
# Configuring Local Authentication and Authorization

Cisco Prime Access Registrar (Prime Access Registrar) allows user information to be stored in its own internal database or external stores such as an LDAP directory or Oracle database. This chapter describes how to configure Prime Access Registrar to perform authentication and authorization using the Prime Access Registrar internal database and how to verify and troubleshoot a local service and userlist configuration.

In RADIUS, an Access Request packet is a request for authentication and authorization (AA). Authentication checks username and password credentials, while authorization typically involves returning the correct information to allow the service a user is authorized to have.
Prime Access Registrar performs AA and returns the appropriate RADIUS attributes in an Access Accept packet.

This chapter contains the following sections:



## Configuring a Local Service and UserList

Prime Access Registrar uses services configured under **/Radius/Services** to process RADIUS requests. To process RADIUS access requests locally, you must configure a service and set its type to **local**. A local service references an Prime Access Registrar userlist.

The following sections show the commands you enter and the expected responses from the Prime Access Registrar server to do the following:

Throughout this chapter, the **aregcmd** commands you enter are shown in **bold** font, and the server responses are shown in `smaller plain font` as shown in the following:

**command you enter**

```
server response
```

# Configuring a Local Service

Prime Access Registrar maintains **Services** under **/Radius**.

To configure a local service:

**Step 1** Use the **add** command at **/Radius/Services** to create a Service.

**cd /Radius/Services**

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

**add SouthBay**

```
Added SouthBay
```

**Step 2** Change directory to the new service and set its type to local.

**cd SouthBay**

```
[ //localhost/Radius/Services/SouthBay ]
```

**set type local**

```
Set Type local
```

**Step 3** Use the **set** command to associate a userlist with the service.

**set userlist SouthUsers**

```
Set UserList SouthUsers
```

# Configuring a Userlist

Prime Access Registrar maintains **UserLists** under **/Radius**.

To configure a userlist:

**Step 1** Use the **add** command at **/Radius/UserLists** to create a userlist.

**cd /Radius/UserLists**

```
[ //localhost/Radius/UserLists ]
```

**add SouthUsers**

```
Added SouthUsers
```

**Step 2** Change directory to the userlist and add users.

**cd SouthUsers**

```
[ //localhost/Radius/UserLists/SouthUsers ]
```

**add user1**

```
Added user1
```

**Step 3** Change directory to each user you add and set the user's password.

**cd user1**

```
[ //localhost/Radius/UserLists/SouthUsers/user1 ]
```

**set Password test**

```
Retype password to confirm:
Set Password <encrypted>
```

# Configuring Cisco Prime Access Registrar to Use the Local Service For AA

To configure Prime Access Registrar to use the local service for authentication and authorization, enter commands to set the DefaultAuthenticationService and DefaultAuthenticationService to the service you created, as shown in the following:

**cd /Radius**

```
[ //localhost/Radius ]
```

**set DefaultAuthenticationService SouthBay**

```
Set DefaultAuthenticationService SouthBay
```

**set DefaultAuthorizationService SouthBay**

```
Set DefaultAuthorizationService SouthBay
```

## Activating the Configuration

To activate the configuration changes you have made, enter the **save** command:

**save**

```
Validating //localhost...

Saving //localhost...
```

After you issue the **save** command, Prime Access Registrar attempts to validate the configuration, checks for all required properties, and ensures there are no logic errors. If the validation is successful, Prime Access Registrar saves the configuration to the MCD database.

# Troubleshooting the Local Service and UserList Configuration

Before you begin troubleshooting, ensure that the current configuration is valid and active. To ensure that any configuration changes you have made are valid and stored in the database, you must issue the **save** command.

**save**

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

To ensure that the current configuration is active, issue the **reload** command.

**reload**

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

## Verifying the Configuration

To verify the configuration changes you have made:

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

**ls /Radius/Services/SouthBay**

```
[ /Radius/Services/SouthBay ]
    Name = SouthBay
        Description =
        Type = local
        IncomingScript~ =
        OutgoingScript~ =
        OutagePolicy~ = RejectAll
        OutageScript~ =
        UserList = SouthUsers
```

**Step 2** Check to see that user **user1** exists under the SouthUsers userlist.

**ls /Radius/UserLists/SouthUsers**

```
[ /Radius/UserLists/SouthUsers ]
    Entries 1 to 1 from 1 total entries
    Current filter: <all>
    Name = SouthUsers
    Description =
    user1/
```

**Step 3** Turn on debugging.

**trace /r 5**

```
Traced "/Radius: Trace level is set to 5"
```

**Step 4** Use **radclient** to send an Access-Request for user **user1**.

**simple user1 test**

---

The debugging output will be sent to the file **name_radius_1_log** in the **/opt/CSCOar/ logs** directory. The following example shows items you should expect in a successful Access-Request.

