CHAPTER 9

Using the Scripting Interface

This section refers to the Scripting Interface (SI) feature for Linux, Mac OS X, and Windows platforms.

The SI feature provides an interface between a third party script, which gathers posture information, and Cisco Trust Agent, which relays the posture information to the Cisco Secure Access Control Server (ACS).

The SI is provided for customers who do not want to create a posture plugin for their application but still want the application to provide posture information.

Certain filenames and file locations provided in this chapter are platform-dependent. To keep the text platform impartial, the platform-specific names were removed and replaced with a generic name. Table 9-1 provides the actual platform-independent names and how they correspond to those on specific platforms.

Table 9-1 Platform-specific Names and Corresponding Platforms

Impartial platform file Names	Windows file names	Linux and Mac OS X file names
ctasi	ctasi.exe	ctasi
ctascriptpp	ctascriptPP.dll	ctascriptpp.so

This chapter contains the following sections:

Scripting Interface Overview

This section describes how the NAC environment is configured to support posture scripts, how the Scripting Interface sends posture information to Cisco Secure Access Control Server (ACS), and it explains the scripting interface components in more detail.

CTA Scripting Interface components enable third party scripts to relay posture credentials, collected from the system, to CTA. The Scripting Interface functionality requires the following components:

- **ctasi** - The scripting Interface (SI) file.
- **ctascriptpp** - The posture plugin file that sends posture information to ACS.
- **Information** file (.inf) - This is a file that associates the script with ctascriptpp posture plugin.
- **Posture script** - The script that gathers posture data information and creates a posture data file.
- **Posture data files** - The plain text file that contains the posture information retrieved by a posture script.

How the Scripting Interface Relays Posture Credentials to ACS

Once the NAC environment has been configured to retrieve posture data from posture scripts, the CTA Scripting Interface can relay posture credentials to CTA and CTA relays the posture credentials to ACS. (See "Configuring the NAC Environment to Use Your Posture Script" section on page 9-13 for more information on configuring the NAC environment.)

The Scripting Interface functionality follows this workflow to gather posture data and pass it on to ACS:

To relay posture credentials, third party scripts must follow these steps:

Step 1 The third party script runs and generates a posture data file. The posture data file is a plain text file. See the "Posture Data Files" section on page 9-7.

Step 2 The third party script invokes the ctasi file. ctasi stores the information in a posture database record.

Step 3 During a posture request, the ACS requests the same posture information that the posture script requested. The ACS can request the same information as the script because the posture attributes of the script were added to the ACS dictionary. See, "Configuring the NAC Environment to Use Your Posture Script" section on page 9-13.

Step 4 To fulfill the posture request, ctascriptPP Posture Plugin gathers the information stored in the posture database record and forwards it to CTA.

Step 5 CTA sends the requested posture credentials to ACS.

Step 6 The ACS checks the posture credentials against its rules and policies and determines a posture for the application and for the system. Examples of posture are Healthy, Transition, Quarantine, Infected, or Unknown.

ctasi Scripting Interface File

The ctasi executable file is a component supplied by Cisco Systems. It provides the SI external interface to posture scripts. Each posture script invokes the ctasi file. This script passes ctasi the full path to the posture data file where the collected posture data is stored. The ctasi file then stores this information in a posture database record. See "Posture Scripts Invoking ctasi" section on page 9-18 for more information about how posture scripts invoke ctasi and "Status Change" section on page 9-20 for information about how ctasi acts on a change in posture status.

ctascriptpp Posture Plugin File

The ctascriptpp file is a Cisco Systems supplied component. It is a Posture Plugin (PP) that interfaces with CTA. The ctascriptpp retrieves posture credentials requested by a posture script, responds to status change queries, and handles posture notifications by CTA. (For more information the ctascriptpp file see, "CTA Scripting Posture Plugin" section on page 7-9.)

An information file created for use with a posture script must set the value of PluginName to ctascriptPP.dll on Windows platforms or ctascriptPP.so for non-Windows platforms. For more information about information files see, "Information Files".

