CHAPTER 2

Understanding Cisco Access Registrar

This chapter describes the Cisco Access Registrar object structure, and explains when Cisco AR references each of these objects during the processing of client requests.

Cisco AR lets you manipulate configuration objects, which define the properties or behavior of the RADIUS server. Cisco AR also lets you invoke custom scripts to affect the behavior of the RADIUS server.

To better understand the role each of these objects plays in the program, it is helpful to look at the steps Cisco AR performs from receipt of an Access-Request packet to the sending of an Access-Response packet.

Cisco Access Registrar Hierarchy

Cisco Access Registrar's operation and configuration is based on a set of *objects*. These objects are arranged in a hierarchical structure much like the Windows 95 Registry or the UNIX directory structure. Cisco Access Registrar's objects can themselves contain subobjects, just as directories can contain subdirectories.

These objects include the following:

- Radius— the root of the configuration hierarchy
- UserLists—contains individual UserLists which in turn contain users
- UserGroups—contains individual UserGroups
- Clients—contains individual Clients
- Vendors—contains individual Vendors
- Scripts—contains individual Scripts
- Services—contains individual Services
- SessionManagers—contains individual Session Managers
- ResourceManagers—contains individual Resource Managers
- Profiles—contains individual Profiles
- RemoteServers—contains individual RemoteServers
- Advanced—contains Ports, Interfaces, Reply Messages, and the Attribute dictionary.

UserLists and Groups

Cisco Access Registrar lets you organize your user community through the configuration objects **UserLists**, **users**, and **UserGroups**.

- Use **UserLists** to group users by organization, such as Company A and Company B. Each list contains the actual names of the users.
- Use **users** to store information about particular users, such as name, password, group membership, base profile, and so on.
- Use **UserGroups** to group users by function, such as PPP, Telnet, or multiprotocol users. Groups allow you to maintain common authentication and authorization requirements in one place, and have them referenced by many users.

For more information about **UserLists** and **UserGroups**, refer to Access Registrar Server Objects in the *Cisco Access Registrar User's Guide*.

Profiles

Cisco Access Registrar uses **Profiles** that allow you to group RADIUS attributes to be included in an Access-Accept packet. These attributes include values that are appropriate for a particular user class, such as PPP or Telnet user. The user's base profile defines the user's attributes, which are then added to the response as part of the authorization process.

Although you can use Group or Profile objects in a similar manner, choosing whether to use one rather than the other depends on your site. If you require some choice in determining how to authorize or authenticate a user session, then creating specific profiles, and specifying a group that uses a script to choose among the profiles 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.

For more information about **Profiles**, refer to Access Registrar Server Objects in the *Cisco Access Registrar User's Guide*.

Scripts

Cisco Access Registrar allows you to create scripts you can execute at various points within the processing hierarchy.

- Incoming scripts—enable you to read and set the attributes of the request packet, and set or change the Environment dictionary variables. You can use the environment variables to control subsequent processing, such as specifying the use of a particular authentication service.
- Outgoing scripts—enable you to modify attributes returned in the response packet.

For more information about **Scripts**, refer to Access Registrar Server Objects in the *Cisco Access Registrar User's Guide*.

Services

Cisco Access Registrar uses *Services* to let you determine how authentication, authorization, and/or accounting are performed.

For example, to use Services for authentication:

- When you want the authentication to be performed by the Cisco Access Registrar RADIUS server, you can specify the **local** service. In this, case you must specify a specific **UserList**.
- When you want the authentication performed by another server, which may run an independent application on the same or different host than your RADIUS server, you can specify either a **radius**, **ldap**, or **tacacs-udp** service. In this case, you must list these servers by name.

When you have specified more than one authentication service, Cisco Access Registrar determines which one to use for a particular Access-Request by checking the following:

- When an incoming script has set the Environment dictionary variable **Authentication-Service** with the name of a Service, Cisco Access Registrar uses that service.
- Otherwise, Cisco Access Registrar uses the default authentication service. The default authentication service is a property of the **Radius** object.

Cisco Access Registrar chooses the authentication service based on the variable **Authentication-Service**, or the default. The properties of that Service, specify many of the details of that authentication service, such as, the specific user list to use or the specific application (possibly remote) to use in the authentication process.

