



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 = AuthentictionServiceB
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. Note 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 \${e|realm}{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:

```

${eq|attribute-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.scriptics.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.