When **Authorization-Service** is not set, the **DefaultAuthorizationService** is used instead.

## Current-Group-Count

**Current-Group-Count** is set and read for group session management. If set, the group-session-limit resource manager sets **Current-Group-Count** to be the new value of the group-session-limit counter.

## Dynamic-Search-Path

**Dynamic-Search-Path** is read for LDAP searching. If set, the server uses it as its LDAP search path rather than the value set in the remote server configuration.

## Group-Session-Limit

**Group-Session-Limit** is set and read for group session management. The group-session-limit resource manager sets this environment variable to be the limit of the group-session-limit counter as set by the configuration.

## Ignore-Accounting-Signature

**Ignore-Accounting-Signature** is set after calling server and client incoming scripts and is used to ignore missing or incorrect accounting signatures from NASs. If set, Cisco Access Registrar does not check whether the account request packet has been signed with the same shared secret as the NAS.

**Ignore-Accounting-Signature** is used to work with RADIUS implementations that did not sign Accounting-Requests. A script was provided in the distribution (for USR NASs) that could be set in the IncomingScript extension point for the USR Vendor that simply set this environment variable.

## Incoming-Translation-Groups

**Incoming-Translation-Groups** is read for authentication. If set, **Incoming-Translation-Groups** specifies the translation groups to be used to filter attributes on requests.

## Misc-Log-Msg-Info

**Misc-Log-Msg-Info** is read for packet event logging. If a log message is generated, the value of **Misc-Log-Msg-Info** is inserted into the middle of the log message.

## PAGER Environment Variable

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

## Reject-Reason

**Reject-Reason** is set when a request is being rejected and contains the **Reject-Reason**. Cisco Access Registrar uses the value of **Reject-Reason** to look up the reject reason in the reply message table.

If **Reject-Reason** is set to one of: UnknownUser, UserNotEnabled, UserPasswordInvalid, UnableToAcquireResource, ServiceUnavailable, InternalError, MalformedRequest, ConfigurationError, IncomingScriptFailed, OutgoingScriptFailed, IncomingScriptRejectedRequest, OutgoingScriptRejectedRequest, or TerminationAction, then the value set in the configuration under **/Radius/Advanced/ReplyMessages** will be returned.

## Remote-Server

**Remote-Server** is set and read for logging a rejected packet from a remote server. **Remote-Server** records the name and IP address of the remote server to which the request has been forwarded.

## Request-Authenticator

**Request-Authenticator** is set for every packet upon reception. Getting the **Request-Authenticator** from a script returns the value of the request authenticator.

## Request-Type

**Request-Type** is set when a request is first received to the type of request, such as one of Access-Request, Access-Accept, Access-Reject, Accounting-Request, Accounting-Response, or Access-Challenge before calling any extension points.

The request contains a string representation of the RADIUS packet type (code). When Cisco Access Registrar does not recognize the packet type, it is represented as “Unknown-Packet-Type-*<N>*”, where *<N>* is the numeric value of the packet type (for example “Unknown-Packet-Type-9”). The known packet types are listed in Table B-1.

**Table B-1 Request-Type Packets**

String	Packet Code
Access-Request	(1)
Access-Accept	(2)
Access-Reject	(3)
Accounting-Request	(4)
Accounting-Response	(5)
Access-Challenge	(11)
Status-Server	(12)
Status-Client	(13)
USR-Resource-Free-Request	(21)
USR-Resource-Free-Response	(22)
USR-Resource-Query-Request	(23)
USR-Resource-Query-Response	(24)
USR-NAS-Reboot-Request	(26)
USR-NAS-Reboot-Response	(27)
Ascend-IPA-Allocate	(50)
Ascend-IPA-Release	(51)
USR-Enhanced-Radius	(254)



**Note** **Request-Type** is to be used only by scripts.

## Require-User-To-Be-In-Authorization-List

**Require-User-To-Be-In-Authorization-List** is read for authorization. If we are authorizing with a different service than we authenticated with (not usually done) and the user is not known by the authorization service, the default is to continue on unless this environment variable is set, in which case we reject the request with a cause of Unknown-user.

## Response-Type

**Response-Type** is set and read throughout processing and used to determine whether the request should be accepted, rejected, or challenged. When **Response-Type** is set to "Access-Reject at any time during the processing of a request, no more processing of the request is done, and an Access-Reject response is sent. For other valid values for **Response-Type**, see Table B-1.

## Retrace-Packet

If set, **Retrace-Packet**, causes a trace the packet to be displayed during the incoming and outgoing scripts. If set, will cause a second trace of the request packet's contents after running all the incoming scripts and/or a second trace of the response packet's contents before running the outgoing scripts.

## Skip-Session-Management

When set to TRUE in a request, **Skip-Session-Management** causes session management to be skipped for the request, even if session management might normally occur.

**Note**

---

There is a #define in **rex.h** named REX\_ENV\_VAR\_SKIP\_SESSION\_MANAGEMENT.

---

## Session-Key

**Session-Key** is read for session management. If set, the server uses it as the key to look up the session associated with the current request, if any. If not set, the server uses the NAS IP Address and NAS Port to create a session key.

## Session-Manager

**Session-Manager** is read after user authorization and determines which dynamic resources to allocate for this user, when one is needed. If set, the server directs the request to be processed by the specified session manager. When not set, the **SessionManager** (as defined in **DefaultSessionManager**) is used when needed.

## Session-Notes

**Session-Notes** is a comma-separated list set to make session information available to scripts.

**Session-Notes** contains the names of other environment variables. If set, these variables are stored on a Session as notes.

## Session-Service

**Session-Service** is set and read during session management. If set, the server will direct the request to be processed by the specified session service.

## Source-IP-Address

**Source-IP-Address** is set when a request is first received to the IP address from which the IP request was received before calling any extension points. **Source-IP-Address** contains the IP address of the NAS or proxy server that sent the request to this server.



Note

---

**Source-IP-Address** is to be used only by scripts.

---

## Source-Port

**Source-Port** is set when a request is first received to the port from which the request was received. **Source-Port** is set for each request before calling any extension points and contains the port on the NAS or proxy server that was used to send the request to this server.



Note

---

**Source-Port** is to be used only by scripts.

---

## Subnet-Size-If-No-Match

**Subnet-Size-If-No-Match** is set to one of BIGGER, SMALLER or EXACT, determines the behavior of the subnet-dynamic resource manager if a pool of the requested size is not available.

## Trace-Level

**Trace-Level** is set for each request before calling any extension points. **Trace-Level** is set to the current trace level as specified through **aregcmd**. If set by a script, **Trace-Level** changes the trace level used to determine what level of information is traced.

## Unavailable-Resource

**Unavailable-Resource** is set during session management. If the request is being rejected because one of the resource managers failed to allocate a resource, **Unavailable-Resource** is set to the name of the resource manager that failed.

## Unavailable-Resource-Type

**Unavailable-Resource-Type** is set during session management. If the request is being rejected because one of the resource managers failed to allocate a resource, **Unavailable-Resource-Type** is set to the type of the resource manager that failed.

## UserDefined1

**UserDefined1** is set to the value of the UserDefined1 property of the user from a local user list during password matching of local users.

## User-Authorization-Script

**User-Authorization-Script** is read in local services during authorization. If set, the server calls the specified script to do additional user authorization after authentication succeeds.

## User-Group

**User-Group** is read in local services during authorization. If set, species the UserGroup to which the current user belongs.

## User-Group-Session-Limit

**User-Group-Session-Limit** is read during session management. If set, **User-Group-Session-Limit** overrides the limit specified for the group-session-limit resource manager.

## User-Name

**User-Name** is read by a local service during authentication. When **User-Name** is set, it is the name used to authenticate or authorize the request and overrides the **User-Name** in the Request dictionary.

## User-Profile

**User-Profile** is read in local services during authorization. If set, **User-Profile** specifies the Profile from which the current user should receive attributes.

## User-Session-Limit

**User-Session-Limit** is read during session management. If set, **User-Session-Limit** overrides the limit specified for the user-session-limit resource manager.



## RADIUS Attributes

---

This appendix lists the attributes Cisco Access Registrar 3.0 supports with their names and values. RADIUS attributes carry the specific authentication, authorization information, and configuration details for requests and replies. For more detailed information about specific attributes, refer to the appropriate RFC as listed Table C-1.

**Table C-1** RFCs for RADIUS Attributes

RFC Subject	RFC Number
Standard RADIUS Attributes	2865
RADIUS Accounting Attributes	2866
Accounting Modifications for Tunnel Protocol Support	2867
Attributes for Tunnel Protocol Support	2868
RADIUS Extensions	2869
RADIUS for IPv6	3162

This appendix has two sections:

**RADIUS Attributes**—This section provides an alphabetic list of all RADIUS attributes Cisco Access Registrar 3.0 supports and a list of all RADIUS attributes in numeric order.

**Vendor Specific Attributes**—This section provides lists of RADIUS vendor-specific attributes (VSAs).

## RADIUS Attributes

This section lists the RADIUS attributes supported in Cisco Access Registrar 3.0. RADIUS attributes carry specific authentication, authorization, information, and configuration details in the Access-Request and the RADIUS server response.

## Cisco Access Registrar 3.0 Attributes

This section provides an alphabetical list of all attributes used in Cisco Access Registrar 3.0 and the attribute number.

**Table C-2 RADIUS Attributes Alphabetical List**

<b>Attribute Name</b>	<b>Attribute Number</b>
Acct-Authentic	45
Acct-Delay-Time	41
Acct-Input-Gigawords	52
Acct-Input-Octets	42
Acct-Input-Packets	47
Acct-Interim-Interval	85
Acct-Link-Count	51
Acct-Multi-Session-Id	50
Acct-Output-Gigawords	53
Acct-Output-Octets	43
Acct-Output-Packets	48
Acct-Session-Id	44
Acct-Session-Time	46
Acct-Status-Type	40
Acct-Terminate-Cause	49
Acct-Tunnel-Connection	68
Acct-Tunnel-Packets-Lost	86
ARAP-Challenge-Response	84
ARAP-Features	71
ARAP-Password	70
ARAP-Security	73
ARAP-Security-Data	74
ARAP-Zone-Access	72
Callback-Id	20
Callback-Number	19
Called-Station-Id	30
Calling-Station-Id	31
Change-Password	17
CHAP-Challenge	60
CHAP-Password	3
Class	25
Configuration-Token	78

**Table C-2 RADIUS Attributes Alphabetical List (continued)**

<b>Attribute Name</b>	<b>Attribute Number</b>
Connect-Info	77
EAP-Message	79
Event-Timestamp	55
Filter-Id	11
Framed-AppleTalk-Link	37
Framed-AppleTalk-Network	38
Framed-AppleTalk-Zone	39
Framed-Compression	13
Framed-Interface-Id	96
Framed-IP-Address	8
Framed-IP-Netmask	9
Framed-IPv6-Pool	100
Framed-IPv6-Prefix	97
Framed-IPv6-Route	99
Framed-IPX-Network	23
Framed-MTU	12
Framed-Pool	88
Framed-Protocol	7
Framed-Route	22
Framed-Routing	10
Idle-Timeout	28
Login-IP-Host	14
Login-IPv6-Host	98
Login-LAT-Group	36
Login-LAT-Node	35
Login-LAT-Port	63
Login-LAT-Service	34
Login-Service	15
Login-TCP-Port	16
Message-Authenticator	80
NAS-Identifier	32
NAS-IP-Address	4
NAS-IPv6-Address	95
NAS-Port	5
NAS-Port-ID	87
NAS-Port-Type	61

**Table C-2 RADIUS Attributes Alphabetical List (continued)**

Attribute Name	Attribute Number
Password-Expiration	21
Password-Retry	75
Port-Limit	62
Prompt	76
Proxy-State	33
Reply-Message	18
Service-Type	6
Session-Timeout	27
State	24
Termination-Action	29
Text-Ascend-Data-Filter	225
Tunnel-Assignment-ID	82
Tunnel-Client-Auth-ID	90
Tunnel-Client-Endpoint	66
Tunnel-Medium-Type	65
Tunnel-Password	69
Tunnel-Preference	83
Tunnel-Private-Group-ID	81
Tunnel-Server-Auth-ID	91
Tunnel-Server-Endpoint	67
Tunnel-Type	64
User-Name	1
User-Password	2
Vendor-Specific Attributes	26

## RADIUS Attributes Numeric List

Table C-3 lists all RFC-defined RADIUS attributes in numeric order.