Information Files

Information files (.inf files) are plain text files with an .inf file extension; they are supplied by the author of the posture script with which they are associated.

Similar to all posture plugins, the ctascriptpp file needs to be associated with at least one information file for CTA to register it as a posture plugin.

The information files associated with the posture scripts must always point to the ctascriptpp file, list VendorID=9, and list VendorIDName=Cisco Systems. The VendorID and VendorIDName identify Cisco Systems as the provider of the credential information.

Sample Information File for Windows

```
[main]
PluginName=ctascriptPP.dll
VendorID=9
VendorIDName=Cisco Systems
Styles=SupportAsync
AppList=script_z
[script_z]
AppType=61440
```

Sample Information File for Linux and Mac OS X

```
[main]
PluginName=ctascriptpp.so
VendorID=9
VendorIDName=Cisco Systems
Styles=SupportAsync
AppList=script_z
```

```
[script_z]
AppType=61440
```

Table 9-2 information File Parameter Descriptions

Parameter	Description	Values	Required
[main]	Defines the first section of the .inf file.	[main]	YES
PluginName	Name of the plugin that the third party script uses.	ctascriptPP.dll (Windows) ctascriptpp.so (Linux or Mac OS X)	YES
VendorID	Defines Cisco as the provider of the credential using a number.	9	YES
VendorIDName	Defines Cisco as the provider of the credentials using a name.	Cisco Systems	YES
Styles	Indicates to the CTA that Scripting Interface reports status changes asynchronously.	SupportAsync	NO
AppList	This field lists all the applications defined later in the .inf file. If there is more than one application, defined in the .inf file, list all applications in this field and separate their names with a comma. For example `AppList=script_z,script_a,script_b`	A string. No spaces are permitted.	YES
[AppListSection]	The name of this section corresponds to a value defined in the AppList parameter. For example [script_z] or [script_a] or [script_b].	A string. No spaces are permitted.	YES
AppType	Numbers in this range are reserved for Cisco Systems, Inc. and indicate to the ACS that a script collected the posture credentials.	61440 (0xF000) - 65535 (0xFFFF).	YES

Posture Scripts

A posture script is written for one application, operating system, or other object. The script defines which properties of that object are going to be used to determine the "posture" of the application. Examples of posture are Healthy, Transition, Quarantine, Infected, or Unknown.

The posture script must adhere to these guidelines and performs these tasks:

- Posture scripts can be written in any scripting language.
- Posture scripts define what attributes to collect.
- The script must produce a posture data file that conforms to the syntax defined in "Posture Data Files" section on page 9-7. The posture data file contains the values of the attributes the posture script collected.
- The script must produce one posture data file for each AppType defined by the script's information file.
- On Linux and Mac OS X operating systems, the posture script must write the posture data file into the /var/tmp/CiscoTrustAgent/pdata directory.
- On Windows operating systems, the posture script can write the posture data file to any directory.
- The posture script must specify that the posture data file it creates can be read by ciscotauser and that the directory in which it is stored can be read by ciscotauser.
- For each posture data file that a script produces, the script must invoke the ctasi scripting interface file separately so that the posture data file may be converted to the posture database record. See "Posture Scripts Invoking ctasi" section on page 9-18 for more information about how posture scripts invoke ctasi.

Posture Data Files

A posture data file is a repository for the posture information gathered by a posture script. This posture data is stored as **plain text** and grouped into posture validation attributes.

A posture data file can only contain data for a single AppType, as defined by the information file used by the posture script. If the posture script creates a combined posture data file for several AppTypes, ctasi will discard the file as invalid and no avpdata file will be created.

A posture validation attribute is composed of a definition header and attribute-value pairs that describe the characteristics of the attribute. (See Table 9-3 on page 9-11, and Table 9-4 on page 9-12).

The posture validation attribute is uniquely defined by combining the values of the vendor-id, application-id, and the attribute-id. These values are defined in the posture data file.

