Using Extension Points

This chapter describes how to use Cisco Prime Access Registrar (Prime Access Registrar) scripting to customize your RADIUS server. At specific points during a Prime Access Registrar Request-Response packet flow, service providers can initiate scripts using REX (C/C++), Java, TCL, or internal script interfaces, to customize or modify the packet flow. When the script finishes, the packet flow continues with the next step.You can write scripts to affect the way Prime Access Registrar handles and responds to requests and to change the behavior of Prime Access Registrar after a script is run.

All scripts reference the three dictionaries listed below, which are data structures that contain key/value pairs.

- **Request dictionary**—contains all of the attributes from the access-request or other incoming packets, such as the username, password, and service hints.
- **Response dictionary**—contains all of the attributes in the access-accept or other response packets. As these are the attributes the server sends back to the NAS, you can use this dictionary to add or remove attributes.
- **Environment dictionary**—contains well-known keys whose values enable scripts to communicate with Prime Access Registrar or to communicate with other scripts.

This chapter contains the following sections:

Client Scripting

Cisco is not liable for scripts developed by clients.

Though Prime Access Registrar allows external code (Tcl/C/C++/Java) to be used by means of a script, custom service, policy engine, and so forth, while processing request, response, or while working with the environment dictionaries, it shall not be responsible for the scripts used and will not be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of the script.

Determining the Goal of the Script

The goal of the script and its scripting point are tied together. For example, when you want to create a script that performs some special processing of a username before it is processed by the Prime Access Registrar server, you would reference this script as an *incoming* script.

When on the other hand, you would like to affect the response, such as setting a specific timeout when there is not one, you would reference the script as an *outgoing* script.

In order to be able to create a script, you need to understand the way Prime Access Registrar processes client requests. Prime Access Registrar processes requests and responses in a hierarchical fashion; incoming requests are processed from the most general to the most specific levels, whereas, outgoing responses are processed from the most specific to the most general levels. Extension points are available at each level.

An incoming script can be referenced at each of the following extension points:

- RADIUS server
- Vendor (of the immediate client)
- Client (individual NAS)
- NAS-Vendor-Behind-the-Proxy
- Client-Behind-the-Proxy
- Remote Server (of type RADIUS)
- Service

An authentication or authorization script can be referenced at each of the following extension points:

- Group Authentication
- User Authentication
- Group Authorization
- User Authorization

The outgoing script can be referenced at each of the following extension points:

- Service
- Client-Behind-the-Proxy
- NAS-Vendor-Behind-the-Proxy
- Client (individual NAS)
- NAS Vendor
- RADIUS server

Writing the Script

You can write scripts in Tcl, REX, Java, or as shared libraries using C or C++. In this section, the scripts are shown in Tcl.

Writing the Script

To write a script:

Step 1 Create the Tcl source file using an editor.

Step 2 Give it a name.

Step 3 Define one or more procedures, using the following syntax:

```
proc name {request response environment}
{Body}
```

Step 4 Create the body of the script.

Step 5 Save the file and copy it to the **/opt/CSCOar/scripts/radius/tcl** directory, or to the location you chose when you installed Prime Access Registrar.

Note You can also use the Prime Access Registrar GUI or CLI to write internal scripts, by which you can add, modify, or delete attribute in the request, response, and environment dictionaries for RADIUS, Diameter, and TACACS+. For more information, see Internal Scripts, page 7-14.

Choosing the Type of Script

When you create a script, you can use any one or all of the three dictionaries: Request, Response, or Environment. Here is what each dictionary does it for you,

- When you use the Request dictionary, you can modify the contents of a NAS request. Scripts that use the Request dictionary are usually employed as incoming scripts.
- When you use the Response dictionary, you can modify what the server sends back to the NAS. These scripts are consequently employed as outgoing scripts.
- When you use the Environment dictionary, you can do the following:
  - Affect the behavior of the server after the script is run. For example, you can use the Environment dictionary to decide which of the multiple services to use for authorization, authentication, and accounting.
  - Communicate among scripts, as the scripts all share these three dictionaries. For example, when a script changes a value in the Environment dictionary, the updated value can be used in a subsequent script.