**Table C-3 RADIUS Attributes Numeric List**

Number	Attribute Name
1	User-Name
2	User-Password
3	CHAP-Password
4	NAS-IP-Address
5	NAS-Port

**Table C-3 RADIUS Attributes Numeric List (continued)**

<b>Number</b>	<b>Attribute Name</b>
6	Service-Type
7	Framed-Protocol
8	Framed-IP-Address
9	Framed-IP-Netmask
10	Framed-Routing
11	Filter-Id
12	Framed-MTU
13	Framed-Compression
14	Login-IP-Host
15	Login-Service
16	Login-TCP-Port
17	Change-Password
18	Reply-Message
19	Callback-Number
20	Callback-Id
21	Password-Expiration
22	Framed-Route
23	Framed-IPX-Network
24	State
25	Class

**Table C-3 RADIUS Attributes Numeric List (continued)**

<b>Number</b>	<b>Attribute Name</b>
26	Vendor-Specific Attributes (VSAs) Refer to “Vendor Specific Attributes” section on page C-13 or the specific vendor’s VSA list: <ul style="list-style-type: none"> <li>• 3GPP VSAs</li> <li>• 3GPP2 VSAs</li> <li>• ACC VSAs</li> <li>• Altiga VSAs</li> <li>• Ascend VSAs</li> <li>• Bay Networks VSAs</li> <li>• Cabletron VSAs</li> <li>• Cisco VSAs</li> <li>• Compatible VSAs</li> <li>• Microsoft VSAs</li> <li>• Nomadix VSAs</li> <li>• RedBack VSAs</li> <li>• RedCreek VSAs</li> <li>• Telebit VSAs</li> <li>• Unisphere VSAs</li> <li>• USR VSAs</li> </ul>
27	Session-Timeout
28	Idle-Timeout
29	Termination-Action
30	Called-Station-ID (DNIS)
31	Calling-Station-ID (CLID)
32	NAS-Identifier
33	Proxy-State
34	Login-LAT-Service
35	Login-LAT-Node
36	Login-LAT-Group
37	Framed-AppleTalk-Link
38	Framed-AppleTalk-Network
39	Framed-AppleTalk-Zone
40	Acct-Status-Type
41	Acct-Delay-Time
42	Acct-Input-Octets

**Table C-3 RADIUS Attributes Numeric List (continued)**

<b>Number</b>	<b>Attribute Name</b>
43	Acct-Output-Octets
44	Acct-Session-Id
45	Acct-Authentic
46	Acct-Session-Time
47	Acct-Input-packets
48	Acct-Output-packets
49	Acct-Terminate-Cause
50	Acct-Multi-Session-Id
51	Acct-Link-Count
52	Acct-Input-Gigawords
53	Acct-Output-Gigawords
54	unassigned
55	Event-Timestamp
56	unassigned
57	unassigned
58	unassigned
59	unassigned
60	CHAP-Challenge
61	NAS-Port-Type
62	Port-Limit
63	Login-LAT-PortNo
64	Tunnel-Type
65	Tunnel-Medium-Type
66	Tunnel-Client-Endpoint
67	Tunnel-Server-Endpoint
68	Acct-Tunnel-Connection
68	Tunnel-ID
69	Tunnel-Password
70	ARAP-Password
71	ARAP-Features
72	ARAP-Zone-Access
73	ARAP-Security
74	ARAP-Security-Data
75	Password-Retry
76	Prompt
77	Connect-Info

**Table C-3 RADIUS Attributes Numeric List (continued)**

Number	Attribute Name
78	Configuration-Token
79	EAP-Message
80	Message-Authenticator
81	Tunnel-Private-Group-ID
81	Ascend-Auth-Type
82	Tunnel-Assignment-ID
83	Tunnel-Preference
84	ARAP-Challenge-Response
85	Acct-Interim-Interval
85	Ascend-IP-Pool-Chaining
86	Acct-Tunnel-Packets-Lost
87	NAS-Port-ID
88	Framed-Pool
88	Ascend-IP-TOS
89	Ascend-IP-TOS-Precedence
90	Tunnel-Client-Auth-ID
90	Ascend-IP-TOS-Apply-To
91	Tunnel-Server-Auth-ID
91	Ascend-Filter
92	Ascend-Dsl-Rate-Type
93	Ascend-Redirect-Number
94	Ascend-ATM-Vpi
95	Ascend-ATM-Vci
96	Ascend-Source-IP-Check
97	Ascend-Dsl-Rate-Mode
98	Ascend-Dsl-Upstream-Limit
99	Ascend-Dsl-Downstream-Limit
100	Ascend-Dsl-CIR-Recv-Limit
101	Ascend-Dsl-CIR-Xmit-Limit
102	Ascend-VRouter-Name
103	Ascend-Source-Auth
104	Ascend-Private-Route
105	unassigned
106	Ascend-FR-Link-Status-DLCI
107	unassigned
108	Ascend-Callback-Delay

**Table C-3 RADIUS Attributes Numeric List (continued)**

<b>Number</b>	<b>Attribute Name</b>
109	unassigned
110	unassigned
111	Ascend-Multicast-GLeave-Delay
112	Ascend-CBCP-Enable
113	Ascend-CBCP-Mode
114	unassigned
115	Ascend-CBCP-Trunk-Group
116	Ascend-Appletalk-Route
117	Ascend-Appletalk-Peer-Mode
118	Ascend-Route-Appletalk
119	unassigned
120	Ascend-Modem-PortNo
121	Ascend-Modem-SlotNo
122	unassigned
123	unassigned
124	unassigned
125	Ascend-Maximum-Call-Duration
126	Ascend-Preference
127	Tunneling-Protocol
128	Ascend-Shared-Profile-Enable
129	Ascend-Primary-Home-Agent
130	Ascend-Secondary-Home-Agent
131	Ascend-Dialout-Allowed
132	Ascend-Client-Gateway
133	Ascend-BACP-Enable
134	Ascend-DHCP-Maximum-Leases
135	Ascend-Client-Primary-DNS
136	Ascend-Client-Secondary-DNS
137	Ascend-Client-Assign-DNS
138	Ascend-User-Acct-Type
139	Ascend-User-Acct-Host
140	Ascend-User-Acct-Port
141	Ascend-User-Acct-Key
142	Ascend-User-Acct-Base
143	Ascend-User-Acct-Time
144	Ascend-Assign-IP-Client

**Table C-3 RADIUS Attributes Numeric List (continued)**

<b>Number</b>	<b>Attribute Name</b>
145	Ascend-Assign-IP-Server
146	Ascend-Assign-IP-Global-Pool
147	Ascend-DHCP-Reply
148	Ascend-DHCP-Pool-Number
149	Ascend-Expect-Callback
150	Ascend-Event-Type
151	Ascend-Session-Svr-Key
152	Ascend-Multicast-Rate-Limit
153	Ascend-IF-Netmask
154	Ascend-Remote-Addr
155	Ascend-Multicast-Client
156	Ascend-FR-Circuit-Name
157	Ascend-FR-LinkUp
158	Ascend-FR-Nailed-Grp
159	Ascend-FR-Type
160	Ascend-FR-Link-Mgt
161	Ascend-FR-N391
162	Ascend-FR-DCE-N392
163	Ascend-FR-DTE-N392
164	Ascend-FR-DCE-N393
165	Ascend-FR-DTE-N393
166	Ascend-FR-T391
167	Ascend-FR-T392
168	Ascend-Bridge-Address
169	Ascend-TS-Idle-Limit
170	Ascend-TS-Idle-Mode
171	Ascend-DBA-Monitor
172	Ascend-Base-Channel-Count
173	Ascend-Minimum-Channels
174	Ascend-IPX-Route
175	Ascend-FT1-Caller
176	Ascend-backup
177	Ascend-Call-Type
178	Ascend-Group
179	Ascend-FR-DLCI
180	Ascend-FR-Profile-Name

**Table C-3 RADIUS Attributes Numeric List (continued)**

<b>Number</b>	<b>Attribute Name</b>
181	Ascend-Ara-PW
182	Ascend-IPX-Node-Addr
183	Ascend-Home-Agent-IP-Addr
184	Ascend-Home-Agent-Password
185	Ascend-Home-Network-Name
186	Ascend-Home-Agent-UDP-Port
187	Ascend-Multilink-ID supported
188	Ascend-Num-In-Multilink
189	Ascend-First-Dest (Not supported)
190	Ascend-Pre-Input-Octets
191	Ascend-Pre-Output-Octets
192	Ascend-Pre-Input-packets
193	Ascend-Pre-Output-packets
194	Ascend-Maximum-Time
195	Ascend-Disconnect-Cause
196	Ascend-Connect-Progress
197	Ascend-Data-Rate
198	Ascend-PreSession-Time
199	Ascend-Token-Idle
200	Ascend-Token-Immediate
201	Ascend-Require-Auth
202	Ascend-Number-Sessions
203	Ascend-Authen-Alias
204	Ascend-Token-Expiry
205	Ascend-Menu-Selector
206	Ascend-Menu-Item
207	unassigned
208	Ascend-PW-Lifetime
209	Ascend-IP-Direct
210	Ascend-PPP-VJ-Slot-Comp
211	Ascend-PPP-VJ-1172
212	Ascend-PPP-Async-Map
213	Ascend-Third-Prompt
214	Ascend-Send-Secret
215	Ascend-Receive-Secret
216	Ascend-IPX-Peer-Mode

**Table C-3 RADIUS Attributes Numeric List (continued)**

<b>Number</b>	<b>Attribute Name</b>
217	Ascend-IP-Pool-Definition
218	Ascend-Assign-IP-Pool
219	Ascend-FR-Direct
220	Ascend-FR-Direct-Profile
221	Ascend-FR-Direct-DLCI
222	Ascend-Handle-IPX
223	Ascend-Netware-timeout
224	Ascend-IPX-Alias
225	Ascend-Metric
226	Ascend-PRI-Number-Type
227	Ascend-Dial-Number
228	Ascend-Route-IP
229	Ascend-Route-IPX
230	Ascend-Bridge
231	Ascend-Send-Auth
232	Ascend-Send-Passwd
233	Ascend-Link-Compression
234	Ascend-Target-Util
235	Ascend-Maximum-Channels
236	Ascend-Inc-Channel-Count
237	Ascend-Dec-Channel-Count
238	Ascend-Seconds-Of-History
239	Ascend-History-Weigh-Type
240	Ascend-Add-Seconds
241	Ascend-Remove-Seconds
242	Ascend-Data-Filter
243	Ascend-Call-Filter
244	Ascend-Idle-Limit
245	Ascend-Preempt-Limit
246	Ascend-Callback
247	Ascend-Data-Svc
248	Ascend-Force-56
249	Ascend-Billing-Number
250	Ascend-Call-By-Call
251	Ascend-Transit-Number
252	Ascend-Host-Info

**Table C-3 RADIUS Attributes Numeric List (continued)**

Number	Attribute Name
253	Ascend-PPP-Address
254	Ascend-MPP-Idle-Percent
255	Ascend-Xmit-Rate

## Vendor Specific Attributes

This section lists all vendor-specific attributes (VSAs) supported by Cisco Access Registrar 3.0.

### 3GPP VSAs

Table C-4 lists the 3GPP VSAs. The vendor ID for 3GPP VSAs is 10415.

**Table C-4 3GPP VSAs**

SubAttr	VSA Name
1	3GPP-IMSI
2	3GPP-Charging-Id
3	3GPP-PDPType
4	3GPP-OG-Address
5	3GPP-GPRS-QoS-Profile
6	3GPP-SGSN-Address
7	3GPP-GGSN-Address
8	3GPP-IMSI-MCC-MNC
9	3GPP-GGSN-MCC-MNC
10	3GPP-NSAPI
11	3GPP-Session-Stop-Indicator
12	3GPP-Selection-Mode
13	3GPP-Charging-Characteristics

### 3GPP2 VSAs

Table C-5 lists the 3GPP2 VSAs. The vendor ID for 3GPP2 VSAs is 5535.

**Table C-5 3GPP2 VSAs**

SubAttr	VSA Name
1	CDMA-Security-Status
2	CDMA-Security-Level
3	CDMA-Pre-Shared-Secret

Table C-5 3GPP2 VSAs (continued)

SubAttr	VSA Name
4	CDMA-Reverse-Tunnel-Spec
5	CDMA-Diff-Svc-Class-Opt
6	CDMA-Container
9	CDMA-PCF-IP-Addr
10	CDMA-BS-MS-Addr
11	CDMA-User-ID
12	CDMA-Forward-MUX
13	CDMA-Reverse-MUX
14	CDMA-Forward-Rate
15	CDMA-Reverse-Rate
16	CDMA-Service-Option
17	CDMA-Forward-Type
18	CDMA-Reverse-Type
19	CDMA-Frame-Size
20	CDMA-Forward-RC
21	CDMA-Reverse-RC
22	CDMA-IP-Technology
23	CDMA-Comp-Flag
24	CDMA-Release-Ind
25	CDMA-Dropped-Octets
26	CDMA-Start-Date
27	CDMA-Start-Time
28	CDMA-Stop-Date
29	CDMA-Stop-Time
30	CDMA-Num-Active
31	CDMA-SDB-Input-Octets
32	CDMA-SDB-Output-Octets
33	CDMA-NumSDB-Input
34	CDMA-NumSDB-Output
35	CDMA-Alt-Billing
36	CDMA-IP-QoS
37	CDMA-Interconnect-IP
38	CDMA-Interconnect-QoS
39	CDMA-Air-QoS
40	CDMA-Airlink-Record-Type
41	CDMA-R-P-Link-ID

**Table C-5 3GPP2 VSAs (continued)**

<b>SubAttr</b>	<b>VSA Name</b>
42	CDMA-Airlink-Record-Type
43	CDMA-PPP-Bytes-Received
44	CDMA-Correlation-ID
45	CDMA-Mobile-Terminate-Originated
48	CDMA-Session-Continue
49	CDMA-Active-Time
50	CDMA-DCCH-Frame-Format
51	CDMA-Beginning-Session
52	CDMA-ESN
54	CDMA-S-Attribute
55	CDMA-S-Request-Attribute
56	CDMA-S-Lifetime-Attribute
57	CDMA-MN-HA-Shared-Key-Request
58	CDMA-MN-HA-Shared-Key
59	CDMA-Remote-IPv4-Address
60	CDMA-HRPD-Access-Authentication
70	CDMA-Remote-IPv6-Address
71	CDMA-Remote-Address-Table-Index
72	CDMA-Address-Octet-Count
73	CDMA-Allowed-Differentiated-Service-Marking
74	CDMA-Allowed-Air-Link-QoS
75	CDMA-DNS-Update-Required

## ACC VSAs

Table C-6 lists the ACC VSAs. The vendor ID for ACC VSAs is 5.

