Administrator Guide for Cisco Trust Agent, Release 2.1 Without Bundled Supplicant

Revised: May 23, 2008

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

Text Part Number: OL-13680-01

Administrator Guide for Cisco Trust Agent 2.1, Wthout Bundled Supplicant

CONTENTS

Preface

Cisco Trust Agent (CTA) collects and reports posture credentials from clients in a Network Admission Control (NAC) environment.

Posture credentials are information about a NAC-compliant software application or a client on which it runs. These are examples of posture credentials that can be collected from the client: software application versions, machine name, operating system, and the client's MAC Address.

CTA reports the posture information it gathers to the Cisco Secure Access Control Server (ACS) which then determines application posture and an overall client posture. Examples of client posture could be "Healthy," "Quarantine," or "Unknown."

Based on the client's posture a NAC-compliant Network Access Device (NAD), such as a Cisco Switch or Cisco Router, provide the client access to a network.

Cisco Trust Agent 2.1 Release

The goals of Cisco Trust Agent, Release 2.1.103.0 (CTA 2.1) are to improve on the CTA 2.1.18.0 selective availability release by resolving outstanding product defects and to provide new functionality from that offered in the CTA 2.0.0.30 release. Cisco Trust Agent release 2.1 is an integral component of the Network Admission Control Framework 2.1 solution.

This offering of CTA 2.1.103.0 does not include a bundled supplicant as the previous offering of CTA 2.1.103.0 did. We recommend that customers who want to perform 802.1x authentication install the Cisco Secure Services Client, version 4.1.2 or later in addition to CTA 2.1.103.0.

Qualified Deployments of CTA 2.1

Cisco Trust Agent 2.1.103.0 will be distributed to existing customers of CTA and those customers evaluating the NAC Framework 2.1 programs.

CTA 2.1 is not intended for distribution to new customers of CTA nor new customers of the NAC 2.1 Framework solution. New customers to CTA and NAC should work with their Cisco Account Team representative to evaluate their NAC Framework-qualified infrastructure and use-case scenarios.

We are making an extra effort to qualify our customers' infrastructure and to ensure that the components in their network are compatible with the NAC Framework, that their goals will be met by the NAC Framework, and that the deployment of the NAC Framework will be successful.

Product Versioning

The full version number of this release is CTA 2.1.103.0. The full release number is used in installation files names and in the text of the *Administrator Guide for Cisco Trust Agent, Release 2.1, Without Bundled Supplicant* and the *Release Notes for Cisco Trust Agent, Release 2.1, Without Bundled Supplicant* when it is important to distinguish the version of CTA being discussed. Any references in the documentation to CTA 2.1 are referring to CTA 2.1.103.0 unless otherwise noted.

CTA 2.1.103.0 Installation File for Windows Operating System

In this offering of CTA 2.1.103.0, there is one installation file for Windows operating systems: CtaAdminEx-win-2.1.103.0.exe. This contains the ctasetup-win-2.1.103.0.msi file which allows administrators to accept the end user license agreement and install CTA 2.1.103.0. CtaAdminEx-win-2.1.103.0.exe does not contain CTA 802.1x Wired Client or Cisco Secure Services Client.

In the previous offering of CTA 2.1.103.0, there was an additional installation file: CtaAdminEx-supplicant-win-2.1.103.0.exe. This file allowed an administrator to install the CTA 802.1x Wired Client as well as CTA. CtaAdminEx-supplicant-win-2.1.103.0.exe is not provided in this offering of CTA 2.1.103.0.

When migrating from the CTA 802.1x Wired Client to Cisco Secure Services Client, you must uninstall CTA 2.1.103.0 and the CTA 802.1x Wired Client first and then re-install CTA 2.1.103.0 alone using the CtaAdminEx-win-2.1.103.0.exe file.

Cisco Secure Services Client

We recommend the use of the Cisco Secure Services Client, version 4.1.2 with CTA 2.1.103.0 for those customers who perform 802.1x authentication. For customers who deployed the CTA 802.1x Wired Client, with the previous offering of CTA 2.1.103.0, we recommend that you migrate to the Cisco Secure Services Client, version 4.1.2 or later.

Audience

The *Administrator Guide for Cisco Trust Agent, Release 2.1, Without Bundled Supplicant* provides installation, configuration, and monitoring information to administrators responsible for deploying Cisco Trust Agent to network clients.

Conventions

This document uses the following conventions:

Item	Convention
Commands, keywords, special terminology, and options that should be selected during procedures	**boldface** font
Variables for which you supply values and new or important terminology	*italic* font
Displayed session and system information, paths and filenames	`screen` font
Information you enter	**`boldface screen`** font
Variables you enter	*`italic screen`* font

Item	Convention
Menu items and button names	**boldface** font
Indicates menu items to select, in the order you select them	**Option > Network Preferences**

Tip Identifies information to help you get the most benefit from your product.

Note Means *reader take note*. Notes identify important information that you should reflect upon before continuing, contain helpful suggestions, or provide references to materials not contained in the document.

Caution Means *reader be careful*. In this situation, you might do something that could result in equipment damage, loss of data, or a potential breach in your network security.

Warning **Identifies information that you must heed to prevent injuring yourself or damaging the state of the software or equipment. Warnings identify definite security breaches that will result if the information presented is not followed carefully.**

Related Documentation

Note Although every effort has been made to validate the accuracy of the information in the printed and electronic documentation, you should also review Cisco Trust Agent documentation on Cisco.com for any updates.

You can find the documentation for Cisco Trust Agent, Release 2.1.103.0 by navigating Cisco.com starting at this link: http://www.cisco.com/en/US/products/ps5923/tsd_products_support_series_home.html. These are the documents that describe this offering of Cisco Trust Agent 2.1.103.0:

- *Migrating from CTA 802.1x Wired Client to Cisco Secure Services Client*
- *Administrator Guide for Cisco Trust Agent, Release 2.1, Without Bundled Supplicant*
- *Release Notes for Cisco Trust Agent, Release 2.1, Without Bundled Supplicant*

You can find the documentation for Cisco Secure Services Client, Release 4.1.2 by navigating Cisco.com starting at this link: http://www.cisco.com/en/US/products/ps7034/tsd_products_support_series_home.html. These are the documents that describe Cisco Secure Services Client:

- *Cisco Secure Services Client Administrator Guide*, for release 4.1.2.
- *Cisco Secure Services Client User Guide*, for release 4.1.2.
- *Release Notes for Cisco Secure Services Client,* for release 4.1.2.

For documentation of other Cisco Network Admission Control (NAC) Framework components follow this link http://www.cisco.com/en/US/netsol/ns617/networking_solutions_sub_solution_home.html.

Obtaining Documentation, Obtaining Support, and Security Guidelines

For information on obtaining documentation, obtaining support, providing documentation feedback, security guidelines, and also recommended aliases and general Cisco documents, see the monthly What's New in Cisco Product Documentation, which also lists all new and revised Cisco technical documentation, at:

http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html

CHAPTER 1

Cisco Trust Agent Overview

Cisco Trust Agent (CTA) is a component of the Cisco Network Admission Control (NAC) program. NAC enables network access devices to permit or deny network clients access to the network based on the posture of the software on the host. This process is called posture validation.

This chapter contains the following sections:

Cisco NAC Overview

The four main components of the Cisco Trust Agent (CTA) posture validation process are as follows:

- **Network client running Cisco Trust Agent**—Cisco Trust Agent collects security posture information from the NAC-compliant applications running on the network client and reports them to the Cisco Secure Access Control Server (ACS). These are some NAC-compliant applications:
  - Antivirus applications
  - Personal firewalls

  - Host-based intrusion protection applications, such as Cisco Security Agent (CSA)

- **Network Access Device (NAD)**—The NAD permits or denies network access. Typically, the NAD is a Cisco router or switch.
- **Authentication Server (Cisco Secure Access Control Server (ACS))**—The authentication server is responsible for obtaining and evaluating the security posture credentials from a network client, determining the overall client posture, and providing the appropriate posture token and network access policy to the NAD based on the client's posture.
- **Posture Validation Servers (optional)**—Posture validation servers support Cisco Secure ACS in determining the overall system posture. They are typically third-party applications that support the validation of the security posture credentials for a specific NAC-compliant application. For example, an antivirus software company may prefer to maintain its own posture validation server rather than store and update posture validation information on Cisco's ACS server.

Posture Validation Process Overview

Initial Posture Validation Process

The following provides an overview of how the posture validation components work together during an initial posture validation process.

1. The computer sends a DHCP (Dynamic Host Configurative Protocol) request to obtain an IP address or an ARP (Address Resolution Protocol) request to convert an IP address into a physical address.
2. The Network Access Device (NAD), a Cisco network router or switch, requests a network access policy from the Cisco Secure Access Control Server (ACS).
3. ACS requests the computer's posture credentials from Cisco Trust Agent (CTA) which was previously installed.
4. CTA receives the security posture credential request and, in turn, prompts the posture plugins installed on the computer to gather the posture credentials from the NAC-compliant applications on the client.

5. CTA aggregates all of the posture credentials from the client and returns the information to the NAD.
   - If the posture credentials are sent using the EAP over UDP protocol, the posture information is sent directly from the CTA to ACS.
   - If the posture credentials are sent using the IEEE 802.1x protocol, CTA hands the posture credentials to the Cisco Secure Services Client (CSSC), also known as the "supplicant" and CSSC forwards the information to the ACS. (Learn more about CSSC on Cisco.com at, http://www.cisco.com/en/US/products/ps7034/index.html.)
6. ACS evaluates the security posture credentials for each application that on the computer that reports them. ACS can perform the credential evaluation using rules in the local database or can relay application credentials to an application-specific posture validation server for evaluation. The result of the evaluation is an application posture token for each evaluated application and an optional user notification.
7. ACS aggregates the application posture credentials and defines an overall system posture token for the client. The system posture token equals the least trusted posture of all the application posture credentials collected from the client.

   These are the default system posture tokens; they are ranked from the most trusted posture to the least trusted posture:
   - Healthy
   - Checkup
   - Quarantine
   - Transition
   - Infected
   - Unknown
8. ACS maps the system posture token to a network access policy and, optionally, a user notification.
9. ACS sends the result of the security posture validation back to the NAD, along with the appropriate network access policy for that client, and any user notifications.

10. The NAD implements the security policy for the client and forwards the posture information back to CTA on the computer. Depending upon how you have configured CTA, the results of the posture validation are logged and any user notifications are displayed on-screen in a dialog box.

    Based on the access policy, the network client can be permitted on the network, denied access to the network, or quarantined to a remediation network until the NAC-compliant applications have been updated to the required levels.

Posture Revalidation Process

Posture revalidation is caused by one of three events and once it occurs posture is validated following the same workflow as described in Posture Validation Process Overview, starting with step 2.

Network Access Device Requests New Posture. In the case of NAC L2 IP or NAC L3 IP network admission methods, switches and routers maintain the posture for a configured amount of time and then that posture expires. When the posture expires, the NAD requests posture again.

An Internal Timer in CTA Expires. The SQTimer parameter defines the interval at which CTA requests the posture plugins for their status. If a posture plugin reports a change in posture status to CTA, CTA alerts the network access device which triggers a re-posturing of the host. This is one implementation of the asynchronous posture status query feature. This feature is only available if Cisco Secure Services Client (CSSC) is installed on the host machine.

See "Configuring Asynchronous Posture Status Query" section on page 5-19 for more information on how to configure the SQTimer.

Posture Plugins Detect a Change in Status. Some posture plugins monitor the status of their applications and report status changes to CTA upon detection. Such plugins are considered "asynchronous" plugins. When CTA receives the status change from an asynchronous plugin, CTA alerts the network access device which triggers a re-posturing of the host. For example, the posture plugin for Cisco Security Agent (CSA) detects when the CSA security has been turned off. This is one implementation of the asynchronous posture status query feature. This feature is only available if Cisco Secure Services Client (CSSC) is installed on the host machine.

Cisco Trust Agent Deployment Considerations

Network clients are the hosts on your network. This includes PCs, laptops, workstations, and servers. You may have hundreds or thousands of network clients on which you are going to install CTA.

To decrease the time spent installing and configuring CTA throughout your network, you can create a custom installation package. The custom installation package lets you provision the CTA settings at installation time. These settings include:

- Communication settings
- Posture notification settings
- Logging and notification setting

CHAPTER 2

Installing the Cisco Trust Agent on Linux Operating Systems

This chapter provides system requirement and installation information for installing Cisco Trust Agent (CTA) on Red Hat Linux operating systems. Read this entire chapter before beginning the installation. There are advanced installation options detailed later in this chapter that you may want to use. For example, before deploying CTA on your network, you can create a custom installation package which allows you to set CTA configuration parameters and provision certificates and plug-ins. Proceeding in this manner could save you time during the CTA deployment process.

See these chapters for installation instructions for other operating systems:

- Chapter 3, "Installing the Cisco Trust Agent on Macintosh Operating Systems."
- Chapter 4, "Installing the Cisco Trust Agent on Windows Operating Systems"

This chapter contains the following sections:

Verifying System Requirements on Linux

Before installing Cisco Trust Agent on a Linux operating system, verify that the target system meets the following requirements:

System Component	Requirement
System	• Pentium class processor or better • Network connection
Operating System and Language Support	All available internationalized versions of these Linux operating systems support CTA 2.1.: • Red Hat Enterprise Linux v3 (Enterprise, Advanced Server, and Workstation) • Red Hat Enterprise Linux v4 (Enterprise, Advanced Server, and Workstation) **Note** **Support for a localized operating system is different from localized version of CTA. The CTA interface and messages are presented in English.**
Linux Installers	Red Hat Package Management (RPM) v4.2 or greater.
Hard Disk Space	20 MB

System Component	Requirement
Memory	256 MB Red Hat Enterprise Linux v3 (Enterprise, Advanced, Workstation) 256 MB Red Hat Enterprise Linux v4 (Enterprise, Advanced, Workstation)
Listening Port	By default, Cisco Trust Agent communicates on UDP port 21862. You can configure this port number. See, "The ctad.ini Configuration File" section on page 5-2 for more information.

Installation File

The installation files for CTA for Linux are contained in the **ctaadminex-linux-2.1.103-0.tar.gz** file**.** That file may be downloaded from Cisco.com. Follow the procedures in "Installing Cisco Trust Agent" to use the file.

Initial Deployments

For large enterprise Linux deployments, administrators may want to deploy CTA with a customized package. This way all required certificates and plug-ins are administratively configured with no end-user interaction or interference. The customized packages can be delivered to many users at once using an automated software deployment mechanism.

CTA Scripting Interface Feature

The Scripting Interface feature allows software developers to write their own scripts to relay posture information, collected on the system, to CTA. The scripts would perform the equivalent function of a posture plugin. Users will not need this feature unless they intend to write posture scripts.

The Scripting Interface is installed by default on Linux installations.

Installing Cisco Trust Agent

In order to install Cisco Trust Agent you log in as the administrative user on the computer.

General Installation Instructions

Step 1 Download the ctaadminex-linux-2.1.103-0.tar.gz from Cisco.com. For the sake of this example, we will store the ctaadminex-linux-2.1.103-0.tar.gz file in /tmp.

Step 2 Follow the procedures in "Extracting the Installation File and Accepting the EULA" section on page 2-4.

Step 3 Install CTA using either of these methods:

- Installing CTA from the Command Line, page 2-5
- Creating a Custom CTA Installation Package, page 2-5

Extracting the Installation File and Accepting the EULA

After downloading the ctaadminex-linux-2.1.103-0.tar.gz file from Cisco.com, use this procedure to extract the CTA installation files and accept the end-user license agreement (EULA).

Step 1 Open a terminal window.

Step 2 Change the directory to the one that contains the ctaadminex-linux-2.1.103-0.tar.gz file. In our example, this directory is /tmp.

Step 3 At the prompt, type the following command and press <**Enter**>.

tar -zxvf ctaadminex-linux-2.1.103-0.tar.gz

The ctaadminex-linux-2.1.103-0.sh file is extracted and placed in the same directory as the ctaadminex-linux-2.1.103-0.tar.gz file.

Step 4 At the prompt, type **./ctaadminex-linux-2.1.103-0.sh** and press <**Enter**>.

Step 5 When prompted, accept the EULA by typing "**y**" and pressing <**Enter**>. The CTA-2.1.103-0 subdirectory is created and the cta-linux-2.1.103-0.i386.rpm is unpacked and placed in that directory. In our example, this new directory would be /tmp/CTA-2.1.103-0.

Installing CTA from the Command Line

Step 1 Follow the procedure in the "Extracting the Installation File and Accepting the EULA" section on page 2-4.

Step 2 Open a terminal window on the client.

Step 3 Change the directory to the directory that contains the cta-linux-2.1.103-0.i386.rpm file. In our example, the directory is /tmp/CTA-2.1.103-0.

Step 4 At the prompt, type **rpm -ivh cta-linux-2.1.103-0.i386.rpm** and press <**Enter**>. You receive these messages indicating that CTA was installed.

Preparing... ################ 100%

1:cta-linux ################ 100%

Step 5 Verify CTA installation using the procedures in "Verifying Cisco Trust Agent Installation on Linux" section on page 2-8.

Step 6 If a CA certificate or a matching root certificate from the Cisco Secure ACS server has not been installed on the network client on which you just installed CTA, you must install one or the other certificates. This enables CTA to establish a secure form of communication with the Cisco Secure ACS server. Refer to "Installing or Updating a Certificate on Linux Operating Systems, page 8-4" for information.

Creating a Custom CTA Installation Package

Use this section as an example of how to create a customized CTA installation on Linux systems.

The CTA installation file is a Red Hat Packet Manager (rpm) file and is installed with standard RPM commands. To create a custom installation package, you create a directory structure which includes the CTA installation file, .ini files,

plugin subdirectory and certificate subdirectory. This directory structure can then be distributed by a software deployment mechanism, such as a script or a software deployment tool.

After the software deployment mechanism distributes the directory structure to the remote network clients, it runs the CTA installation file. The CTA installation file copies the contents of the directory structure to the proper locations on the remote network client. The software deployment mechanism does not need to recompile the CTA installation file to create a custom installation.

The customization choices in this procedure are optional. However, you will find that including some of these customizations is worthwhile. CTA is not a centrally managed product. If you do not plan to use the product defaults, it is to your benefit to pre-configure all available product settings before deploying CTA.

Please read this entire procedure before beginning. There are options detailed later in the instructions that you should be aware of before beginning.

Step 1 Before you create the custom installation package, install CTA on the client you will use to create the custom package. This will install the template ctad.ini file and give you exposure to the CTA installation process. Begin with the "General Installation Instructions" section on page 2-4 to install CTA.

Step 2 Perform the procedure in the "Extracting the Installation File and Accepting the EULA" section on page 2-4.

Step 3 Change the directory to the one that contains the **cta-linux-2.1.103-0.i386.rpm** file. In our example, this is the /tmp/CTA-2.1.103-0 directory.

Step 4 Create a **certs** subdirectory. For example: /tmp/CTA-2.1.103-0/**certs**

Copy the root certificate for your Cisco Secure ACS server to this directory. During installation, any certificates in this directory are added to the systems root certificate store.

If your Cisco Secure ACS server uses self-signed certificates, see the Cisco Secure ACS documentation for information about obtaining the certificate; if you use a CA server, refer to your CA server documentation.

Note This step is optional if a CA certificate or ACS root certificate have already been distributed to the network clients receiving this customized CTA installation. If these certificates have not been distributed, this step is required.

Step 5 Create a **plugins** subdirectory. For example: /tmp/CTA-2.1.103-0/**plugins**

Copy any third party plugins that you want to provision at installation time into this directory.

Step 6 Create a new **ctad.ini** file and store it in the installation directory at the same level as the **cta-linux-2.1.103-0.i386.rpm** file. In our example, this is the /tmp/CTA-2.1.103-0 directory.

The ctad.ini file is used to configure CTA communication settings, user notifications, and certificate validation rules. If you want to change the default communication settings, such as the port number CTA listens over, the maximum number of sessions, and session time-out values, include this file. Refer to Chapter 5, "Configuring Cisco Trust Agent" for instructions on how you should create and format this file.

Step 7 Create a new ctalogd.ini file and store it in the installation directory at the same level as the **cta-linux-2.1.103-0.i386.rpm** file. In our example, this is the /tmp/CTA-2.1.103-0 directory. Refer to Chapter 6, "Cisco Trust Agent Event Logging" for instructions on how you should create and format this file.

Step 8 A software deployment mechanism deploys the customized /tmp/CTA-2.1.103-0 directory to the appropriate network clients and saves it in a local directory.