Note The vendor-id value in the posture data file is the same as the VendorID value in the information file. The application-id value in the posture data file is the same as the AppType value in the information file.

A line in the posture data file can be no more than 511 characters long. Any character in the line after the 511th character is truncated.

The syntax of the posture data file is derived from the Posture Validation Attribute Definition file employed by the ACS CSUtil database utility. (For more information about the ACS CSUtil database utility, see the *User Guide for Cisco Secure ACS for Windows Server*.) Each individual script must produce posture data files that conforms to the format and the syntax of the posture data file (see the Sample Posture Data File, page 9-9).

Each time a script runs, prior to invoking the ctasi file, it writes the collected posture information into its corresponding posture data file. The ctasi executable file, when instructed by the posture script, reads the posture information from the posture data file.

Creating Posture Data Files on Linux and Mac OS X Operating Systems

On a Linux or Mac OS X operating system, the posture scripts must create posture data files in the following directory:

```
/var/tmp/CiscoTrustAgent/pdata
```

Note This directory is created during the SI feature installation.

Creating Posture Data Files on Windows Operating Systems

On a Windows system, the posture scripts can create posture data files anywhere on the host disk.

Sample Posture Data File

The sample posture data file is a plain text file that lists the mandatory and the optional keys in each section. The following example shows a sample posture data file to illustrate the required syntax. This sample is valid for all operating systems. Table 9-3 explains each entry and indicates whether it is required or optional.

```
[attr#0]
vendor-id=9
vendor-name=Cisco
application-id=61440
application-name=Script-001
attribute-id=32768
attribute-name=Script-Name
attribute-profile=in
attribute-type=string
attribute-value=Script "posture_file_01"

[attr#1]
vendor-id=9
vendor-name=Cisco
application-id=61440
application-name=Script-001
attribute-id=32769
attribute-name=Unsigned-Integer
attribute-profile=in
attribute-type=unsigned integer
attribute-value=31

[attr#2]
vendor-id=9
vendor-name=Cisco
application-id=61440
application-name=Script-001
attribute-id=32770
attribute-name=Host-IP-Address
attribute-profile=in
attribute-type=ipaddr
attribute-value=196.68.1.99

[attr#3]
```

```
vendor-id=9
vendor-name=Cisco
application-id=61440
application-name=Script-001
attribute-id=32772
attribute-name=Date-the-Script-was-written
attribute-profile=in
attribute-type=date
attribute-value=1077771601

[attr#4]
vendor-id=9
vendor-name=Cisco
application-id=61440
application-name=Script-001
attribute-id=32773
attribute-name=Script-Version
attribute-profile=in
attribute-type=version
attribute-value=1.0.3.5

[attr#5]
vendor-id=9
vendor-name=Cisco
application-id=61440
application-name=Script-001
attribute-id=32774
attribute-name=octet-array-sample
attribute-profile=in
attribute-type=octet-array
attribute-value=0x11 0xbf 0x0a 0x0c

[attr#6]
vendor-id=9
vendor-name=Cisco
application-id=61440
application-name=Script-001
attribute-id=32776
attribute-name=Integer-(signed)
attribute-profile=in
attribute-type=integer
attribute-value=-5
```

Table 9-3 Posture Data File Definitions

Keyword	Description	Values	Required
[attr#n]	The attribute identifier. Open and closed square brackets denote the key value pairs for a single attribute.	Append letters “attr” with a running index.	YES
vendor-id	A globally unique vendor identifier (used by RADIUS) assigned by the Internet Assigned Numbers Authority (IANA).	Nine corresponds to Cisco Systems.	NO
vendor-name	The vendor name represented by the vendor-id.	Cisco	NO
application-id	The posture App-Type integer.	A value from the SI values of 61440 (0xF000) to 65535 (0xFFFF). If it is not one of these values, the posture data file is discarded as invalid.	YES
application-name	The textural representation of the posture App-Type.	Use the name of the third party script that created the file.	NO
attribute-id	The attribute-code integer value.	Any value from private AVPs (32768 to 65535). Any attribute-id with a value outside of this range is discarded as illegal.	YES
attribute-name	The textual representation of the attribute-name.	Free text. No spaces allowed.	NO
attribute-profile	Consistent with the configuration file format used by ACS.	Always ignored by the ctasi file.	NO