**Table C-6 ACC VSAs**

SubAttr	VSA Name	Type	Min-Max Value
1	Acc-Reason-Code	ENUM: no reason given/no failure resource shortage protocol error invalid attribute invalid service type invalid framed protocol invalid attribute value invalid user information invalid IP address invalid integer syntax invalid NAS port	0-56
1	Acc-Reason-Code ( <i>Continued</i> )	ENUM: requested by user session already open network disconnect service interruption physical port error idle timeout session timeout administrative reset NAS reload or reset NAS error NAS request	0-56
1	Acc-Reason-Code ( <i>Continued</i> )	ENUM: undefined reason given too many RADIUS users conflicting attributes port limit exceeded facility not available internal configuration error bad route specification	0-56

Table C-6 ACC VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
1	Acc-Reason-Code ( <i>Continued</i> )	Access Partition bind failure security violation request type conflict configuration disallowed missing attribute no authentication server invalid request missing parameter invalid parameter call cleared with cause inopportune config request invalid config parameter missing config parameter incompatible service profile administrative reset	0-56
1	Acc-Reason-Code ( <i>Continued</i> )	administrative reload no authentication response port unneeded port preempted port suspended service unavailable callback user error host request no accounting server no accounting response access denied temporary buffer shortage	0-56
2	Acc-Ccp-Option	ENUM: Disabled Enabled	1-2
3	Acc-Input-Errors	UINT32	0-253
4	Acc-Output-Errors	UINT32	0-253
5	Acc-Access-Partition	String	0-253
6	Acc-Customer-Id	String	0-253
7	Acc-Ip-Gateway-Pri	IPADDR	0-253
8	Acc-Ip-Gateway-Sec	IPADDR	0-253
9	Acc-Route-Policy	ENUM : Funnel Direct	1-2
10	Acc-ML-MLX-Admin-State	ENUM: Enabled Disabled	1-2
11	Acc-ML-Call-Threshold	UINT32	0-253
12	Acc-ML-Clear-Threshold	UINT32	0-253

Table C-6 ACC VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
13	Acc-ML-Damping-Factor	UINT32	0-253
14	Acc-Tunnel-Secret	String	0-253
15	Acc-Clearing-Cause	ENUM: cause unspecified unassigned number invalid information element c message incompatible with sta recovery on timer expiration mandatory information element protocol error interworking normal clearing user busy no user responding user alerted no answer	0-127
15	Acc-Clearing-Cause (Continued)	ENUM: no route to transit network call rejected number changed non selected user clearing destination out of order invalid or incomplete number facility rejected no route to destination response to status inquiry normal unspecified cause no circuit or channel availab network out of order	0-127
15	Acc-Clearing-Cause (Continued)	ENUM: temporary failure switching equipment congestio access information discarded circuit or channel unavailabl circuit or channel preempted resources unavailable quality of service unavailabl facility not subscribed outgoing calls barred incoming calls barred bearer capability unauthorize bearer capability not availab	0-127

Table C-6 ACC VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
15	Acc-Clearing-Cause (Continued)	ENUM: channel unacceptable service not available bearer capability not impleme channel type not implemented facility not implemented call awarded being delivered restricted digital informatio service not implemented invalid call reference identified channel does not e call identity does not exist call identity in use no call suspended	0-127
15	Acc-Clearing-Cause (Continued)	ENUM: suspended call cleared incompatible destination invalid transit network selec invalid message mandatory information element message not implemented inopportune message information element not imple	0-127
16	Acc-Clearing-Location	ENUM: local or remote user private network serving local beyond interworking point public network serving local transit network private network serving remot public network serving remote international network	0-10
17	Acc-Service-Profile	String	0-253
18	Acc-Request-Type	ENUM: Ring Indication Dial Request User Authentication Tunnel Authentication User Accounting Tunnel Accounting	1-6
19	Acc-Framed-Bridge	ENUM : Disabled Enabled	0-1
20	Acc-Vpsm-Oversubscribed	ENUM : False True	1-2

Table C-6 ACC VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
21	Acc-Acct-On-Off-Reason	ENUM : NAS Reset NAS Reload Configuration Reset Configuration Reload Enabled Disabled	0-5
22	Acc-Tunnel-Port	UINT32	0-253
23	Acc-Dns-Server-Pri	IPADDR	0-253
24	Acc-Dns-Server-Sec	IPADDR	0-253
26	Acc-Nbns-Server-Sec	IPADDR	0-253
27	Acc-Dial-Port-Index		
28	Acc-Ip-Compression	ENUM: Disabled Enabled	0-1
29	Acc-Ipx-Compression	ENUM: Disabled Enabled	0-1
30	Acc-Connect-Tx-Speed	UINT32	0-253
31	Acc-Connect-Rx-Speed	UINT32	0-253
32	Acc-Modem-Modulation-Type	String	0-253
33	Acc-Modem-Error-Protocol	String	0-253
34	Acc-Callback-Delay	UINT32	0-253
35	Acc-Callback-Num-Valid	String	0-253
36	Acc-Callback-Mode	ENUM: User-Auth User-Specified-E-164 CBCP-Callback CLI-Callback	0-7
37	Acc-Callback-CBCP-Type	ENUM: CBCP-None CBCP-User-Specified CBCP-Pre-Specified	1-3
38	Acc-Dialout-Auth-Mode	ENUM: PAP CHAP CHAP-PAP NONE	1-4
39	Acc-Dialout-Auth-Password	String	0-253
40	Acc-Dialout-Auth-UserName	String	0-253

Table C-6 ACC VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
42	Acc-Access-Community	ENUM: PUBLIC NETMAN	1-2
43	Acc-Vpsm-Reject-Cause	ENUM: No-Access-Partition Access-Partition-Disabled Partition-Portlimit-Exceeded License-Portlimit-Exceeded Home-Server-Down Rejected-By-Home-Server NAS-Administratively-Disabled	1-7
44	Acc-Ace-Token	String	0-253
45	Acc-Ace-Token-Ttl	UINT	0-253
46	Acc-Ip-Pool-Name	String	0-253
47	Acc-Igmp-Admin-State	ENUM : Enabled Disabled	1-2
48	Acc-Igmp-Version	ENUM : V1 V2	1-2

## Altiga VSAs

Table C-7 lists the Altiga VSAs. The vendor ID for Altiga VSAs is 3076.

Table C-7 Altiga VSAs

SubAttr	VSA Name	Type	Min-Max Value
1	Altiga-General-Access-Hours	String	0-253
2	Altiga-General-Simultaneous-Logic	UINT32	0-253
3	Altiga-General-Minimum-Password-Length	UINT32	0-253
4	Altiga-General-All-Alphabetic-Only-Passwords	ENUM	0-1
5	Altiga-General-Primary-DNS	IP Address	0-253
6	Altiga-General-Secondary-DNS	IP Address	0-253
8	Altiga-General-Secondary-WINS	IP Address	0-253
9	Altiga-General-SEP-Card-Assignment	UINT32	0-253
10	Altiga-General-Priority-On-SEP	UINT32	0-253
11	Altiga-General-Tunneling-Protoco	UNIT32	0-253
12	Altiga-IPSec-Security-Associatio	String	0-253

Table C-7 Altiga VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
13	Altiga-IPSec-Authentication	ENUM: None RADIUS LDAP NT Domain SDI Internal	0-5
15	Altiga-IPSec-Banner	String	0-253
16	Altiga-IPSec-Allow-Password-Storage-On-Client	ENUM: False True	0-1
17	Altiga-PPTP-L2TP-Use-Client-Specified-Address	ENUM: False True	0-1
18	Altiga-PPTP-Minimal-Authentication-Protocol	UINT32	0-253
19	Altiga-L2TP-Minimal-Authentication	UINT32	0-253
20	Altiga-PPTP-Encryption	UINT32	0-253
21	Altiga-L2TP-Encryption	UINT32	0-253
22	Altiga-Argument-Authentication-Server-Type	ENUM: First Active Server RADIUS LDAP NT SDI Internal	0-5
23	Altiga-Argument-Authentication-Server-Password	String	0-253
24	Altiga-Argument-Request-Authenticator-Vector	String	0-253
25	Altiga-IPSec-LTL-Keepalives	ENUM: False True	0-1
26	Altiga-Argument-IPSec-Group-Name	String	0-253
27	Altiga-IPSec-Split-Tunneling	String	0-253
28	Altiga-IPSec-Default-Domain	String	0-253
28	Altiga-IPSec-Secondary-Domain-List	String	0-253
30	Altiga-IPSec-Tunnel-Type	ENUM: LAN to LAN Remote Access	1-2
31	Altiga-IPSec-Mode-Configuration	ENUM: False True	0-1
32	Altiga-Argument-Authentication-Server-Priority	UINT32	0-253

Table C-7 Altiga VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
33	Altiga-IPSec-Group-Lock-Of-User	ENUM: False True	0-1
34	Altiga-IPSec-IPSec-Over-UDP	ENUM: False True	0-1
35	Altiga-IPSec-UDP-Port-For-IPSec	UINT32	0-253
128	Altiga-Partitioning-Primary-DHCP		
129	Altiga-Partitioning-Secondary-DHCP	IP Address	0-253
131	Altiga-Partitioning-Premise-Rout	IP Address	0-253
132	Altiga-Partitioning-Partition-Max-Sessions	String	0-253
133	Altiga-Partitioning-Mobile-IP-Key	String	0-253
134	Altiga-Partitioning-Mobile-IP-Address	IP Address	0-253
135	Altiga-Partitioning-Mobile-IP-SPI	IP Address	0-253
136	Altiga-Partitioning-Strip-Realm	ENUM: False True	0-1
137	Altiga-Partitioning-Group-ID	UINT32	0-253
250	Altiga-Group-Name	String	0-253

## Ascend VSAs

Table C-8 lists the Ascend VSAs. The vendor ID for Ascend VSAs is 529.

Table C-8 Ascend VSAs

SubAttr	VSA Name	Type	Min-Max Value
17	Ascend-Change-Password	String	0 - 253
18	Ascend-Session-Type	ENUM: Unused Unknown G711-Ulaw G711-Alaw G723 G729 G723-64KPS G728 RT24	0 - 8
19	Ascend-H323-Gatekeeper	IP Address	0 - 253
21	Ascend-H323-Conference-ID	String	0-253
22	Ascend-H323-Destination-NAS-ID	IP Address	0-65535

Table C-8 Ascend VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
23	Ascend-H323-Dialed-Time	UINT32	0-253
24	Ascend-H323-Dialed-Number	String	0-253
25	Ascend-Inter-Arrival-Jitter	UINT32	0-253
26	Ascend-Dropped-Octets	UINT32	0-253
27	Ascend-Dropped-Packets	UINT32	0-253
48	Ascend-Call-Direction	ENUM: Incoming Outgoing	0-1
49	Ascend-Service-Type	ENUM	0 - 23; NotUsed None EuUi Telnet TelnetBin RawTcp TermServer MP VirtualConn X25DChan PseuTunPPP IpFax Other ATM HdlcNrm VoIp Visa2 PPP Slip MPP X25 Combinet FR EuRaw
68	Ascend-Tunnel-ID	String	0 - 253
126	Ascend-Route-Preference	ENUM: Interface, OSPF-Internal, RIP, Down-WAN, OSPF-ASE, Infinite, ICMP	0-225
132	Ascend-Client-Gateway	IP Address	0 - 253
144	Ascend-Assign-IP-Client	IP Address	0-0

Table C-8 Ascend VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
145	Ascend-Assign-IP-Server	IP Address	0-0
152	Ascend-Multicast-Rate-Limit	UINT32	0-65535
162	Ascend-FR-DCE-N392	UINT32	0-65535
163	Ascend-FR-DTE-N392	UINT32	0-65535
164	Ascend-FR-DCE-N393	UINT32	0-65535
165	Ascend-FR-DTE-N393	UINT32	0-65535
166	Ascend-FR-T391	UINT32	0-65535
167	Ascend-FR-T392	UINT32	0-65535
168	Ascend-Bridge-Address	UINT32	1-253
169	Ascend-TS-Idle-Limit	UINT32	0-65535
170	Ascend-TS-Idle-Mode	ENUM; TS-Idle-None TS-Idle-Input TS-Idle-Input-Output	0-2
171	Ascend-DBA-Monitor	ENUM; Transmit Transmit-Receive None	0-2
172	Ascend-Base-Channel-Count	UINT32	0-65535
173	Ascend-Minimum-Channels	UINT32	0-65535
174	Ascend-IPX-Route	String	1-253
175	Ascend-FT1-Caller	ENUM; FT1-No FT1-Yes	0-1
176	Ascend-Backup	String	1-253
177	Ascend-Call-Type	ENUM; Nailed Nailed/MPP Perm/Switched	0-2
178	Ascend-Group	String	1-253
179	Ascend-FR-DLCI	UINT32	0-65535
180	Ascend-FR-Profile-Name	String	1-253
181	Ascend-Ara-PW	String	1-253
182	Ascend-IPX-Node-Address	String	1-253
183	Ascend-Home-Agent-IP-Addr	IP Address	0-0
184	Ascend-Home-Agent-Password	String	1-253
185	Ascend-Home-Network-Name	String	1-253
186	Ascend-Home-Agent-UDP-Port	UINT32	0-65535

Table C-8 Ascend VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
187	Ascend-Multilink-ID	UINT32	0-65535
188	Ascend-Num-In-Multilink	UINT32	0-65535
189	Ascend-First-Dest	IP Address	0-0
190	Ascend-Pre-Input-Octets	UINT32	0-65535
191	Ascend-Pre-Output-Octets	UINT32	0-65535
192	Ascend-Pre-Input-Packets	UINT32	0-65535
193	Ascend-Pre-Output-Packets	UINT32	0-65535
194	Ascend-Maximum-Time	UINT32	0-65535
195	vAscend-Pre-Output-Packets (continued)	ENUM: No-Reason, Not-Applicable, Modem-No-DCD, Session-Timeout, Invalid-Incoming-User, Disconnect-Due-To-Callback, DCD-Detected-Then-Inactive, Modem-Invalid-Result-Codes, Protocol-Disabled-Or-Unsuppor, Disconnect-Req-By-RADIUS, Disconnect-Req-By-Local-Admin, V110-Timeout-Or-Sync-Retry-Ex, PPP-Auth-Timeout-Exceeded, User-Executed-Do-Hangup, Remote-End-Hung-Up, Resource-Has-Been-Quiesced, Max-Call-Duration-Reached, Unknown, (continued)	0-195