Step 9 The software deployment mechanism installs CTA and its customizations by following the procedure in the "Installing CTA from the Command Line" section on page 2-5.

Upgrading to Cisco Trust Agent, Release 2.1

Use this procedure to upgrade your CTA installation:

Step 1 Download the ctaadminex-linux-2.1.103-0.tar.gz from Cisco.com. For the sake of this example, we will store the ctaadminex-linux-2.1.103-0.tar.gz file in /tmp.

Step 2 Follow the procedures in "Extracting the Installation File and Accepting the EULA" section on page 2-4.

Step 3 Install CTA using either of these methods:

- Installing CTA from the Command Line, page 2-5. However, when you are ready to install the upgrade, change the installation command to the upgrade command. This is an example:

  rpm -Uvh cta-linux-2.1.103-0.i386

- Creating a Custom CTA Installation Package, page 2-5

Verifying Cisco Trust Agent Installation on Linux

After Cisco Trust Agent has been installed you will find the following directory structures containing CTA's executable files:

- /opt/CiscoTrustAgent
- /opt/PostureAgent

You may also verify which version of CTA is installed by following this procedure:

Step 1 Open a terminal window on the system.

Step 2 Type **rpm -q cta-linux** and press <Enter>.

The version of CTA is returned. In the case of CTA 2.1.103.0, this information will be returned: **cta-linux-2.1.103-0**.

Verifying CTA is Running

Step 1 Open a terminal window on the system.

Step 2 Type rpm -q cta-linux and press <Enter>.

Step 3 Type **`ps -A | grep cta`** and press <**Enter**.>

Step 4 Verify that the following daemons are running:

- ctad

- ctalogd
- ctapsd
- ctaeoud

If these daemons are not running, try rebooting the system. If the daemons still do not run, try reinstalling the application.

Verifying CTA Package Information

Step 1 Open a terminal window on the system.

Step 2 At any prompt, type **rpm -q cta-linux** and press <**Enter**>.

The full package name is returned, for example, **cta-linux-2.1.103-0**

Uninstalling Cisco Trust Agent on Linux

To uninstall Cisco Trust Agent, follow this procedure:

Step 1 Log in to the client as the root user.

Step 2 Open a terminal window.

Step 3 At the prompt, run the following command and press <**Enter**>.

```
#rpm -e cta-linux
```

Cisco Trust Agent is uninstalled. You do not need to reboot the system.

Note Certificates and plugin files are not deleted when CTA is uninstalled; they remain on the client.

CHAPTER 3

Installing the Cisco Trust Agent on Macintosh Operating Systems

This chapter provides system requirement and installation information for installing Cisco Trust Agent (CTA) on Macintosh operating systems. Read this entire chapter before beginning the installation. There are advanced installation options detailed later in this chapter that you may want to use. For example, before deploying CTA on your network, you can create a custom installation package which allows you to set CTA configuration parameters and install certificates and plug-ins. Proceeding in this manner could save you time during the CTA deployment process.

See these chapters for installation instructions for other operating systems:

- Chapter 2, "Installing the Cisco Trust Agent on Linux Operating Systems"
- Chapter 4, "Installing the Cisco Trust Agent on Windows Operating Systems"

This chapter contains the following sections:

System Requirements for Mac OS X

Before installing Cisco Trust Agent on a Mac OS X Operating system, verify that the target system meets the following requirements:

System Component	Requirement
System	• G3 processor and later • Network connection
Free Hard Disk Space	20 MB minimum
Memory	256 MB RAM
Listening Port	By default, Cisco Trust Agent listens on UDP port 21862. You can change this port number, if necessary. See "The ctad.ini Configuration File" section on page 5-2.
Operating System and Language Support	All available internationalized versions of Mac OS X 10.3.9 and 10.4 operating systems support CTA 2.1. **Note** Support for a localized operating system is different from localized version of CTA. The CTA interface and messages are presented in English.

CTA Scripting Interface Feature

The Scripting Interface feature allows software developers to write their own scripts to relay posture information, collected on the system, to CTA. The scripts would perform the equivalent function of a posture plugin. Users will not need this feature unless they intend to write posture scripts.

The Scripting Interface is an optional Feature of CTA.

Installation Files

The installation files for CTA for Mac are contained in the **ctaadminex-darwin-2.1.103.0.tar.gz file.** That file may be downloaded from Cisco.com. Follow the procedures in "Installing Cisco Trust Agent" for descriptions of the files contained in the **ctaadminex-darwin-2.1.103.0.tar.gz** file and their uses.

Installing Cisco Trust Agent

In order to install Cisco Trust Agent you log in as the administrative user on the machine.

General Installation Instructions

Step 1 Download the ctaadminex-darwin-2.1.103.0.tar.gz from Cisco.com and follow the procedures in "Extracting the Installation Image and Accepting the EULA" section on page 3-4.

Step 2 Install CTA using any of these methods:

- Installing CTA from the Command Line, page 3-4
- Installing CTA Using an Installation Wizard, page 3-6
- Installing CTA Using a Custom Installation Package, page 3-12

Step 3 Install Cisco Secure Access Control Server (ACS) root certificate on the end-point if it is not distributed as part of a custom installation package. See "About The ACS Server Root Certificate" section on page 8-3 for information about installing this certificate separately.

Step 4 Verify CTA installation using the "Verifying Cisco Trust Agent Installation" section on page 3-16.

Extracting the Installation Image and Accepting the EULA

After downloading the ctaadminex-darwin-2.1.103.0.tar.gz file from Cisco.com, use this procedure to extract the CTA installation files and accept the end-user license agreement (EULA).

Step 1 Open a terminal window.

Step 2 Using the CD command, change the directory to that which contains the ctaadminex-darwin-2.1.103.0.tar.gz file.

Step 3 At the prompt, type:

tar zxvf ctaadminex-darwin-2.1.103.0.tar.gz

and press <**Return**>. The ctaadminex.sh file is extracted and placed in the same directory as the ctaadminex-darwin-2.1.103.0.tar.gz file.

Step 4 At the prompt, type **./ctaadminex.sh** and press <**Return**>.

Step 5 When prompted, accept the EULA agreement by entering "**y**" and pressing <**Return**>. The cta-darwin-2.1.103.0.dmg is unpacked and placed in the same directory as the ctaadminex.sh file.

Step 6 At the prompt, type **open cta-darwin-2.1.103.0.dmg** and press <**Return**>. The CiscoTrustAgent volume icon is placed on the desktop and the cta-darwin-2.1.103.0.dmg disk image is visible in Finder.

Installing CTA from the Command Line

Step 1 Follow the procedure in the "Extracting the Installation Image and Accepting the EULA" section on page 3-4.

Step 2 Open a terminal window on the end point.

Step 3 At the prompt, use the CD command to change the directory to /Volumes/CiscoTrustAgent.

Step 4 To install CTA, at the prompt, on one line, type the following:

sudo installer -verbose -pkg /Volumes/CiscoTrustAgent/CiscoTrustAgent.mpkg/ -target /Volumes/Macintosh\ /HD

and press <**Return**>.

Step 5 When prompted, enter the Administrative user's password. CTA is installed. At the end of a successful installation you will see the message, "The install was successful."

Step 6 (Optional) To install Cisco Trust Agent Scripting Interface feature, at the prompt, on one line, type the following:

sudo installer -verbose -pkg /Volumes/CiscoTrustAgent/CiscoTrustAgent.mpkg/CiscoTrustAgentSI.pkg/ -target /Volumes/Macintosh\ /HD

and press <**Return**>. At the end of a successful installation you will see the message, "The install was successful."

Step 7 Verify the installation of CTA using the "Verifying Cisco Trust Agent Installation" section on page 3-16.

Step 8 Exit the /Volumes/CiscoTrustAgent directory by typing **CD ..** at the prompt and pressing <**Return**>.

Step 9 At the prompt, type **hdiutil detach /Volumes/CiscoTrustAgent** and press <**Return**>. You will receive messages in the terminal window that the volume was unmounted and ejected. Also the CiscoTrustAgent volume icon will be removed.

Step 10 If a CA certificate or a matching root certificate from the Cisco Secure ACS server has not been installed on the network client on which you just installed CTA, you must install one or the other certificates. This enables CTA to establish a secure form of communication with the Cisco Secure ACS server. Refer to the "Installing or Updating a Certificate on Mac OS X Operating System" section on page 8-4 for more information.

Installing CTA Using an Installation Wizard

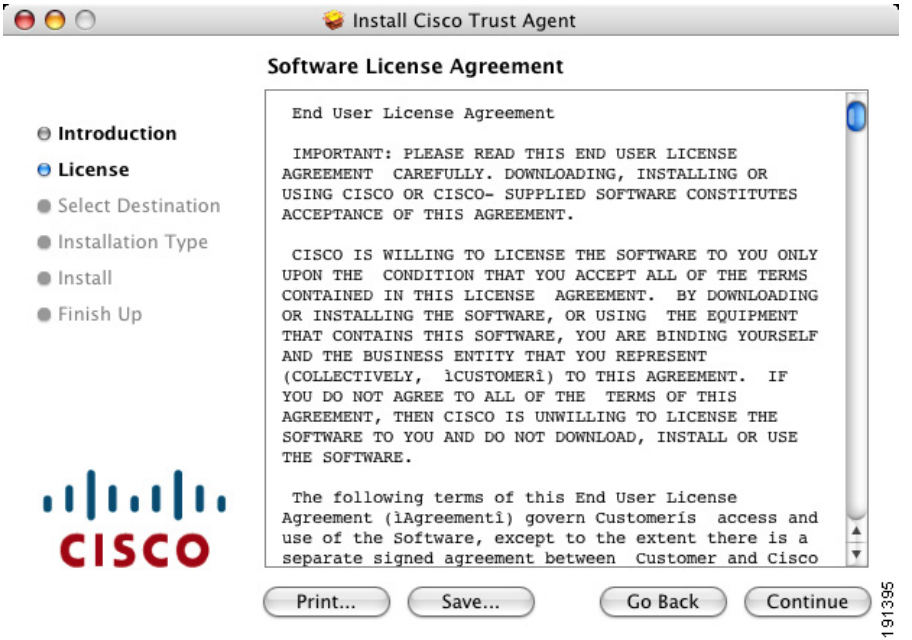
Step 1 Follow the procedure in the "Extracting the Installation Image and Accepting the EULA" section on page 3-4.

Step 2 Double-click the **CiscoTrustAgent** volume icon on the desktop.

Step 3 Double-click the **CiscoTrustAgent.mpkg** icon in the Finder window that opens.

Step 4 Click **Continue** in the Welcome window.

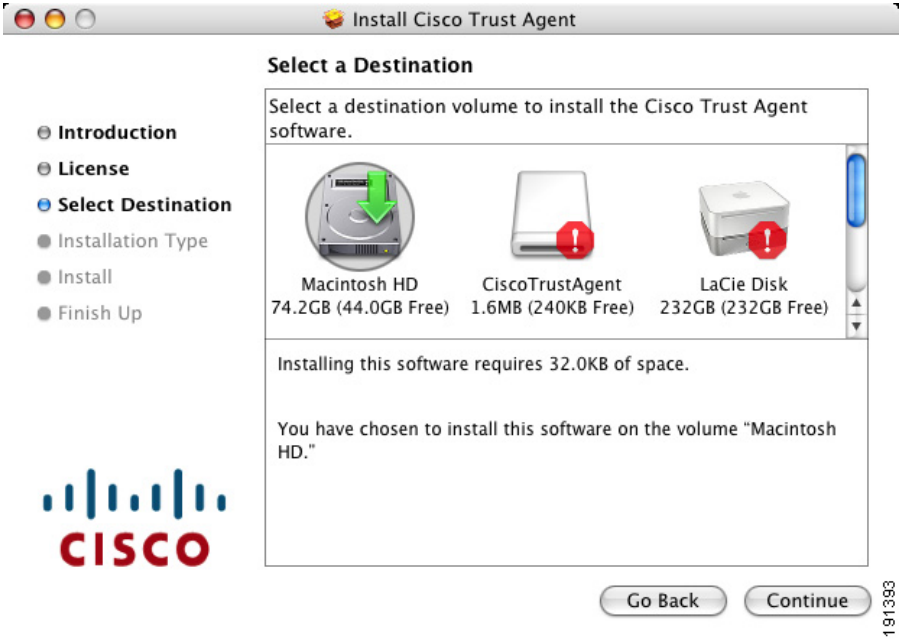
Step 5 Click **Continue** in the Software License Agreement window.



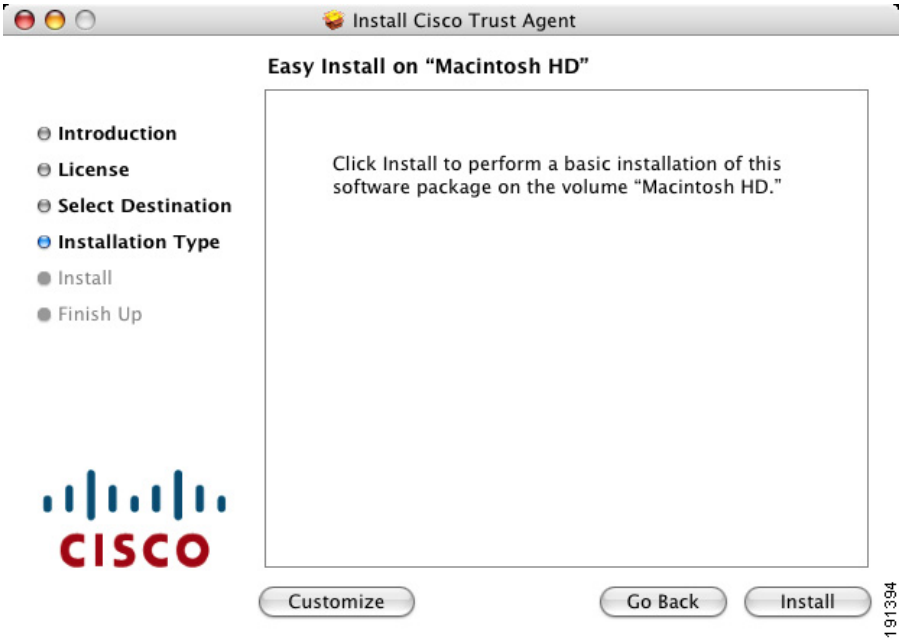
Step 6 Click **Agree** to accept the license for Cisco Trust Agent.

Step 7 **Select the drive** on which you want to install CTA in the Select a Destination window.

Note You may not install CTA on drives or disk images that display the red exclamation mark symbol.

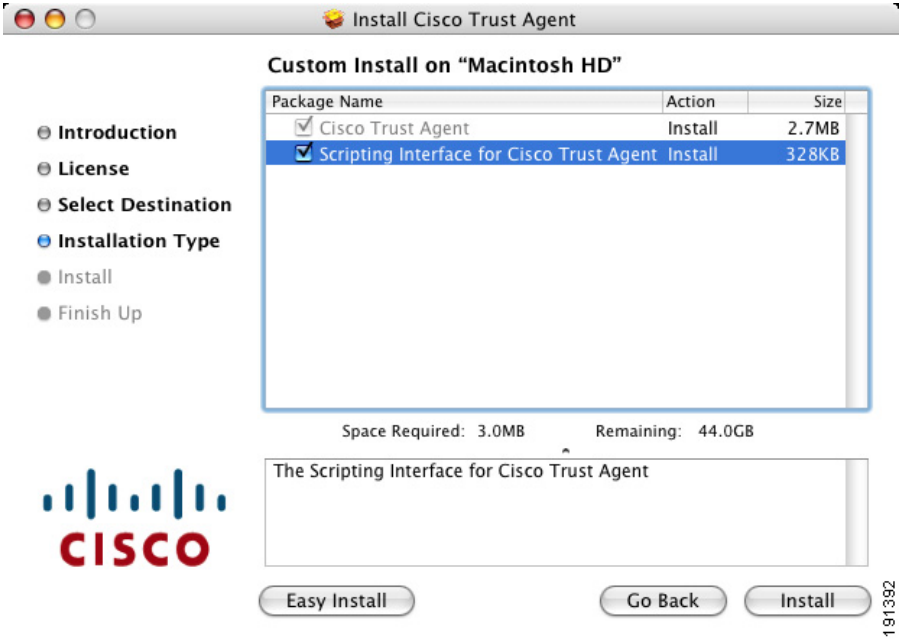


Step 8 Click **Continue**. The Easy Install window opens.



Step 9 Click **Customize** to install the Scripting Interface or skip to Step 11.

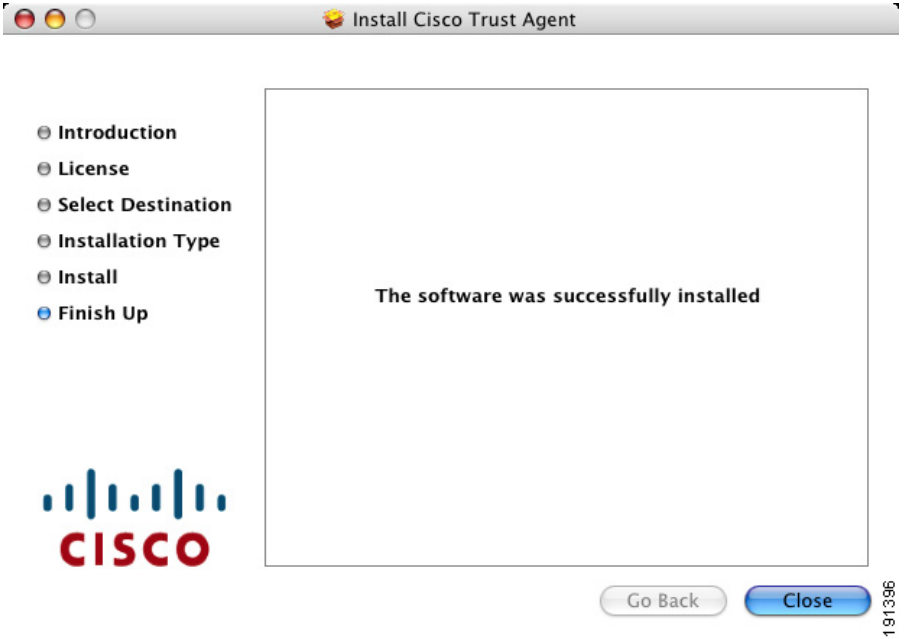
Step 10 In the Custom Install window, check the box next to **Scripting Interface for Cisco Trust Agent** to install that feature.



Step 11 Click **Install** to continue with the installation.

Step 12 Type the Administrator's password when prompted and click **OK**.

Step 13 After CTA has been installed you receive the message, "The software was successfully installed."



Step 14 Click **Close**.

Step 15 Eject the CiscoTrustAgent volume on the desktop by dragging its icon to the trash.

Installing CTA Using a Custom Installation Package

Use this section as an example of how to create a customized CTA installation on Macintosh systems.

The cta-darwin-2.1.103.0.dmg disk image contains the CiscoTrustAgent.mpkg package which you use to install Cisco Trust Agent. The cta-darwin-2.1.103.0.dmg disk image can be modified by adding configuration .ini files, posture plugins, and certificate files to it. This customized Cisco Trust Agent package can be distributed by a software deployment mechanism, such as a script or a software deployment tool.

After the software deployment mechanism distributes the customized cta-darwin-2.1.103.0.dmg disk image to the remote network clients, it runs the CiscoTrustAgent.mpkg package. The CTA installation file copies the contents of the disk image to the proper locations on the remote network client. The software deployment mechanism does not need to recompile the CTA installation file to create a custom installation.

The customization choices in this procedure are optional. However, you will find that including some of these customizations is worthwhile. CTA is not a centrally managed product. If you do not plan to use the product defaults, it is to your benefit to pre-configure all available product settings before deploying CTA.

Please read this entire procedure before beginning. There are options detailed later in the instructions that you should be aware of before beginning.

Creating a Custom Installation Package

Step 1 Before you create the custom installation package, install CTA on the client you will use to create the custom package. This will install the template ctad.ini file and give you exposure to the CTA installation process. Use the "Installing CTA Using an Installation Wizard" section on page 3-6 to install CTA.

Step 2 Use the procedure in "Extracting the Installation Image and Accepting the EULA" section on page 3-4 to extract the cta-darwin-2.1.103.0.dmg disk image and accept the EULA for all users.

Step 3 Double-click the CiscoTrustAgent volume on the desktop.