For more information about Services, refer to Access Registrar Server Objects in the *Cisco Access Registrar User's Guide*.

Session Management Using Resource Managers

Cisco Access Registrar lets you track user sessions and 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 resources of a particular type as described below.

- IP-Dynamic—manages a pool of IP addresses and allows you to dynamically allocate IP addresses from that pool
- 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
- 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
- USR-VPN—manages Virtual Private Networks (VPNs) that use USR NAS Clients.

For more information about Session Managers, refer to Access Registrar Server Objects in the *Cisco Access Registrar User's Guide*.

If necessary, you can create a complex relationship between the Session Managers and the Resource Managers.

When you need to share a resource among Session Managers, you can create multiple Session Managers that refer to the same Resource Manager. For example, if one pool of IP addresses is shared by two departments, but each department has a separate policy about how many users can be logged in

concurrently, you might create two Session Managers and three Resource Managers: one dynamic IP Resource Manager that is referenced by both Session Managers, and two concurrent session Resource Managers, one for each Session Manager.

In addition, Cisco Access Registrar lets you pose queries about sessions. For example, you can query Cisco Access Registrar about which session (and thus which NAS-Identifier, NAS-Port and/or User-Name) owns a particular resource, as well as query Cisco Access Registrar about how many resources are allocated or how many sessions are active.

Program Flow

When a NAS sends a request packet to Cisco Access Registrar with a name and password, Cisco Access Registrar performs the following actions. Note, Table 2-1 describes the flow without regard to scripting points.

Table 2-1 From Access-Request to Access-Accept

CAR Server Action	Explanation
Receives an Access-Request.	The Cisco Access Registrar server receives an Access-Request packet from a NAS.
Determines whether to accept the request.	The Cisco Access Registrar server checks to see if the client's IP address is listed in **/Radius/Clients/**<*Name*>**/**<*IPAddress*>.
Invokes the policy SelectPolicy if it exists.	The CAR Policy Engine provides an interface to define and configure a policy and to apply the policy to the corresponding access-request packets.
Performs authentication and/or authorization.	Directs the request to the appropriate service, which then performs authentication and/or authorization according to the type specified in **/Radius/Services/**<*Name*>**/**<*Type*>.
Performs session management.	Directs the request to the appropriate Session Manager.
Performs resource management for each Resource Manager in the SessionManager.	Directs the request to the appropriate resource manager listed in **/Radius/SessionManagers/**<*Name*>**/**<*ResourceManagers*>**/**<*Name*>, which then allocates or checks the resource according to the type listed in **/Radius/**<*ResourceManagers*>**/**<*Name*>**/**<*Type*>.
Sends an Access-Accept.	Creates and formats the response, and sends it back to the client (NAS).

Scripting Points

Cisco Access Registrar lets you invoke scripts you can use to affect the Request, Response, or Environment dictionaries.

Client or NAS Scripting Points

Table 2-2 shows the location of the scripting points within the section that determines whether to accept the request from the client or NAS. Note, the scripting points are indicated with the asterisk (*) symbol.

Table 2-2 Client or NAS Scripting Points

Action	Explanation
Receives an Access-Request.	The Cisco Access Registrar RADIUS server receives an Access-Request packet from a NAS.
Determines whether to accept the request.	The client's IP address listed in **/Radius/Clients/**<*Name*>**/IPAddress**.
*Executes the server's incoming script.	A script referred to in **/Radius/IncomingScript**.
*Executes the vendor's incoming script.	The vendor listed in /Radius/Clients/*Name*/Vendor, and is a script referred to in **/Radius/Vendors/**<*Name*>**/IncomingScript**.
*Executes the client's incoming script.	A script referred to in **/Radius/Clients/**<*Name*>**/IncomingScript**.
Determines whether to accept requests from this specific NAS.	
	/Radius/Advanced/RequireNASsBehindProxyBeInClientList set to TRUE.
	The NAS's Identifier listed in **/Radius/Clients/**<*Name*>, or its NAS-IP-Address listed in **/Radius/Clients/**<*Name*>**/IPAddress**.
If the client's IP address listed in /Radius/Clients/<*Name*>**/IPAddress is different:**	
*Executes the vendor's incoming script.	The vendor listed in **/Radius/Clients/***Name*/Vendor, and is a script referred to in **/Radius/Vendors/**<*Name*>**/IncomingScript**.
*Executes the client's incoming script.	The client listed in the previous /Radius/Clients/*Name*, and is a script referred to in /Radius/Clients/*Name*/IncomingScript.