Table 9-3 Posture Data File Definitions (continued)

Keyword	Description	Values	Required
attribute-type	The data type.	string integer unsigned integer ipaddr date version octet-array	YES
attribute-value	The attribute value.	The correct syntax for each attribute datatype is described in Table 9-4.	YES

Table 9-4, Syntax for Attribute Datatype Values is shown below:

Table 9-4 Syntax for Attribute Datatype Values

Data Type	Description	Example
octet-array	A space separated stream of strings representing the values of each octet in the array in hexadecimal format.	Octet array {10,32,17,1} will be represented by 0x0A 0x20 0x11 0x01.
integer	A signed value representing 32-bit signed integer value. Valid Range: [-2147483647 ... 2147483646]	Value of -5 will be represented by -5. Value of 10 can be represented by 10 or +10.
unsigned integer	An unsigned value representing the 32-bit unsigned value.	Value of 31 will be represented by a value 31.
string	A free text string.	Script "posture_file_01"

Table 9-4, Syntax for Attribute Datatype Values is shown below:

Table 9-4 Syntax for Attribute Datatype Values

Data Type	Description	Example
ipaddr	The conventional decimal representation of an IP address version 4. Each one of the 4 octets of the IP address is represented by the decimal value of the octet, and they are separated by a period.	196.68.1.99
date	32-bit unsigned value representing the number of seconds elapsed since midnight (00:00:00), January 1, 1970, coordinated universal time (UTC) represented by decimal digits. No negative values allowed.	0 corresponding to Midnight (00:00:00), January 1, 1970.
version	A string representation of a version. Ideally, this string conforms with the NAC concept of version, which is comprised of 4 integers separated by periods. major.minor.revision.build	2.1.0

Configuring the NAC Environment to Use Your Posture Script

These are the tasks you need to perform in order for your posture script to return posture information to the ACS. These tasks assume that the NAC environment itself is already configured and that the endpoint can return posture based only on the CTA Posture Plugin and the Host Posture Plugin.

Step 1 Write a Posture Script

Step 2 Write an Information File for the Posture Script

Step 3 Register Posture Scripts

Step 4 Add Script Interface Attributes to the ACS Dictionary

Step 5 Configure ACS Rules to Determine Posture Based on the Script's Posture Attributes

Write a Posture Script

Write a script to collect posture information from an application. The script identifies the application's attributes which will determine the application's "posture," the script outputs the posture information in the standard format of the posture data file, and the script invokes CTA's ctasi executable file. The script can be written in any scripting language. See "Posture Scripts" section on page 9-7 to learn about the requirements for writing posture scripts.

Write an Information File for the Posture Script

Write an information file that associates the posture script with the ctascriptpp posture plugin. See the "Information Files" section on page 9-5 for a description of this file.

Register Posture Scripts

Posture scripts must be registered with CTA before they can communicate with this application. To register scripts, place the information file that corresponds to the posture script in the appropriate Linux, Mac OS X, or Windows directory before the script runs. See the "Information Files" section on page 9-5 for more information about the information file.

To register scripts on Linux or Mac OS X systems, copy the information file to the following directory:

```
/opt/PostureAgent/Plugins/install
```

To register scripts on Windows platforms, copy the information files to the following directory:

```
%COMMONPROGRAMFILES%\PostureAgent\Plugins\Install
```

On Windows operating systems, `%COMMONPROGRAMFILES%` is the profile location for any logged on user. For example, on Windows XP, `%COMMONPROGRAMFILES%` is assigned the following default directory:
`\Program Files\Common Files.`

For all operating systems, ensure that **ciscotauser** has permission to read the information file.