**Note** Lines of interest are in **bold** font.

```
11/12/2013 18:34:35: P1144: Packet received from 127.0.0.1
11/12/2013 18:34:35: P1144: Trace of Access-Request packet
11/12/2013 18:34:35: P1144:    identifier = 4
11/12/2013 18:34:35: P1144:    length = 62
11/12/2013 18:34:35: P1144:    reqauth = f5:37:f7:04:99:85:c7:63:8f:bc:f4:44:ab:03:4e:1a
11/12/2013 18:34:35: P1144:    User-Name = user1
11/12/2013 18:34:35: P1144:    User-Password = 59:fb:2e:a9:34:de:0e:15:60:8d:4b:64:77:6a:57:d8
11/12/2013 18:34:35: P1144:    NAS-Port = 2
11/12/2013 18:34:35: P1144:    NAS-Identifier = localhost
11/12/2013 18:34:35: P1144: Using Client: localhost (127.0.0.1)
11/12/2013 18:34:35: P1144: Using NAS: localhost (127.0.0.1)
11/12/2013 18:34:35: P1144: Request is directly from a NAS: TRUE
11/12/2013 18:34:35: P1144: Authenticating and Authorizing with Service SouthBay
11/12/2013 18:34:35: P1144: Getting User user1's UserRecord from UserList SouthUsers
11/12/2013 18:34:35: P1144: User user1's password matches
11/12/2013 18:34:35: P1144: No default Remote Session Service defined.
11/12/2013 18:34:35: P1144: Trace of Access-Accept packet
11/12/2013 18:34:35: P1144:    identifier = 4
11/12/2013 18:34:35: P1144:    length = 20
```

```
11/12/2013 18:34:35: P1144:    reqauth = 36:88:34:0c:cc:ea:9e:d8:6d:f5:14:f7:ab:26:e7:f6
11/12/2013 18:34:35: P1144: Sending response to 127.0.0.1
11/12/2013 18:34:35: Log: Request from localhost (127.0.0.1): User user1 accepted
```

The following example shows a trace for an unsuccessful Access-Request due to an invalid password.

**Note** Lines of interest are in **bold** font.

```
11/12/2013 19:05:13: P1527: Packet received from 127.0.0.1
11/12/2013 19:05:13: P1527: Trace of Access-Request packet
11/12/2013 19:05:13: P1527:11/12/2013 19:05:13: P1527:11/12/2013 19:05:13: P1527:
11/12/2013 19:05:13: P1527:11/12/2013 19:05:13: P1527:
11/12/2013 19:05:13: P1527:11/12/2013 19:05:13: P1527:11/12/2013 19:05:13: P1527: Using Client: localhost
(127.0.0.1)
11/12/2013 19:05:13: P1527: Using NAS: localhost (127.0.0.1)
11/12/2013 19:05:13: P1527: Request is directly from a NAS: TRUE
11/12/2013 19:05:13: P1527: Authenticating and Authorizing with Service SouthBay
11/12/2013 19:05:13: P1527: Getting User user1's UserRecord from UserList SouthUsers
11/12/2013 19:05:13: P1527: User user1's password does not match
11/12/2013 19:05:13: P1527: Rejecting request
11/12/2013 19:05:13: P1527: Rejecting request
11/12/2013 19:05:13: P1527: Trace of Access-Reject packet
11/12/2013 19:05:13: P1527:11/12/2013 19:05:13: P1527:11/12/2013 19:05:13: P1527:
11/12/2013 19:05:13: P1527:11/12/2013 19:05:13: P1527: Sending response to 127.0.0.1
11/12/2013 19:05:13: Log: Request from localhost (127.0.0.1): User user1 rejected (UserPasswordInvalid)
```

If a user's password is invalid, reset the password to ensure it was entered correctly. Also check that the shared secret being used by the RADIUS client and the Prime Access Registrar server match.