Step 4 Customize the **/Volumes/CiscoTrustAgent/certs** directory by copying the root certificate for your Cisco Secure ACS server, or other certificates, to this directory. **During installation, any certificates in this directory are added to the systems root certificate store**.

If your Cisco Secure ACS server uses self-signed certificates, see the Cisco Secure ACS documentation for information about obtaining the certificate; if you use a CA server, refer to your CA server documentation.

Note This step is optional if a CA certificate or ACS root certificate have already been distributed to the network clients receiving this customized CTA installation. If these certificates have not been distributed, and you choose not to add them to the customized disk image, you will need to distribute and install either the CA certificate or ACS root certificate before the client can communicate with the NAC infrastructure.

Step 5 Customize the **/Volumes/CiscoTrustAgent/plugins** directory by copying third party plugins that you want to provision at installation time into this directory.

Step 6 Create a new **ctad.ini** file and copy it into the /Volumes/CiscoTrustAgent directory at the same level as the CiscoTrustAgent.mpkg package. This file is used to configure CTA communication settings, user notifications, and certificate validation rules. If you want to change the default communication settings, such as the port number CTA listens over, the maximum number of sessions, or session time-out values, include this file. Refer to Chapter 5, "Configuring Cisco Trust Agent" for instructions on how you should create and format this file.

Step 7 Create a new ctalogd.ini file and copy it to the /Volumes/CiscoTrustAgent directory at the same level as the CiscoTrustAgent.mpkg package. This file is used to enable and disable CTA logging. Refer to Chapter 6, "Cisco Trust Agent Event Logging"for instructions on how you should create and format this file.

Step 8 A software deployment mechanism deploys the custom-cta-darwin-2.1.103.0.dmg disk image to the appropriate network clients and saves it in a local directory.

Step 9 The software deployment mechanism installs CTA and its customizations by following the "Installing CTA from the Command Line" section on page 3-4.

Repairing or Upgrading an Existing CTA Installation

Use this procedure to reinstall CTA or to add the CTA Scripting Interface to an existing CTA installation.

Step 1 Use the procedure in "Extracting the Installation Image and Accepting the EULA" section on page 3-4 to extract the cta-darwin-2.1.103.0.dmg disk image and accept the EULA.

Step 2 Open Finder and navigate to the directory where cta-darwin-2.1.103.0.dmg is stored.

Step 3 Double-click **cta-darwin-2.1.103.0.dmg**. The CiscoTrustAgent volume icon appears on the desktop.

Step 4 Double-click the **CiscoTrustAgent** volume icon.

Step 5 Double-click the **CiscoTrustAgent.mpkg** icon in the Finder window that opens.

Step 6 Click **Continue** in the Welcome window.

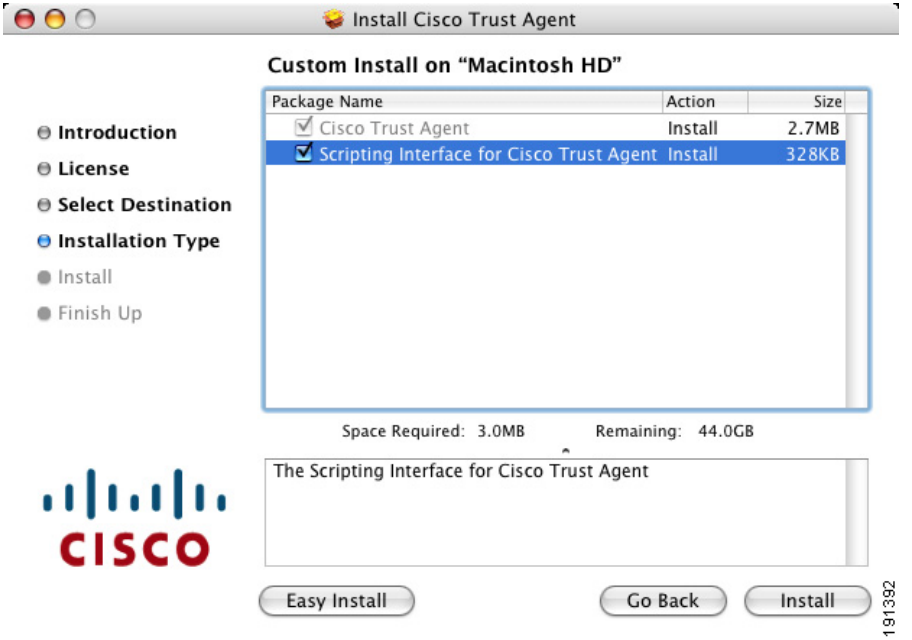
Step 7 Click **Continue** in the Software License Agreement window.

Step 8 Click **Agree** to accept the license for Cisco Trust Agent.

Step 9 **Select the drive** on which you want to install CTA in the Select a Destination window.

Step 10 Click **Continue**. The Easy Install window opens.

Step 11 To install the Scripting Interface feature, click **Customize** or skip to step Step 13.



Step 12 In the Custom Install window, check the box next to Scripting Interface for Cisco Trust Agent to install that feature.

Step 13 Click **Upgrade** to continue with the easy installation of the window.

Step 14 Type the Administrator's password when prompted and click **OK**.

Step 15 After CTA has been installed you receive the message, "The software was successfully installed."

Step 16 Click **Close**.

Step 17 Eject the CiscoTrustAgent volume on the desktop by dragging its icon to the trash.

Verifying Cisco Trust Agent Installation

After Cisco Trust Agent has been installed you will find the following directory structures containing CTA's executable files:

- /opt/CiscoTrustAgent
- /opt/PostureAgent

To verify that the Cisco Trust Agent is running, follow this procedure:

Step 1 Open a terminal window on the system.

Step 2 Type **`ps -ax | grep cta`** and press <**Enter**.>

Step 3 Verify that the following daemons are running:

- /opt/CiscoTrustAgent/sbin/ctad
- ctalogd
- ctapsd
- ctaeoud

If these daemons are not running, try rebooting the system. If the daemons still do not run, try reinstalling the application.

Uninstalling Cisco Trust Agent

There are two uninstallation procedures for CTA. You may uninstall CTA and the Scripting Interface or you may uninstall the Scripting Interface alone.

Uninstalling CTA and the CTA Scripting Interface

Step 1 Open a terminal window on the target system.

Step 2 At the prompt, type **CD /opt/CiscoTrustAgent** and press <**Return**>.

Step 3 At the prompt, run the following command:

```
sudo ./cta_uninstall.sh
```

Step 4 Enter the Administrative user's password when prompted.

Step 5 To uninstall CTA and the CTA Scripting Interface without further prompting, type **y** when prompted.

After Cisco Trust Agent has been successfully uninstalled, you receive the message, "Cisco Trust Agent has been successfully uninstalled."

Note Certificates, plugin files, and customized configuration files are not deleted when CTA is uninstalled; they remain on the client.

Uninstalling Only the CTA Scripting Interface

Step 1 Open a terminal window on the target system.

Step 2 At the prompt, type **CD /opt/CiscoTrustAgent** and press <**Return**>.

Step 3 At the prompt, run the following command:

```
sudo ./cta_uninstall.sh --SI
```

Step 4 Enter the Administrative user's password when prompted.

Step 5 When prompted, type **y** to uninstall Cisco Trust Agent.

Step 6 After a successful uninstallation, you receive the message, "Cisco Trust Agent Scripting Interface has been successfully uninstalled."

CHAPTER 4

Installing the Cisco Trust Agent on Windows Operating Systems

This chapter provides system requirement and installation information for installing Cisco Trust Agent (CTA) on Windows operating systems. Read this entire chapter before beginning the installation. There are advanced installation options detailed later in this chapter that you may want to use. For example, before deploying CTA on your network, you can create a custom installation package which allows you to set CTA configuration parameters and provision certificates, and posture plug-ins. Proceeding in this manner could save you time during the CTA deployment process.

See these other chapters for installation instructions for different operating systems:

- Chapter 2, “Installing the Cisco Trust Agent on Linux Operating Systems.”
- Chapter 3, “Installing the Cisco Trust Agent on Macintosh Operating Systems.”

This chapter contains the following sections:

System Requirements for Installation

Before installing Cisco Trust Agent on a Windows operating system, verify that the target system meets the requirements in Table 4-1.

Note CTA 2.1 does not support Windows NT 4.0 Server or Windows NT 4.0 Workstation. CTA 2.0 was the last release to support Windows NT 4.0.

Table 4-1 CTA System Requirements

System Component	Requirement
System	• Pentium II class processor or better • Network connection
Windows Installer (MSI)	Version 2.0 or later.
Free Hard Disk Space	20 MB minimum
Memory	256 MB of RAM
Listening Port	By default, Cisco Trust Agent listens on UDP port 21862. You can configure this port number if necessary. See the "Configuring EAP over UDP Communication" section on page 5-12.
Recommended Supplicant	Cisco Secure Services Client, version 4.1.2 or later

System Component	Requirement
Windows Operating Systems on which CTA 2.1 runs.	• Windows 2000 Professional and Advanced Server, SP4 and Update Rollup 1 • Windows XP Professional, SP1, SP2, and SP3 • Windows XP Home, SP1, SP2,and SP3 • Windows 2003 Server, SP1 and R2
Windows Operating Systems on which CTA 2.1 and Cisco Secure Services Client run.	• Windows 2000 Professional and Advanced Server, SP4 • Windows XP Professional, SP1, SP2 and SP3 • Windows 2003 Server **Note** See the *Cisco Secure Services Client Administrator Guide* for a complete list of operating systems that support CSSC.
Language Support for localized operating systems	All available localized versions of these operating systems support this release of CTA. **Note** Support for a localized operating system is different from localized version of CTA. The CTA interface and messages are presented in English. • Windows 2000 Professional and Advanced Server, SP4 and Update Rollup 1 • Windows XP Professional, SP1, SP2, and SP3 • Windows XP Home, SP1, SP2, and SP3 • Windows 2003 Server, SP1 and R2

CTA Scripting Interface

The Cisco Trust Agent Scripting Interface feature allows software developers to write their own scripts to relay posture information, collected on the system, to CTA. The scripts can be written to perform the equivalent functions of a posture plugin. End-users will not need this feature unless they intend to write posture validation scripts.

The Scripting Interface is an optional feature of CTA. See "Installing Optional Features During CTA Installation" section on page 4-8 or "Installing CTA Using an Installation Wizard" section on page 4-11 for information on installing this feature along with CTA.

Cisco Secure Services Client

We recommend the use of the Cisco Secure Services Client, version 4.1.2 (CSSC) with CTA 2.1.103.0 for those customers who perform 802.1x authentication. For customers who deployed the CTA 802.1x Wired Client, with the previous offering of CTA 2.1.103.0, we recommend that you migrate to the Cisco Secure Services Client, version 4.1.2.

CSSC sends posture and authentication information, collected by CTA, using the IEEE 802.1x transport protocol to 802.1x-enabled access devices, such as an Ethernet switch, to the Cisco Secure Access Control Server (ACS). After successful client-server authentication, the port access control on the Ethernet switch allows the end-user to connect to the network.

If the NAC deployment in your enterprise uses network routers, or if your network switches communicate with CTA using the EAPoverUDP protocol, you do not need to install CSSC on the same computer as CTA.

Note CSSC is only available for Windows installations. Its default configuration supports only wired network access but it may be upgraded to support wireless network access as well. Learn more about Cisco Secure Services Client at http://www.cisco.com/en/US/products/ps7034/index.html.

Installation File

The CtaAdminEx-win-2.1.103.0.exe installation file is available for download at http://www.cisco.com/cgi-bin/tablebuild.pl/cta by registered users of Cisco.com.

Note These files are not available for download with this release of CTA 2.1.103.0:

- ctasetup-win-2.0.x.y.exe

- ctasetup-supplicant-2.0.x.y.exe
- CtaAdminEx-supplicant-win-2.1.103.0.exe

CtaAdminEx-win-2.1.103.0.exe contains the CTA end-user license agreement (EULA) and the ctasetup-win-2.1.103.0.msi installation file. By running the CtaAdminEx-win-2.1.103.0.exe file, you accept the EULA for all users and extract the ctasetup-win-2.1.103.0.msi installation file. You can use standard MSI commands or an installation wizard to install CTA 2.1.103.0 with or without the Scripting Interface.

If you intend to perform 802.1x authentication, we recommend you use the Cisco Secure Services Client (CSSC) as the supplicant. CSSC is downloaded and installed separately from CTA 2.1.103.0. Registered users of Cisco.com can download CSSC 4.1.2 by clicking this link and following the link to "Download Software" http://www.cisco.com/en/US/products/ps7034/index.html.

Installing Cisco Trust Agent

Cisco Trust Agent installation files are standard Microsoft Windows Installation (MSI) files. Once deployed to the end-point, you can use standard MSI commands to install CTA silently, without user-interaction, or allow users to perform the installation using an installation wizard.

General Installation Instructions

This is the outline of tasks required to install Cisco Trust Agent.

Step 1 Run the CtaAdminex-win-2.1.103.0.exe file and accept the EULA. The ctasetup-win-2.1.103.0.msi file is extracted. See the "Installation File" section on page 4-4 for an explanation of these files.

Step 2 (Optional) Create a custom installation package which could contain ACS root certificate, posture plugins, or a customized CTA configuration file. See the "Installing CTA Using a Custom Installation Package" section on page 4-19 for an explanation of these procedures.

Step 3 Install CTA by distributing the ctasetup-win-2.1.103.0.msi file to end-users alone or as part of a custom installation package. You can use standard MSI commands to specify the features installed with CTA and the level of user interaction.

See the "Installing CTA Using MSI Commands" section on page 4-6 and "Installing CTA Using an Installation Wizard" section on page 4-11 for descriptions of the different installation methods.

Step 4 Install Cisco Secure Access Control Server (ACS) root certificate on the end-point if not distributed as part of a custom installation package. See "About The ACS Server Root Certificate" section on page 8-3 for information about installing this certificate separately.

Step 5 Verify CTA installation.

Installing CTA Using MSI Commands

Standard MSI commands can be passed to the Microsoft Windows Installer through command-line options. These commands determine what features to install as well as the level of user interaction.

This section describes the most common MSI commands.

Note For more information on MSI installation commands see the Microsoft Windows Installer SDK at http://msdn.microsoft.com/library/default.asp?url=/library/en-us/msi/setup/about_windows_installer.asp

- Installation Command
- Uninstallation Command
- Reinstalling or Repairing CTA
- Creating a Log File While Installing CTA
- Installing Optional Features During CTA Installation
- Specifying Installation Directory
- Specifying Reboot Options
- Setting User Interface Mode

Installation Command

To install CTA using MSI command line options, you must know the name and path of the ctasetup-win-2.1.103.0.msi installation file and use the "/I" option with the Msiexec.exe command. The command can be entered from any prompt. See the following example:

Msiexec.exe /I "*C:\Path_To_MSI\ctasetup-win-2.1.103.0.msi*"

This command installs CTA using an installation wizard. Users accept the EULA, choose what features to install, and the installation directory.

Uninstallation Command

To uninstall CTA using MSI command line options, you must know CTA's ProductCode or "GUID." To find the GUID, follow this procedure:

Step 1 Open the Windows Registry Editor.

Step 2 Navigate to **HKEY_LOCAL_MACHINE**\Software\Cisco Systems\Cisco Trust Agent.

The value of the **ProductCode** registry key, including the curly brackets, is the GUID.

To uninstall Cisco Trust Agent, use the /X option with Msiexec.exe command. The command can be entered from any prompt. See the following example:

```
Msiexec.exe /X {GUID}
```

Reinstalling or Repairing CTA

To reinstall or repair CTA from using MSI command line options, run the MSI installation file using the MSI "/F" option. The full command can be run from any prompt. See the following example:

```
Msiexec.exe /fmosu "C:\Path_To_MSI\ctasetup-win-2.1.103.0.msi"
```

The /fmosu argument performs these actions:

f – Reinstalls package

m - Rewrites all required computer-specific registry entries.

o - Reinstalls if file is missing or if an older version is installed.

s - Overwrites all existing shortcuts.

u - Rewrites all required user specific registry entries

Using this command users see messages asking them to wait while CTA is being configured.

Creating a Log File While Installing CTA

To create a log file during installation, run the MSI installation file using the MSI "/L" option. The full command can be entered from any prompt. This logging option requires that the log directory exist and is writable. The log file itself may not exist but the file name must be specified in the command. See the following example:

```
Msiexec.exe /I "C:\Path_To_MSI\ctasetup-win-2.1.103.0.msi" /L*V
"C:\ctalogfile.txt"
```

The `/L*V` option performs these actions:

L – Creates a log file

*V – Specifies verbose logging.

Installing Optional Features During CTA Installation

The ADDLOCAL option allows you to install the Scripting Interface feature along with CTA. This example shows using the ADDLOCAL option to install the Scripting Interface feature:

```
Msiexec.exe /I "C:\Path_To_MSI\ctasetup-win-2.1.103.0.msi"
ADDLOCAL=Scripting_Interface
```

Note The ADDLOCAL command can be used with an interactive installation or a silent installation. (See the "Setting User Interface Mode" section on page 4-10 for information about "silent" and "interactive" installations.)

Specifying Installation Directory

By default, the ctasetup-win-2.1.103.0.msi installation file installs CTA in the "\ProgramFiles\Cisco Systems" directory of the local drive. You can use the "INSTALLDIR" MSI command to specify a different directory. The directory does not have to exist before you issue the command. See the following example:

```
Msiexec.exe /I "C:\Path_To_MSI\ctasetup-win-2.1.103.0.msi"
INSTALLDIR="D:\NewDirectory"
```

This command shows users the installation wizard. During the installation, users will still have an opportunity to change the destination directory.

Specifying Reboot Options

By default the Microsoft Windows Installer determines when a reboot of the system is necessary and automatically prompts the user to reboot at the end of the installation. You can customize this action by using the Microsoft Windows Installer property called "REBOOT." This property forces or suppresses certain system prompts for a reboot. The behavior of the REBOOT option also depends on whether the end-user is following an installation wizard or the installation is being performed silently.

The REBOOT property has three options:

- **Force** - If end-users perform the installation using an installation wizard, they will be prompted to reboot the system after the installation. If the installation is silent, the system reboots automatically without prompting the user.
- **Suppress** - If end-users perform the installation using an installation wizard, they will not be prompted to reboot the system at the end of the installation. If a reboot is required in the middle of an installation, end-users will be prompted to reboot system. If the installation is silent, end-users will not be prompted to reboot at the end of the installation. If a reboot is required in the middle of an installation, the system will be rebooted automatically without prompting the user.
- **ReallySuppress** - All prompts to reboot the system at the end or during an installation, whether the installation is being performed with an installation wizard or is silent, are suppressed.

This is an example of using the REBOOT option with the Force value:

```
Msiexec.exe /I "C:\ctasetup-win-2.1.103.0.msi" REBOOT=Force
```

Setting User Interface Mode

By default, CTA's MSI files provide users with an installation wizard. Using various MSI commands, you can control how much the user is involved in CTA's installation.

For a full description of how the installation wizard works, see the "Installing CTA Using an Installation Wizard" section on page 4-11.

The command options in the following table specify the amount of end-user interaction with the CTA installation:

Table 4-2 User Interface MSI Command Line Options

Command Option	Description
/q, /qn	There is no user interaction. This provides a silent installation. Example: `Msiexec.exe /I "C:\ctasetup-win-2.1.103.0.msi" /q`
/qb	Users see messages alerting them that CTA is being configured, however, users are not prompted perform any action. Example: `Msiexec.exe /I "C:\ctasetup-win-2.1.103.0.msi" /qb`
/qr	Users see some of the installation wizard windows including a progress bar showing installation, however, users are not prompted perform any action. Example: `Msiexec.exe /I "C:\ctasetup-win-2.1.103.0.msi" /qr`
/qf	Users are fully involved in the installation of CTA. They install CTA using the installation wizard. Example: `Msiexec.exe /I "C:\ctasetup-win-2.1.103.0.msi" /qf`

Command Option	Description
/qn+	Users receive a pop-up message at the end of the installation specifying the success or failure of the installation. Example: `Msiexec.exe /I "C:\ctasetup-win-2.1.103.0.msi" /qn+`
/qb+	Users see messages alerting them that CTA is being configured, however, users are not prompted perform any action during the installation. At the end of the installation users receive a pop-up message that specifies the success or failure of the installation. Example: `Msiexec.exe /I "C:\ctasetup-win-2.1.103.0.msi" /qb+`

Tip When combining MSI options, specify the user interface command at the end of the entire command. For example, the following command installs CTA with the Scripting Interface, logging is turned on, and users experience basic user interaction with a final pop-up message.

```
Msiexec.exe /I "C:\ctasetup-win-2.1.103.0.msi"
ADDLOCAL=Scripting_Interface /L*V "C:\logfile.txt" /qb+
```

Installing CTA Using an Installation Wizard

This section describes installing CTA and the Scripting Interface by following an installation wizard. You must have administrator privileges on the client to install CTA.