Note Choose unique filenames to prevent them from being overwritten by another script's registration process.

Add Script Interface Attributes to the ACS Dictionary

Add the application's posture validation attributes, identified in the script, to the ACS attribute dictionary. ACS will then be able to recognize them when their values are reported by CTA.

The posture data collected by a posture script is identified by the vendor-id, application-id of the application, and attribute-id of the posture attribute. The values of application-id and attribute-id are not assigned to a company or an application; they are chosen from a range of valid values by the author of the posture script.

The ACS dictionary does not contain all combinations of application-id and attribute-id values by default. If your script uses these values, you must add these values to the ACS dictionary.

For a more thorough description of how the CSUtil.exe utility is used in this process, see *Appendix D:CSUtil Database Utility*, in the *User Guide for Cisco Secure Access Control Server.*

To add the application-id and attribute-id values your script uses to the ACS dictionary, perform the tasks in these sections:

Step 1 Create a Posture-Validation Attribute Definition File

Step 2 Add the Attributes to the ACS Dictionary

Step 3 Verify the Contents of the ACS Dictionary

Step 4 Restart ACS Services

Create a Posture-Validation Attribute Definition File

A posture-validation attribute definition file is a text file that contains one or more posture-validation attribute definitions. Each definition comprises a definition header and several values.

Tip Use a semicolon (;) to identify lines that are comments.

The only difference between the posture-validation attribute definition file and the posture data file is that the posture data file uses one additional attribute-value pair, `attribute-value=`, to the description of each attribute. See "Sample Posture Data File" section on page 9-9 for an example of this attribute-value pair file and a descriptions of the attribute.

```
[attr#0]
vendor-id=9
vendor-name=Cisco
application-id=61440
application-name=Script-001
attribute-id=32768
attribute-name=Script-Name
attribute-profile=in
attribute-type=string

[attr#1]
vendor-id=9
vendor-name=Cisco
application-id=61440
application-name=Script-001
attribute-id=32769
attribute-name=Unsigned-Integer
attribute-profile=in
attribute-type=unsigned integer

[attr#2]
vendor-id=9
vendor-name=Cisco Systems
application-id=61440
application-name=Script-001
attribute-id=32770
attribute-name=Host-IP-Address
attribute-profile=in
attribute-type=ipaddr
```

Add the Attributes to the ACS Dictionary

The **-addAVP** option imports posture-validation attribute definitions into ACS from the posture-validation attribute definition file described in, Create a Posture-Validation Attribute Definition File, page 9-16.

Before You Begin

Because completing this procedure requires restarting the **CSAuth** service, which temporarily suspends authentication services, consider performing this procedure when demand for ACS services is low. Use the steps in "Verify the Contents of the ACS Dictionary" section on page 9-17 to create a backup of posture-validation attribute definitions. You can also use the exported attribute definition file to double-check the vendor ID, application ID, and attribute ID of current posture-validation attributes.

Add attributes to the ACS dictionary using the CSUtil utility. This utility is provided with your ACS.

Step 1 On the ACS server, open a command prompt window.

Step 2 Connect to the directory \Program Files\CiscoSecure ACS V4.0\bin.

Step 3 At the prompt type: `CSUtil.exe -addAVP` *`filename`*

In the -addAVP command above filename indicates the posture-validation attribute definition file you created in the "Create a Posture-Validation Attribute Definition File" section on page 9-16. If the posture-validation attribute definition file is not put in the same directory as the CSUtil.exe application, you must specify the full path to the file.

Verify the Contents of the ACS Dictionary

To verify that your attributes were added to the ACS dictionary, follow this procedure:

Step 1 On the ACS server, open a command prompt window.

Step 2 Connect to the directory \Program Files\CiscoSecure ACS V4.0\bin.