Table C-8 Ascend VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
195	vAscend-Pre-Output-Packets	ENUM: TermSrv-User-Quit, TermSrv-Idle-Timeout, TermSrv-Exit-Telnet, TermSrv-No-IPaddr, TermSrv-Exit-Raw-TCP, TermSrv-Exit-Login-Failed, TermSrv-Exit-Raw-TCP-Disabled, TermSrv-CTRL-C-In-Login, TermSrv-Destroyed, TermSrv-User-Closed-VCon, Call-Disconnected, TermSrv-VCon-Destroyed, TermSrv-Exit-Rlogin, TermSrv-Bad-Rlogin-Option, TermSrv-Not-Enough-Resources, MPP-No-NULL-Msg-Timeout, CLID-Authentication-Failed, (continued)	0-195
195	vAscend-Pre-Output-Packets	ENUM: PPP-LCP-Timeout, PPP-LCP-Negotiation-Failed, PPP-PAP-Auth-Failed, PPP-CHAP-Auth-Failed, PPP-Rmt-Auth-Failed, PPP-Rcv-Terminate-Req, PPP-Rcv-Close-Event, PPP-No-NCPs-Open, PPP-MP-Bundle-Unknown, PPP-LCP-Close-MP-Add-Fail, CLID-RADIUS-Timeout (continued)	0-195
195	vAscend-Pre-Output-Packets (continued)	Out-Of-Resources, Invalid-IP-Address, Hostname-Resolution-Failed, Bad-Or-Missing-Port-Number, Host-Reset, Connection-Refused, Connection-Timeout, Connection-Closed, Network-Unreachable, Host-Unreachable, Network-Unreachable-Admin, Host-Unreachable-Admin, Port-Unreachable,	

Table C-8 Ascend VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
196	Ascend-Connect-Progress	ENUM: No-Progress, unknown1, Call-Up, unknown2, Modem-Up, Modem-Awaiting-DCD, Modem-Awaiting-Codes, TermSrv-Started, TermSrv-Raw-TCP-Started, TermSrv-Telnet-Started, TermSrv-Raw-TCP-Connected, TermSrv-Telnet-Connected, TermSrv-Rlogin-Started, TermSrv-Rlogin-Connected, TermSrv-Authentication-Begin, Modem-Outdial-Call-Up	0-94
196	Ascend-Connect-Progress	ENUM: LAN-Session-Up, LCP-Opening, CCP-Opening, IPNCP-Opening, NCP-Opening, LCP-Opened, CCP-Opened, IPNCP-Opened, BNCP-Opened, LCP-State-Initial, LCP-State-Starting, LCP-State-Closed, LCP-State-Stopped, BACP-Opened, LCP-State-Stopping, LCP-State-Request-Sent, LCP-State-Ack-Received, LCP-State-Ack-Sent, IPXNCP-Opened, ATNCP-Opened, BACP-Opening, V110-Up, V110-State-Opened, V110-State-Carrier, V110-State-Reset, V110-State-Closed	0-94
197	Ascend-Data-Rate	UINT32	0-65535
198	Ascend-PreSession-Time	UINT32	0-65535
199	Ascend-Token-Idle	UINT32	0-65535

Table C-8 Ascend VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
200		ENUM: Tok-Imm-No, Tok-Imm-Yes	0-1
201	Ascend-Require-Auth	ENUM: Not-Require-Auth Require-Auth Pap-Only Pap-Only Pap-Login-Only Pap-Framed-Only Pap-Outbound-Only CHAP-Only CHAP-Only CHAP-Login-Only CHAP-Framed-Only CHAP-Outbound-Only MS-CHAP-Only MS-CHAP-Only MS-CHAP-Login-Only MS-CHAP-Framed-Only MS-CHAP-Outbound-Only	0-55
210	Ascend-PPP-VJ-Slot-Comp	ENUM: VJ-Slot-Comp-No	1-1
211	Ascend-PPP-VJ-1172	ENUM: PPP-VJ-1172	1-1
212	Ascend-PPP-Async-Map	UINT32	0-65535
213	Ascend-Third-Prompt	String	1-253
214	Ascend-Send-Secret	String	1-253
215	Ascend-Receive-Secret	String	1-253
216	Ascend-IPX-Peer-Mode	ENUM: IPX-Peer-Router, IPX-Peer-Dialin	1-1
217	Ascend-IP-Pool-Definition	String	1-253
218	Ascend-Assign-IP-Pool	UINT32	0-65535
219	Ascend-FR-Direct	ENUM: FR-Direct-No, FR-Direct-Yes	1-1
220	Ascend-FR-Direct-Profile	String	1-253
221	Ascend-FR-Direct-DLCI	UINT32	0-65535
222	Ascend-Handle-IPX	ENUM: Handle-IPX-None, Handle-IPX-Client, Handle-IPX-Server	0-2
223	Ascend-Netware-timeout	UINT32	0-65535

Table C-8 Ascend VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
224	Ascend-IPX-Alias	UINT32	0-65535
225	Ascend-Metric	UINT32	0-65535
226	Ascend-PRI-Number-Type	ENUM: Unknown-Number, Intl-Number, National-Number, Local-Number Abbrev-Number	0-5
227	Ascend-Dial-Number	String	1-253
228	Ascend-Route-IP	ENUM: Unknown-Number, Intl-Number, National-Number, Local-Number, Abbrev-Number	0-5
229	Ascend-Route-IPX	ENUM: Route-IPX-No Route-IPX-Yes	0-1
230	Ascend-Bridge	ENUM: Bridge-No, Bridge-Yes	0-1
231	Ascend-Send-Auth	ENUM: Send-Auth-None, end-Auth-PAP, Send-Auth-CHAP	0-2
232	Ascend-Send-Passwd	String	1-253
233	Ascend-Link-Compression	ENUM: Link-Comp-None, Link-Comp-Stac, Link-Comp-Stac-Draft-9, Link-Comp-MS-Stac	0-3
234	Ascend-Target-Util	UINT32	0-65535
235	Ascend-Maximum-Channels	UINT32	0-65535
236	Ascend-Inc-Channel-Count	UINT32	0-65535
237	Ascend-Dec-Channel-Count	UINT32	0-65535
238	Ascend-Seconds-Of-History	UINT32	0-65535
239	Ascend-History-Weigh-Type	ENUM: History-Constant, History-Linear, History-Quadratic	0-2
240	Ascend-Add-Seconds	UINT32	0-65535
241	Ascend-Remove-Seconds	UINT32	0-65535
242	Ascend-Data-Filter	String	1-253

Table C-8 Ascend VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
243	Ascend-Call-Filter	String	1-253
244	Ascend-Idle-Limit	UINT32	0-65535
245	Ascend-Idle-Limit	UINT32	0-65535
246	Ascend-Callback	ENUM: Callback-No, Callback-Yes	0-1
247	Ascend-Data-Svc	ENUM: Switched-Voice-Bearer, Switched-56KR, Switched-192K, Switched-256K, Switched-320K, Switched-384K-MR, Switched-448K, Switched-512K, Switched-566K, Switched-640K, Switched-704K, Switched-768K, Switched-64K, Switched-832K, Switched-896K, Switched-960K, Switched-1024K, Switched-1088K, Switched-1152K, Switched-1216K, Switched-1280K, Switched-1344K, Switched-1408K, Switched-64KR, Switched-1472K, Switched-1600K, Switched-1664K, Switched-1728K, Switched-1792K, Switched-1856K, Switched-1920K, Switched-inherited, Switched-restricted-bearer-x3, Switched-clear-bearer-v110, Switched-56K, Switched-restricted-64-x30, Switched-clear-56-v110, Switched-modem, Switched-atmodem, Switched-384KR, Switched-384K, Switched-1536K, Switched-1536KR, Switched-128K	0-43
248	Ascend-Force-56	ENUM: Force-56-No, Force-56-Yes	0-1
249	Ascend-Billing-Number	String	1-253
250	Ascend-Call-By-Call	UINT32	0-65535
251	Ascend-Transit-Number	String	1-253
252	Ascend-Host-Info	String	1-253
253	Ascend-PPP-Address	IP Address	0-0
254	Ascend-MPP-Idle-Percent	UINT32	0-65535

## Bay Networks VSAs

Table C-9 Lists the Bay Networks VSAs. The vendor ID for Bay Networks VSAs is 1584.

**Table C-9 Bay Networks VSAs**

SubAttr	VSA Name	Type	Min-Max Value
28	Annex-Filter	String	1-253
29	Annex-CLI-Command	String	1-253
30	Annex-CLI-Filter	String	1-253
31	Annex-Host-Restrict	String	1-253
32	Annex-Host-Allow	String	1-253
33	Annex-Product-Name	String	1-253
34	Annex-SW-Version	String	1-253
35	Annex-Local-IP-Address	IPADDR	1-253
36	Annex-Callback-Portlist	UINT32	0-0
44	Annex-System-Disc-Reason	UINT32	0-0
45	Annex-Modem-Disc-Reason	UINT32	0-0
46	Annex-Disconnect-Reason	UINT32	0-0
50	Annex-Transmit-Speed	UINT32	0-0
51	Annex-Receive-Speed	UINT32	0-0

## Cabletron VSAs

Table C-10 lists the Cabletron VSAs. The vendor ID for Cabletron VSAs is 52.

**Table C-10 Cabletron VSAs**

SubAttr	VSA Name	Type	Min-Max Value
192	Cabletron-Framed-Data-Rate	ENUM: Rate-56KB Rate-64KB Rate-112KB Rate-128KB	0-4
193	Cabletron-Phone-Number	String	0-253
194	Cabletron-Caller-Id	String	0-253
196	Cabletron-Connection-Reference	UINT32	0-253
198	Cabletron-Initial-Rate	UINT32	0-253

Table C-10 Cabletron VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
199	Cabletron-Maximum-Rate	UINT32	0-253
192	Cabletron-Framed-Data-Rate	Enum: Rate-56KB Rate-64KB Rate-112KB Rate-128KB	192

## Cisco VSAs

Table C-11 lists the Cisco VSAs. The vendor ID for Cisco VSAs is 9.

Table C-11 Cisco VSAs

SubAttr	VSA Name	Type	Min-Max Value
1	Cisco-AVPair	String	0-253
2	Cisco-NAS-Port	String	0-253
3	Cisco-Fax-Account-ID-Origin	String	0-253
4	Cisco-Fax-Message-ID	String	0-253
5	Cisco-Fax-Pages	String	0-253
6	Cisco-FAX Cover Page Flag	String	0-253
7	Cisco-Fax-Modem-Time	String	0-253
8	Cisco-Fax-Connect-Speed	String	0-253
9	Cisco-Fax-Recipient-Count	String	0-253
10	Cisco-Fax-Process-Abort-Flag	String	0-253
11	Cisco-Fax-DSN-Address	String	0-253
12	Cisco-Fax-DSN-Flag	String	0-253
13	Cisco-Fax-MDN-Address	String	0-253
14	Cisco-Fax-MDN-Flag	String	0-253
15	Cisco-Fax-Auth-Status	String	0-253
16	Cisco-Email-Server-Address	IP Address	
17	Cisco-Email-Server-ACK Flag	String	0-253
18	Cisco-Gateway-ID	String	0-253
19	Cisco-Call-Type	String	0-253
20	Cisco-Port-Used	String	0-253
21	Cisco-Abort-Cause	String	0-253
22	Cisco-CRS-Info	String	0-253
23	Cisco-h323-Remote-Address	String	0-253
24	Cisco-h323-Conf-ID	String	0-253

Table C-11 Cisco VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
25	Cisco-h323-Setup-Time	String	0-253
26	Cisco-h323-Call-Origin	String	0-253
27	Cisco-h323-Call-Type	String	0-253
28	Cisco-h323-Connect-Time	String	0-253
29	Cisco-h323-Disconnect-Time	String	0-253
30	Cisco-h323-Disconnect-Cause	String	0-253
31	Cisco-h323-Voice-Quality	String	0-253
32	Cisco-h323-Generic-IVR-Out	String	0-253
33	Cisco-h323-Gateway-ID	String	0-253
34	Cisco-3GPP2-AVPair	String	0-253
35	Cisco Connection ID-h323-incoming-connection-ID	String	0-253
100	Cisco-h323-Generic-IVR-In	String	0-253
101	Cisco-h323-Amount-Balance		
102	Cisco-h323-Time-Balance	String	0-253
103	Cisco-h323-Return-Code	String	0-253
104	Cisco-h323-Prompt-ID	String	0-253
105	Cisco-h323-Time-of-Day	String	0-253
106	Cisco-h323-Redirect-Number	String	0-253
107	Cisco-h323-Preferred-Language	String	0-253
108	Cisco-h323-Redirect-IP-Address	String	0-253
109	Cisco-h323-Billing-Model	ENUM: postpaid prepaid	0-1
110	Cisco-h323-Currency	String	0-253
128	Cisco-UCP-IP-Pool-ID	String	0-253
129	Cisco-UCP-User-Max-Sessions	String	0-253
130	Cisco-UCP-User-Session-Count	String	0-253
131	Cisco-UCP-Next-Session-ID	String	0-253
132	Cisco-UCP-VPDN-Max-Sessions	String	0-253
133	Cisco-UCP-VPDN-Session-Count	String	0-253
134	Cisco-UCP-B-Channel-Max-Sessions	String	0-253
135	Cisco-UCP-B-Channel-Session-Coun	String	0-253
136	Cisco-UCP-Status	String	0-253
137	Cisco-UCP-BLOB-Attribute-Length	String	0-253
138	Cisco-UCP-Disable-Statu	String	0-253
139	Cisco-UCP-Block-Access-Range	String	0-253
140	Cisco-UCP-Home-POP-ID	String	0-253

Table C-11 Cisco VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
175	Cisco-UCP-IP-Addresses	IP Address	0-253
176	Cisco-UCP-Session-Info	String	0-253
211	Cisco-Ascend AV pairs	String	0-253
250	Cisco-SSG-Account-Info	String	0-253
251	Cisco-SSG-Service-Info	String	0-253
252	Cisco-SSG-Command-Code	String	0-253
253	Cisco-SSG-Control-Info	String	0-253

## Compatible VSAs

Table C-12 lists the Compatible VSAs. The vendor ID for Compatible VSAs is 255.