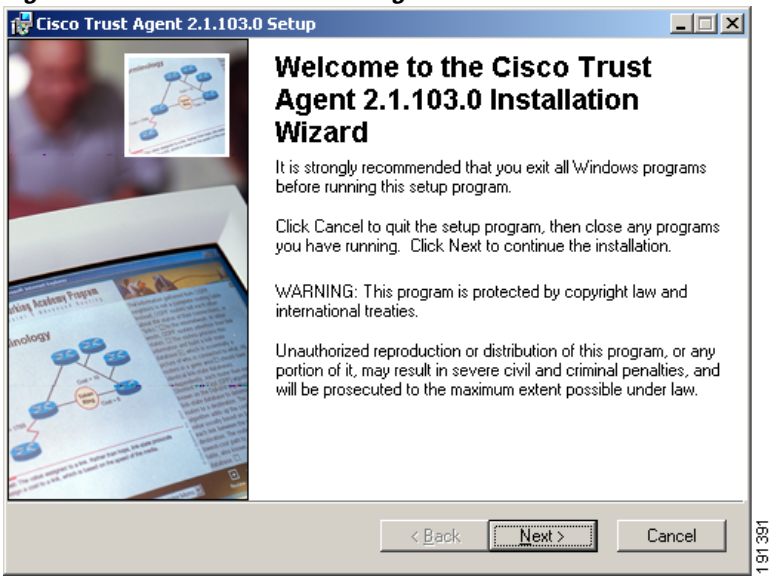
Note If the group policy for the target system allows for elevated privileges for the MSI, then users with Standard or Restricted privileges can install CTA. To use the elevated privileges, MSI 2.0 must be installed before you begin the CTA installation. You cannot use a custom installation package to install the MSI unless you have administrator privileges.

Step 1 Read the "General Installation Instructions" section on page 4-5.

Step 2 Exit all Windows programs.

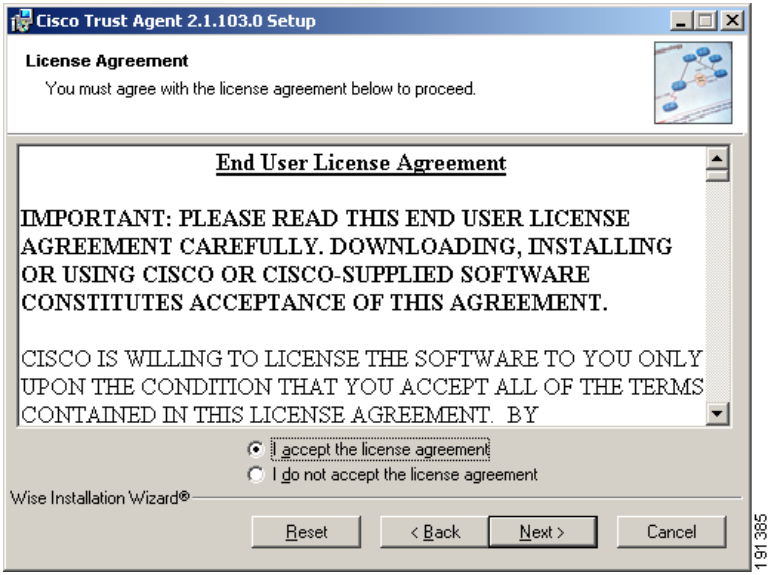
Step 3 Launch the appropriate ctasetup-win-2.1.103.0.msi file by issuing the proper MSI command line option or by double-clicking the file. The Cisco Trust Agent **Installation Wizard** opens as shown in Figure 4-1.

Figure 4-1 The Cisco Trust Agent Installation Wizard



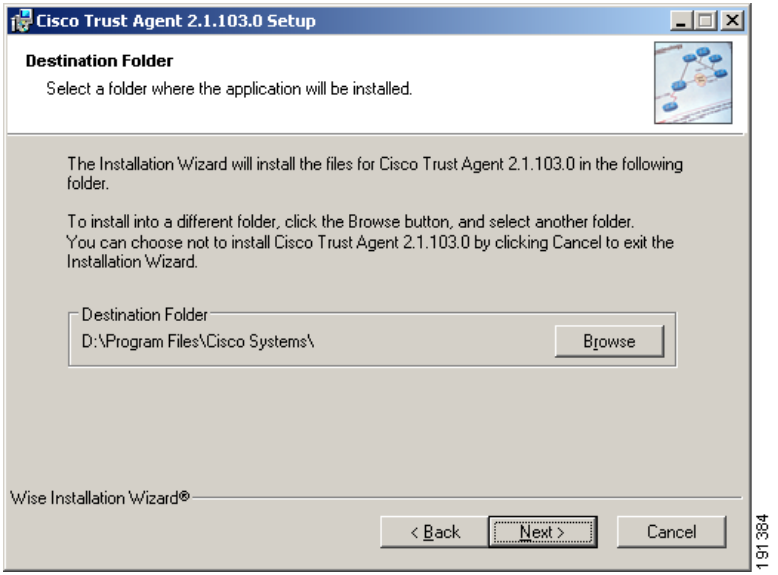
Step 4 Click **Next.** The **License Agreement** window opens as shown in Figure 4-2.

Figure 4-2 The License Agreement on Windows



Step 5 Accept the license agreement by selecting the **I accept the license agreement** radio button and by clicking **Next**. The Destination Folder window opens as shown in Figure 4-3.

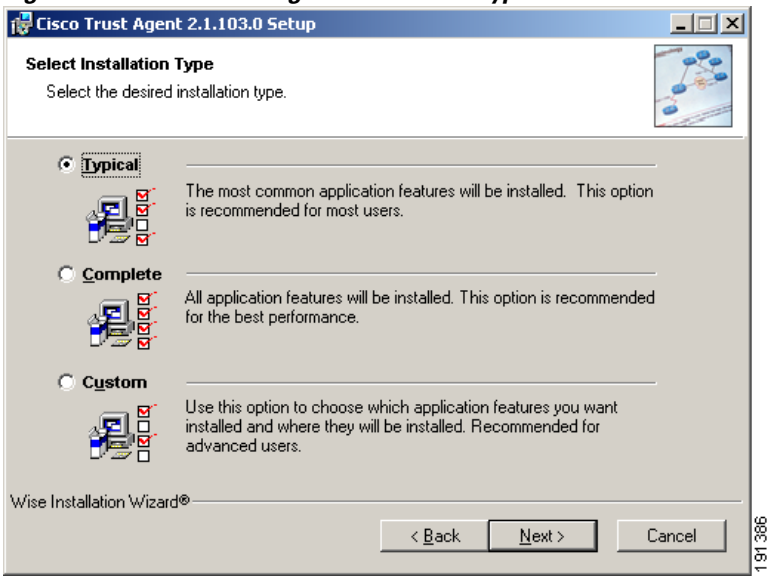
Figure 4-3 The Destination Window



Step 6 Click **Next** to accept the default destination folder or click **Browse** and navigate to the desired drive and folder and then click **OK**. The new install location appears in the **Destination Folder** pane. Click **Next** to install CTA in the folder you specified.

Step 7 The **Select Installation Type** dialog box opens (Figure 4-4).

Figure 4-4 Selecting an Installation Type

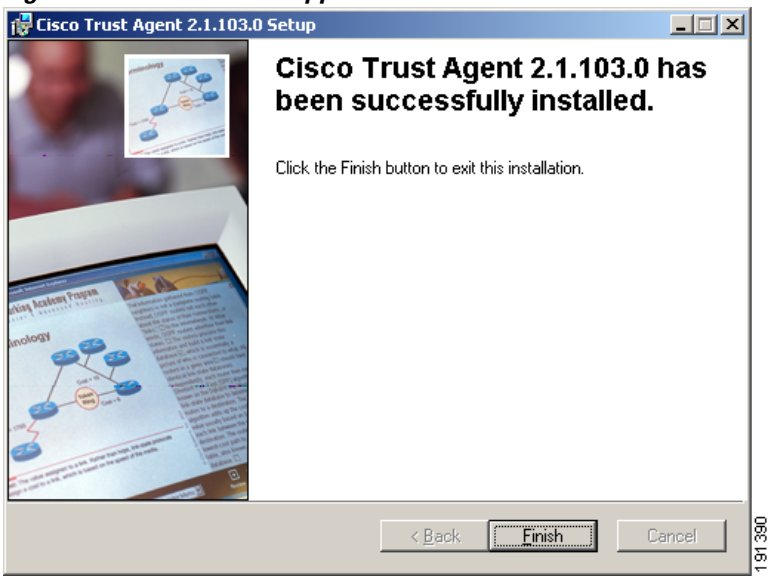


Selecting **Typical** will install CTA. The Scripting Interface is an optional feature and it is not installed during a **Typical** installation.

Selecting **Complete** installation will install the Scripting Interface feature along with CTA.

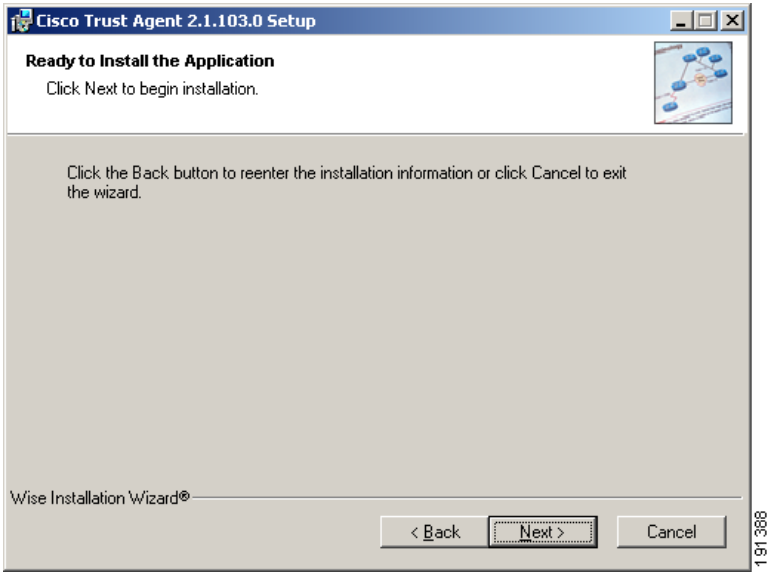
Selecting **Custom** installation will allow you to include or exclude any features available with the installation file. Figure 4-5 shows how you can select the Scripting Interface feature during a **Custom** installation.

Figure 4-5 Choose Application Features



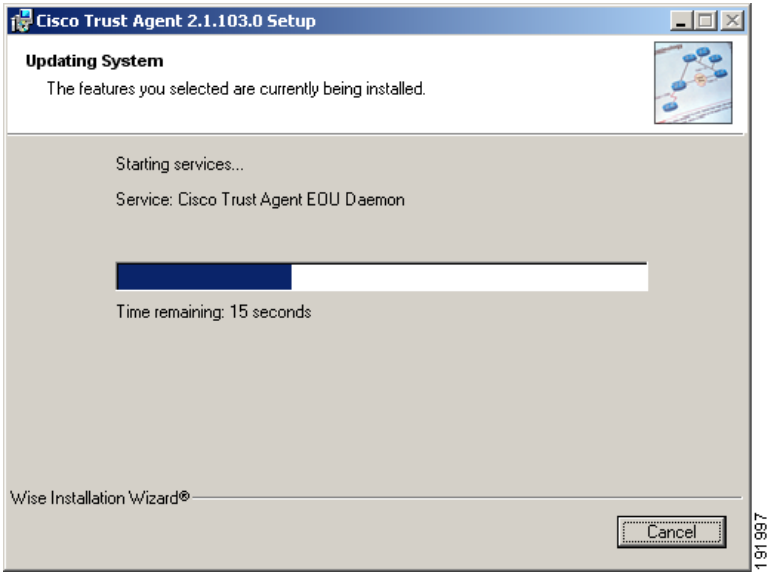
After choosing an installation type and selecting a CTA feature, click **Next**. The **Installing the Application** window opens as shown in Figure 4-6.

Figure 4-6 Installing the Application



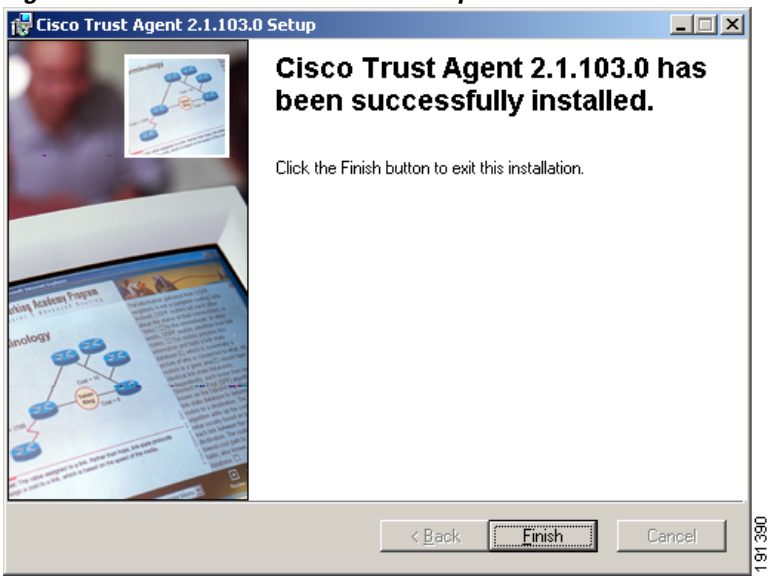
Step 8 Click **Next.** The application installs in the selected directory. Figure 4-7 illustrates the window that shows the progress of the installation.

Figure 4-7 The System is Being Updated



When the installation has completed, the installer displays the **Installation Completed** window as shown in Figure 4-8.

Figure 4-8 The Installation is Complete



Step 9 Click **Finish**. You do not need to restart your machine after installation.

Note For CTA to establish a secure form of communication with the Cisco Secure ACS server, you must have either a CA certificate or a matching root certificate from the Cisco Secure ACS server installed on the system. Refer to "About The ACS Server Root Certificate" section on page 8-3 for information.

Installing CTA Using a Custom Installation Package

Use this section as an example of how to create a customized CTA installation.

The CTA install application is a single executable file. To create a custom installation package, you create a directory structure which includes the desired CTA installation file, .ini files, plugin subdirectories and certificate subdirectories. This directory structure can then be distributed by a software deployment mechanism, such as a script or a software deployment tool.

After the software deployment mechanism distributes the directory structure to the remote network clients, it runs the CTA installation file with the desired MSI command line options. The CTA installation file copies the contents of the directory structure to the proper locations on the remote network client. The software deployment mechanism does not recompile the CTA installation file to create a custom installation.

Please read this entire procedure before beginning. There are options detailed later in the instructions that you should be aware of before beginning.

These are the procedures you need to follow to create a custom installation package:

Step 1 Install CTA on the Administrator's Client

Step 2 Create the Custom Installation Directory Structure

Step 3 Customize the Installation Directory

Step 4 Run the CTA Installation File to Install the Custom Package

Install CTA on the Administrator's Client

Before you create the custom installation package, install CTA on the client you will use to create the custom installation package. This will install the template ctad.ini file and familiarize you with CTA's installation process. Use the "Installing CTA Using an Installation Wizard" section on page 4-11 to install CTA.

Create the Custom Installation Directory Structure

Step 1 Create an empty directory on your network client. For example, D:\CTA\Custom_Package

Step 2 Open a command prompt.

Step 3 CD to the directory which contains the **CtaAdminEx-win-2.1.103.0.exe** file. (See the "Installation File" section on page 4-4 for a description of this file.)

Step 4 After the prompt, on one line, type the name of the CtaAdminEx-win-2.1.103.0.exe file followed by a -p switch and the path to the directory you created in Step 1 of this procedure. This will extract the ctasetup-win-2.1.103.0.msi file to the new directory. For example:

```
D:\CTA>CtaAdminEx-win-2.1.103.0.exe -p D:\CTA\Custom_Package
```

Note If you do not use the -p switch in the command, the ctasetup-win-2.1.103.0.msi is extracted to the same directory that contains the CtaAdminEx-win-2.1.103.0.exe file.

Note If you want to create a custom installation package for a wizard-driven installation, copy the ctasetup-win-2.1.103.0.msi to the Custom_Package directory.

Step 5 When prompted, read and accept the End User License Agreement (EULA) on behalf of all users by typing **Y** and pressing **<Enter>**. After the CTA installation file has been extracted, a message is returned showing the path to which the installation file was extracted. In our example, you should now have a D:\CTA\Custom_Package directory with one file in it: ctasetup-win-2.1.103.0.msi

Step 6 Proceed to "Customize the Installation Directory."

Customize the Installation Directory

The customization choices in this procedure are optional. However, you will find that including some of these customizations is worthwhile. CTA is not a centrally managed product. If you do not plan to use the product defaults, it is to your benefit to pre-configure all available product settings before deploying CTA.

Step 1 Create a **certs** subdirectory. For example: D:\CTA\Custom_Package**certs**

Copy the root certificate for your Cisco Secure ACS server to this directory. During installation, any certificates in this directory are added to the systems root certificate store.

If your Cisco Secure ACS server uses self-signed certificates, see the Cisco Secure ACS documentation for information about obtaining the certificate; if you use a CA server, refer to your CA server documentation.

Note This step is not required if a CA certificate or ACS root certificate have already been distributed to the network clients receiving this customized CTA installation. If these certificates have not been distributed, this step is required.

Step 2 Create a **plugins** subdirectory. For example: D:\CTA\Custom_Package**plugins**

Copy any third-party plugins that you want to provision at installation time into this directory.

Step 3 Create a new **ctad.ini** file and store it in the D:\CTA\Custom_Package directory. This file is used to configure CTA communication settings, user notifications, and certificate validation rules. Refer to Chapter 5, "Configuring Cisco Trust Agent" for instructions on how you should create and format this file.

Step 4 Create a new ctalogd.ini file and store it in the D:\CTA\Custom_Package directory. This file is used to enable and disable CTA logging. Refer to Chapter 6, "Cisco Trust Agent Event Logging"for instructions on how you should create and format this file.

Step 5 Proceed to "Run the CTA Installation File to Install the Custom Package."

Run the CTA Installation File to Install the Custom Package

For the sake of this procedure, we assume that the custom package is deployed to the appropriate network clients by a software deployment mechanism such as a software deployment tool or script.

Note The custom package consists of the customized installation directories and the CTA installation .msi file. The installation .msi does not recompile the customized files into a new installation .msi file, it installs CTA and the customized files you created in the "Customize the Installation Directory" section on page 4-21.

Step 1 The software deployment mechanism deploys the custom package to the appropriate network clients and saves it to a local directory. For the sake of this example, we assume that the custom installation package is saved to the C:\Temp directory. There would now be a C:\Temp\Custom_Package directory on the client.

Step 2 The software deployment mechanism can then run the CTA installation file and employ whatever MSI command line options you choose. (See the "Installing CTA Using MSI Commands" section on page 4-6 for a summary of common MSI commands and examples of how they are used with the CTA installation files.)

Here are two different examples of the CTA installation:

Run the CTA installation file as it is, this installs CTA and your customizations. To do so, the software deployment mechanism would run the following commands:

a. CD to the C:\Temp\Custom_Package directory.

b. From the prompt, run the **ctasetup-win-2.1.103.0.msi** file. For example:

C:\Temp\Custom_Package>**ctasetup-win-2.1.103.0.msi**

Run the CTA installation file with MSI command line options. The command in the procedure installs CTA with your customizations, the Scripting Interface, it logs the installation, and stores the log file in "C:\Custom_Package\logfile.txt, and it is a silent installation.

a. CD to the C:\Temp\Custom_Package directory.

b. From the prompt, run the **ctasetup-win-2.1.103.0.msi** file with the MSI command line options. For example:

```
C:\Temp\Custom_Package>Msiexec.exe /I ctasetup-win-2.1.103.0.msi
ADDLOCAL=Scripting_Interface /L*V "C:\Custom_Package\logfile.txt" /qn
```

Upgrading to Cisco Trust Agent, Release 2.1

Cisco Trust Agent supports upgrade installations from versions 1.0, 2.0, 2.0.1, and Selective Availability and Beta 2.1 releases to CTA 2.1.103.0.