# Configuring Return Attributes and Check-Items

Prime Access Registrar supports RADIUS check item attributes at the user and group levels. You can configure Prime Access Registrar to check for attributes that must be present or attributes that must not be present in the Access-Request packet for successful authentication. For a complete list of attributes supported in Prime Access Registrar, see Cisco Prime Access Registrar 8.0 Reference Guide.

When using check item attributes, Prime Access Registrar rejects Access-Requests if either of the following conditions exist:

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

This section contains the following topics:

## Configuring Per User Return Attributes

User return attributes are attributes that are specific for a given user each time they log in. To configure a user's return attributes, change directory to the user's Attributes subdirectory and configure the desired attributes.

**cd /Radius/UserLists/SouthUsers/User1/Attributes**

```
[ //localhost/Radius/UserLists/SouthUsers/user1/Attributes ]
```

**set Session-Timeout 60**

```
Set Session-Timeout  60
```

**set Callback-Number 5551124**

```
Set Callback-Number  5551124
```

## Configuring Per User Check-Items

Check Items are a way to check that certain attribute/values exist in a user‘s access-request. If the attribute/values are not present in the access-request, the Prime Access Registrar server rejects the access-request.

To check that an access-request for `user1` has the Calling-Station-Id attribute set to 5555678, enter the following:

**cd /Radius/UserLists/SouthUsers/User1/CheckItems**

```
[ //localhost/Radius/UserLists/SouthUsers/user1/CheckItems ]
```

**set Calling-Station-Id 5555678**

```
Set Calling-Station-Id  5555678
```

Be sure to **save** your configuration to preserve your changes.

## Verifying the Per User Return Attributes and Check-Items Configuration

A successful request will produce a trace similar to the following:

```
11/12/2013 14:08:07: P1539: Packet received from 127.0.0.1
11/12/2013 14:08:07: P1539: Trace of Access-Request packet
11/12/2013 14:08:07: P1539:    identifier = 1
11/12/2013 14:08:07: P1539:    length = 71
11/12/2013 14:08:07: P1539:    reqauth = d6:86:c5:1e:0e:a0:20:4f:9a:1a:2c:35:27:16:12:36
11/12/2013 14:08:07: P1539:    User-Name = user1
11/12/2013 14:08:07: P1539:    User-Password = 99:dc:4a:22:ef:f6:8b:90:a2:3a:50:f0:a6:03:6e:b3
11/12/2013 14:08:07: P1539:    NAS-Port = 1
11/12/2013 14:08:07: P1539:    Calling-Station-Id = 5555678
11/12/2013 14:08:07: P1539:    NAS-Identifier = localhost
11/12/2013 14:08:07: P1539: Using Client: localhost (127.0.0.1)
```

```
11/12/2013 14:08:07: P1539: Using NAS: localhost (127.0.0.1)
11/12/2013 14:08:07: P1539: Request is directly from a NAS: TRUE
11/12/2013 14:08:07: P1539: Authenticating and Authorizing with Service SouthBay
11/12/2013 14:08:07: P1539: Getting User user1's UserRecord from UserList SouthUsers
11/12/2013 14:08:07: P1539: User user1's password matches
11/12/2013 14:08:07: P1539: Processing User user1's check items
11/12/2013 14:08:07: P1539: Merging User user1's Attributes into response Dictionary
11/12/2013 14:08:07: P1539: Merging attributes into the Response Dictionary:
11/12/2013 14:08:07: P1539:   Adding attribute Callback-Number, value = 5551124
11/12/2013 14:08:07: P1539:   Adding attribute Session-Timeout, value = 60
11/12/2013 14:08:07: P1539: No default Remote Session Service defined.
11/12/2013 14:08:07: P1539: Trace of Access-Accept packet
11/12/2013 14:08:07: P1539:    identifier = 1
11/12/2013 14:08:07: P1539:    length = 35
11/12/2013 14:08:07: P1539:    reqauth = cc:2d:51:71:b5:49:0e:e6:f1:eb:1c:61:51:7a:f1:cb
11/12/2013 14:08:07: P1539:    Callback-Number = 5551124
11/12/2013 14:08:07: P1539:    Session-Timeout = 60
11/12/2013 14:08:07: P1539: Sending response to 127.0.0.1
11/12/2013 14:08:07: Log: Request from localhost (127.0.0.1): User user1 accepted
```

# Configuring Profiles to Group Attributes

You can use thePrime Access Registrar profile object to group attributes. For example, you might want to group attributes for all PPP users. All PPP users could then be assigned the profile and the attributes contained in the profile would be returned in their access-accepts.