Table C-12 Compatible VSAs

SubAttr	VSA Name	Type	Min-Max Value
0	Compatible-Tunnel-Delay	UNIT32	0-253
1	Compatible-Tunnel-Throughput	UNIT32	0-253
3	Compatible-Tunnel-Server-Endpoint	IP Address	0-253
4	Compatible-Tunnel-Group-Info	String	0-253
5	Compatible-Tunnel-Password	String	0-253
6	Compatible-Echo	UNIT32	0-253
7	Compatible-Tunnel-Client-IPX	UNIT32	0-253

## Microsoft VSAs

Table C-13 lists the Microsoft VSAs. The vendor ID for Microsoft VSAs is 311.

Table C-13 Microsoft VSAs

SubAttr	VSA Name	Type	Min-Max Value
1	MS-CHAP-Response	String	50-50
2	MS-CHAP-Error	String	0-253
3	MS-CHAP-CPW1	String	70-70
4	MS-CHAP-CPW2	String	84-84
5	MS-CHAP-LM-Enc-PW	String	4-253
6	MS-CHAP-NT-Enc-PW	String	4-253
7	MS-MPPE-Encryption-Policy	ENUM: Encryption-Allowed Encryption-Required	1-2

## Vendor Specific Attributes

Table C-13 Microsoft VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
8	MS-MPPE-Encryption-Types	String	0-4
9	MS-RAS-Vendor	UINT32	0-253
10	MS-CHAP-Domain	String	0-253
11	MS-CHAP-Challenge	String	0-253
12	MS-CHAP-MPPE-Keys	String	32-32
13	MS-BAP-Usage	ENUM: Not allowed Allowed Required	0-2
14	MS-Link-Utilization-Threshold	UINT32	0-253
15	MS-Link-Drop-Time-Limit	String	0-253
16	MS-MPPE-Send-Key	String	0-253
17	MS-MPPE-Recv-Key	String	0-253
18	MS-RAS-Version	String	0-253
19	MS-Old-ARAP-Password	String	0-253
20	MS-New-ARAP-Password	String	0-253
21	MS-ARAP-Password-Change-Reason	ENUM: Just-Change-Password Expired-Password Admin-Requires-Password-Chang Password-Too-Short	1-4
22	MS-Filter	String	0-253
23	MS-Acct-Auth-Type	ENUM: PAP CHAP MS-CHAP-1 MS-CHAP-2 EAP	1-5
26	MS-CHAP2-Success	String	43-43
27	MS-CHAP2-CPW8	String	68-68
29	MS-Secondary-DNS-Server	IP Address	68-68
31	MS-Secondary-NBNS-Server	IP Address	70-70
33	MS-ARAP-Challenge	String	8-8

## Nomadix VSAs

Table C-14 lists the Nomadix VSAs. The vendor ID for Nomadix VSAs is 3309.

**Table C-14 Nomadix VSAs**

SubAttr	VSA Name	Type	Min-Max Value
1	Nomadix-Bw-Up 0 253	UINT32	0-253
2	Nomadix-Dw-Down	UINT32	0-253

## RedBack VSAs

Table C-16 lists the RedBack VSAs. The vendor ID for RedBack VSAs is 2352.

**Table C-15 RedBack VSAs**

SubAttr	VSA Name	Type	Min-Max Value
1	RedBack-Client-DNS-Pri	String	0-253
2	RedBack-Client-DNS-Sec	String	0-253
3	RedBack-DHCP-Max-Leases	String	0-253
4	RedBack-Context-Name	String	0-253
5	RedBack-Bridge-Group	String	0-253
6	RedBack-BG-Aging-Time	String	0-253
7	RedBack-BG-Path-Cost	String	0-253
8	RedBack-BG-Span-Dis	String	0-253
9	RedBack-BG-Trans-BPDU	String	0-253
10	RedBack-Rate-Limit-Rate	String	0-253
11	RedBack-Rate-Limit-Burst	String	0-253
12	RedBack-Police-Rate	String	0-253
13	RedBack-Police-Burst	String	0-253
14	RedBack-Source-Validation	String	0-253
15	RedBack-Tunnel-Domain	String	0-253
16	RedBack-Tunnel-Local-Name	String	0-253
17	RedBack-Tunnel-Remote-Name	String	0-253
18	RedBack-Tunnel-Function	String	0-253
21	RedBack-Tunnel-Max-Sessions	String	0-253
22	RedBack-Tunnel-Max-Tunnels	String	0-253
23	RedBack-Tunnel-Session-Auth	String	0-253
24	RedBack-Tunnel-Window	String	0-253
25	RedBack-Tunnel-Retransmit	String	0-253
26	RedBack-Tunnel-Cmd-Timeout	String	0-253

## Vendor Specific Attributes

Table C-15 RedBack VSAs

SubAttr	VSA Name	Type	Min-Max Value
27	RedBack-PPPOE-URL	String	0-253
28	RedBack-PPPOE-MOTM	String	0-253
29	RedBack-Tunnel-Group	String	0-253
30	RedBack-Tunnel-Context	String	0-253
31	RedBack-Tunnel-Algorithm	String	0-253
32	RedBack-Tunnel-Deadtime	String	0-253
33	RedBack-Mcast-Send	String	0-253
34	RedBack-Mcast-Receive	String	0-253
35	RedBack-Mcast-MaxGroups	String	0-253
36	RedBack-Ip-Address-Pool-Name	String	0-253
37	RedBack-Tunnel-DNIS	String	0-253
38	RedBack-Medium-Type	String	0-253
39	RedBack-PVC-Encapsulation-Type	String	0-253
40	RedBack-PVC-Profile-Name	String	0-253
41	RedBack-PVC-Circuit-Padding	String	0-253
42	RedBack-Bind-Type	String	0-253
43	RedBack-Bind-Auth-Protocol	String	0-253
44	RedBack-Bind-Auth-Max-Sessions	String	0-253
45	RedBack-Bind-Bypass-Bypass	String	0-253
46	RedBack-Bind-Auth-Context	String	0-253
47	RedBack-Bind-Auth-Service-Grp	String	0-253
48	RedBack-Bind-Bypass-Context	String	0-253
49	RedBack-Bind-Int-Context	String	0-253
50	RedBack-Bind-Tun-Context	String	0-253
51	RedBack-Bind-Ses-Context	String	0-253
52	RedBack-Bind-Dot1q-Slot	String	0-253
53	RedBack-Bind-Dot1q-Port	String	0-253
54	RedBack-Bind-Dot1q-Vlan-Tag-Id	String	0-253
55	RedBack-Bind-Int-Interface-Name	String	0-253
56	RedBack-Bind-L2TP-Tunnel-Name	String	0-253
57	RedBack-Bind-L2TP-Flow-Control	String	0-253
58	RedBack-Bind-Sub-User-At-Context	String	0-253
59	RedBack-Bind-Sub-Password	String	0-253
60	RedBack-Ip-Host-Addr	String	0-253
61	RedBack-IP-TOS-Field	String	0-253
62	RedBack-NAS-Real-Port	String	0-253

Table C-15 RedBack VSAs

SubAttr	VSA Name	Type	Min-Max Value
63	RedBack-Tunnel-Session-Auth-Ctx	String	0-253
64	RedBack-Tunnel-Session-Auth-Service-Grp	String	0-253
65	RedBack-Tunnel-Rate-Limit-Rate	String	0-253
66	RedBack-Tunnel-Rate-Limit-Burst	String	0-253
67	RedBack-Tunnel-Police-Rate	String	0-253
68	RedBack-Tunnel-Police-Burst	String	0-253
69	RedBack-Tunnel-L2F-Second-Password	String	0-253
128	RedBack-Acct-Input-Octets-64	String	0-253
129	RedBack-Acct-Output-Octets-64	String	0-253
130	RedBack-Acct-Input-Packets-64	String	0-253
131	RedBack-Acct-Output-Packets-64	String	0-253
132	RedBack-Assigned-IP-Address	String	0-253
133	RedBack-Acct-Mcast-In-Octets	String	0-253
134	RedBack-Acct-Mcast-Out-Octets	String	0-253
135	RedBack-Acct-Mcast-In-Packets	String	0-253
136	RedBack-Acct-Mcast-Out-Packets	String	0-253
137	RedBack-LAC-Port	String	0-253
138	RedBack-LAC-Real-Port	String	0-253
139	RedBack-LAC-Port-Type	String	0-253
140	RedBack-LAC-Real-Port-Type	String	0-253

## RedCreek VSAs

Table C-16 lists the RedCreek VSAs. The vendor ID for RedCreek VSAs is 1958.

Table C-16 RedCreek VSAs

SubAttr	VSA Name	Type	Min-Max Value
6	RedCreek-Tunneled-IP-Netmask	IP Address	0-253
7	RedCreek-Tunneled-Gateway	IP Address	0-253
9	RedCreek-Tunneled-WINS-Server1	String	0-253
10	RedCreek-Tunneled-WINS-Server2	String	0-253
11	RedCreek-Tunneled-HostName	String	0-253
12	RedCreek-Tunneled-DomainName	String	0-253
13	RedCreek-Tunneled-Search-List	String	0-253

## Telebit VSAs

Table C-17 lists the Telebit VSAs. The vendor ID for Telebit VSAs is 117.

**Table C-17 Telebit VSAs**

SubAttr	VSA Name	Type	Min-Max Value
1	Telebit-Login-Command	String	0-253
2	Telebit-Port-Name	String	0-253
3	Telebit-Activate-Command	String	0-253
4	Telebit-Accounting-Info	String	0-253
5	Telebit-Login-Option	String	0-253

## Unisphere VSAs

Table C-18 lists the Unisphere VSAs. The vendor ID for RedBack VSAs is 4874.

**Table C-18 Unisphere VSAs**

SubAttr	VSA Name	Type	Min-Max Value
1	Unisphere-Virtual-Router	String	0-253
2	Unisphere-Local-Address-Pool	String	0-253
3	Unisphere-Local-Interface	String	0-253
4	Unisphere-Primary-DNS	String	0-253
5	Unisphere-Secondary-DNS	String	0-253
6	Unisphere-Primary-WINS	String	0-253
7	Unisphere-Secondary-WINS	String	0-253
8	Unisphere-Tunnel-Virtual-Router	String	0-253
9	Unisphere-Tunnel-Password	String	0-253
10	Unisphere-Ingress-Policy-Name	String	0-253
11	Unisphere-Egress-Policy-Name	String	0-253
12	Unisphere-Ingress-Statistics	String	0-253
13	Unisphere-Egress-Statistics	String	0-253
14	Unisphere-Service-Category	String	0-253
15	Unisphere-PCR	String	0-253
16	Unisphere-SCR	String	0-253
17	Unisphere-MBS	String	0-253
18	Unisphere-Init-CLI-Access-Level	String	0-253
19	Unisphere-Allow-All-VR-Access	String	0-253
20	Unisphere-Alt-CLI-Access-Level	String	0-253
21	Unisphere-Alt-CLI-VR-Router-Name	String	0-253

**Table C-18 Unisphere VSAs**

<b>SubAttr</b>	<b>VSA Name</b>	<b>Type</b>	<b>Min-Max Value</b>
22	Unisphere-SA-Validate	String	0-253
23	Unisphere-IGMP-enable	String	0-253
24	Unisphere-PPPoE-Description	String	0-253
25	Unisphere-Redirect-VRouter-Name	String	0-253

## USR VSAs

Table C-19 lists the USR VSAs. The vendor ID for USR VSAs is 429.

**Table C-19 USR VSAs**

SubAttr	VSA Name	Type	Min-Max Value
1	USR-DTE-Data-Idle-Timeout	UINT32	0-0
2	USR-Default-DTE-Data-Rate	ENUM: 110_BPS 300_BPS 600_BPS 1200_BPS 2400_BPS 4800_BPS 7200_BPS 9600_BPS 12K_BPS 14.4K_BPS 16.8_BPS 19.2K_BPS 38.4K_BPS 75_BPS 450_BPS UNKNOWN_BPS 57.6K_BPS 21.6K_BPS 24K_BPS 26K_BPS 28K_BPS 115K_BPS 31K_BPS 33K_BPS 25333_BPS 110_BPS 300_BPS 600_BPS 1200_BPS 2400_BPS 26666_BPS 28000_BPS 29333_BPS 30666_BPS 32000_BPS	1-54

Table C-19 USR VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
2	USR-Default-DTE-Data-Rate	33333_BPS 34666_BPS 36000_BPS 37333_BPS 38666_BPS 40000_BPS 41333_BPS 42666_BPS 44000_BPS 45333_BPS 46666_BPS 48000_BPS 49333_BPS 50666_BPS 52000_BPS 53333_BPS 54666_BPS 56000_BPS 57333_BPS 58666_BPS 60000_BPS 61333_BPS 62666_BPS 64000_BPS	
3	USR-Last-Number-Dialed-Out	String	1-253
4	USR-Sync-Async-Mode	ENUM: Asynchronous Synchronous	1-2
5	USR-Originate-Answer-Mode	ENUM: Originate_in_Originate_Mode Originate_in_Answer_Mode Answer_in_Originate_Mode Answer_in_Answer_Mode	1-4
6	USR-Failure-to-Connect-Reason	ENUM:	1-67