The behavior of an upgrade reflects the kind of installation being used. If the upgrade is performed using an installation wizard, CTA 2.1.103.0 recognizes the previous installation of CTA and prompts users to upgrade. In the case of a silent installation, it is assumed that the user intends to perform an upgrade and the installation proceeds without prompting the user.

Upgrade procedures are the same as the installation procedures described in "Installing CTA Using MSI Commands" section on page 4-6, "Installing CTA Using an Installation Wizard" section on page 4-11, and "Installing CTA Using a Custom Installation Package" section on page 4-19.

Upgrading from Cisco Trust Agent, Release 1.0

During an upgrade installation of CTA from 1.0 to CTA 2.1, existing certificates remain in the certificate store in which they were installed during the CTA 1.0 installation. Posture plugins and the ctalogd.ini file are moved to their new location in the CTA 2.1.103.0 directory structure. The ctad.ini file used in CTA 1.0 remains in the directory in which it was originally installed and CTA 2.1.103.0 recognizes the file in its original location.

Upgrading from Cisco Trust Agent, Release 2.0.0.30

During an upgrade installation of CTA from 2.0.0.30 to CTA 2.1.103.0, certificates, third-party posture plugins, ctad.ini, ctalogd.ini, and log files remain in the directories in which they were installed and they are used by CTA 2.1.103.0.

Upgrading from Cisco Trust Agent, Release 2.0.1

Cisco Trust Agent 2.0.1 was a release supported on Windows XP platforms only. During an upgrade installation of CTA from 2.0.1 to CTA 2.1, certificates, third-party posture plugins, ctad.ini, ctalogd.ini, and log files remain in the directories in which they were installed and they are used by CTA 2.1.103.0.

Upgrading from CTA 2.1 Selective Availability and Beta Releases to CTA 2.1.103.0

Some customers of Cisco's Network Admission Control program participated in testing "selective availability" or "limited availability" releases and Beta releases of CTA 2.1 to test its functionality in their NAC environments.

These builds, numbered 2.1.18.0, 2.1.100.0, 2.1.101.0, and 2.1.102.0 may be upgraded to CTA 2.1.103.0. The certificates, third-party posture plugins, ctad.ini, ctalogd.ini, and log files remain in the directories in which they were installed and they are used by CTA 2.1.103.0.

Upgrading Cisco Trust Agent with the CTA 802.1x Wired Client to Cisco Secure Services Client

The *Migrating from CTA 802.1x Wired Client to Cisco Secure Servics Client* document describes how to upgrade 802.1x supplicant used with CTA from the CTA 802.1x Wired Client to Cisco Secure Services Client.

Verifying the Cisco Trust Agent Installation

After Cisco Trust Agent has been installed you will find the following directory structures containing CTA's executable files:

- \Program Files\Cisco Systems\CiscoTrustAgent
- \Program Files\Common Files\PostureAgent

After installing CTA, to verify that CTA is running, follow this procedure:

Step 1 Open a command prompt window on the target system.

Step 2 Type **net start** and then click **Enter**.

Step 3 Verify that the following services are running:

Current Service Names:

- Cisco Posture Server Daemon
- Cisco Systems Inc. CTA Posture State Daemon
- Cisco Trust Agent EoU Daemon
- Cisco Trust Agent Logger Daemon

If these services are not running, reboot the computer and check again. If the services still do not run, reinstall the application.

Uninstalling Cisco Trust Agent on Windows

To uninstall Cisco Trust agent, follow these steps:

Step 1 Choose **Start > Settings > Control Panel > Add/Remove Programs**.

Step 2 Choose **Cisco Trust Agent** from the list of installed applications.

Step 3 Click **Remove**.

A confirmation dialog box appears.

Step 4 Click **Yes** to continue the removal.

Note Certificates and plugin files are not deleted when CTA is uninstalled; they remain on the client.

CHAPTER 5

Configuring Cisco Trust Agent

Cisco Trust Agent (CTA) may be configured by making changes to its ctad.ini file. The tasks in this chapter describe configuring CTA's communication of posture data to the Cisco Secure Access Control Server as well as the communication of that posture to the client. To configure CTA Logging, see Chapter 6, "Cisco Trust Agent Event Logging".

This chapter contains the following sections:

The ctad.ini Configuration File

The ctad.ini configuration file is a plain text file that contains the parameters for the Cisco Trust Agent's communication settings, user notifications, certificate filtering rules, and the scripting interface feature.

Some parameters are shared by Linux, Mac OS X, and Windows environments while other are used only for a particular operating system. See the "Sample Windows ctad.ini File" section on page 5-28, "Sample Linux ctad.ini File" section on page 5-32, and "Sample Mac OS X ctad.ini File" section on page 5-37 for examples of ctad.ini files.

The template ctad.ini file is named ctad-temp.ini and is installed during the Linux, Mac OS X, and Windows installations. The ctad-temp.ini file for Linux and Mac OS X are stored in the /etc/opt/CiscoTrustAgent/ directory. The ctad-temp.ini file for Windows is stored in the \Program Files\Cisco Systems\CiscoTrustAgent\ directory.

Editing the ctad.ini Configuration File

In order to edit the ctad.ini file you must have administrative privileges.

Step 1 Locate the ctad.ini file or the ctad-temp.ini template file on the host.

- For Linux and Mac OS X operating systems, navigate to the /etc/opt/CiscoTrustAgent directory.
- For Windows Operating systems, navigate to the Program Files\CiscoSystems\CiscoTrustAgent directory.

Step 2 If there is a ctad.ini file in the directory, you can edit that file directly. If there is **only** a ctad-temp.ini file, make a copy of it and save the copy as "ctad.ini".

Step 3 Open the ctad.ini file in a plain text editor.

Step 4 Locate the section you want to edit.

Step 5 Delete the semicolon (;) in front of the parameter you want to edit.

Step 6 Adhering to the value ranges defined in the ctad.ini file, change the value of the parameter.

Step 7 Save the ctad.ini file and close the file.

Step 8 Activate changes in the CTA environment:

- If your changes were to the [main], [UserNotifies] or [ServerCertDNVerification] sections of the ctad.ini file, the new values will be re-read when they are needed.
- If your changes were to the [EAPoUDP] section, reboot the system on which CTA runs.

ctad.ini Configuration Parameters

Table 5-1 explains the sections and parameters used in the ctad.ini file by Linux, Mac OS X, and Windows operating systems.

Table 5-1 ctad.ini Configuration Parameters

Keyword	Description	Operating System(s)
[main]	Heading for main section of ctad.ini file.	Linux, Mac OS X, Windows
EnableVFT	The “EnableVFT” parameter indicates if Validation-Flag TLV is enabled on the version of IOS running on the Network Access Device (NAD). “EnableVFT” enables CTA to operate with the NAD wether it has support for Validation-Flag TLV or not. Default Value: 0 Range of Values: 0, 1 0 = IOS does not support Validation-Flag TLV 1 = IOS does support Validation-Flag TLV	Linux, Mac OS X, Windows

Table 5-1 ctad.ini Configuration Parameters (continued)

Keyword	Description	Operating System(s)
PPInterfaceType	The PPInterfaceType parameter describes how CTA gathers posture plugin information. Default Value: Block Range of Values: Block, NonBlockConcurrent, NonBlockSerial • Block: CTA will request posture information from one plug-in at a time. It will not request information from the next plugin until it has received the posture credentials from the current plug-in. • NonBlockConcurrent: CTA requests posture information from all posture plugins simultaneously. The operation of gathering posture credentials ends when all the posture information is returned or the PPWaitTimeout value is reached, whichever is sooner. • NonBlockSerial: CTA requests posture information from one plug-in at a time. It will not request information from the next plugin until it has received the posture credentials from the current plug-in. The operation of gathering posture credentials ends when all the posture information is returned or the PPWaitTimeout value is reached, whichever is sooner.	Linux, Mac OS X, Windows
PPWaitTimeout	The PPWaitTimeout is only applicable if the PPInterfaceType is set to NonBlockConcurrent or NonBlockSerial. The parameter defines the maximum time allowed, in seconds, to complete the processing of all plugin queries. Should this timer expire while waiting for responses, CTA will send all of the data that it received thus far in the exchange. Default value: 5 seconds Range of values: 1 - 300 seconds	Linux, Mac OS X, Windows

Table 5-1 ctad.ini Configuration Parameters (continued)

Keyword	Description	Operating System(s)
PPMsgSize	The PPMsgSize parameter allows the administrator to modify the maximum message size that a posture plugin can send from 1k to a maximum value of 6k. Default value: 1024 bytes Range of values: 1024 – 6144 bytes	Linux, Mac OS X, Windows
*PluginName*_PPMsgSize	An application-specific posture plugin message size may be added to the ctad.ini file. When added, the posture plugin this parameter references will provide a posture message of this size and it will ignore the value provided by PPMsgSize. See Configuring an Application-Specific Posture Plug-in Message Size, page 5-17. Range of values: 1024 – 6144 bytes	Linux, Mac OS X, Windows
SQTimer	The status query timer (SQTimer) parameter defines how often CTA queries the posture plugins to detect a change in their status. Default value: 300 seconds Range of values: 5 - 4294967295 seconds	Linux, Mac OS X, Windows
[EAPoUDP]	Section head for Cisco Trust Agent using EAP over UDP protocol to communicate with Cisco Secure ACS.	Linux, Mac OS X Windows
LocalPort	The LocalPort parameter defines the port on which the NAD initiates posture validation with CTA. Changing this value requires changes to the NAD configuration. Default value: 21862 Rang of values: 1 - 65550	Linux, Mac OS X, Windows

Table 5-1 ctad.ini Configuration Parameters (continued)

Keyword	Description	Operating System(s)
MaxSession	CTA supports one established session per NAD. But CTA can support concurrent sessions with multiple NADs. MaxSession is the number of sessions you allow CTA to support. Default value: 8 Range of values: 1 - 256	Linux, Mac OS X, Windows
SessionIdleTimeout	The SessionIdleTimeout parameter defines the number of seconds an EAP over UDP session can remain idle before timing out. Default value: 3600 Range of values: 60 - 172800	Linux, Mac OS X, Windows
BootTimeUDPExemptions	The BootTimeUDPExemptions parameter alters the Windows Firewall policy and enable the CTA to receive packets when the Windows XP SP2 and SP3-based computer is starting. Default value: 1 Range of values: 0, 1 0 = Windows Firewall Boot Time Exemptions are disabled. 1 = Windows Firewall Boot Time Exemptions are enabled. **Note** Use of the BootTimeUDPexemptions parameter is relevant only when used in conjunction with Microsoft's hot fix for Windows XP (KB17730)	Windows
[UserNotifies]	The [UserNotifes] section defines the behavior of pop-up windows containing messages sent from ACS to CTA.	Linux, Mac OS X, Windows

Table 5-1 ctad.ini Configuration Parameters (continued)

Keyword	Description	Operating System(s)
SysModal	SysModal specifies whether the user notification dialog boxes are modal or not. If the notification dialog boxes are modal, the user must close the notification dialog box to continue working. Also, if the dialog boxes are modal, and there is more than one notification, only the last notification appears. Default value: 1 Range of Windows values: 0, 1 0 = Modal dialog boxes are disabled. 1 = Modal dialog boxes are enabled.	Windows
UserActionDelayTimeout	If the browser that displays the posture message is launched before the host obtains an IP address, the browser will fail to open the URL contained in the posture message. Setting this parameter allows you to delay the launch of the browser window so that the host has more time to obtain an IP address. Default value: 25 seconds Range of values: 0 - 4294967295 seconds	Linux, Mac OS X, Windows
EnableNotifies	The EnableNotifies parameter enables or disables user notifications. If the EnableNotifies parameter is enabled, a clickable URL may also be presented in the pop-up browser window that contains the posture result. The end user can click the URL link to go to a browser page that contains additional information provided by the ACS administrator. This parameter applies to logged-in users. Default value: 1 Range of values 0, 1 0 = User notifications are disabled. 1 = User notifications are enabled.	Linux, Mac OS X, Windows

Table 5-1 ctad.ini Configuration Parameters (continued)

Keyword	Description	Operating System(s)
MsgTimeout	The MsgTimeout parameter specifies how long, in seconds, user notification dialog boxes are displayed. Default value: 300 Range of values: 30 - 4294967 Special value: 0 (disables the timeout)	Linux, Mac OS X, Windows
EnableLogonNotifies	Enables or disables user notification received before the user is logged on. Default value: 1 Range of values: 0, 1 0 = User notifications received before the user is logged on are discarded. 1 = User notifications received before the user is logged on are saved and displayed to the user when they log on.	Linux, Mac OS X, Windows
LogonMsgTimeout	Specifies how long, in seconds, a message is saved when no user is logged on and when EnableLogonNotifies enabled. Default value: 86400 Range of values: 30 - 4294967 Special value: 0 (disables the timeout)	Linux, Mac OS X, Windows
DisplayType	The DisplayType parameter determines if messages sent from ACS to CTA are displayed in a graphic user interface or in a terminal window. Default value: gui Range of Linux values: term, gui Range of Mac OS X values: gui	Linux

Table 5-1 ctad.ini Configuration Parameters (continued)

Keyword	Description	Operating System(s)
TermFont	The TermFont parameter sets the font in which to display the terminal screen text. Use the default value. The TermFont value for Asian Languages will be implemented in a future release. Default value: -misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso10636-1 TermFont entry below for Asian languages: TermFont=-misc-zysong18030-medium-r-normal--0-0-0-0-c-0-iso10646-1	Linux
ClearOldNotification	The ClearOldNotification parameter clears or saves notification messages. Default value: 1 Range of values: 0,1 0 = Notification messages are saved. 1 = CTA clears the old notification message before displaying the new window.	Linux, Mac OS X
BrowserPath	The BrowserPath parameter specifies the full path of the browser on Linux systems. • For Red Hat Enterprise Linux v3 (Enterprise, Advanced, Workstation) use this path: **/usr/bin/mozilla** • For Red Hat Enterprise Linux v4 (Enterprise, Advanced, Workstation) use this path: **/usr/bin/firefox**	Linux

Table 5-1 ctad.ini Configuration Parameters (continued)

Keyword	Description	Operating System(s)
[ServerCertDNVerification]	The [ServerCertDNVerification] section contains configurable parameters for distinguished name (DN) matching. When using CA certificates to validate your Cisco Secure ACS server certificate, you can implement additional security using distinguished name matching. See Configuring Certificate DN Matching, page 5-27 for more information on these parameters.	Linux, Mac OS X, Windows
TotalRules	The TotalRules parameter defines the number of DN matching rules that follow it. If the number of rules is greater than 1 the rules are connected by an OR statement. Default value: (none) Range of values: 0 - 64 Special value: 0 (Disables DN matching)	Linux, Mac OS X, Windows
RuleX	The RuleX parameters are DN matching rules, where X is the index for the rule. These are examples: Rule1=CN*”Server”, ISSUER-CN*”Finance” Rule2=CN=”Finance posture Cert”, OU*”Finance”, ISSUER-CN*”ACME”	Linux, Mac OS X, Windows
[Scripting_Interface]	The parameters in the [Scripting_Interface] section define the scripting interface	Linux, Mac OS X, Windows
delta_stale	The delta_stale parameter specifies how long, in minutes, before the posture database record is deemed outdated. Default value: 43200 Range of values: 1-5256000 Special value: 0 (the database will never expire)	Linux, Mac OS X, Windows

Configuring EAP over UDP Communication

CTA can communicate with a router or switch using the Extensible Authentication Protocol over User Datagram Protocol (EAP over UDP). When CTA uses EAP over UDP to communicate with a router, this is referred to as the NAC L3 IP method. When CTA uses EAP over UDP to communicate with a switch, this is referred to as the NAC L2 IP method. In these cases, you can configure the NAC L3 IP and NAC L2 IP communication in the [EAPoUDP] section of the ctad.ini configuration file.

These configurations are optional. You are not required to change the default values of these parameters in order for CTA to run properly after installation.

Note On Windows systems, CTA can also communicate with a switch, through a supplicant such as the Cisco Secure Services Client using the IEEE 802.1x protocol. That communication is not configurable in the ctad.ini file.

To configure EAP over UDP communication, use the Editing the ctad.ini Configuration File, page 5-3 procedure, reference the sample ctad.ini files, and reference the ctad.ini Configuration Parameters, page 5-4.

You can configure the following communication settings for CTA:

- LocalPort — By default, CTA listens on UDP port 21862. If you change this setting, you need to configure your network access device to initiate the posture validation process on the new port number.
- MaxSession — CTA only supports one established session per network access device, but can support sessions from multiple, different network access devices at the same time. CTA can support up to 255 sessions at one time.
- SessionIdleTimeout — This specifies the number of seconds an EAPoUDP session can remain idle before timing out.
- BootTimeUDPExemptions — The BootTimeUDPExemptions parameter alters the Windows Firewall policy and enables CTA to receive packets when the Windows XP SP2 or SP3-based computer is starting.

  By enabling BootTimeUDPExemptions you alter the Windows XP Firewall setting by adding our local EAPoUDP port to the Windows XP Firewall boot time UDP exemptions policy. This enables CTA to communicate with ACS over the network.

Note Use of the BootTimeUDPexemptions parameter is relevant only when used in conjunction with Microsoft's hot fix for Windows XP (KB17730)

Configuring Posture Plugins

These are the aspects of posture plugin behavior that are configurable:

- The interaction between CTA and posture plugins for the collection of posture data, transferring notifications, and status updates.
- The message size a posture plugin provides to CTA.
- Reporting behavior of legacy posture plugins.

Configuring CTA and Posture Plugin Interaction

CTA and the posture plugins interact for the transfer of posture data, posture notifications, and status updates. The PPInterfaceType and PPWaitTimeout parameters are used together to determine how CTA interacts with the plugins and how long the interaction with all plugins lasts. The procedure below describes how to set the values of PPInterfaceType and PPWaitTimeout.

Step 1 Locate the ctad.ini file or the ctad-temp.ini template file on the host.

- For Linux and Mac OS X operating systems, navigate to the /etc/opt/CiscoTrustAgent directory.
- For Windows Operating systems, navigate to the Program Files\CiscoSystems\CiscoTrustAgent directory.

Step 2 If there is a ctad.ini file in the directory, you can edit that file directly. If there is **only** a ctad-temp.ini file, make a copy of it and save the copy as "ctad.ini".

Step 3 Locate the [Main] section in the ctad.ini file.

Step 4 Delete the semicolon (;) in front of the **PPInterfaceType** and set the parameter to Block, NonBlockSerial, or NonBlockConcurrent. See the description of these values in the "ctad.ini Configuration Parameters" section on page 5-4.

Step 5 Delete the semicolon (;) in front of the **PPWaitTimeout** parameter and set the PPWaitTimeout period to the number of seconds you require for all posture plugins to return their posture information.

Note The maximum setting for PPWaitTimeout cannot be more than the maximum time that the network access device allows the host to respond to posture requests. If PPWaitTimeout is set at too high a value, the entire posture request will timeout and posture will be re-requested.

Step 6 Save and close the ctad.ini file.

See Example 5-1 for a description of how these parameters interact during a posture request.

Example 5-1 Interaction of PPInterfaceType and PPWaitTimeout parameters

The examples that follow describe the effect of the values of PPInterfaceType and PPWaitTimeout during a request for posture information. The interaction between CTA and the plugin for the transfer of a notification or a status update would also follow the same workflow.

For these examples, assume that these are the posture plugins on the client that return posture credentials and that they take the stated amount of time to return those posture credentials to CTA:

Posture Plugin	Time to collect posture credentials and send them to CTA
CiscoHostPP	0.5 seconds
CTAPP	0.5 seconds
AntivirusPP	3.0 seconds
ApplicationPP	2.0 seconds

Scenario 1—PPInterfaceType is set to “Block” and PPWaitTimeout is set to 5 seconds.

Because PPInterfaceType is set to “Block” one plugin must collect its credential information before the next plugin can collect its posture credential information. Once all the posture information is collected it is sent to CTA.

In this case, it would take 6 seconds for the full amount of posture credentials to be collected before being sent to CTA. The PPWaitTimeout has no effect on the collection of this data because that parameter is only relevant when PPInterfaceType is set to either NonBlockConcurrent or NonBlockSerial.

Scenario 2—PPInetrfaceType is set to NonBlockConcurrent and PPWaitTimeout is set to 5 seconds.

Because PPInterfaceType is set to "NonBlockConcurrent" all the plugins can collect their posture data simultaneously.

In this example, it would take 3 seconds for all the posture credential information to be collected. The PPWaitTimeout parameter is greater than 3 seconds, so it has no effect on the transaction.