The following sections show scripts examples of all the three dictionaries:

- Request Dictionary Script
- Response Dictionary Script
- Environment Dictionary Script

Request Dictionary Script

The Request Dictionary script is referenced from the server's IncomingScript scripting point. It checks to see whether the request contains a **NAS-Identifier** or a **NAS-IP-Address**. When it does not, it sets the **NAS-IP-Address** from the request's source IP address.

```
proc MapSourceIPAddress {request response environment}
{
    if { ! ( [ $request containsKey NAS-Identifier ] ||
```

```
        [ $request containsKey NAS-IP-Address ] ) } {
        $request put NAS-IP-Address [ $environment get Source-IP-Address ]
    }
```

Tcl scripts interpret **$request** arguments as active commands that can interpret strings from the Request dictionary, which contains keys and values.

The **containsKey** method has the syntax: <*$dict*> **containsKey** <*attribute*>. In this example, <*$dict*> refers to the Request dictionary and the attributes **NAS-identifier** and **NAS-IP-Address**. The **containsKey** method returns **1** when the dictionary contains the attribute, and **0** when it does not. Using the **containsKey** method prevents you from overwriting an existing value.

The **put** method has the syntax: <*$dict*> **put** <*attribute value*>[<*index*>]. In this example, <*$request*> refers to the Request dictionary and the attribute is **NAS-IP-Address**. The **put** method sets the NAS's IP address attribute.

The **get** method has the syntax: <*$dict*> get <*attribute*>. In this example, <*$dict*> refers to the Environment dictionary and <*attribute*> is the **Source-IP-Address**. The **get** method returns the value of the attribute from the environment dictionary.

For a list of the methods you can use with scripts, see
Cisco Prime Access Registrar 9.0 Reference Guide. They include **get**, **put**, and others.

Response Dictionary Script

This script is referenced from either the user record for users whose sessions are always PPP, or from the script, **AuthorizeService**, which checks the request to determine which service is desired. The script merges the Profile named **default-PPP-users** into the Response dictionary.

```
proc AuthorizePPP {request response environment}
{
    $response addProfile default-PPP-users
```

The **addProfile** method has the syntax: <*$dict*> **addProfile** <*profile*>[<*mode*>]. In this example, <*$dict*> refers to the Response dictionary and the profile is **default-PPP-users**. The script copies all of the attributes of the Profile <*profile*> into the dictionary.

Environment Dictionary Script

This script is referenced from the NAS Incoming Script scripting point. It looks for a realm name on the username attribute to determine which AAA services should be used for the request. When it finds `@radius`, it selects a set of AAA services that will proxy the request to a remote RADIUS server. When it finds `@tacacs`, it selects the Authentication Service that will proxy the request to a TACACS server for authentication. For all of the remaining usernames, it uses the default Service (as specified in the configuration by the administrator).

Note the function, **regsub**, is a Tcl function.

```
proc ParseProxyHints {request response environment} {
    set userName [ $request get User-Name ]
    if { [ regsub "@radius" $userName "" newUserName ] } {
    $request put User-Name $newUserName
    $radius put Authentication-Service "radius-proxy"
    $radius put Authorization-Service "radius-proxy"
    $radius put Accounting-Service "radius-proxy"
    } else {
    if { [ regsub "@tacacs" $userName "" newUserName ] } {
    $request put User-Name
    $radius put Authentication-Service "tacacs-client"
```

Adding the Script Definition

After you have written the script, you must add the script definition to the Prime Access Registrar's script **Configuration** directory so it can be referenced. Adding the script definition involves:

- Specifying the script definition; it must include the following:
  - **Name**—used in other places in the configuration to refer to the script. It must be unique among all other scripts.
  - **Language**—can be either Tcl or REX (shared libraries)
  - **Filename**—the name you used when you created the file
  - **EntryPoint**—the function name.

  The **Name** and the **EntryPoint** can be the same name, however they do not have to be.
- Choosing a scripting point; nine exist for incoming and outgoing scripts. These include:
  - the server
  - the vendor of the immediate client
  - the immediate client
  - the vendor of the specific NAS
  - the specific NAS
  - the service (rex or tcl)
  - the group (only AA scripts
  - the user record (only AA scripts)
  - remote server (only type RADIUS)

The rule of thumb to use in determining where to add the script is the most general scripts should be on the outermost points, whereas the most specific scripts should be on the innermost points.

Note The client and the NAS are the same entity, unless the immediate client is acting as a proxy for the actual NAS.

