

Wireless Support

Revised: March 20, 2009, OL-17222-03

This chapter provides information about using Cisco Access Registrar (CAR) for wireless support. The following topics are included in this chapter:

- [“Mobile Node-Home Agent Shared Key” section on page 18-1](#)
- [“3GPP2 Home Agent Support” section on page 18-2](#)
- [“Session Correlation Based on User-Defined Attributes” section on page 18-5](#)
- [“Managing Multiple Accounting Start/Stop Messages” section on page 18-5](#)
- [“NULL Password Support” section on page 18-6](#)
- [“New 3GPP2 VSAs in the Cisco Access Registrar Dictionary” section on page 18-5](#)

Mobile Node-Home Agent Shared Key

In a mobile wireless environment, a Home Agent (HA) can request a Mobile Node-Home Agent (MN-HA) shared key from the home CAR RADIUS server during a mobile IP registration request (RRQ) from a Packet Data Serving Node (PDSN). CAR supports distribution of the shared key in this environment. CAR encrypts the shared key using MD5 encryption before sending the key back to the HA in an Access-Accept packet.

When an HA receives an RRQ from a PDSN, the HA authenticates the RRQ using a MN-HA shared key. If the HA does not have the MN-HA shared key, it retrieves the MN-HA shared key from the CAR server by sending an Access-Request packet containing the 3GPP2 VSA CDMA-MN-HA-SPI (SPI attribute). CAR then sends the CDMA-MN-HA-Shared-Key corresponding to the user if the user has been successfully authenticated.

Use Case Example

When HA receives an RRQ from a PDSN, it authenticates the RRQ by using a MN-HA shared key. If the HA does not have the MN-HA shared key, it retrieves the MN-HA shared key from the CAR server by sending an Access-Request packet containing the 3GPP2 vendor-specific attribute (VSA) CDMA-MN-HA-SPI, the Security Parameter Index (SPI attribute).

The CAR server then sends the CDMA-MN-HA-Shared-Key corresponding to the user if the user has successfully authenticated subject to the following rules:

1. If there is an incoming SPI and no configured SPI, the CAR server authenticates the user as usual and does not include a configured shared-key (if there is one) in the reply.
2. If the incoming SPI does not match the configured SPI, the CAR server authenticates the user as usual, but does not include the configured shared-key (if there is one) in the reply.
3. If the incoming SPI matches the configured SPI, but there is no shared-key configured, the CAR server proceeds with normal authentication. Since there is no shared-key, it will not be included in the reply.
4. If the incoming SPI matches the configured SPI and a configured shared-key exists, the CAR server proceeds to encrypt the MCD5 shared-key and include it in the Access-Accept.

The key to including the shared key in an Access-Accept is in matching the values of the SPI attribute.

Configuring User Attributes

To configure a user with the CDMA-MN-HA-SPI VSA to request a MN-HA shared key, complete the following steps:

-
- Step 1** Log in to the CAR server and launch **aregcmd**.
Log in as a user with administrative rights such as user **admin**.
- Step 2** Change directory to the attribute directory of the user.
cd /Radius/UserLists/Default/bob/Attributes
- Step 3** Set the CDMA-MN-HA-SPI VSA to the appropriate shared-key value.
set CDMA-MN-HA-SPI 1234
`set CDMA-MN-HA-SPI 1234`
- Step 4** Set the CDMA-MN-HA-SPI VSA to the appropriate shared-key value.
set CDMA-MN-HA-Shared-Key secret123
`set CDMA-MN-HA-Shared-Key secret123`
- Step 5** Validate and save your changes.
validate
save

3GPP2 Home Agent Support

The CAR server supports 3GPP2 home agents. This support enables mobile IP clients that authenticate through a CAR 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

CAR 1.7 and above 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 CAR 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 might select the 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.

New 3GPP2 VSAs in the Cisco Access Registrar Dictionary

CAR 4.2 supports 3GPP2 vendor-specific attributes (VSAs) in the vendor-specific dictionary in `/Radius/Advanced/Attribute Dictionary`.



Note

There is no planned support for the Accounting-Container (3GPP2/6) attribute because it has different syntax than other vendor-specific attributes (VSAs) and requires special processing.

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 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 introduced 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 CAR 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 which allows the decision to be made based on run-time conditions, such as attributes in the access-request or policies.