Scenario 3—PPInterfaceType is set to NonBlockSerial and PPWaitTimeout is set to 5 seconds.

Because PPInetrfaceType is set to NonBlockSerial the plugins collect their posture data one at a time. In this scenario, the following would occur:

1. CTA requests posture credentials from CiscoHostPP.
2. CiscoHostPP collects its posture credentials.
3. CTA request posture credentials from CTAPP.
4. CTAPP collects its posture credentials.
5. CTA requests posture credentials from AntivirusPP.
6. AntivirusPP collects its posture credentials.
7. CTA requests posture credentials from ApplicationPP.
8. ApplicationPP begins collecting its posture credentials.
9. The PPWaitTimeout expires before ApplicationPP can complete collecting posture credentials.
10. CTA sends the posture credentials for CiscoHostPP, CTAPP, and AntivirusPP to the ACS.
11. CTA considers ApplicationPP an "unresponsive plugin" and will not request its posture credentials again until ApplicationPP supplies its posture credentials for the last request. When ApplicationPP does reply with its posture credentials it is no longer considered an "unresponsive plugin."

Tip These scenarios describe how the posture plugin sends posture credentials to CTA. Once CTA has the posture credentials, it forwards them to ACS. ACS then evaluates the posture credentials against the posture validation network access profile.

In the specific case of scenario 3, if the posture credentials of ApplicationPP are required in order for the client to be granted network access, then access may not be granted because ApplicationPP did not have enough time to report its credentials to CTA and CTA did not have the credentials to send them to ACS.

Configuring the Default Posture Plug-in Message Size

By default, plug-ins are permitted to provide 1024 bytes of information to CTA. This number can be increased to allow all plug-ins to provide up to 6KB of information. However, limiting the size of the posture message to as close to 1KB of information as possible allows more applications to provide a posture message and optimizes the reporting time of all the posture messages. The PPMsgSize parameter in the ctad.ini sets the posture plugin message size for all plugins.

You can also set an application-specific posture plugin message size that is different than the default PPMsgSize value for an application that has its own plugin. See "Configuring an Application-Specific Posture Plug-in Message Size" section on page 5-17 for more information.

The maximum amount of posture data that CTA can send to ACS at one time is 16KB.

Note If there is a Symantec posture plugin installed on the client, the ctad.ini file must be configured in one of two ways:

- PPMsgSize must be set to 1024 bytes.
- The Symantec posture plugin must use an application-specific posture plugin set to 1024 bytes

Without using one of these configurations posture plugin messages from any posture plugin will not be transferred to CTA.

Step 1 Locate the ctad.ini file or the ctad-temp.ini template file on the host.

- For Linux and Mac OS X operating systems, navigate to the /etc/opt/CiscoTrustAgent directory.
- For Windows Operating systems, navigate to the Program Files\CiscoSystems\CiscoTrustAgent directory.

Step 2 If there is a ctad.ini file in the directory, you can edit that file directly. If there is **only** a ctad-temp.ini file, make a copy of it and save the copy as "ctad.ini."

Step 3 Locate the [Main] section in the ctad.ini file.

Step 4 Delete the semicolon (;) in front of the **PPMsgSize**.

Step 5 Increase the **PPMsgSize** to up to 6144 bytes (6KB).

Step 6 Save and close the ctad.ini file.

Configuring an Application-Specific Posture Plug-in Message Size

The posture plugin message size may be customized for any posture plugin. The default posture plugin message size for all plugins is equal to the value of the PPMsgSize parameter in the ctad.ini file. If that default value is not appropriate for a specific posture plug-in, you can specify an application-specific PPMsgSize parameter value.

The application-specific PPMsgSize parameter is added to the ctad.ini file in the [main] section and it uses this naming convention: ***PluginName*_PPMsgSize,** where ***PluginName*** is the name of plugin as specified in the posture plugin's information file.

For example, assume you need to set XYZApplication's unique posture plugin message size to 4096 bytes. Suppose XYZApplication posture plugin's information file defines its posture plugin .dll file name like this:

```
[main]
PluginName=XYZApplicationPP.dll
```

In this example, the name of the new PPMsgSize parameter for XYZApplication you would create would be **XYZApplicationPP_PPMsgSize**.

Note If there is a Symantec posture plugin installed on the client, the ctad.ini file must be configured in one of two ways to maintain the transfer of posture plugin messages from any posture plugin to CTA:

- PPMsgSize must be set to 1024 byte
- The Symantec posture plugin must use an application-specific posture plugin, set to 1024 bytes.

Step 1 In the \Program Files\Common Files\PostureAgent\Plugins or /opt/PostureAgent/Plugins directory, find the .inf file for the posture plugin that requires an application-specific PPMsgSize.

Step 2 Make note of the plugin name in the PluginName field of .inf file.

Step 3 Locate the ctad.ini file or the ctad-temp.ini template file on the host.

- For Linux and Mac OS X operating systems, navigate to the /etc/opt/CiscoTrustAgent directory.
- For Windows Operating systems, navigate to the \Program Files\CiscoSystems\CiscoTrustAgent directory.

Step 4 If there is a ctad.ini file in the directory, you can edit that file directly. If there is **only** a ctad-temp.ini file, make a copy of it and save the copy as "ctad.ini."

Step 5 Locate the [main] section in the ctad.ini file.

Step 6 Add a new parameter to the ctad.ini file following the naming convention described previously in this section. The minimum and maximum value for all PPMsgSize parameters remains 1024 bytes and 6144 bytes respectively.

Step 7 Save and close the ctad.ini file.

Configuring PPMsgSize for Host Posture Plugin

If PPMsgSize is less than 4096 bytes and there is no application-specific PPMsgSize parameter set for the host posture plugin, then CTA internally sets the value of CiscoHostPlugin_PPMsgSize at 4096 bytes.

If you create a specific CiscoHostPP_PPMsgSize parameter for the Host posture plugin, it must be set between 4096 bytes and 6144 bytes.

Configuring PPMsgSize for Symantec Posture Plugin

If there is a Symantec posture plugin installed on the client, PPMsgSize must be set to 1024 bytes or the Symantec posture plugin must use an application-specific posture plugin set to 1024 bytes to maintain the transfer of posture plugin messages from any posture plugin to CTA.

Configuring Asynchronous Posture Status Query

CTA can be configured to query posture plugins at regular intervals to determine if there has been a change to their application's status. If a posture plugin alerts CTA that there has been a change in posture status, CTA alerts the network access device which triggers a re-posturing of the host. This is called "asynchronous posture status query." This feature is available on NAC L2 802.1x networks. To use this feature the Cisco Secure Services Client must installed on the client along with CTA.

Asynchronous posture status query can not be used on NAC L2 IP or NAC L3 IP networks. To configure the regular interval at which CTA queries the resident plugins for posture status, perform the following procedure:

Step 1 Locate the ctad.ini file or the ctad-temp.ini template file on the host.

- For Linux and Mac OS X operating systems, navigate to the /etc/opt/CiscoTrustAgent directory.
- For Windows Operating systems, navigate to the Program Files\CiscoSystems\CiscoTrustAgent directory.

Step 2 If there is a ctad.ini file in the directory, you can edit that file directly. If there is only a ctad-temp.ini file, make a copy of it and save the copy as "ctad.ini".

Step 3 Locate the [Main] section in the ctad.ini file.

Step 4 Delete the semicolon (;) in front of the **SQTimer** parameter.

Step 5 Change the value of SQTimer to reflect the interval at which you want CTA to query posture plugins for a change in posture status.

Configuring User Notifications

User notifications report the "posture" of the host. The messages are sent from Cisco Secure Access Control Server (ACS) to Cisco Trust Agent (CTA). The notifications are displayed as pop-up messages on the desktop, or login screen, of the system on which CTA is installed. These notifications are the only interactive end-user functionality of CTA.

User notifications are configured in the [UserNotifies] section of the ctad.ini configuration file. Any changes made to the [UserNotifies] section of the ctad.ini configuration file are detected and implemented by CTA the next time a notification is received.

The Windows, Linux, and Mac OS X user notification configurations are optional. You do not need to change the default configuration of user notifications in order for CTA to run properly. After reading Configuring Windows User Notifications or Configuring Linux User Notifications, follow the Editing the ctad.ini Configuration File, page 5-3 procedure to make the appropriate changes to the ctad.ini file. Use the sample ctad.ini files and the section on ctad.ini Configuration Parameters, page 5-4 as references.

Configuring Windows User Notifications

To configure user notifications, use the Editing the ctad.ini Configuration File, page 5-3 procedure, reference the sample ctad.ini files, and reference the ctad.ini Configuration Parameters, page 5-4.

You can configure the following notification properties on Windows platforms:

- Where the notifications appear.
  - If the **EnableNotifies** parameter is enabled then user notifications are displayed on the desktop, after the user has logged in.
  - If the **EnableLogonNotifies** parameter in the ctad.ini is enabled, then user notifications received before the user is logged on are saved and displayed to the user when they log on.
- How long the notification dialog box displays before it closes automatically.
  - The **MsgTimeout** parameter defines how long the message is displayed on the desktop.

  - The **LogonMsgTimeout** Specifies how long, in seconds, a message is saved when no user is logged on and when EnableLogonNotifies is enabled.

- The **UserActionDelayTimeout** parameter allows you to delay the launch of the browser window so that the host has more time to obtain an IP address. If the browser that displays the posture message is launched before the host obtains an IP address, the browser will fail to open the URL contained in the posture message.
- **SysModal** window behavior is enabled by default. The behavior requires a user to close a notification dialog box to continue working. You can change this behavior by creating a ctad.ini file and disabling the parameter in the file.

Note If the user notification dialog box is set to system modal, and there is more than one notification message, only the newest message appears. Responding to the message closes all of the message dialog boxes.

Configuring Linux User Notifications

To configure user notifications, use the Editing the ctad.ini Configuration File, page 5-3 procedure, reference the sample Linux ctad.ini file, and reference the ctad.ini Configuration Parameters, page 5-4.

You can configure the following notification properties:

- If the **EnableLogonNotifies** parameter in the ctad.ini is enabled, then user notifications received before the user is logged on are saved and displayed to the user when they log on.

  If a GUI is not installed with the Linux operating system, these notifications are written to a message file in the /var/opt/CiscoTrustAgent/msg directory.
- How long the notification dialog box displays before it closes automatically.
  - The **MsgTimeout** parameter defines how long the message is displayed on the desktop.
  - The **LogonMsgTimeout** specifies how long, in seconds, a message is saved when no user is logged on and when EnableLogonNotifies enabled.

- The **DisplayType** parameter allows you to choose between receiving messages in a terminal window or in the GUI.
- The **TermFont** parameter specifies the font that will be displayed in the terminal window.
- The **ClearOldNotifications** parameter clears or saves old notification messages. If ClearOldNotifications is enabled, an old notification is cleared before showing a new notification.
- The **BrowserPath** parameter specifies the full path to the browser CTA should use.
- The **UserActionDelayTimeout** parameter allows you to delay the launch of the browser window so that the host has more time to obtain an IP address. If the browser that displays the posture message is launched before the host obtains an IP address, the browser will fail to open the URL contained in the posture message**.**

Configuring Mac OS X User Notifications

To configure user notifications, use the Editing the ctad.ini Configuration File, page 5-3 procedure, reference the sample Mac OS X ctad.ini file, and reference the ctad.ini Configuration Parameters, page 5-4.

You can configure the following notification properties:

- If the **EnableLogonNotifies** parameter in the ctad.ini is enabled, then user notifications received before the user is logged on are saved and displayed to the user when they log on.
- How long the notification dialog box displays before it closes automatically.
  - The **MsgTimeout** parameter defines how long the message is displayed on the desktop.
  - The **LogonMsgTimeout** specifies how long, in seconds, a message is saved when no user is logged on and when EnableLogonNotifies enabled.
- The **ClearOldNotifications** parameter clears or saves old notification messages. If ClearOldNotifications is enabled, an old notification is cleared before showing a new notification.

- The **UserActionDelayTimeout** parameter allows you to delay the launch of the browser window so that the host has more time to obtain an IP address. If the browser that displays the posture message is launched before the host obtains an IP address, the browser will fail to open the URL contained in the posture message**.**
- **SysModal** window behavior is enabled by default. The behavior requires a user to close a notification dialog box to continue working. You can change this behavior by creating a ctad.ini file and disabling the parameter in the file.

Note If the user notification dialog box is set to system modal, and there is more than one notification message, only the newest message appears. Responding to the message closes all of the message dialog boxes.

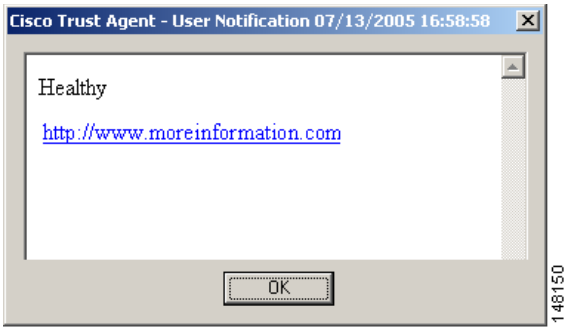
Configuring Clickable URL and Browser Auto-Launch Features

After the overall posture of the client has been determined, the Cisco Secure ACS can be configured to notify the user of that posture. Notifications may be a message in a pop-up window or the user can be directed to a particular URL.

- **Clickable URL**

  The pop-up window that notifies the user will contain a posture value, such as “Healthy” or “Quarantine.” It can also contain a clickable URL for the user to follow. For example, if the client requires remediation, the clickable URL can direct the user to an area where a particular remediation solution is available.

Figure 5-1 Clickable URL Notification Pop-up



- **Browser Auto-launch**

  Instead of receiving a pop-up window with a clickable URL, the notification can be configured to automatically open a browser window which is already pointing to the URL the user requires.

Logging Notifications

If logging is enabled, the following notification events are logged:

- Notification failures, such as a failure to create the notification dialog box.
- User notification messages.

The messages that are logged are filtered by the logging level assigned to the notification component of CTA. This logging level is specified by the CTAMsg parameter in the [LogLevel] section of the ctalogd.ini configuration file. If this parameter does not appear in the configuration file, or if the configuration file does not exist, then a default level of 3-High is used. This level allows all informational, warning, and critical messages to be written to the log file. The message severity is assigned by the component or plugin that sent the message.

For more information about logging, see Chapter 6, “Cisco Trust Agent Event Logging”.

Certificate Distinguished Name Matching

Certificate distinguished name (DN) matching is performed when CTA communicates with ACS using the NAC L3 IP and NAC L2 IP methods. Similar functionality can be obtained for NAC L2 802.1x environments by configuring Cisco Secure Services Client to validate trusted servers.

When using CA certificates to validate your ACS server certificate to CTA, you can implement additional security using DN matching to validate the certificate. This prevents other servers or processes that may be using the same root certificate from gaining a trust relationship with the host.

DN matching occurs at the end of the transport layer security (TLS) handshake, after the certificate chain is built. After CTA has received ACS's certificate, this rule is employed to ensure that the properties of the certificate fields have the expected values.

If any rule in the [ServerCertDNVerification] section of the ctad.ini file succeeds then the ACS server is authenticated to CTA.

Note There are no default settings for the [ServerCertDNVerification] section; that section in the sample file contains example information. If you use this sample file as the basis for your own ctad.ini file, delete the section or change it to suit your environment.

DN Matching Rule Syntax

Each rule can contain multiple sub-rules, separated by a comma. If a single sub-rule within a rule fails, then the entire rule fails. The maximum length for a single rule is 255 characters.

The following shows the general format for a rule:

```
Rule1=subrule, subrule, subrule, ...
```

Each sub-rule consists of a certificate subject attribute followed by a value:

```
attribute[operator]"value"
```

The following operators are supported:

- **= (equals)**—The value must exactly match the value given in the subrule.
- *** (contains)**—The value must contain the value given in the subrule.

The following DN attributes are supported:

- **CN**—Common Name
- **SN**—Surname
- **GN**—Given Name
- **N**—Unstructured Name
- **I**—Initials
- **GENQ**—Generation Qualifier
- **C**—Country
- **L**—Locality
- **SP**—Province
- **ST**—State
- **O**—Organization
- **OU**—Organization Unit
- **T**—Title
- **EA**—E-mail Address

Note You can also create sub-rules that check certificate issuer attributes by adding the ISSUER- prefix to the attributes listed above. For example, to validate the Common Name of certificate issuer, you would use the attribute ISSUER-CN.

Keeping the syntax in mind, here is an example of two rules:

```
Rule1=CN*"Server", ISSUER-CN*"Finance"
Rule2=CN="Finance posture Cert", OU*"Finance", ISSUER-CN*"ACME"
```

At the end of the TLS handshake the Subject field or Issuer field in the Cisco Secure ACS server certificate are compared to these rules. The DN matching will succeed if one or the other rules succeed.

Rule 1 indicates that the Subject field CN (Common Name) must contain “Server” and the Issuer field CN must contain “Finance.” If both fields do not have the correct information the rule fails.

Rule 2 indicates that the Subject field CN must equal “Finance posture Cert”, the Organization Unit (OU) field must contain “Finance”, and the Issuer CN field must contain “ACME”. If all of these subrules do not have the correct information, the rule fails.

Configuring Certificate DN Matching

Step 1 Locate the ctad.ini file or the ctad-temp.ini template file on the host.

- For Linux and Mac OS X operating systems, navigate to the /etc/opt/CiscoTrustAgent directory.
- For Windows Operating systems, navigate to the Program Files\CiscoSystems\CiscoTrustAgent directory.

Step 2 If there is a ctad.ini file in the directory, you can edit that file directly. If there is **only** a ctad-temp.ini file, make a copy of it and save the copy as “ctad.ini”.

Step 3 Locate the [ServerCertDNVerification] section in the ctad.ini file.

Step 4 Delete the semicolon (;) in front of the TotalRules parameter and equate the parameter to the number of rules you are going to write.

Step 5 Following the example in the ctad.ini file create a new line for each rule. Begin each line with “RuleX=” where X is the number of the rule.

Step 6 Write each rule following the “DN Matching Rule Syntax” section on page 5-25.

Step 7 Save and close the file. The new rules will be implemented the next time DN matching occurs.

Configuring the Scripting Interface

The delta_stale parameter in the [ScriptingInterface] section of the ctad.ini file is the only configurable parameter for the Scripting Interface. This parameter defines the time, in minutes, before the posture data file is considered out of date.

To configure the CTA Scripting Interface, use the Editing the ctad.ini Configuration File, page 5-3 procedure, reference the sample ctad.ini files, and reference the “ctad.ini Configuration Parameters” section on page 5-4.