This section contains the following topics:

Adding the Example Script Definition

In the server configuration a **Scripts** directory exists. You must add the script you created to this directory. To add the **ParseProxyHints** script to the Prime Access Registrar server, enter the following command and supply the following information:

Name=**ParseProxyHints**
Description=**ParseProxyHints**
Language=**tcl**

Filename=**ParseProxyHints**
Entrypoint=**ParseProxyHints**

aregcmd add /Radius/Scripts/ParseProxyHints ParseProxyHints tcl ParseProxyHints ParseProxyHints

Choosing the Scripting Point

As the example script, **ParseProxyHints**, applies to a specific NAS (NAS1), the entry point should be that NAS. To specify the script at this scripting point, enter:

aregcmd set /Radius/Clients/NAS1/IncomingScript ParseProxyHints

Testing the Script

To test the script, you can use the **radclient** command, which lets you create and send packets. For more information about the **radclient** command, see Chapter 4, "Setting the CPAR Configurable Option."

About the Tcl/Tk 8.3 Engine

Prime Access Registrar uses Tcl engine is version Tcl/Tk8.3. Since the Tcl/Tk8.3 engine supports a multi-threading application environment, it can achieve much better performance than Tcl/Tk7.6.

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

Cisco Prime Access Registrar Scripts

The Prime Access Registrar scripts are stored in **/localhost/Radius/Scripts**. Most of the scripts are written in the RADIUS Extension language (REX). Some scripts are provided in both REX and Tcl. The scripts written in Tcl all begin with the letter **t** followed by their functional name. The Tcl scripts are listed below:

tACME
tAuthorizePPP
tAuthorizeService
tAuthorizeTelnet
tMapSourceIPAddress
tParseAARealm
tParseAASRealm
tParseProxyHints
tParseServiceAndAAARealmHints
tParseServiceAndAAASRealmHints
tParseServiceAndAARealmHints
tParseServiceAndAASRealmHints
tParstServiceAndProxyHints
tParseServiceHints

You can use the Prime Access Registrar GUI to write internal scripts, by which you can add, modify, or delete attribute in the request, response, and environment dictionaries for RADIUS, Diameter, and TACACS+. For more information about configuring internal scripts by using the GUI, see Cisco Prime Access Registrar 9.0 User Guide.

This section contains the following topics:

- ParseServiceHints
- ParseTranslationGroupsByCLID
- ParseTranslationGroupsByDNIS
- ParseTranslationGroupsByRealm
- UseCLIDAsSessionKey
- USRIncomingScript
- USRIncomingScript-IgnoreAccountingSignature
- USROutgoingScript
- Internal Scripts, page 7-14
- Blacklisting Script, page 7-16

ACME

ACME is referenced from Vendor ACME for the outgoing script. If the Prime Access Registrar server accepts this Access-Request and the response does not yet contain a Session-Timeout, set it to 3600 seconds.

AltigaIncomingScript

AltigaIncomingScript maps Altiga-proprietary attributes to Prime Access Registrar's global attribute space.

AltigaOutgoingScript

AltigaOutgoingScript maps Altiga attributes from Prime Access Registrar's global attribute space to the appropriate Altiga-proprietary attributes.

ANAAAOutgoing

ANAAAOutgoing can be referenced from either the client or vendor outgoing scripting point to be used in HRPD/EV-DO networks where Prime Access Registrar is the Access Network (AN) AAA server. ANAAAOutgoing checks to see if the response contains the Callback-Id attribute. If the response contains the Callback-Id attribute and the value is less than 253 characters, ANAAAOutgoing prefixes a zero (0) to the value. For example, it changes "112" into "0112." The ANAAAOutgoing script always returns REX_OK.

AscendIncomingScript

AscendIncomingScript maps Ascend-proprietary attributes to Prime Access Registrar's global attribute space.

AscendOutgoingScript

AscendOutgoingScript maps Ascend attributes from Prime Access Registrar's global attribute space to the appropriate Ascend-proprietary attributes.

AuthorizePPP

AuthorizePPP is referenced from either the use record for users who's sessions are always PPP or from the from the script AuthorizeService, which checks the request to determine which service is desired. This script merges in the Profile named "default-PPP-users" into the response dictionary.

AuthorizeService

AuthorizeService is referenced from user record for users who's sessions might be PPP, SLIP or Telnet depending on how they are connecting to the NAS. This script checks the request to determine which service is desired. If it is telnet, it calls the script AuthorizeTelnet. If it is PPP, it calls the script AuthorizePPP. If it is SLIP, it calls the script AuthorizeSLIP. If it is none of these, it rejects the request.