Authentication and/or Authorization Scripting Points

Table 2-3 shows the location of the scripting points within the section that determines whether to perform authentication and/or authorization.

Table 2-3 Authentication and Authorization Scripting Points

Action	Explanation
Determines Service to use for authentication and/or authorization.	The Service name defined in the Environment dictionary variable **Authentication-Service**, and is the same as the Service defined in the Environment dictionary variable **Authorization-Service**.
	The Service name referred to by **/Radius/DefaultAuthenticationService**, and is the same as the Service defined in **/Radius/DefaultAuthorizationService**.

Action	Explanation
Performs authentication and/or authorization.	If the Services are the same, perform authentication and authorization.
	If the Services are different, just perform authentication.
*Executes the Service's incoming script.	A script referred to in **/Radius/Services/**<*Name*>**/IncomingScript**.
Performs authentication and/or authorization.	Based on the Service type defined in **/Radius/Services/**<*Name*>**/**<*Type*>.
*Executes the Service's outgoing script.	A script referred to in **/Radius/Services/**<*Name*>**/OutgoingScript**.
Determines whether to perform authorization.	The Service name defined in **/Radius/DefaultAuthorizationService**, if different than the Authentication Service.
*Executes the Service's incoming script.	A script referred to in **/Radius/Services/**<*Name*>**/IncomingScript**.
Performs authorization.	Checks that the Service type is defined in **/Radius/Services/**<*Name*>**/**<*Type*>.
*Executes the Service's outgoing script.	A script referred to in **/Radius/Services/**<*Name*>**/OutgoingScript**.

Session Management

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

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

Table 2-4 describes how Cisco Access Registrar handles session management.

Table 2-4 Session Management Processing

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

Action	Explanation
Performs resource management.	Directs the request to the appropriate Resource manager listed in **/Radius/SessionManagers/**<*Name*>**/ResourceManagers/**<*Name*>, which then allocates or checks the resource according to the type listed in **/Radius/ResourceManagers/**<*Name*>**/**<*Type*>.
Sends an Access-Accept.	Creates and formats the response, and sends it back to the client (NAS).

Failover by the NAS and Session Management

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

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

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

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

Script Processing Hierarchy

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

Table 2-5, Table 2-6, and Table 2-7 show the overall processing order and flow: (1-6) Incoming Scripts, (7-11) Authentication/Authorization Scripts, and (12-17) Outgoing Scripts.

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

Table 2-5 Cisco Access Registrar Processing Hierarchy for Incoming Scripts

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

Table 2-6 Cisco Access Registrar Processing Hierarchy for Authentication/Authorization Scripts

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

Table 2-7 Cisco Access Registrar Processing Hierarchy for Outgoing Scripts

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

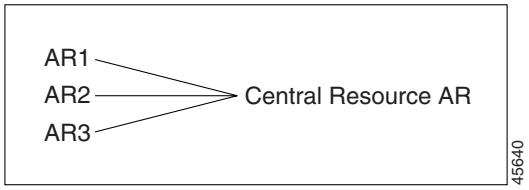
Cross Server Session and Resource Management

Prior to CAR 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 CAR servers. It also meant that, to ensure accurate session tracking, all packets relating to one user session were required to go to the same CAR server.

Overview

Cisco 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 2-1. This enables accurate tracking of User and Group session L across multiple AAA servers, and IP addresses allocated to sessions are managed in one place.

Figure 2-1 Multiple CAR Servers



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

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

When the Accounting-Stop packet arrives at the frontline CAR, it does the regular accounting processing. Then, if the front line CAR server is configured to use Central Resource CAR, a proxy packet will be sent to Central Resource CAR server 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 policy engine to set the Session-Service environment variable.

Configuring a Front Line Cisco 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 policy engine. The value in the Session-Service variable overrides the DefaultSessionService.

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

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 = 1.7R0
    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.