Sample Windows ctad.ini File

```
; ***************************************
; CTAD.INI FILE DEFINITION
; This file defines communication parameters between a
; Network Access Device (NAD) and Cisco Trust Agent (CTA).
; It also defines variables for notifications and certificate
; filtering rules.
;
; This file can be edited with a plain text editor and used
; in a custom installation of CTA.
;
; The default location for the ctad.ini file is
; the Program Files\CiscoSystems\CiscoTrustAgent\
; directory.
;
; GENERAL EDITING INSTRUCTIONS
; To "turn on" a parameter in a section, delete the ; before
; the ParameterName.
; To "turn off" a parameter, type a ; before the ParameterName.
; ***************************************

; ***************************************
[main]

;The EnableVFT parameter indicates if Validation-Flag TLV
;   is enabled on the version of IOS running on the Network Access
;   Device (NAD). “EnableVFT” enables CTA to operate with the NAD
;   wether it has support for Validation-Flag TLV or not.
;Default Value: 0
;Range of Values: 0, 1
;   0 = IOS does not support Validation-Flag TLV
;   1 = IOS does support Validation-Flag TLV
;EnableVFT=0

;The PPInterfaceType parameter describes how CTA gathers posture
;   plugin information.
;   Default value: Block
;   Range of values: Block, NonBlockConcurrent, NonBlockSerial
:   Block = CTA will request posture information from one plug-in at a time.
;   NonBockConcurrent = Request posture information from all posture plugins
;   simultaneously.
;   NonBlockSerial = Requests posture information from posture plugins one at
;   a time and waits for either the return of the posture credentials
;   or the end of the PPWaitTimeout value before
;   requesting posture credentials from the next posture plugin.
;PPInterfaceType=Block
```

```
;The PPWaitTimeout parameter represents the maximum time allowed,
;   in seconds, to complete the processing of all plug-ins.
;   Default value: 5 seconds
;   Range of values: 1 - 300 seconds
;PPWaitTimeout=5

; The PPMsgSize parameter allows the administrator to modify
;   the maximum message size that a posture plugin can send
;   from 1k to a maximum value of 6k.
;   Default value: 1024 bytes
;   Range of values: 1024 - 6144 bytes
;PPMsgSize=1024

;The SQTimer (status query timer) parameter defines
;   how often CTA queries the posture plugins to detect a change
;   in their status.
;   Default value: 300 seconds
;   Range of values: 5 - 4294967925 seconds
;SQTimer=300

;The [EAPoUDP] section defines the communication settings between CTA
;   and a Layer 3 Network Access Device (NAD), such as a router.
[EAPoUDP]

;The LocalPort parameter defines the port on which the NAD initiates
;   posture validation with CTA. Changing this value requires
;   changes to the NAD configuration.
;   Default value: 21862
;   Rang of values: 1 - 65550
;LocalPort=21862

;CTA supports one established session per NAD. But CTA can support
;   concurrent sessions with multiple NADs. MaxSessions is
;   the number of sessions you allow CTA to support.
;   Default value: 8
;   Range of values: 1 - 256
;MaxSession=8

;The SessionIdleTimeout parameter defines the number of seconds
;   an EAP over UDP session can remain idle before timing out.
;   Default value: 3600
;   Range of values: 60 - 172800
;SessionIdleTimeout=3600

;The BootTimeUDPExemptions parameter alters the Windows Firewall
;   policy and enables CTA to receive packets when the
;   Windows XP SP2 or SP3-based computer is starting.
```

```
;   Default value: 1
;   Range of values: 0, 1
;   0 = Windows Firewall Boot Time Exemptions are disabled.
;   1 = Windows Firewall Boot Time Exemptions are enabled.
;BootTimeUDPExemptions=1

;The [UserNotifes] section defines the behavior of pop-up windows
;   containing messages sent from ACS to CTA
[UserNotifies]

;SysModal specifies whether the user notification dialog boxes are
;   modal or not. If the notification dialog boxes are modal,
;   the user must close the notification dialog box to continue
;   working. Also, if the dialog boxes are modal, and there is
;   more than one notification, only the last notification appears.
;   Default value: 1
;   Range of values: 0, 1
;   0 = Modal dialog boxes are disabled.
;   1 = Modal dialog boxes are enabled.
;SysModal=1

;Setting this UserActionDelayTimeout parameter allows you to delay the launch of
;   the browser window so that the host has more time to obtain an IP address.
;   If the browser that displays the posture message is launched before the host
;   obtains an IP address, the browser will fail to open the URL contained in the
;   posture message.
;   Default value: 25 seconds
;   Range of values: 0 - 4294967295 seconds
;UserActionDelayTimeout=25

;The EnableNotifies parameter enables or disables user
;   notifications. This parameter applies to logged-in users.
;   Default value: 1
;   Range of values 0, 1
;   0 = User notifications are disabled.
;   1 = User notifications are enabled.
;EnableNotifies=1

;The MsgTimeout parameter specifies how long, in seconds,
;   user notification dialog boxes are displayed.
;   Default value: 300
;   Range of values: 30 - 4294967
;   Special value: 0 (disables the timeout)
;MsgTimeout=300

;The EnableLogonNotifies parameter enables or disables user
;   notifications received before the user is logged on.
;   Default value: 1
```

```
;   Range of values: 0, 1
;   0 = User notifications received before the user is logged on are
;   discarded.
;   1 = User notifications received before the user is logged on are
;   saved and displayed to the user when the log on.
;EnableLogonNotifies=1

;The LogonMsgTimeout Specifies how long, in seconds, a message
;   is saved when no user is logged on and when
;   EnableLogonNotifies is enabled.
;   Default value: 86400
;   Range of values: 30 - 4294967
;   Special value: 0 (disables the timeout)
;   LogonMsgTimeout=86400

;The [ServerCertDNVerification] section contains configurable
;   parameters for distinguished name (DN) matching. When
;   using CA certificates to validate your Cisco Secure ACS
;   server certificate, you can implement additional security
;   using distinguished name matching.
[ServerCertDNVerification]

;The TotalRules parameter defines the number of DN matching rules
;   that follow it. If the number of rules is greater than 1
;   the rules are connected by an OR statement.
;   Default value: (none)
;   Range of values: 0 - 64
;   Special values: 0 (Disables DN matching)
;TotalRules=2

;The RuleX parameters are DN matching rules, where X is the index
;   for the rule.
;   NOTE: THE RULES BELOW ARE EXAMPLES ONLY. DO NOT USE THEM
;   WITHOUT MODIFYING THEM TO SUIT YOUR ENVIRONMENT.
;Rule1=CN*"Server", ISSUER-CN*"Finance"
;Rule2=CN="Finance posture Cert", OU*"Finance", ISSUER-CN*"ACME"

;The parameters in the [Scripting_Interface] section define the
;   scripting interface behavior.
[Scripting_Interface]

; The delta_stale parameter specifies how long, in minutes, before
;   the posture database record is deemed outdated.
;   Default value: 43200
;   Range of values: 1-5256000
;   Special value: 0 (the database will never expire)
;delta_stale=43200
; ***************************************
```

Sample Linux ctad.ini File

```
; ************************************
; CTAD.INI FILE DEFINITION
; This file defines communication parameters between a
; Network Access Device (NAD) and Cisco Trust Agent (CTA).
; It also defines variables for notifications and certificate
; filtering rules.
;
; This file can be edited with a plain text editor and used
; in a custom installation of CTA.
;
; The default location of the ctad.ini file is the
; /etc/opt/CiscoTrustAgent/ directory.

; GENERAL EDITING INSTRUCTIONS
; To "turn on" a parameter in a section, delete the ; before
; the ParameterName.
; To "turn off" a parameter, type a ; before the ParameterName.
; ************************************

; ************************************
[main]
;The “EnableVFT” parameter indicates if Validation-Flag TLV
;   is enabled on the version of IOS running on the Network Access
;   Device (NAD). “EnableVFT” enables CTA to operate with the NAD
;   wether it has support for Validation-Flag TLV or not.
;Default Value: 0
;Range of Values: 0, 1
;   0 = IOS does not support Validation-Flag TLV
;   1 = IOS does support Validation-Flag TLV
;EnableVFT=0

;The PPInterfaceType parameter describes how CTA gathers posture
;   plugin information.
;   Default value: Block
;   Range of values: Block, NonBlockConcurrent, NonBlockSerial
:   Block = CTA will request posture information from one plug-in at a time.
;   NonBockConcurrent = Request posture information from all posture plugins
;   simultaneously.
;   NonBlockSerial = Requests posture information from posture plugins one at
;   a time and waits for either the return of the posture credentials
;   or the end of the PPWaitTimeout value before
;   requesting posture credentials from the next posture plugin.
;PPInterfaceType=Block
```

```
;The PPWaitTimeout parameter represents the maximum time allowed,
;   in seconds, to complete the processing of all plug-ins.
;   Default value: 5 seconds
;   Range of values: 1 - 300 seconds
;PPWaitTimeout=5

; The PPMsgSize parameter allows the administrator to modify
;   the maximum message size that a posture plugin can send
;   from 1k to a maximum value of 6k.
;   Default value: 1024 bytes
;   Range of values: 1024 - 6144 bytes
;PPMsgSize=1024

;The SQTimer (status query timer) parameter defines how often CTA queries the posture
;   plugins to detect a change in their status.
;   Default value: 300 seconds
;   Range of values: 5 - 4294967925 seconds
;SQTimer=300

; The [EAPoUDP] section defines the communication settings
;   between CTA and the Network Access Device (NAD).
[EAPoUDP]

;The LocalPort defines the port on which the NAD initiates posture
;   validation with CTA. Changing this value requires changes
;   to the NAD configuration.
;   Default value: 21862
;   Rang of values: 1 - 65550
;LocalPort=21862

;CTA supports one established session per NAD. But can support
;   concurrent sessions with multiple NADs. MaxSessions is
;   the number of sessions you allow CTA to support.
;   Default value: 8
;   Range of values: 1 - 256
;MaxSession=8

;SessionIdleTimeout defines the number of seconds an EAP over UDP
;   session can remain idle before timing out.
;   Default value: 3600
;   Range of values: 60 - 172800
;SessionIdleTimeout=3600

;The [UserNotifes] section defines the behavior of pop-up windows
;   containing messages sent from ACS to CTA
[UserNotifies]
```

```
;Setting this UserActionDelayTimeout parameter allows you to delay the launch of
;   the browser window so that the host has more time to obtain an IP address.
;   If the browser that displays the posture message is launched before the host
;   obtains an IP address, the browser will fail to open the URL contained in the
;   posture message.
;   Default value: 25 seconds
;   Range of values: 0 - 4294967295 seconds
;UserActionDelayTimeout=25

;The EnableNotifies parameter enables or disables user
;   notifications. This parameter applies to logged-in users.
;   Default value: 1
;   Range of values 0, 1
;   0 = User notifications are disabled.
;   1 = User notifications are enabled.
;EnableNotifies=1

;The MsgTimeout parameter specifies how long, in seconds,
;   user notification dialog boxes are displayed.
;   Default value: 300
;   Range of values: 30 - 4294967
;   Special value: 0 (disables the timeout)
;MsgTimeout=300

;The EnableLogonNotifies parameter enables or disables user
;   notifications received before the user is logged on.
;   Default value: 1
;   Range of values: 0, 1
;   0 = User notifications received before the user is logged on are
;   discarded.
;   1 = User notifications received before the user is logged on are
;   saved and displayed to the user when the log on.
;EnableLogonNotifies=1

;The LogonMsgTimeout Specifies how long, in seconds, a message
;   is saved when no user is logged on and when
;   EnableLogonNotifies is enabled.
;   Default value: 86400
;   Range of values: 30 - 4294967
;   Special value: 0 (disables the timeout)
;   LogonMsgTimeout=86400

;The DisplayType parameter determines if messages sent from
;   ACS to CTA are displayed in a graphic user interface or
;   in a terminal window.
;   Default value: gui
;   Range of values: term, gui
;DisplayType=gui
```

```
;The TermFont parameter sets the font in which to display
;   the terminal screen text. Use the default value until
;   Cisco Secure ACS can forward localized messages to CTA.
;   Default value: -misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso10636-1
;TermFont=-misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso10636-1
;TermFont entry below for Asian languages
;TermFont=-misc-zysong18030-medium-r-normal--0-0-0-0-c-0-iso10646-1

; The BrowserPath parameter specifies the full path of the browser on Linux systems.
;   For Red Hat Enterprise Linux v3 (Enterprise, Advanced, Workstation)
;   use this path: /usr/bin/mozilla
;   For Red Hat Enterprise Linux v4 (Enterprise, Advanced, Workstation)
;   use this path: /usr/bin/firefox
; BrowserPath=

;The ClearOldNotification parameter clears or saves notification
;   messages.
; Default value: 1
; Range of Values:
;   0 = Notification messages are saved.
;   1 = CTA clears the old notification message before displaying
;       the new window.
;ClearOldNotification=1

;The [ServerCertDNVerification] section contains configurable
;   parameters for distinguished name (DN) matching. When
;   using CA certificates to validate your Cisco Secure ACS
;   server certificate, you can implement additional security
;   using distinguished name matching.
[ServerCertDNVerification]

;The TotalRules parameter defines the number of DN matching rules
;   that follow it. If the number of rules is greater than 1
;   the rules are connected by an OR statement.
;   Default value: (none)
;   Range of values: 0 - 64
;   Special values: 0 (Disables DN matching)
;TotalRules=2

;The RuleX parameters are DN matching rules, where X is the index
;   for the rule.
;   NOTE: THE RULES BELOW ARE EXAMPLES ONLY. DO NOT USE THEM
;   WITHOUT MODIFYING THEM TO SUIT YOUR ENVIRONMENT.
;Rule1=CN*"Server", ISSUER-CN*"Finance"
;Rule2=CN="Finance posture Cert", OU*"Finance", ISSUER-CN*"ACME"
```

```
;The parameters in the [Scripting_Interface] section define the
;   scripting interface behavior.
[Scripting_Interface]

; The delta_stale parameter specifies how long, in minutes, before
;   the posture database record is deemed outdated.
;   Default value: 43200
;   Range of values: 1-5256000
;   Special value: 0 (the database will never expire)
;delta_stale=43200
```

Sample Mac OS X ctad.ini File

```
; ************************************
; CTAD.INI FILE DEFINITION
; This file defines communication parameters between a
; Network Access Device (NAD) and Cisco Trust Agent (CTA).
; It also defines variables for notifications and certificate
; filtering rules.
;
; This file can be edited with a plain text editor and used
; in a custom installation of CTA.
;
; The default location of the ctad.ini file is the
; /etc/opt/CiscoTrustAgent/ directory.

; GENERAL EDITING INSTRUCTIONS
; To "turn on" a parameter in a section, delete the ; before
; the ParameterName.
; To "turn off" a parameter, type a ; before the ParameterName.
; ************************************

; ************************************
[main]
;The “EnableVFT” parameter indicates if Validation-Flag TLV
;   is enabled on the version of IOS running on the Network Access
;   Device (NAD). “EnableVFT” enables CTA to operate with the NAD
;   wether it has support for Validation-Flag TLV or not.
;Default Value: 0
;Range of Values: 0, 1
;   0 = IOS does not support Validation-Flag TLV
;   1 = IOS does support Validation-Flag TLV
;EnableVFT=0

;The PPInterfaceType parameter describes how CTA gathers posture
;   plugin information.
;   Default value: Block
;   Range of values: Block, NonBlockConcurrent, NonBlockSerial
:   Block = CTA will request posture information from one plug-in at a time.
;   NonBockConcurrent = Request posture information from all posture plugins
;   simultaneously.
;   NonBlockSerial = Requests posture information from posture plugins one at
;   a time and waits for either the return of the posture credentials
;   or the end of the PPWaitTimeout value before
;   requesting posture credentials from the next posture plugin.
;PPInterfaceType=Block
```

```
;The PPWaitTimeout parameter represents the maximum time allowed,
;   in seconds, to complete the processing of all plug-ins.
;   Default value: 5 seconds
;   Range of values: 1 - 300 seconds
;PPWaitTimeout=5

; The PPMsgSize parameter allows the administrator to modify
;   the maximum message size that a posture plugin can send
;   from 1k to a maximum value of 6k.
;   Default value: 1024 bytes
;   Range of values: 1024 - 6144 bytes
;PPMsgSize=1024

;The SQTimer (status query timer) parameter defines how often CTA queries the posture
;   plugins to detect a change in their status.
;   Default value: 300 seconds
;   Range of values: 5 - 4294967925 seconds
;SQTimer=300

; The [EAPoUDP] section defines the communication settings
;   between CTA and the Network Access Device (NAD).
[EAPoUDP]

;The LocalPort defines the port on which the NAD initiates posture
;   validation with CTA. Changing this value requires changes
;   to the NAD configuration.
;   Default value: 21862
;   Rang of values: 1 - 65550
;LocalPort=21862

;CTA supports one established session per NAD. But can support
;   concurrent sessions with multiple NADs. MaxSessions is
;   the number of sessions you allow CTA to support.
;   Default value: 8
;   Range of values: 1 - 256
;MaxSession=8

;SessionIdleTimeout defines the number of seconds an EAP over UDP
;   session can remain idle before timing out.
;   Default value: 3600
;   Range of values: 60 - 172800
;SessionIdleTimeout=3600

;The [UserNotifes] section defines the behavior of pop-up windows
;   containing messages sent from ACS to CTA
[UserNotifies]
```

```
;Setting this UserActionDelayTimeout parameter allows you to delay the launch of
;   the browser window so that the host has more time to obtain an IP address.
;   If the browser that displays the posture message is launched before the host
;   obtains an IP address, the browser will fail to open the URL contained in the
;   posture message.
;   Default value: 25 seconds
;   Range of values: 0 - 4294967295 seconds
;UserActionDelayTimeout=25

;The EnableNotifies parameter enables or disables user
;   notifications. This parameter applies to logged-in users.
;   Default value: 1
;   Range of values 0, 1
;   0 = User notifications are disabled.
;   1 = User notifications are enabled.
;EnableNotifies=1

;The MsgTimeout parameter specifies how long, in seconds,
;   user notification dialog boxes are displayed.
;   Default value: 300
;   Range of values: 30 - 4294967
;   Special value: 0 (disables the timeout)
;MsgTimeout=300

;The EnableLogonNotifies parameter enables or disables user
;   notifications received before the user is logged on.
;   Default value: 1
;   Range of values: 0, 1
;   0 = User notifications received before the user is logged on are
;   discarded.
;   1 = User notifications received before the user is logged on are
;   saved and displayed to the user when the log on.
;EnableLogonNotifies=1

;The LogonMsgTimeout Specifies how long, in seconds, a message
;   is saved when no user is logged on and when
;   EnableLogonNotifies is enabled.
;   Default value: 86400
;   Range of values: 30 - 4294967
;   Special value: 0 (disables the timeout)
;   LogonMsgTimeout=86400

;The ClearOldNotification parameter clears or saves notification
;   messages.
; Default value: 1
; Range of Values:
;   0 = Notification messages are saved.
;   1 = CTA clears the old notification message before displaying
```

```
;       the new window.
;ClearOldNotification=1

;The [ServerCertDNVerification] section contains configurable
;   parameters for distinguished name (DN) matching. When
;   using CA certificates to validate your Cisco Secure ACS
;   server certificate, you can implement additional security
;   using distinguished name matching.
[ServerCertDNVerification]

;The TotalRules parameter defines the number of DN matching rules
;   that follow it. If the number of rules is greater than 1
;   the rules are connected by an OR statement.
;   Default value: (none)
;   Range of values: 0 - 64
;   Special values: 0 (Disables DN matching)
;TotalRules=2

;The RuleX parameters are DN matching rules, where X is the index
;   for the rule.
;   NOTE: THE RULES BELOW ARE EXAMPLES ONLY. DO NOT USE THEM
;   WITHOUT MODIFYING THEM TO SUIT YOUR ENVIRONMENT.
;Rule1=CN*"Server", ISSUER-CN*"Finance"
;Rule2=CN="Finance posture Cert", OU*"Finance", ISSUER-CN*"ACME"

;The parameters in the [Scripting_Interface] section define the
;   scripting interface behavior.
[Scripting_Interface]

; The delta_stale parameter specifies how long, in minutes, before
;   the posture database record is deemed outdated.
;   Default value: 43200
;   Range of values: 1-5256000
;   Special value: 0 (the database will never expire)
;delta_stale=43200
```

CHAPTER 6

Cisco Trust Agent Event Logging

CTA logging is disabled by default because CTA is intended to be a transparent application to end users. If you enable logging, CTA logs events generated by CTA components and the posture plugins for the NAC-compliant applications that reside on the system.

This chapter contains the following sections:

How Logging Works

Event logging is implemented as a service on the network client. When the CTA Event Logger service starts, it reads the logging configuration file, ctalogd.ini and uses any logging levels and settings that are specified. If no logging levels or settings are specified in the ctalogd.ini file, the logging service uses its default values.

If logging is enabled, the events are evaluated against the default logging levels or those defined in the ctalogd.ini file. Events that meet the logging level are formatted and written to the log file.

Note CTA logging is disabled by default and can be enabled by using the clogcli utility, see "The clogcli Logging Utility" section on page 6-4 for more information.

CTA Log Files

CTA log files are text files created by the Cisco Trust Agent Event Logging service. They are ASCII text files that you can view using any text editor. Log file names are automatically generated by the event logger whenever a log file is created. A new log file is created when one of the following events occur:

- Logging is changed from disabled to enabled.
- The log file is cleared while logging is enabled.
- The current log file reaches the maximum file size.

Note The creation of the log file does not occur until the first event is received while logging is enabled.

On **Windows** operating systems, this is the default location of log files:

```
\Program Files\Cisco Systems\CiscoTrustAgent\Logging\Logs
```

On **Linux** and **Mac OS X** operating systems, this is the default location of log files:

```
/var/log/CiscoTrustAgent
```

These log file directory locations can be changed using the clogcli utility. See, "The clogcli Logging Utility" section on page 6-4 for this procedure.

The logfile names use the date and time the event logger was started to create unique file names. The log file names contain the following format:

CTALOG-YYYY-MM-DDTHH-MM-SS_N.txt.