AuthorizeSLIP

AuthorizeSLIP is referenced from either the user record for users who's sessions are always SLIP or from the from the script AuthorizeService, which checks the request to determine which service is desired. This script merges in the Profile named "default-SLIP-users" into the response dictionary.

AuthorizeTelnet

AuthorizeTelnet is referenced from either the user record for users who's sessions are always telnet or from the from the script AuthorizeService, which checks the request to determine which service is desired. This script merges in the Profile named "default-Telnet-users" into the response dictionary.

CabletronIncoming

CabletronIncoming maps Cabletron-proprietary attributes to Prime Access Registrar's global attribute space.

CabletronOutgoing

Use CabletronOutgoing to map Cisco-proprietary attributes from Prime Access Registrar's global attribute space to the appropriate Cabletron-proprietary attributes.

CiscoIncoming

Use CiscoIncoming to map Cisco-proprietary attributes to Prime Access Registrar's global attribute space.

CiscoOutgoing

Use CiscoOutgoing to map Cisco-proprietary attributes from Prime Access Registrar's global attribute space to the appropriate Cabletron-proprietary attributes.

CiscoWithODAPIncomingScript

Use CiscoWithODAPIncomingScript to map Cisco-proprietary attributes to Prime Access Registrar's global attribute space and to map ODAP requests to the appropriate services and session managers.

CiscoWithODAPIncomingScript checks the incoming NAS-Identifier sent by the client. If the NAS-Identifier does not equal odap-dhcp, the request is not an ODAP request. If the request is not an ODAP request, the script does no more ODAP-specific processing, and calls CiscoIncomingScript to allow it to process the request.

If the request is an ODAP request, CiscoWithODAPIncomingScript removes the NAS-Identifier attribute because it is no longer required. The script then sets the Authentication-Service and the Authorization-Service to odap-users and sets the Accounting-Service to odap-accounting.

ExecCLIDRule

ExecCLIDRule is referenced from the policy engine to determine the authentication and authorization service and policy based on the CLID set in the policy engine.

ExecDNISRule

ExecDNISRule is referenced from the policy engine to determine the authentication and authorization service and policy based on the DNIS set in the policy engine.

ExecFilterRule

ExecFilterRule is referenced from the policy engine to determine whether a user packet should be rejected or not based on whether a special character like "*", "/", "\" or "?" shows up in the packet.

ExecNASIPRule

ExecNASIPRule is referenced from the policy engine to enable configuration of policies based on the incoming NAS-IP-Address. You can configure two attributes, *client-ip-address* and *subnetmask*, to match the incoming NAS-IP-Address and its subnet mask. If the attributes match, ExecNASIPrule sets the environment variables (if they are configured in that rule).

ExecRealmRule

ExecRealmRule is referenced from the policy engine to determine the authentication and authorization service and policy based on the realm set in the policy engine.

ExecTimeRule

ExecTimeRule either rejects or accepts Access Request packets based on the time range specified in a user's login profile. You can configure the TimeRange and AcceptedProfile attributes.

The format for the TimeRange is to set the allowable days followed by the allowable times, as in:

TimeRange = dateRange, timeRange

The dateRange can be in the form of a date, a range of allowable dates, a day, or a range of allowable days. The timeRange should be in the form of hh:mm-hh:mm.

Here are a few examples:

mon-fri,09:00-17:00

Allows access Monday through Friday from 9 AM until 5 PM.

mon,09:00-17:00;tue-sat,12:00-13:00

Allows access on Monday from 9 AM until 5 PM and from 12 noon until 1 PM on Tuesday through Saturday

mon,09:00-24:00;tue,00:00-06:00

Allows access on Monday from 9 AM until Tuesday at 6 AM

1-13,10-17:00; 15,00:00-24:00

Allows access from the first of the month until the thirteenth of the month from 10 AM until 5 PM and all day on the fifteenth of the month

LDAPOutage

LDAPOutage is referenced from LDAP Services as OutageScript. LDAPOutage logs when the LDAP binding is lost.

MapSourceIPAddress

MapSourceIPAddress is referenced from the Prime Access Registrar server's IncomingScript scripting point. MapSourceIPAddress checks to see if the request contains either a NAS-Identifier or a NAS-IP-Address. If not, this script sets the NAS-IP-Address from the request's source IP address.

The Tcl version of this script is tMapSourceIPAddress.