Table C-19 USR VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
7	USR-Initial-Tx-Link-Data-Rate	ENUM: 110_BPS 14.4K_BPS 16.8_BPS 19.2K_BPS 38.4K_BPS 75_BPS 450_BPS UNKNOWN_BPS 57.6K_BPS 21.6K_BPS 24K_BPS 300_BPS 26K_BPS 28K_BPS 115K_BPS 31K_BPS 33K_BPS 25333_BPS 26666_BPS 28000_BPS 29333_BPS 30666_BPS 600_BPS 32000_BPS 33333_BPS 34666_BPS 36000_BPS 37333_BPS 38666_BPS 40000_BPS 41333_BPS 42666_BPS 44000_BPS 1200_BPS 45333_BPS 46666_BPS 48000_BPS 49333_BPS 50666_BPS 52000_BPS 53333_BPS 54666_BPS 56000_BPS 57333_BPS 2400_BPS 58666_BPS 60000_BPS	1-54

Table C-19 USR VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
7	USR-Initial-Tx-Link-Data-Rate (continued)	61333_BPS 62666_BPS 64000_BPS 4800_BPS 7200_BPS 9600_BPS 12K_BPS	
8	USR-Final-Tx-Link-Data-Rate	ENUM: 110_BPS 14.4K_BPS 16.8_BPS 19.2K_BPS 38.4K_BPS 75_BPS 450_BPS UNKNOWN_BPS 57.6K_BPS 21.6K_BPS 24K_BPS 300_BPS 26K_BPS 28K_BPS 115K_BPS 31K_BPS 33K_BPS 25333_BPS 26666_BPS 28000_BPS 29333_BPS 30666_BPS 600_BPS	1-54
8	USR-Final-Tx-Link-Data-Rate	32000_BPS 33333_BPS 34666_BPS 36000_BPS 37333_BPS 38666_BPS 40000_BPS 41333_BPS 42666_BPS 44000_BPS 1200_BPS 45333_BPS 46666_BPS 48000_BPS 49333_BPS 50666_BPS 52000_BPS 53333_BPS	1-54

Table C-19 USR VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
8	USR-Final-Tx-Link-Data-Rate	54666_BPS 56000_BPS 57333_BPS 2400_BPS 58666_BPS 60000_BPS 61333_BPS 62666_BPS 64000_BPS 4800_BPS 7200_BPS 9600_BPS	
9	USR-Modulation-Type	ENUM: usRoboticsHST bell208b v21FaxClass1 v27FaxClass1 v29FaxClass1 v17FaxClass1 v21FaxClass2 v27FaxClass2 v29FaxClass2 v17FaxClass2 v32Terbo ccittV32 v34 vFC v34plus x2 v110 v120 x75 ayncSyncPPP clearChannel ccittV22bis bell103 ccittV21 bell212 ccittV32bis ccittV23 negotiationFailed	1-28
9	USR-Modulation-Type	ENUM:	
10	USR-Equalization-Type	ENUM: Long Short	1-2
112	USR-Characters-Sent	UINT32	0-0
13	USR-Characters-Received	UINT32	0-0

Table C-19 USR VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
14	USR-Blocks-Sent	UINT32	0-0
15	USR-Blocks-Received 0	UINT32	0-0
16	USR-Blocks-Resent	UINT32	0-0
17	USR-Retrains-Requested	UINT32	0-0
18	USR-Retrains-Granted	UINT32	
19	USR-Line-Reversals	UINT32	
20	USR-Number-Of-Characters-Lost0	UINT32	0-0
21	USR-Back-Channel-Data-Rate	ENUM : 450BPS 300BPS None	1-3
22	USR-Number-of-Blers	UINT32	0-0
23	USR-Number-of-Link-Timeouts	UINT32	0-0
24	USR-Number-of-Fallbacks	UINT32	0-0
25	USR-Number-of-Upshifts	UINT32	0-0
26	USR-Number-of-Link-NAKs	UINT32	0-0
27	USR-Simplified-MNP-Levels	ENUM: Unknown NON_ARQ MNP10ec LAPMAC V42ETC2 V42SREJ PIAFS V120 X75 MNP3 MNP4 V42 HST synchronous MNP2 MNP10(Cellular) V42ETC	0-16

Table C-19 USR VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
28	USR-Connect-Term-Reason	ENUM: dtrDrop retransmitLimit linkDisconnectMsgReceived noLoopCurrent invalidSpeed unableToRetrain managementCommand noDialTone keyAbort lineBusy noAnswer escapeSequence voice noAnswerTone noCarrier undetermined v42SabmeTimeout v42BreakTimeout v42DisconnectCmd v42IdExchangeFail v42BadSetup v42InvalidCodeWord athCommand v42StringToLong v42InvalidCommand none v32Cleardown dialSecurity	1-67

Table C-19 USR VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
28	USR-Connect-Term-Reason	remoteAccessDenied loopLoss ds0Teardown promptNotEnabled noPromptingInSync carrierLoss nonArqMode modeIncompatible noPromptInNonARQ dialBackLink linkAbort autopassFailed pbGenericError pbLinkErrTxPreAck pbLinkErrTxTardyACK pbTransmitBusTimeout inactivityTimeout pbReceiveBusTimeout pbLinkErrTxTAL pbLinkErrRxTAL pbTransmitMasterTimeout pbClockMissing pbReceivedLsWhileLinkUp pbOutOfSequenceFrame pbBadFrame pbAckWaitTimeout pbReceivedAckSeqErr mnpIncompatible pbReceiveOvrflwRNRFail pbReceiveMsgBufOvrflw rcvdGatewayDiscCmd tokenPassingTimeout dspInterruptTimeout mnpProtocolViolation	
28	USR-Connect-Term-Reason	class2FaxHangupCmd hstSpeedSwitchTimeout undefined remotePassword linkPassword	
29	USR-DTR-False-Timeout	UINT32	0-0
30	USR-Fallback-Limit	UINT32	0-0
31	USR-Block-Error-Count-Limit	UINT32	0-0
32	USR-Simplified-V42bis-Usage	ENUM: None ccittV42bis mnpLevel5	1-3
33	USR-DTR-True-Timeou	UINT32	0-0

## Vendor Specific Attributes

Table C-19 USR VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
34	USR-Last-Number-Dialed-In-DNIS	String	1-253
35	USR-Last-Callers-Number-ANI	String	1-253
36	USR-Mbi-Ct-PRI-Card-Slot	UINT32	0-0
37	USR-Mbi-Ct-TDM-Time-Slot	UINT32	0-0
38	USR-Mbi-Ct-PRI-Card-Span-Line	UINT32	0-0
39	USR-Mbi-Ct-BChannel-Used	UINT32	0-0
40	USR-IP-Input-Filter	String	1-253
41	USR-IPX-Input-Filter	String	1-253
42	USR-IP-Output-Filter	String	1-253
43	USR-IPX-Output-Filter	String	1-253
44	USR-SAP-Output-Filter	String	1-253
45	USR-VPN-ID	UINT32	0-0
46	USR-VPN-Name	String	1-253
47	USR-VPN-Neighbor	String	1-253
48	USR-Framed-Routing-V2	ENUM: RIP-V2-Off RIP-V2-On	1-2
49	USR-VPN-Gateway	String	1-253
50	USR-Tunnel-Authenticato	String	1-253
51	USR-Packet-Index	String	1-253
52	USR-Cutoff	String	1-253
53	USR-Access-Accept-Packet	String	1-253
54	USR-Primary-DNS-Server	String	1-253
55	USR-Secondary-DNS-Server	String	1-253
56	USR-Primary-NBNS-Server	String	1-253
57	USR-Secondary-NBNS-Server	String	1-253
58	USR-Syslog-Tap	UINT32	0-0
59	USR-Chassis-Call-Slot	UINT32	0-0
60	USR-Chassis-Call-Span	UINT32	0-0
61	-Chassis-Call-Channel	UINT32	0-0
62	USR-Keypress-Timeout	UINT32	0-0
63	USR-Unauthenticated-Time	UINT32	0-0
64	USR-Bearer-Capabilities	UINT32	0-0
65	USR-Speed-Of-Connection	UINT32	0-0
66	USR-Max-Channels	UINT32	0-0
67	USR-Channel-Expansion	UINT32	0-0
68	USR-Channel-Decrement	UINT32	0-0

Table C-19 USR VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
69	USR-Expansion-Algorithm	UINT32	0-0
70	USR-Compression-Algorithm	UINT32	0-0
71	USR-Receive-Acc-Map	UINT32	0-0
72	USR-Transmit-Acc-Map	UINT32	0-0
73	USR-Compression-Reset-Mode	UINT32	0-0
74	USR-Min-Compression-Size	UINT32	0-0
75	USR-IP	UINT32	0-0
76	USR-IPX	UINT32	0-0
77	USR-Filter-Zones	UINT32	0-0
78	USR-Appletalk	UINT32	0-0
79	USR-Bridging	UINT32	0-0
80	USR-Spoofing	UINT32	0-0
81	USR-Host-Type	String	1-253
82	USR-Send-Name	UINT32	0-0
83	USR-Send-Password	String	1-253
84	USR-Start-Time	UINT32	0-0
85	USR-End-Time	UINT32	0-0
86	USR-Send-Script1	String	1-253
87	USR-Reply-Script1	String	1-253
88	USR-Send-Script2	String	1-253
89	USR-Reply-Script2	String	1-253
90	USR-Send-Script3	String	1-253
91	USR-Send-Script3 USR-Reply-Script3	String	1-253
92	USR-Send-Script4	String	1-253
93	USR-Reply-Script4	String	1-253
94	USR-Send-Script5	String	1-253
95	USR-Reply-Script5	String	1-253
96	USR-Send-Script6	String	1-253
97	USR-Reply-Script6	String	1-253
98	USR-Terminal-Type	String	1-253
99	USR-Appletalk-Network-Range	UINT32	0-0
100	USR-Local-IP-Address	String	1-253
101	USR-Routing-Protocol	UINT32	0-0
102	USR-Modem-Group	UINT32	0-0
103	USR-IPX-Routing	UINT32	0-0
104	USR-IPX-Wan	UINT32	0-0

Table C-19 USR VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
105	USR-IP-RIP-Policies	UINT32	0-0
106	USR-IP-RIP-Simple-Auth-Password	String	0-253
107	USR-IDS0-Call-Type	UINT32	0-0
108	USR-Call-Terminate-in-GMT	UINT32	0-0
109	USR-Call-Connect-in-GMT	UINT32	0-0
110	USR-Call-Arrival-in-GMT	UINT32	0-0
111	USR-Channel-Connected-To	UINT32	0-0
112	USR-Slot-Connected-To	UINT32	0-0
113	USR-Device-Connected-To	ENUM: None isdGateway quadModem	1-3
114	USR-NFAS-ID	UINT32	0-0
115	USR-Q931-Call-Reference-Value	UINT32	0-0
116	USR-Call-Event-Code	ENUM: notSupported noFreeIGW igwRejectCall igwSetupTimeout noFreeTdmTs bcReject ieReject chidReject progReject callingPartyReject calledPartyReject setup blocked analogBlocked digitalBlocked outOfService busy congestion protocolError noFreeBchannel inOutCallCollision usrSetup telcoDisconnect usrDisconnect noFreeModem modemsNotAllowed modemsRejectCall modemSetupTimeout	1-28
117	USR-DS0	UINT32	0-0
118	USR-DS0s	String	1-253

Table C-19 USR VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
119	USR-Gateway-IP-Address	IP Address	0-0
120	USR-Physical-State	UINT32	0-0
121	USR-Chassis-Temp-Threshold	UINT32	0-0
122	USR-Card-Type	ENUM: SlotEmpty QuadV32DigitalModemNAC DualT1NIC DualAlogMdmNIC QuadDgtlMdmNIC QuadAlogDgtlMdmNIC TokenRingNIC SingleT1NIC EthernetNIC ShortHaulDualT1NIC DualAlogMgdIntlMdmNIC X25NIC	
122	USR-Card-Type (continued)	ENUM: QuadAlogNonMgdMdmNIC QuadAlogMgdIntlMdmNIC QuadAlogNonMgdIntlMdmNIC QuadLsdLiMgdMdmNIC QuadLsdLiNonMgdMdmNIC QuadLsdLiMgdIntlMdmNIC QuadLsdLiNonMgdIntlMdmNIC EthernetWithV35NIC HSEthernetWithoutV35NIC DualHighSpeedV35NIC QuadV35RS232LowSpeedNIC DualE1NIC ShortHaulDualE1NIC BellcoreLongHaulDualT1NIC BellcoreShrtHaulDualT1NIC SCSIEdgeServerNIC QuadV32AnalogModemNAC QuadV32DigAnlModemNAC QuadV34DigModemNAC QuadV34AnlModemNAC QuadV34DigAnlModemNAC SingleT1NAC EthernetGatewayNAC AccessServer 486TrGatewayNAC SlotUnknown	

Table C-19 USR VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
122	USR-Card-Type (continued)	ENUM: 486EthernetGatewayNAC DualRS232NAC 486X25GatewayNAC ApplicationServerNAC ISDNGatewayNAC ISDNpriT1NAC ClkedNetMgtCard ModemPoolManagementNAC NetwMgtCard ModemPoolNetserverNAC (continued)	1-1027
122	USR-Card-Type (continued)	ModemPoolV34ModemNAC ModemPoolISDNnNAC NTServerNAC QuadV34DigitalG2NAC QuadV34AnalogG2NAC QuadV34DigAnlgG2NAC NETServerFrameRelayNAC NETServerTokenRingNAC X2524ChannelNAC DualT1NAC WirelessGatewayNac EnhancedAccessServer EnhancedISDNGatewayNAC DualModemNAC QuadModemNAC TrGatewayNAC X25GatewayNAC DualV34ModemNAC	
123	USR-Security-Login-Limit	UINT32	0-0
124	USR-Security-Resp-Limit	UINT32	0-0
125	USR-Packet-Bus-Session	UINT32	0-0
126	USR-DTE-Ring-No-Answer-Limit	UINT32	0-0