Step 3 At the prompt, type the following:

`CSUtil.exe -dumpAVP` *`avp_new_dump.txt`*

In the command above the avp_new_dump.txt represents the name you choose for the dump file. The command creates the file you name and saves it in the \Program Files\CiscoSecure ACS V4.0\bin directory.

Restart ACS Services

Once you have confirmed that the attributes you intended to add to the ACS dictionary were properly added you need to restart these ACS services for the changes to take affect:

- csauth
- cslog
- csadmin

You can restart these services using the Windows Services window.

Configure ACS Rules to Determine Posture Based on the Script's Posture Attributes

On the ACS, configure the rules and policies that determine the application's posture. These rules and policies are based on the posture attributes you added to the ACS attribute dictionary in "Add Script Interface Attributes to the ACS Dictionary" section on page 9-15. For example, a rule might determine that the application's posture is "Healthy" if the application's version is equal to or greater than "2.0.0.0." See the *User Guide for Cisco Secure Access Control Server* documentation for more information about Posture Validation rules and Network Access Profiles.

At this point a posture script will be able to relay posture credentials to ACS.

Posture Scripts Invoking ctasi

After the posture script has run and collected the relevant posture information, it invokes the ctasi file and provides one of two input parameters.

On Windows platforms, invoke ctasi with the command below:

```
ctasi.exe <dir_path>posture_file n
```

ctasi is located in the following Windows directory:
\Program Files\Common Files\PostureAgent

On Linux and Mac OS X platforms, invoke ctasi with the command below:
`ctasi <dir_path>posture_file` *n*

For non-windows operating systems, ctasi is located in this directory:
/opt/PostureAgent/

Note In either case of the posture script invoking ctasi, there can be no spaces in the directory path. If there are spaces in the name of the directory path, enclose the entire path and file name in quotation marks. For example:

```
ctasi.exe "c:\Posture Data Files\posture_file.txt" 1
```

The options for the commands that invoke ctasi have the same function in all operating system environments.

The first command option, `<dir_path>posture_file`, is required. This is the full path where the corresponding file with the posture data information was saved. On Windows platforms this path may be to any location. On Linux and Mac OS X platforms, this path leads to this directory:

```
/var/tmp/CiscoTrustAgent/pdata
```

The second command option, `n`, is optional. The second option is a number ranging from 1 to 2. This option allows the third party to do the following:

- When the script passes the value n=1, it instructs ctasi that the script has already detected a status change and CTA should accept this status change irrespective of the posture data file contents.
- When the script passes the value n=2, it instructs ctasi that the script determined that there was NO status change and CTA should accept this fact irrespective of the posture data file contents.

If the second optional parameter is omitted, CTA determines if there was a status change or not. CTA makes this decision by comparing the most recent posture reported by the script with the previous posture reported by the script for the same Application Type.

Note There is one exception where the ctasi ignores the no status change instruction by a posture script. This occurs when the ctasi finds no corresponding pre-existing posture database record on the system. The rationale is because no pre-existing posture database record exists on the system, posture for the Application Type may not have been collected, or it was collected so long ago that it was removed as stale.

Status Change

If a Status Change has been relayed to or detected by the ctasi file, then ctasi file asynchronously notifies CTA about this status change because the SupportAsync notification style is always performed by the Scripting Interface.

However, ctasi can only notify CTA and not the NAD. It is the CTA that has the responsibility to notify the NAD about this status change.

In the layer 2 implementation of NAC, this means that once ctasi reports the status change to CTA, CTA will report the status change to the NAD and a new revalidation will be initiated immediately because the underlying protocol allows the CTA to asynchronously notify the NAD about the Status Change.

In a layer 3 implementation of NAC, the underlying protocol does not allow CTA to asynchronously relay this or any other Status Change to the NAD. CTA has to wait for the Status Change Query request from the NAD in order to report it. Therefore, a new revalidation will not be attempted until the Status Change Query comes from the NAD.