ParseAAARealm

ParseAAARealm is referenced from the NAS IncomingScript scripting point. It looks for a realm name on the username attribute as a hint of which AAA service should be used for this request. If @<realm> is found, the AAA service is selected which has the same name as the realm.

ParseAAASRealm

ParseAAASRealm is referenced from the NAS incoming script extension point. ParseAAASRealm looks for a realm name on the username attribute as a hint of which AAA service and which SessionManager should be used for this request. If @<realm> is found, the AAA service and SessionManager which have the same name as the realm are selected.

ParseAARealm

ParseAARealm is referenced from the NAS IncomingScript scripting point. It looks for a realm name on the username attribute as a hint of which authentication and authorization service should be used for this request. If @<realm> is found, it selects the AA service that has the same name as the realm and the DefaultAccountingService (as specified in the configuration by the administrator).

The Tcl version of this script is named tParseAARealm.

ParseAASRealm

ParseAASRealm is referenced from the NAS IncomingScript scripting point. It looks for a realm name on the username attribute as a hint of which AA service and which SessionManager should be used for this request. If @<realm> is found, the AA service and the SessionManager which have the same name as the realm are selected. The Accounting service will be the DefaultAccountingService (as specified in the configuration by the administrator).

The Tcl version of this script is named tParseAASRealm.

ParseProxyHints

ParseProxyHints is referenced from the NAS IncomingScript scripting point. It looks for a realm name on the username attribute as a hint of which AAA services should be used for this request. If @radius is found, a set of AAA services is selected which will proxy the request to a remote radius server. If @tacacs is found, the AuthenticationService is selected that will proxy the request to a tacacs server for authentication. For any services not selected, the default service (as specified in the configuration by the administrator) will be used.

The Tcl version of this script is named tParseProxyHints.

ParseServiceAndAAARealmHints

ParseServiceAndAAARealmHints is referenced from the NAS IncomingScript scripting point. It calls both ParseServiceHints and ParseAAARealm.

The Tcl version of this script is named tParseServiceAndAAARealmHints.

ParseServiceAndAAASRealmHints

ParseServiceAndAAASRealmHints is referenced from the NAS IncomingScript scripting point. It calls both ParseServiceHints and ParseAAASRealm.

The Tcl version of this script is named tParseServiceAndAAASRealmHints.

ParseServiceAndAARealmHints

ParseServiceAndAARealmHints is referenced from the NAS IncomingScript scripting point. It calls both ParseServiceHints and ParseAARealm.

The Tcl version of this script is named tParseServiceAndAARealmHints.

ParseServiceAndAASRealmHints

ParseServiceAndAASRealmHints is referenced from the NAS IncomingScript scripting point. It calls both ParseServiceHints and ParseAASRealm.

The Tcl version of this script is named tParseServiceAndAASRealmHints.

ParseServiceAndProxyHints

ParseServiceAndProxyHints is referenced from the NAS IncomingScript scripting point. It calls both ParseServiceHints and ParseProxyHints.

The Tcl version of this script is named tParseServiceAndProxyHints.

ParseServiceHints

ParseServiceHints is referenced from the NAS IncomingScript scripting point. Check to see if we are given a hint of the service type or the realm. If so, set the appropriate attributes in the request or radius dictionary to record the hint and rewrite the username to remove the hint.

The Tcl version of this script is named tParseServiceHints.

ParseTranslationGroupsByCLID

ParseTranslationGroupsByCLID is referenced from the policy engine to determine the incoming and outgoing translation groups based on CLID set in the policy engine so that the attributes can be added and/or filtered out by the configuration data set in MCD.

ParseTranslationGroupsByDNIS

ParseTranslationGroupsByDNIS is referenced from the policy engine to determine the incoming and outgoing translation groups based on realm set in the policy engine so that the attributes can be added/filtered out by the configuration data set in MCD.

ParseTranslationGroupsByRealm

ParseTranslationGroupsByRealm is referenced from the policy engine to determine the incoming and outgoing translation groups based on the realm set in the policy engine. ParseTranslationGroupsByRealm allows the attributes to be added or filtered out by the configuration data set in MCD.

UseCLIDAsSessionKey

UseCLIDAsSessionKey is used to specify that the Calling-Station-Id attribute 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.

USRIncomingScript

USRIncomingScript maps USR-proprietary attributes to Prime Access Registrar's global attribute space.

USRIncomingScript-IgnoreAccountingSignature

USRIncomingScript-IgnoreAccountingSignature maps USR-proprietary attributes to Prime Access Registrar's global attribute space and sets a flag to ignore the signature on Accounting-Request packets. Earlier versions of the USR RADIUS client did not correctly sign Accounting-Request packets.