Table C-19 USR VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
127	USR-Final-Rx-Link-Data-Rate	ENUM: 110_BPS 14.4K_BPS 16.8_BPS 19.2K_BPS 38.4K_BPS 75_BPS 450_BPS UNKNOWN_BPS 57.6K_BPS 21.6K_BPS 24K_BPS 300_BPS 6K_BPS 28K_BPS 115K_BPS 31K_BPS 33K_BPS 25333_BPS 26666_BPS 28000_BPS 62666_BPS 9333_BPS 30666_BPS 600_BPS (continued)	1-54

Table C-19 USR VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
127	USR-Final-Rx-Link-Data-Rate (continued)	32000_BPS 33333_BPS 34666_BPS 36000_BPS 37333_BPS 38666_BPS 40000_BPS 41333_BPS 42666_BPS 44000_BPS 1200_BPS 45333_BPS 46666_BPS 48000_BPS 49333_BPS 50666_BPS 52000_BPS 53333_BPS 54666_BPS 56000_BPS 57333_BPS 2400_BPS 58666_BPS 60000_BPS 61333_BPS 64000_BPS 800_BPS 7200_BPS 9600_BPS 12K_BPS	

Table C-19 USR VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
128	USR-Initial-Rx-Link-Data-Rate	ENUM: 110_BPS 14.4K_BPS 16.8_BPS 19.2K_BPS 38.4K_BPS 75_BPS 450_BPS UNKNOWN_BPS 57.6K_BPS 21.6K_BPS 24K_BPS 300_BPS 26K_BPS 28K_BPS 115K_BPS 31K_BPS 33K_BPS 25333_BPS 26666_BPS	1-54
128	USR-Initial-Rx-Link-Data-Rate	28000_BPS 29333_BPS 30666_BPS 600_BPS 32000_BPS 33333_BPS 34666_BPS 36000_BPS 37333_BPS 38666_BPS 40000_BPS 41333_BPS 42666_BPS 44000_BPS 1200_BPS 45333_BPS 46666_BPS 48000_BPS 49333_BPS 50666_BPS 52000_BPS 53333_BPS	

Table C-19 USR VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
128	USR-Initial-Rx-Link-Data-Rate	54666_BPS 56000_BPS 57333_BPS 2400_XBPS 58666_BPS 60000_BPS 61333_BPS 62666_BPS 64000_BPS 4800_BPS 7200_BPS 9600_BPS 12K_BPS	
129	USR-Event-Date-Time	UINT32	0-0
130	USR-Chassis-Temperature	UINT32	0-0
131	USR-Actual-Voltage	UINT32	0-0
132	USR-Expected-Voltage	UINT32	0-0
133	USR-Power-Supply-Number	UINT32	0-0
134	USR-Channel	UINT32	0-0
135	USR-Chassis-Slot	UINT32	0-0

Table C-19 USR VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
136	USR-Event-Id	ENUM: HUB_Temp_Out_of_Range Fan_Failed Watchdog_Timeout Mgmt_Bus_Failure In_Connection_Est Out_Connection_Est In_Connection_Term Out_Connection_Term Connection_Failed Connection_Timeout DTE_Transmit_Idle DTR_True DTR_False Block_Error_at_Threshold Fallbacks_at_Threshold No_Dial_Tone_Detected No_Loop_Current_Detected Yellow_Alarm Red_Alarm Loss_Of_Signal Rcv_Alm_Ind_Signal Timing_Source_Switch Modem_Reset_by_DTE Modem_Ring_No_Answer DTE_Ring_No_Answer Pkt_Bus_Session_Active Pkt_Bus_Session_Congestion Pkt_Bus_Session_Lost Pkt_Bus_Session_Inactive User_Interface_Reset Gateway_Port_Out_of_Service Gateway_Port_Link_Active Dial_Out_Login_Failure Dial_In_Login_Failure Dial_Out_Restricted_Number Dial_Back_Restricted_Number User_Blacklisted Attempted_Login_Blacklisted Response_Attempt_Limit_Exceed Login_Attempt_Limit_Exceeded Dial_Out_Call_Duration Dial_In_Call_Duration Pkt_Bus_Session_Err_Status NMC_AutoRespns Trap Acct_Server_Contact_Loss Yellow_Alarm_Clear Red_Alarm_Clear Loss_Of_Signal_Clear (Continued)	

Table C-19 USR VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
136	USR-Event-Id ( <i>Continued</i> )	Rcv_Alrm_Ind_Signal_Clear Incoming_Connection_Establish Module_Inserted Outgoing_Connection_Establish Incoming_Connection_Terminate Outgoing_Connection_Terminate Connection_Attempt_Failure Continuous_CRC_Alarm Continuous_CRC_Alarm_Clear Physical_State_Change Module_Removed Gateway_Network_Failed Gateway_Network_Restored Packet_Bus_Clock_Lost Packet_Bus_Clock_Restored D_Channel_In_Service D_Channel_Out_of_Service DSOs_In_Service DSOs_Out_of_Service T1/T1PRI/E1PRI_Call_Event PSU_Voltage_Alarm Psu_Incompatible T1,T1-E1/PRI-Call-Arrive-Even T1,T1-E1/PRI-Call-Connect-Eve T1,T1-E1/PRI-Call-Termina-Eve T1,T1-E1/PRI-Call-Failed-Even	6-84
137	USR-Number-of-Rings-Limit	UINT32	0-0
138	USR-Connect-Time-Limit	UINT32	0-0
139	USR-Call-End-Date-Time	UINT32	0-0
140	USR-Call-Start-Date-Time	UINT32	0-0
141	USR-Server-Time	UINT32	0-0
142	USR-Request-Type	ENUM: Access-Request Access-Challenge Status-Server Status-Client Access-Accept Reserved Access-Reject Accounting-Request Accounting-Response Access-Password-Change Access-Password-Ack Access-Password-Reject	1-255
143	USR-Old-Password	String	0-253
144	USR-Expiration	UINT32	0-0

Table C-19 USR VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
145	USR-Prompt	UINT32	0-1
146	USR-Char-Noecho	UINT32	0-0
147	USR-User-Group-Name	String	0-253
148	148 USR-Call-Reference-Number	UINT32	0-253
149	USR-Dial-In-Sec-Mode	UNIT32	0-0
150	USR-Req-Db-Mdm-Sel	UINT32	0-0
151	USR-Req-Db-Login-Valid	UINT32	0-0
152	USR-Dialback-Group-Names	String	0-253
153	USR-Dial-In-Call-Rest	String	0-253
154	USR-Dial-Out-Call-Rest	String	0-253
155	USR-Logins-Before-Blacklist	UINT32	0-0
156	USR-Failed-Logins	UINT32	0-0
157	USR-Allowed-DB-Modems	String	0-253
158	USR-VPN-Encrypter	String	0-253
159	USR-Acct-VPN-Gateway	String	0-253
160	USR-Re-CHAP-Timeout	UINT32	0-0
161	USR-RMMIE-Manufacture-ID	String	0-253
162	USR-RMMIE-Product-Code	String	0-253
163	USR-RMMIE-Serial-Number	String	0-253
164	USR-RMMIE-Firmware-Version	String	0-253
165	USR-RMMIE-Firmware-Build-Date	String	0-253
166	USR-RMMIE-Status	ENUM: notEnabledInLocalModem notDetectedInRemoteModem ok	1-3
170	USR-RMMIE-Last-Update-Time	UINT32	0-253
171	USR-RMMIE-Last-Update-Event	ENUM: None initialConnection retrain speedShift plannedDisconnect	1-5
172	USR-RMMIE-Rcv-Tot-PwrLvl	UNIT32	0-253
173	USR-RMMIE-Rcv-PwrLvl-3300Hz	UNIT32	0-253
174	USR-RMMIE-Rcv-PwrLvl-3750Hz	UNIT32	0-253
175	USR-RMMIE-PwrLvl-NearEcho-Canc	UNIT32	0-253
176	USR-RMMIE-PwrLvl-FarEcho-Canc	UNIT32	0-253
177	USR-RMMIE-PwrLvl-Noise-Lvl	UNIT32	0-253
178	USR-RMMIE-PwrLvl-Xmit-Lvl	UNIT32	0-253

## Vendor Specific Attributes

Table C-19 USR VSAs (continued)

SubAttr	VSA Name	Type	Min-Max Value
179	USR-IPX-SAP	String	0-253
180	USR-MIC	UNIT32	0-253
181	USR-Call-Tracking-ID	UNIT32	0-253
182	USR-Log-Filter-Packet	UNIT32	0-253
183	USR-CCP-Algorithm	UNIT32	0-253
184	USR-ACCM-Type	UNIT32	0-253
185	USR-Connect-Speed	UNIT32	0-253
186	USR-Framed-IP-Address-Pool-Name	UNIT32	0-253
187	USR-MP-EDO	String	0-253
188	USR-Local-Framed-IP-Addr	UNIT32	0-253
189	USR-IP-RIP-Input-Filter	String	0-253
190	USR-IP-Call-Input-Filter	String	0-253
191	USR-IPX-Call-Input-Filter	String	0-253
192	USR-AT-Input-Filter	String	0-253
193	USR-AT-RTMP-Input-Filter	String	0-253
194	USR-AT-Zip-Input-Filter	String	0-253
195	USR-AT-Call-Input-Filter	String	0-253
196	USR-ET-Bridge-Input-Filter	String	0-253
197	USR-IP-RIP-Output-Filter	String	0-253
198	USR-IP-Call-Output-Filter	String	0-253
199	USR-IPX-RIP-Output-Filter	String	0-253
200	USR-IPX-Call-Output-Filter	String	0-253
201	USR-AT-Output-Filter	String	0-253
202	USR-ET-RTMP-Output-Filter	String	0-253
203	USR-AT-Zip-Output-Filter	String	0-253
204	USR-AT-Call-Output-Filter	String	0-253
205	USR-ET-Bridge-Output-Filter	String	0-253
206	USR-ET-Bridge-Call-Output-Filter	String	0-253
207	USR-IP-Default-Route-Option	UINT32	0-253
208	USR-MP-EDO-HIPER	String	0-253
209	USR-MP-MRRU	UINT32	0-253

**A**

## AA

- services 3-10

- AAAFileServiceSyncInterval 3-28

- AcceptAll 3-8

- Accepted-Profiles B-1

- Access-Challenge 1-2

- Access Registrar

- backups 14-1

- definition 1-1

- dictionaries 6-1

- internal database 14-1

- objects 3-1, 3-3

- server 3-1

- Access-Reject B-5

- Access-Request 4-2, 4-6, 4-7

- Accounting

- attributes 1-4

- database 1-1

- definition 1-1

- log file 3-10

- Accounting-Service B-1

- ACKaccounting 3-23

- addProfile method A-2

- AdvancedDuplicateDetectionMemoryInterval 3-29

- Advanced objects 3-1

- APPEND A-2, A-5, A-7, A-8

- aregcmd

- Access Registrar command 2-1

- commands 2-3

- add 2-3

- cd 2-3

- delete 2-4

- exit 2-4

- filter 2-4

- find 2-5

- help 2-5

- insert 2-5

- login 2-5

- logout 2-6

- ls 2-6

- next 2-6

- prev 2-7

- pwd 2-7

- query-sessions 2-7

- quit 2-7

- release-sessions 2-7

- reload 2-8

- save 2-8

- set 2-9

- start 2-10

- stats 2-10

- status 2-11

- stop 2-11

- trace 2-12

- unset 2-13

- validate 2-13

- definition 2-1

- error codes 2-15

- save 5-3

- session management commands 3-13

- syntax 2-1

- ARIsCaseInsensitive 3-30

- Attribute Dictionary 1-4, 3-33, A-1

- methods A-1

- put method A-2

- Attributes 3-20, C-1

- alphabetical list C-2

- check item 9-22

- numeric list C-4

- AUGMENT A-2, A-5, A-7, A-8

- Authorization

- definition 1-1

**B**

- Backups 14-1

BaseProfile 3-4, 3-5  
 BindName 3-23  
 BindPassword 3-24

## C

Callback-Number 1-4  
 callsPerSecond 4-10  
 Case insensitive commands  
   see also aregcmd  
 cd command 2-1  
 CertificateDBPath 3-29  
 change directory command  
   see also aregcmd  
 CHAP  
   Access Request packet 4-2  
 CHAP\_ PASSWORD  
   attribute type 3-33  
 Check item attributes 9-22  
 CiscoWithODAPIncomingScript 8-1, 8-3  
 clear method A-2  
 Client/server model 1-1  
 Client-Behind-the-Proxy 6-2  
 Clients  
   IPAddress 3-6  
   list 3-30  
   objects 3-1  
   vendor properties 3-6  
 ConfigurationError reply message 3-32  
 Configuration Objects 2-2  
 Configuring  
   check item attributes 9-23  
   LDAP RemoteServer 12-2  
   local service 5-2  
   ODBC RemoteServer 13-3  
 containsKey method A-2

## D

Database  
   Access Registrar backups 14-1  
   MCD 5-3  
 DefaultAccountingService 3-3  
 DefaultAuthenticationService 3-2  
 DefaultAuthorizationService 3-2  
 DefaultReturnedSubnetSizeIfNoMatch 3-29  
 DefaultSessionManager 3-3  
 DefaultSessionService 3-3  
 Dictionaries  
   Types of 6-1  
 Dictionary  
   attribute 3-30  
 DNSLookupAndLDAPRebindInterval 12-5  
 DropPacket. 3-8

## E

EAP 3-9  
   authentication mechanism 3-9  
 EAP-LEAP 3-10  
 EAP-MD5 3-10  
 Easysoft Open Source 13-6  
 Empty string 2-1  
 EntryPoint 3-12  
 ENUM  
   attribute type 3-34  
 Environment Dictionary 6-1, 6-3  
 Environment Dictionary script 6-4  
 Environment variable  
   Accounting-Service B-1  
   Acquire-Group-Session-Limit B-2  
   Acquire-IP-Dynamic B-2  
   Acquire-IP-Per-NAS-Port B-2  
   Acquire-IPX-Dynamic B-2  
   Acquire-Subnet-Dynamic B-2  
   Acquire-User-Session-Limit B-2