Stale Posture Data

Because the ctasi file may not have been invoked for an extended period of time, you may want to invalidate your posture credentials. For example, if a posture script has been deleted or blocked it can not update the posture data file. As a result, the last collected posture information is preserved in the posture data file and used whether it is current or out of date.

You can use the Write-TimeStamp Attribute (attribute-id=15) type to minimize problems occurring from stale posture data.

Note attribute-id 15 is defined to be a data type date. It contains the time, in UTC format, when the posture database record was last updated by the ctasi file. This AVP is paramount to Stale Posture management.

Each entry in the posture database that corresponds to a unique application-id is associated with a Write-TimeStamp attribute. The attribute value is set by the ctasi file. This file updates the attribute value with the system UTC clock value each time it updates the posture database record that correlates to the particular application-id.

To eliminate stale posture data, see the Managing Stale Posture Database with CTA, page 9-21, and the Managing Stale Posture Database on the ACS Server, page 9-22.

Managing Stale Posture Database with CTA

You can add a configurable time value, delta_stale, that is set in the [Scripting Interface] section of the ctad.ini file to help manage stale posture data, as shown in example 9-1.

Example 9-1 Sample ctad.ini File

```
; ctad.ini sample

[Scripting_Interface]
delta_stale=20
```

This is a universal value that is defined in minutes. It corresponds to all posture database records that are related to the SI. If this entry is missing from the ctad.ini file, a default value of (60*24*30 = 43200) one full, 30-day month is used by default. Immediately before the ctascriptpp file opens the relevant posture data record, it uses this value and the current time from the operating system to decide whether the posture data record is stale.

Note Network Administrators should configure the delta_stale value to be the largest possible value desired among all application-id's related to the SI.

Based on the fact that Write-TimeStamp (attribute-id=15) denotes the time that the record was last updated, the record is considered stale if it was last updated more than delta_stale minutes before the current time.

If the posture data record is deemed stale, it is deleted. Subsequently, the ctascriptpp file does not return any posture information to the posture request. This feature is beneficial because after the stale time value delta is exceeded, no stale posture data is transmitted from the network client to the ACS. However, it is limited because only one universal delta value can be configured that corresponds to all posture data records for all application-id's related to the SI.

Managing Stale Posture Database on the ACS Server

The ACS can be configured to detect stale posture data per application-id.

Configuring rules on the ACS, per application-id, involving the Write-TimeStamp (attribute-id=15) is an approach that works in coordination with, and in addition to the Managing Stale Posture Database with CTA approach.

Using the Managing Stale Posture Database on the ACS Server approach a rule (most likely with different stale threshold values) can be set on the ACS per application-id. Using the approach described in the Managing Stale Posture Database with CTA section, only one threshold is set for all application-id's within the SI range.

To create a rule with the Write-TimeStamp attribute value on the ACS, add the Write-TimeStamp attribute to the ACS dictionary as you did in the "Add Script Interface Attributes to the ACS Dictionary" section on page 9-15. This is done in the same way that you would add the SI-specific attributes. Use the CSUtil utility with the appropriate input file as shown below. (See *Appendix D:CSUtil Database Utility*, in the *User Guide for Cisco Secure Access Control Server* for more information about the CSUtil utility):

```
CSUtil -addAVP write_time_stamp.ini
```

See the Sample write_time_stamp.ini example for a sample write_time_stamp.ini file. This sample file is consistent with other examples in this chapter in that the application-id and the application-name are the same.

To add the Write-TimeStamp for your application-id, copy the file and replace the application-id and the application-name values with the application-id and the application-name values that correspond to your script.

Example 9-2 Sample write_time_stamp.ini

```
[attr#0]
vendor-id=9
vendor-name=Cisco
application-id=61440
application-name=Script-001
attribute-id=15
attribute-name=WriteTimeStamp
attribute-profile=in
attribute-type=date
```

With the Write-TimeStamp attribute added to the ACS dictionary, you can configure rules that apply different access policies for the host based on the age of the posture data. This can be determined from the Write-TimeStamp and the current time on the ACS.