USROutgoingScript

USROutgoingScript maps USR attributes from Prime Access Registrar's global attribute space to the appropriate USR-proprietary attributes.

Internal Scripts

Prime Access Registrar allows you to write internal scripts, by which you can add, modify, or delete attributes in the request, response, and environment dictionaries for RADIUS, Diameter, and TACACS+. You can use the Prime Access Registrar GUI or CLI to configure the internal scripts.

Prime Access Registrar allows you to create script statements for the following operations:

- Simple Attribute Operation—allows you to add, modify, or delete an attribute value to the request, response, or environment dictionary
- Copy an Attribute—allows you to copy an attribute value from one dictionary to another
- Concatenate Operation—allows you to concatenate an attribute value from one dictionary to another
- Replace Operation—allows you to replace an attribute value from one dictionary to another
- Value Based Manipulations—allows you to manipulate attribute values in a dictionary based on a given text

- Log or Trace Messages—allows you to create different levels of log or trace messages
- I can do it myself—allows you to create your own script for the selected protocol

You can also use internal scripts as part of the FastRules workflow. For more information about FastRules, see "Using FastRules to Process Packet Flow" chapter of the *Cisco Prime Access Registrar 9.0 User Guide.*

To configure internal scripts using the Prime Access Registrar GUI, see the "Using the Graphical User Interface" chapter of the *Cisco Prime Access Registrar 9.0 User Guide*.

CLI to Configure Internal Scripts

A sample CLI to configure internal script statements is given below:

```
--> cd /r/scripts/test

[ //localhost/Radius/Scripts/test ]
    Name = test
    Description =
    Language = internal
    Statements/

--> cd statements/

[ //localhost/Radius/Scripts/test/Statements ]
    1. #req:User-Name=~(.*)(@[a-z]+.[a-z]+)~\1

[ //localhost/Radius/Scripts/test1 ]
    Name = test1
    Description =
    Language = internal
    Statements/

--> cd statements/

[ //localhost/Radius/Scripts/test1/Statements ]
    1. -rsp:Framed-IP-Address=1.1.1.1

-->

[ //localhost/Radius/Scripts/test3/Statements ]
    1. +rsp:Tacacs-AVpair=cmd=show running-config
    2. +rsp:Tacacs-AVPair=aaa
    3. -rsp:Tacacs-AVPair=aaa

-->

[ //localhost/Radius/Scripts/test4/Statements ]
    1. -req:Cisco-AVPair=bbb

--> cd ../../test7/statements/

[ //localhost/Radius/Scripts/test7/Statements ]
    1. #rsp:Framed-IPX-Network=2
    2. +rsp:State=Delivered
    3. -rsp:State
    4. +req:Cisco-AVPair=aaaa
    5. #req:Cisco-AVPair=5
    6. #rsp:Framed-IPX-Network=req:Cisco-AVPair
    7. -req:Cisco-AVPair
```

Blacklisting Script

Prime Access Registrar supports two types of blacklisting:

- IMSI-Based Blacklisting, page 7-16
- IP-Based Blacklisting, page 7-17

Note One instance of Prime Access Registrar can have only one type of blacklisting; either IMSI-based or IP-based.

Blacklisting support is available for the following:

- Diameter remote server—You can choose to configure blacklisting as part of the outgoing script of the remote server
- SIGTRAN-M3UA remote server—You can choose to configure blacklisting as part of the global title translation script of the remote server.

IMSI-Based Blacklisting

Prime Access Registrar allows you to blacklist one or more IMSI values available in the incoming EAP-SIM or EAP-AKA requests. A scripting point option is provided such that you can set an environment dictionary variable Blacklisted-IMSI to ***TRUE*** or ***FALSE*** to blacklist or whitelist IMSI values respectively. An IMSI value marked as blacklisted is rejected and will not be processed further.