Acquire-USR-VPN **B-2**  
 Allow-Null-Password **B-2**  
 Authentication-Service **B-3**  
 Authorization-Service **B-3**  
 Current-Group-Count **B-3**  
 Dynamic-Search-Path **B-3**  
 Group-Session-Limit **B-3**  
 Ignore-Accounting-Signature **B-3**  
 Incoming-Translation-Groups **B-3**  
 Misc-Log-Msg-Info **B-4**  
 Reject-Reason **B-4**  
 Remote-Server **B-4**  
 Request-Authenticator **B-4**  
 Request-Type **B-4**  
 Require-User-To-Be-In-Authorization-List **B-5**  
 Response-Type **B-5**  
 Session-Key **B-6**  
 Session-Manager **B-6**  
 Session-Service **B-6**  
 Source-IP-Address **B-7**  
 Trace-Level **B-7**  
 Unavailable-Resource **B-7**  
 Unavailable-Resource-Type **B-7**  
 User Authorization-Script **B-8**  
 User-Group **B-8**  
 User-Group-Session-Limit **B-8**  
 User-Name **B-8**  
 User-Profile **B-8**  
 User-Session-Limit **B-8**

Error codes

- aregcmd **2-15**

ExecPrefixRule **10-8**

Extension points **6-1**

---

## F

Failover policy **3-10**  
 Filename **3-12**  
 FilenamePrefix **3-11**

file service **3-8, 3-10**  
     FilenamePrefix **3-11**  
     MaxFileAge **3-11**  
     MaxFileSize **3-11**  
 Filter **3-24**  
 firstKey method **A-2**  
 Framed-IP-Address **1-4**  
 Framed Protocol **1-4**  
 FramedRouting **3-19**

---

## G

Gateway  
     Description **3-19**  
     IPAddress **3-19**  
     LocationID **3-19**  
     Name **3-19**  
     SharedSecret **3-19**  
     TunnelRefresh **3-19**  
 Gateways **3-19**  
 get method **A-2**  
 Group-Session-Limit Resource Manager **3-16**

---

## H

HiddenAttributes **3-4**  
 HostName **3-23**

---

## I

Identifier **3-19**  
 IncomingScript **3-2, 3-6, 3-7, 3-23**  
 IncomingScriptFailed reply message **3-32**  
 IncomingScript RejectedRequest reply message **3-32**  
 Incoming scripts **1-2**  
 InitEntryPoint **3-8, 3-12**  
 InitEntryPointArgs **3-12**  
 InitialBackgroundTimerSleepTime **3-28**

InitialTimeout 3-23, 3-26  
 Interfaces properties 3-1  
 InternalError reply message 3-32  
 IPADDR  
   attribute type 3-33  
 IPAddress 3-6  
 IP-Dynamic Resource Manager 3-15  
 IP-Per-NAS-Port Resource Manager 3-15  
 IPX-Dynamic Resource Manager 3-16  
 isEmpty method A-2

---

**J**

JavaVMOptions 3-29

---

**L**

LDAP 12-1  
   MultipleServersPolicy 12-2  
   protocol 3-23  
   RemoteServers 3-22  
 ldap  
   BindName 3-23  
   BindPassword 3-24  
   Filter 3-24  
   HostName 3-23  
   LDAPToEnvironmentMappings 3-25  
   LDAPToRadiusMappings 3-25  
   LimitOutstandingRequests 3-24  
   MaxOutstandingRequests 3-24  
   MaxReferrals 3-24  
   PasswordEncryptionStyle 3-24  
   ReferralAttribute 3-24  
   ReferralFilter 3-24  
   SearchPath 3-24  
   Timeout 3-23, 3-26  
   UserPasswordAttribute 3-24  
   UseSSL 3-25

LDAP Rebind 12-4  
   failures 12-5  
 LDAP RemoteServer 12-2  
 LDAP server 1-4  
 LDAP service 12-1  
 LDAPToCheckItemMappings 3-25, 12-6  
 LDAPToEnvironmentMappings 3-25, 12-6  
 LDAPToRadiusMappings 3-25, 12-5  
 LEAP 3-9  
 Lightweight Directory Access Protocol 12-1  
 LimitOutstandingRequests 3-24  
 local 3-9, B-8  
   UserList type 3-3  
 localhost 4-6  
 Local Service 5-2  
 local service 3-3  
 LogFileCount 3-29  
 LogFileSize 3-29  
 Logging in 2-5  
 Logging out 2-6  
 login command 2-5  
 log method A-2  
 LogServerActivity 3-27

---

**M**

Malformed Request reply message 3-32  
 MarkerList 13-5  
 MaxFileAge 3-11  
 MaxFileSize 3-11  
 MaximumNumberOfRadiusPackets 3-27  
 MaximumNumberOfUDPTacacsPackets 3-28  
 MaximumODBCResultSize 3-29  
 MaxOutstandingRequests 3-24  
 MaxReferrals 3-24  
 MaxTries 3-23, 3-26  
 MCD 14-1  
 mcdcd.d01-d03 14-2  
 mcdConfig.txt 14-2

MCD database 5-3  
 mcddb.dbd 14-2  
 mcddb.k01-k03 14-2  
 mcshadow 14-1  
 MinimumSocketBufferSize 3-28  
 MPLS 8-1  
 MultipleServersPolicy 3-10, 12-2, 13-3

---

## N

Name=value 3-20  
 NAS 1-1  
 NAS IP Address 3-33  
 NAS-IP-Address 1-4  
 NAS-Port 1-4  
 NAS-Vendor-Behind-the-Proxy 6-2  
 Neighbor 3-19  
 nextKey method A-2

---

## O

ODAP  
   accounting service 8-7  
   address ranges 8-2  
   AllowAnonymousPassword property 8-6  
   CiscoIncomingScript 8-3  
   configuration summary 8-4  
   configuring 8-4  
   configuring clients 8-15  
   configuring Session Managers 8-13  
   detailed configuration 8-5  
   on-demand address pool 8-1  
   Resource Managers 8-9  
   service 8-6  
   Session Managers 8-8  
   userlist 8-5  
   users 8-5  
   vendor type 8-4

ODBC.ini file 13-2  
 ODBCDataSource 13-4, 13-6  
 ODBC RemoteServer 13-3  
 ODBC service 13-2  
 ODBCToEnvironmentMappings 13-6  
 ODBCToRadiusMappings 13-5  
 ORACLE\_HOME 13-1  
 Oracle Driver  
   Easysoft Open Source 13-6  
 Oracle functions 13-5  
 order dependent commands  
   see also aregcmd  
 OutagePolicy 3-8  
 OutageScript 3-8  
 OutgoingScript 3-2, 3-6, 3-7, 3-23  
 OutgoingScriptFailed 3-32  
 OutgoingScriptRejectedRequest 3-32  
 Outgoing scripts 1-3

---

## P

Packet fields 1-3  
 packet-identifier 4-3  
 Password  
   length of 3-4  
 PasswordEncryptionStyle 3-24  
 Ports 3-30  
 Ports properties 3-1  
 PPP 1-4, 3-19  
 Prefix feature 10-8  
   attributes 10-8  
 Profile objects 3-1  
 Proxy server 1-4  
 put method A-2

---

## Q

query-sessions command 3-13

**R**

## radclient

- callsPerSecond 4-10
- syntax 4-1
- timetest 4-9

## RADIUS

- attribute name 4-5
- attributes C-1
- messages 1-3
- packet type identifier 4-3
- program flow 1-2
- protocol 1-1
- server 2-2, 2-5, 3-5, 4-3, 6-4
- server test tool 4-1

RADIUS EXtension. See REX

RadiusServer object 3-1

ReactivateTimerInterval 3-22

ReferralAttribute 3-24

ReferralFilter 3-24

RejectAll 3-8

Reject-Reason B-4

release-sessions command 3-13

RemoteLDAPServiceThreadTimerInterval 3-28

RemoteRadiusServerInterface 3-30

RemoteServers 12-2, 13-3

Remote servers

- policy 3-10

RemoteServers objects 3-1

REMOVE\_ALL A-3, A-9

remove method A-3

REPLACE A-2, A-5, A-7, A-8

Replication 7-1

Reply Messages 3-31

RepTransactionArchiveLimit 7-1

RepTransactionSyncInterval 7-1

Request Dictionary 1-2, 6-1

- script 6-3

Request-Type Packets

Access-Accept B-5

Access-Challenge B-5

Access-Reject B-5

Access-Request B-5

Accounting-Request B-5

Accounting-Response B-5

Ascend-IPA-Allocate B-5

Ascend-IPA-Release B-5

Status-Client B-5

Status-Server B-5

USR-Enhanced-Radius B-5

USR-NAS-Reboot-Request B-5

USR-NAS-Reboot-Response B-5

USR-Resource-Free-Request B-5

USR-Resource-Free-Response B-5

USR-Resource-Query-Request B-5

USR-Resource-Query-Response B-5

RequireNASsBehindProxyBeInClientList 3-27, 3-30

Resource Managers 3-1, 3-15, 3-16

Group-Session-Limit 3-18

Home-Agent 3-18

IP-Dynamic 3-16

IP-Per-NAS-Port 3-16

IPX-Dynamic 3-17

subnet-dynamic 3-17

User-Session-Limit 3-18

USR-VPN 3-19

Response Dictionary 1-3, 6-1

script 6-3

Response-Type B-5

REX

scripts 3-7

REX attribute dictionary

getBytes method A-6

putBytes method A-8

REX environment dictionary

allocateMemory A-10

clear A-10

containsKey A-10

- firstKey **A-10**
- get **A-10**
- isEmpty **A-10**
- log **A-10**
- nextKey **A-10**
- put **A-10**
- remove **A-10**
- reschedule **A-11**
- size **A-11**
- trace **A-11**
- rex service
  - EntryPoint **3-12**
  - Filename **3-12**
  - InitEntryPoint **3-12**
  - InitEntryPointArgs **3-12**
- RFC 2138 **3-33, C-1**
- RoundRobin policy **3-10**

---

## S

- Scripting point **6-1**
  - NAS IncomingScript **6-4**
- Script objects **3-1**
- Scripts
  - adding script definition **6-4**
  - choosing the type of script **6-2**
  - determining goal **6-1**
  - extension points **6-1**
  - writing **6-2**
- Search key **13-5**
- SearchPath **3-24**
- Services
  - file **3-10**
  - ldap **3-10**
  - local **3-3, 3-8, 3-9, B-8**
  - proxy requests **3-22**
  - radius **3-10**
  - tacacs-udp **3-10**
- services **3-8**

- Services objects **3-1, 3-8**
- ServiceUnavailable reply message **3-32**
- SessionBackingStoreSynchronizationInterval **3-28**
- Session Management
  - definition **1-1**
- Session Managers **3-13**
- Session Managers objects **3-1**
- Setting attributes
  - spaces in value **2-10, 5-8**
- Shadow backups **14-1**
- Shared libraries **A-1**
- SharedSecret **3-19, 3-23**
- Shared secret **3-6**
  - definition **1-2**
- size method **A-3**
- SLIP **1-4**
- SNMP **3-30**
- SQLDefinition **13-4**
- SQL queries **13-4**
- SQLStatement **13-4**
- SQL syntax restrictions **13-4**
- stats command **B-4**
- sticky commands **2-6**
- STRING
  - attribute type **3-33**

---

## T

- tacacs-udp **3-10, 3-22**
- Tcl attribute dictionary **A-1, A-2**
  - addProfile method **A-2**
  - clear method **A-2**
  - firstKey method **A-2**
  - get method **A-2**
  - isEmpty method **A-2**
  - log method **A-2**
  - nextKey method **A-2**
  - remove method **A-3**
  - size method **A-3**

trace method **A-3**  
 TerminationAction reply message **3-32**  
 Timeout **3-23, 3-26**  
 timetest **4-9**  
 trace method **A-3**  
 TunnelRefresh **3-19**

---

## U

UDPPacketSize **3-27**  
 UINT32  
   attribute type **3-33**  
 UnableToAcquireResource reply message **3-32**  
 UNDEFINED  
   attribute type **3-33**  
 UnknownUser reply message **3-32**  
 use\_challenge parameter **4-2**  
 UseAdvancedDuplicateDetection **3-29**  
 UserDefined **3-4**  
 User extensions. See Scripts.  
 UserGroup objects **3-1**  
 UserGroups  
   check item attributes **9-23**  
 UserList **3-1**  
   check item attributes **9-23**  
 UserNotEnabled reply message **3-32**  
 UserPasswordAttribute **3-24**  
 UserPasswordInvalid **3-32**  
 User-Profile **B-8**  
 User properties **3-4**  
 UserService **3-10**  
 User-Session-Limit **B-8**  
 UserSessionLimit **3-18**  
 User-Session-Limit Resource Manager **3-16**  
 UseSSL **3-25**  
 USR-VPN  
   FramedRouting **3-19**  
   Gateways **3-19**  
   Identifier **3-19**

Neighbor **3-19**  
 USR-VPN Resource Manager **3-16**

---

## V

valueAsInt **4-5**  
 valueAsIPAddress **4-5**  
 Variables  
   environment **B-1**  
   radclient **4-8**  
 VENDOR\_SPECIFIC  
   attribute type **3-34**  
 VendorID **3-34**  
 Vendor objects **3-1**  
 Vendor specific attributes **C-13**  
   Ascend **C-23**  
   Cabletron **C-32**  
   Cisco **C-33**  
   Compatible **C-35**  
   Nomadix **C-37**  
   RedCreek **C-37, C-39**  
 Vendor-specific attributes  
   3GPP2 **C-13**  
   ACC **C-16**  
   Altiga **C-21**  
   Bay Networks **C-32**  
 Vendor-specific attributes  
   Telebit **C-40**  
 VHG/PE router **8-1**  
 VRFs **8-2**  
 VSAs **C-13**