To configure profiles to group attributes:

**Step 1** Change directory to **/Radius/Profiles** and add a profile.

**cd /Radius/Profiles**

```
[ //localhost/Radius/Profiles ]
```

**add PPP-Profile**

```
Added PPP-Profile
```

**Step 2** Change directory to the new profile, then change directory to the profile's Attributes subdirectory.

**cd PPP-Profile**

```
[ //localhost/Radius/Profiles/PPP-Profile ]
```

**cd Attributes**

```
[ //localhost/Radius/Profiles/PPP-Profile/Attributes ]
```

**Step 3** Configure the desired attributes for the profile.

**set Service-Type Framed**

```
Set Service-Type  Framed
```

**set Framed-Protocol PPP**

```
Set Framed-Protocol  PPP
```

**Note** When you need to set an attribute to a value that includes a space, you must double-quote the value, as in the following: *set Framed-Route "192.168.1.0/24  192.168.1.1"*

**Step 4** Assign the profile to a user by setting the user's BaseProfile attribute to the desired profile.

**cd /Radius/UserLists/SouthUsers/User1**

```
[ //localhost/Radius/UserLists/SouthUsers/user1 ]
```

**set BaseProfile PPP-Profile**

```
Set BaseProfile PPP-Profile
```

## Configuring Return Attributes and Check-Items Using UserGroup

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

To configure return attributes and check-items using usergroup:

**Step 1** Change directory to **/Radius/UserGroups** and add a UserGroup.

**cd /Radius/UserGroups**

```
[ //localhost/Radius/UserGroups ]
```

**add PPP-Group**

```
Added PPP-Group
```

**Step 2** Change directory to the new UserGroup and add Return Attributes.

**cd PPP-Group**

```
[ //localhost/Radius/UserGroups/PPP-Group ]
```

**cd Attributes**

```
[ //localhost/Radius/UserGroups/PPP-Group/Attributes ]
```

**set Service-Type Outbound**

```
Set Service-Type Outbound
```

**Step 3** Change directory to the UserGroups' Check-Items subdirectory and add CheckItems.

**cd ../CheckItems/**

```
[ //localhost/Radius/UserGroups/PPP-Group/CheckItems ]
```

**set Service-Type Framed**

```
Set Service-Type  Framed
```

**Step 4** Assign the UserGroup to a User.

**cd /Radius/UserLists/SouthUsers/User2**

```
[ //localhost/Radius/UserLists/SouthUsers/user2 ]
```

**set Group PPP-Group**

```
Set Group PPP-Group
```

# Return Attribute Precedence

Because there are multiple ways of returning attributes, you might at some time have an attribute clash. In case of an attribute clash, the attribute precedence is as follows (from highest to lowest):

1. User attribute
2. User profile
3. UserGroup attribute
4. UserGroup profile

# aregcmd Command Performance

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

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

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

```
[ //localhost/Radius/Services/GroupService ]
    Name = GroupService
```

```
Description =
Type = group
IncomingScript~ =
OutgoingScript~ =
ResultRule = OR
GroupServices/
1. UserService1
2. UserService2
3. UserService3
```

# UserDefined1 Property

The UserDefined1 property of a user object is a free text field. You can use the UserDefined1 property to store additional user information much like the Description property, but its most powerful use is to pass information to an extension point script. The value set in the UserDefined1 property is automatically set to the environment variable of the same name during authentication. Any extension point script that subsequently runs has access the value in that property.

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

# Access-Request Logging

By default, Prime Access Registrar logs all dropped and rejected requests in the name_radius_1_log file. The following are examples of log entries for dropped or rejected requests.

```
11/12/2013 17:38:11 name/radius/1 Warning Protocol 0 Request from localhost (127.0.0.1):
User user1 rejected (UserPasswordInvalid)
```

```
11/12/2013 18:05:12 name/radius/1 Warning Protocol 0 Packet from 128.107.132.106: that
address is not in the Clients list <unknown user>
```

To log all accepted requests as well, set the LogServerActivity advanced property to TRUE:

**set /Radius/Advanced/LogServerActivity TRUE**

**Set /Radius/Advanced/LogServerActivity TRUE**

**save**

```
Validating //localhost...

Saving //localhost...
```

### reload

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


Access-Accept packets are now logged as well:
11/12/2013 18:22:32 name/radius/1 Activity Protocol 0 Request from localhost (127.0.0.1):
User user2 accepted
```