The IMSI-based blacklisting script is shown below:

```
proc CheckBlackList {request response environ}
{
    set imsi [ $environ get IMSI ]
    if { [ string compare $imsi 984579621012345 ] == 0 }
    {
        $environ put Blacklisted-IMSI TRUE
        $environ put Notification-Code 19384
    }
}
```

Where, ***CheckBlackList*** is the entrypoint variable of the global title translation script *checklist*, as shown in the example below:

```
[ //localhost/Radius/Scripts/checklist ]
    Name = checklist
    Description =
    Language = tcl
    Filename = tclscript.tcl
    EntryPoint = CheckBlackList
    InitEntryPoint =
    InitEntryPointArgs =
```

If the environment variable *Blacklisted-IMSI* is set as ***TRUE*** and if the IMSI value available in the incoming script matches the given string, then that IMSI is blacklisted and will not be processed. You can configure a notification code to represent failure. If no notification code is set, 16384 representing *General Failure* is sent upon rejection of an IMSI value. For more information about the Notification-Code variable, see Cisco Prime Access Registrar 9.0 Reference Guide

IP-Based Blacklisting

Prime Access Registrar allows you to blacklist one or more IP addresses available in the incoming EAP-SIM or EAP-AKA requests. A scripting point option is provided such that you can set an environment dictionary variable Blacklisted-Key to the IP address that you want to check. An IP addressed mapped to the Blacklisted-Key variable is rejected and will not be processed.

Run the following script:

```
proc CheckIPRange {request response environ}

{
$environ put "Blacklisted-Key" [ $request get Framed-IP-Address ]
}
```

Where:

- **Blacklisted-Key** is the environment variable that is mapped with the IP address to be checked
- ***CheckIPRange*** is the entrypoint variable of the global title translation script *checkIPList*, as shown in the example below:

```
[ //localhost/Radius/Scripts/CheckIPRange ]
    Name = CheckIPRange
    Description =
    Language = rex
    Filename = librexblacklist.so
    EntryPoint = CheckIPList
    InitEntryPoint = InitIPList
    InitEntryPointArgs =
/opt/CSCOar/logs/Whitelist.txt,Blacklisted-Key,Whitelist,10
```

If the IP address available in the incoming script is mapped as Blacklisted-Key, then that IP address is blacklisted and will not be processed further. You can configure a notification code to represent failure. If no notification code is set, 16384 representing *General Failure* is sent upon rejection of an IP address. For more information about the Notification-Code variable, see
Cisco Prime Access Registrar 9.0 Reference Guide

Extension Points in Cisco Prime Access Registrar

Table 7-1 lists the scripting points used in Prime Access Registrar. You can find the scripting point information in the file /cisco-ar/examples/rexscript/rex.h.

Note the following:

- All the scripting points can be used for processing packets of type RADIUS, Diameter, or TACACS+.
- The packet type—RADIUS, Diameter, or TACACS+—can be obtained from the ‘Request-Type’ or ‘Response-Type’ attribute of the environment dictionary. For more details, see
  Cisco Prime Access Registrar 9.0 Reference Guide

Table 7-1 Scripting Points in Prime Access Registrar

Service	Scripting Point Number	Scripting Point	Description	Sample Use Case (if applicable)
—	1	Server incoming Command-line interface (CLI) configuration path: /Radius/IncomingScript	Script runs for every request packet.	To inspect each packet and determine the kind of service (authentication, authorization, or accounting) to be provided.
		Remote Server incoming CLI configuration path: /Radius/RemoteServers/<remote server name>	Script runs when a packet is received from a remote server.	To add/modify/delete attributes coming from the remote server response.
—	2	Vendor incoming CLI configuration path: /Radius/Vendors/<vendorname>/IncomingScript	Script runs only for requests from all the clients mapped to a particular vendor.	To translate vendor proprietary attributes to RADIUS/Diameter vendor-specific attributes.
—	3	Client incoming CLI configuration path: /Radius/Clients/<clientname>/IncomingScript	Script runs only for requests from the specified client (router, NAS, and so on).	To choose AAA service based on an NAS IP address.
—	5	Service incoming CLI configuration path: /Radius/Services/<servicename>/IncomingScript	Script runs when a service is called.	To add/modify/delete attributes before calling a particular service.
Local	6	User group authentication CLI configuration path: /Radius/UserGroups/<name>/AuthenticationScript	Script runs for authentication requests for a user belonging to a particular user group.	—
	7	User record authentication CLI configuration path: /Radius/UserLists/<Userlistname>/<username>/AuthenticationScript	Script runs for authentication requests for a particular user.	—
	8	User group authorization CLI configuration path: /Radius/UserGroups/<name>/AuthorizationScript	Script runs for authorization requests for a user belonging to a particular user group.	—
	9	User record authorization CLI configuration path: /Radius/UserLists/<Userlistname>/<username>/AuthorizationScript	Script runs for authorization requests for a particular user.	—

Table 7-1 Scripting Points in Prime Access Registrar (continued)

Service	Scripting Point Number	Scripting Point	Description	Sample Use Case (if applicable)
—	10	Service outgoing CLI configuration path: /Radius/Services/<servicename>/OutgoingScript	Script runs after a service is executed.	To add/modify/delete attributes before sending the response.
—	12	Client outgoing CLI configuration path: /Radius/Clients/<clientname>/OutgoingScript	Script runs when sending a response against the request received from a specific client (router, NAS, and so on.)	To add/modify/delete attributes before sending a response to the particular client.
—	13	Vendor outgoing CLI configuration path: /Radius/Vendors/<vendorname>/OutgoingScript	Script runs when sending a response against the requests received from all the clients mapped to a particular vendor.	To translate RADIUS/Diameter vendor-specific attributes to the appropriate vendor proprietary attributes.
—	14	Server outgoing CLI configuration path: /Radius/OutgoingScript	Script runs for every response packet.	To add/modify/delete attributes before sending the response.
		Remote Server outgoing CLI configuration path: /Radius/RemoteServers/<remote server name>	Script runs when a packet is sent out to a remote server.	To add/modify/delete attributes in a packet going to a remote server.
Remote	16	Remote server outage CLI configuration path: /Radius/Services/<service name>	Script runs when the remote server is down.	To add/modify/delete attributes when the remote servers defined in the service are down.
Session Manager	19	Session manager incoming CLI configuration path: /Radius/SessionManagers/<session manager name>	Script runs to direct requests to be processed by a specific session manager after authentication and authorization.	To add attributes to the cache for a user before the session manager is called.
	20	Session manager outgoing CLI configuration path: /Radius/SessionManagers/<session manager name>	Script runs to direct responses from a specific session manager after authentication and authorization.	To translate the session cache attribute before sending the response.

Table 7-1 Scripting Points in Prime Access Registrar (continued)

Service	Scripting Point Number	Scripting Point	Description	Sample Use Case (if applicable)
—	21	Virtual server outgoing	Script set any time during packet run time and runs after Server Outgoing scripting point. Can be triggered from any other scripting point by setting the environment variable **Virtual-Server-Outgoing-Script**. To learn more about environment variables, see Cisco Prime Access Registrar 9.0 R eference Guide. This scripting point is applicable only for RADIUS packets.	—
SIGTRAN/ M3UA	22	Global title translation CLI configuration path: /Radius/RemoteServers/<re mote server name>	Script runs before sending out request to Home Registration Register (HLR) to perform Global Title Translation (GTT).	To define GTT based on incoming International Mobile Subscriber Identity (IMSI) ranges.
EAP-AKA	23	Quintet generation CLI configuration path: /Radius/Services/<EAP-AK A service name>	Script runs to generate quintets by using a simulator for EAP-AKA service.	To fetch quintet information from a configured file based on the incoming IMSI.
Translation	24	IMSI translation CLI configuration path: /Radius/RemoteServers/<re mote server name>	Script runs to change the incoming IMSI.	To modify the incoming IMSI ranges before sending a request to a Signal Transfer Point (STP)/HLR.
FastRules	25	FastRule	Script is configured within a fast rule execution path.	To add an attribute in the same flow after authentication and authorization of a packet.
Request Translation	26	Prerequest translation CLI configuration path: /Radius/Services/<service name>	To add/modify/delete incoming RADIUS/Diameter attribute values in the request before translation.	To add/modify/delete incoming RADIUS/Diameter attribute values in the request before translation.
	27	Postrequest translation CLI configuration path: /Radius/Services/<service name>	To add/modify/delete translated Diameter/RADIUS attributes in the request after translation.	To add/modify/delete translated Diameter/RADIUS attributes in the request after translation.

Table 7-1 Scripting Points in Prime Access Registrar (continued)

Service	Scripting Point Number	Scripting Point	Description	Sample Use Case (if applicable)
Response Translation	28	Preresponse translation CLI configuration path: /Radius/Services/<service name>	To add/modify/delete Diameter/RADIUS attribute values in the response before translation.	To add/modify/delete Diameter/RADIUS attribute values in the response before translation.
	29	Postresponse translation CLI configuration path: /Radius/Services/<service name>	To add/modify/delete RADIUS/Diameter attributes in the response after translation.	To add/modify/delete RADIUS/Diameter attributes in the response after translation.