- **CTALOG**—A fixed prefix indicating that CTA created the log.
- **YYYY**—A four-digit value for the year.
- **MM**—A two-digit value for the month.
- **DD**—A two-digit value for the day.
- **T**—A fixed separator between the date and time.
- **HH**—A two-digit value for the hours, specified in 24-hour time.
- **MM**—A two-digit value for the minutes.
- **SS**—A two-digit value for the seconds.
- **N**—The n^{th} log file created since the event logger was started. This occurs when the current log file reaches the maximum size or when logging is disabled and then enabled.

For example, if the event logger was started on September 20, 2007 at 5:12:58 p.m. the generated log file name would be named `CTALOG-2007-09-20T17-12-58_1.txt`.

Log File Format

The log file contains the following fields:

- **Logging Instance**—An incremental number for the log entry.
- **Date/Time**—The date and time the entry was logged.
- **Severity**—The severity of the logged event. Valid severity values are:
  - Critical
  - Info
  - Warning
- **Error Code**—The error code associated with the event.
- **Message Body**—Text describing the event.

The following displays examples of log file entries:

Example 6-1 CTA Log File Sample Entries

```
Cisco Systems Trust Agent Version 2.0
Copyright (C) 1998-2003 Cisco Systems, Inc. All Rights Reserved.
Trust Agent Type(s): Windows, WinNT Running on: 5.1.2600

1   15:29:57.748  07/13/05 Sev=Info/3  PADaemon/0x6300000C
Starting service:

2   15:30:40.719  07/13/05 Sev=Info/3  NetTrans/0x63100016
NAD proposed AssociationID 56789
```

Logging Considerations

Log files remain on the disk until a user deletes them, the allotted disc space for logs is used up, or the log file reaches a certain age.

If you enable a high-level of logging, you can potentially fill the disk over time. If you enable logging by default, ensure that a policy for regular log file removal or archival is in place.

The clogcli Logging Utility

CTA provides a utility, **clogcli**, which can be run locally on the end user system. clogcli provides a way for users to enable, disable, and configure logging. This utility is useful in situations where local system troubleshooting is required. The clogcli utility runs at the command line.

On Windows operating systems, clogcli is stored in this directory:

```
\Program Files\Cisco Systems\CiscoTrustAgent\
```

On Linux and Mac OS X operating systems, clogcli is stored in this directory:
`/opt/CiscoTrustAgent/sbin`

To run the clogcli utility, follow this procedure:

Step 1 Open a Command Prompt Window (Windows) or a Terminal Window (Linux and Mac OS X).

Step 2 Change the directory to the one in which clogcli is stored:

- For Windows operating systems, change the directory to: `\Program Files\Cisco Systems\CiscoTrustAgent\`
- For Linux and Mac OS X operating systems, change the directory to: `/opt/CiscoTrustAgent/sbin`

Step 3 At the prompt type **clogcli**, followed by the proper command, and press <Enter>. The command syntax is the same for Windows and Linux operating systems and slightly different for Mac OS X operating systems.

Table 6-1 describes all the commands and options for the clogcli utility.

Table 6-1 clogcli Utility Command Option

Option	Description and example
clear	Clears the current log file. A new log file is created if logging is enabled. Linux and Windows example: #> **clogcli clear** Mac OS X example: $ **./clogcli clear**
disable	Disables logging. Parameter name in ctalogd.ini: EnableLog=0 Linux and Windows example: #> **clogcli disable** Mac OS X example: $ **./clogcli disable**

Table 6-1 clogcli Utility Command Option (continued)

Option	Description and example
enable	Enables logging. Parameter name in ctalogd.ini: EnableLog=1 Linux and Windows example: #> **clogcli enable** Mac OS X example: $ **./clogcli enable**
enable -t	Enables logging until the machine is rebooted. Parameter name in ctalogd.ini: EnableLog=2 Linux and Windows example: #> **clogcli enable -t** Mac OS X example: $ **./clogcli enable -t**

Table 6-1 clogcli Utility Command Option (continued)

Option	Description and example
logdir	Sets the log file location for CTA logs. Parameter name in ctalogd.ini: LogDir Default Windows Location: \Program Files\Cisco Systems\CiscoTrustAgent\Logging\Logs Default Linux Location: `/var/log/CiscoTrustAgent` Range of Values: Any existing directory location. Windows example: > **`clogcli logdir c:\Temp\CTALogs`** Linux example: # **`clogcli logdir /tmp/CTALogs`** Mac OS X example: $ **`./clogcli logdir /tmp/CTALogs`** **Note** This note is for **Linux** systems only. For security reasons, ctalogd does not run with "administrator" or "root" permissions. Therefore, it is possible for you to specify a directory that ctalogd does not have permission to write to. If ctalogd is not able to create the log file in the directory you specify, then it will create the log file in the default logging directory. You should also see an error message in the syslog. After you begin logging, check both your chosen and the default directories to verify the location of your log file.

Table 6-1 clogcli Utility Command Option (continued)

Option	Description and example
loglevel	Sets the log level for all CTA components at once and to the same level. See, "Logging Levels" section on page 6-11 for descriptions of the logging levels. Default Value: 3 Range of values: 1-3, 15 Linux and Windows example: #> **clogcli loglevel 2** Mac OS X example: $ **./clogcli loglevel 2**
maxdisk	Sets the maximum number of megabytes of space that may be used for logging. Parameter name in ctalogd.ini: MaxDiskSize Default Value: 50 Range of Values: 50-100 Special Value: 0 - If maxdisk is set to zero then there is no disk space limit for log files. Linux and Windows example: #> **clogcli maxdisk 60** Mac OS X example: $ ./**clogcli maxdisk 60**

Table 6-1 clogcli Utility Command Option (continued)

Option	Description and example
maxfile	Sets the minimum log file size in megabytes. Parameter name in ctalogd.ini: MaxFileSize Default Value: 4 MB Range of Values: 0 - 50 MB. Special Value: 0 - If maxfile is set to zero, then there is no limit on the log size. Linux and Windows example: #> **clogcli maxfile 5** Mac OS X example: $ **./clogcli maxfile 5**

Table 6-1 clogcli Utility Command Option (continued)

Option	Description and example
zipit	This command retrieves log files and configuration files and inserts the files into a zip file which is stored on the Windows desktop and in the user's home directory on Linux or Mac OS X operating systems. Linux and Windows example: #> clogcli **zipit** Mac OS X example: $ **./clogcli zipit** The zip file is named: *NAC-TS-YYYY-MM-DDTHH-MM-SS*.zip Where "NAC-TS" stands for Network Admission Control - Technical Support. The log file follows the Year-Month-DayTHours-Minutes-Seconds convention of the ctalog file. These are the files collected by the **clogcli zipit** command: • CTA log files: \Program Files\Cisco Systems\CiscoTrustAgent\Logging\Logs**CTALOG*.txt** • If CTA 802.1x Wired Client is installed or was once installed and the old log files were not deleted, these files will also be collected: – \Program Files\Cisco Systems\Cisco Trust Agent 802_1x Wired Client \system\log**apiDebug*.txt** – \Program Files\Cisco Systems\Cisco Trust Agent 802_1x Wired Client \system\log**clientDebug*.txt** – \Program Files\Cisco Systems\Cisco Trust Agent 802_1x Wired Client**log*.txt** • Output from ctastat: **ctastat-output.txt** • **ctad.ini file** • **ctalogd.ini file** **Note** Log files generated by Cisco Secure Services Client are not collected by the clogcli zipit command.

Logging Levels

Setting the logging level determines which events are added to the log file. Each logging level provides an incremental addition to the logging information provided by the level below it.

- **1-LOW** — Low-level logging includes critical events. The intent of the low setting is to ensure that your log file does not grow too large while still logging the events that are likely most worth your attention.
- **2-MEDIUM** — Medium-level logging includes warnings and the critical events provided in the low setting.
- **3-HIGH** — High-level logging includes informational events, such as success messages, warnings, and critical events.
- **15-EVERYTHING** — This is the most verbose level of logging. It captures all messages.

HIGH is the default level used when logging is enabled. You can change the logging level for each CTA component using the clogcli utility.

As a general guideline, when troubleshooting problems on systems running CTA, keep the logging level set to **3** to receive the most information about the system operation. If you configure logging to be enabled and there are no issues, you should set the logging level to Medium or Low. Setting the logging level to medium or low prevents the log file from growing rapidly and consuming disk space, yet provides enough information to diagnose any possible problems with CTA, posture plugins, or system posture.

Configuring CTA Logging For Large Deployments

To deploy a specialized level of logging state, logging size, log file locations and log file size, you can create a ctalogd.ini file that can be distributed in a custom installation package.

The ctalogd.ini file contains the configuration parameters for CTA logging. The ctalogd.ini file is not delivered at the time of installation; it is created when you use the clogcli utility to enable logging and change logging attributes.

CTA logging is disabled by default. Once logging is enabled, the default logging attributes are used unless you specify different attributes in the ctalogd.ini file.

To configure CTA logging, follow this procedure:

Step 1 Use the simplest method to install CTA on a test machine. See, Chapter 2, "Installing the Cisco Trust Agent on Linux Operating Systems," Chapter 3, "Installing the Cisco Trust Agent on Macintosh Operating Systems,"or Chapter 4, "Installing the Cisco Trust Agent on Windows Operating Systems" for the appropriate installation method.

Step 2 Verify that CTA has been installed and is running by using the "Verifying Cisco Trust Agent Installation" procedure in the installation chapter.

Step 3 Enable logging by using the **clogcli enable** command as described in the "The clogcli Logging Utility" section on page 6-4.

Step 4 On the test machine, start the resident file management application and change the directory to the /etc/opt/CiscoTrustAgent directory on Linux and Mac OS X operating systems or the \Programs\Cisco Systems\CiscoTrustAgent\Logging directory on Windows operating systems. You should see the ctalogd.ini file in that directory.

Step 5 Read the ctalogd.ini file. You should see one section and one entry in the file:

```
[main]
EnableLog=1
```

Step 6 Close the ctalogd.ini file.

Step 7 Continuing to use the clogcli logging utility, specify the logging attributes and logging levels you desire.

Step 8 When you are done configuring the logging attributes, you can distribute the ctalogd.ini file to an individual machine, by storing it in the /etc/opt/CiscoTrustAgent directory on Linux and Mac OS X operating systems or the \Programs\Cisco Systems\CiscoTrustAgent\Logging directory on Windows operating systems. You can also distribute the ctalogd.ini file to many machines when you perform a CTA installation with a custom installation package. These machines will all use the logging attributes you specified in the file.

Sample ctalogd-temp.ini File

CTA logging is disabled by default. The clogcli utility is designed to create the ctalogd.ini file which defines logging parameters for a local installation. There is also a ctalogd-temp.ini file that is shipped with CTA as a template file. Advanced users can edit the file directly if they so choose.

The ctalogd-temp.ini file is delivered to this location on Windows operating systems: \Program Files\Cisco Systems\CiscoTrustAgent\Logging

The ctalogd-temp.ini file is delivered to this location on Linux and Mac OS X operating systems: /etc/opt/CiscoTrustAgent

See Example 6-2 for an example of the ctalogd-temp.ini file.

Example 6-2 Sample ctalogd-temp.ini File

```
; This file contains Cisco Trust Agent log settings.
; To use these settings, save a coy of this file as ctalogd.ini
; and edit that file.
; This file may be used for Linux, Mac OS X, and Windows operating
; systems.

[main]
; This section turns logging on or off and defines the size, age and
; location of CTA log files.
EnableLog=0
; 0 = disable logging
; 1 = enable logging
MaxFileSize=4
MaxDiskSize=50
FileDeleteAge=30
;LogDir=/var/log/CiscoTrustAgent
LogDir=D:\Program Files\Cisco Systems\CiscoTrustAgent\Logging\Logs

[LogLevel]
; This section allows you to set the logging levels
; for various components of the Cisco Trust Agent.
; 0 = disable
; 1 = log critical events only
; 2 = log critical and warning events
; 3 = log critical, warning, and informational events
; 15 = log all events and messages
```

```
PADaemon=3
NetTrans=3
PAPlugin=3
CTAMsg=3
CTAD=3
PEAP=3
EAPTLV=3
EAPSQ=3
PPMgr=3
PSDaemon=3
HostPP=3
CTASC=3
CTASTATE=3
```

CHAPTER 7

Posture Plugins

Posture plugins are the means by which Cisco Trust Agent (CTA) retrieves posture credentials from NAC-compliant applications installed on a client.

Typically, two files comprise a posture plugin. A posture plugin for Windows consists minimally of a dynamic link library ".dll" file and an information ".inf" file. A posture plugin for Linux consists of a shared object ".so" file and an information ".inf" file.

Posture plugins gather posture credentials from NAC-compliant applications. Posture credentials are information about an application that determines the trust the network should have in the security of that application. Posture credentials may include these kinds of attributes: application name, application version, application release date, or proprietary application settings or configurations. Posture credentials can be different for each NAC-compliant application.

Once the posture plugin gathers the posture credentials, it sends them to the CTA. CTA sends the posture credentials to Cisco Secure Access Control Server (ACS) which determines a posture token for each application that provided credentials and an overall posture token for the entire client. The posture token is communicated to the client in a posture notification message.

In the Network Admission Control (NAC) environment, the value of a client's posture token determines the level of network access the client is allowed. When we refer to a client's "posture," we are referring to the value of the client's posture token.

See "Initial Posture Validation Process" section on page 1-2 and "Posture Revalidation Process" section on page 1-4 for a description of the workflows and conditions surrounding these events.

This chapter contains the following sections:

Also see the "Configuring Posture Plugins" section on page 5-13 for information about configuring posture plugins.

Types of Posture Plugins Installed by Default

When CTA is installed these posture plugins are installed by default.

- Host Posture Plugin
- Cisco Trust Agent Posture Plugin

The CTA Scripting Posture Plugin posture plugin is installed if the scripting interface was installed.

Other plugins may also be installed, such as the posture plugin for Cisco Security Agent or a partner in the Network Admission Control program. These plugins are not discussed in this chapter.

Host Posture Plugin

The Host Posture Plugin retrieves basic information about the host and returns it to the ACS.

The default location of the Windows host posture plugin is the \Program Files\Common Files\PostureAgent\Plugins directory. The plugin consists of two files: CiscoHostPP.dll and CiscoHostPP.inf.

The default location of the host posture plugin on Linux and Mac OS X operating systems is the /opt/PostureAgent/Plugins directory. The plugin consists of two files: CiscoHostPP_unix.inf and ciscohostpp.so.

These are the attributes that are returned by the host posture plugin:

Table 7-1 Host Posture Plugin Attributes

Condition name in Posture Validation Policies page of ACS	Attribute Description	Used by these operating system
Cisco:Host:ServicePacks	Windows service packs that have been installed on the client	Windows
Cisco:Host:HotFixes	Windows hot fixes that have been installed on the client	Windows
Cisco:Host:HostFQDN	Machine name	Windows
Cisco:Host:Package	Version and string version of rpm packages	Linux and Mac OS X
Cisco:Host:MACAddress	Machine's MAC address	Linux, Mac OS X, Windows

Note The Host posture plugin only reports Windows operating system hotfixes. It does not report application hotfixes.

MAC Address Information Returned by Host Posture Plugin

The Host Posture Plugin reports basic information about the client running CTA to the ACS. With the release of CTA 2.1, the Host Posture Plugin can now return the MAC address of the client running CTA, provided that the MacAddress attribute has been added to the Posture-Validation Attribute Definition File employed by the ACS CSUtil database utility. (For more information about the ACS CSUtil database utility and the Posture-Validation Attribute Definition File, see the *User Guide for Cisco Secure ACS for Windows Server.*)

The attribute information for MACAddress is below.

```
[attr#n]
vendor-id=9
vendor-name=Cisco
application-id=2
application-name=Host
attribute-id=00009
attribute-name=MACAddress
attribute-profile=in
```

```
attribute-type=string
```

The plugin will return all the MAC addresses available on the client running CTA and combine them into one string; the MAC addresses will be separated by pipes (|). For example, a wireless network card and a wired network card will each return a MAC address.

If you are defining a posture validation rule in ACS based on only one of these MAC addresses, the posture attribute should "contain" the MAC address you are verifying rather than "equal" or "start with" the MAC address you are verifying.

Package Information Retrieved by Host Posture Plugin for Linux Platforms

In addition to the information defined by the host posture plugin attributes, the host posture plugin for Linux and Mac OS X platforms allows you to retrieve the version number of certain packages pre-defined in ACS.

The version number of the package may be expressed in one of these forms:

- A string.
- A representation of the version number in the form of x.x.x.x. In this case, the version numbers is converted to a 4-octet version number using an algorithm.

Here is an example of how CTA returns package version number using each format: Assume there is a posture validation rule from ACS that requests the version number of OpenSSL. When requested as a string, the version number would be returned as a combination of numbers and letters, such as 0.9.7a. When requested as a 4-octet number, the version number that would be return is 0.9.7.97.

Creating posture validation rules on ACS that request package version information as numbers rather than strings allows you to apply operators in the ACS rule such as "greater than or equal to," "greater than," "less than," or "less than or equal to." These operators do not apply to strings. The string format is beneficial when too much information is lost when using the numeric format. For example, a package might have a version "rockie-mnt-rel-Feb", in this case the converted numeric version is reported as "0.0.0.0". In this case, the string version of the package is more meaningful.

When using the Host Posture Plugin on Linux operating systems to retrieve package information, the string value of a package version can be determined in advance of making the ACS rule by running the “rpm -q package-name” command, for example, “rpm -q openssl”. The 4-octet value of a package version is determined from the same output of the “rpm -q package-name” command.

Package Information Retrieved by Host Posture Plugin for Mac OS X Platforms

For Mac OS X, there are two types of applications that are of concern to CTA: system applications which have receipts in /Library/Receipts/ and user applications which are installed in /Applications directory.

System applications are identified by the first level folder name under /Library/Receipts, like "Danish.pkg", "X11SDK.pkg". User applications are identified by the application name under /Applications directory as displayed in Finder. For example, “Firefox”, “DVD\ Player”.

The applications located in the subfolders of /Applications directory can also be queried, in these cases the package name looks like the relative path to /Applications. For example, "Utilities/Disk\ Utility", "Zinio/Zinio\ Reader".

Note White spaces in package names must be escaped with backslash (“\”).

The version information of system applications is parsed out of the Contents/version.plist file under the package's directory under the /Library/Receipts directory. Version information is in the form of "a.b.c.d". The first three fields of version are from the CFBundleShortVersionString key, and the fourth field is from SourceVersion key. For user application packages, the version information is retrieved from the Info.plist file under the Contents/ directory in the application’s directory. We first look for the value of CFBundleShortVersionString key. If this key is not present we will return the value of CFBundleVersion key. If both keys are missing no information will be returned for the package.

Cisco Trust Agent Posture Plugin

By default, CTA installs and registers a posture plugin that provides information about itself. The CTA posture plugin returns information such as CTA’s name, version number, and the name of the operating system on which it runs.

The default location of the Windows CTA posture plugin is the \Program Files\Common Files\PostureAgent\Plugins directory. The plugin consists of two files: ctapp.dll and ctapp.inf.

The default location for the CTA posture plugin on Linux and Mac OS X operating systems is the /opt/PostureAgent/Plugins directory. The plugin consists of two files: ctapp.so and ctapp_unix.inf.

The CTA plugin returns many of the same attributes for all operating systems. You can use the information in table 7-2 when defining rules for the default Cisco Trust Agent plugin in ACS:

Table 7-2 CTA Posture Plugin Attributes and Definitions

Condition name in Posture Validation Policies page of ACS	Attribute Description	Used by this operating system
Cisco:PA: Application-Posture-Assessment	This is the value of the Application-posture token from CTA.	Linux, Mac OS X, Windows
Cisco:PA:Kernel-Version	Kernel version, which is the same as the output of the “uname -r” command.	Linux, Mac OS X,
Cisco:PA:MachinePostureState	Contains the running status of the machine. Linux and Mac OS X return these values: • Booting (ACS value = 1) • Running (ACS value = 2) Windows platforms return these values: • Booting (ACS value = 1) • Running (ACS value = 2) • Logged in (ACS value = 3 See Machine Posture State, page 7-8 for more information on the use of this attribute.	Linux, Mac OS X, Windows.
Cisco:PA:OS-Release	A string that contains the OS Kernel name, version, and hardware platform.	Linux, Mac OS X,

Condition name in Posture Validation Policies page of ACS	Attribute Description	Used by this operating system
Cisco:PA:OS-Type	Contains the name of the operating system running on the client. Linux platforms return these names: • Red Hat Enterprise Linux WS • Red Hat Enterprise Linux ES • Red Hat Enterprise Linux AS Mac OS X returns these names: • Mac OS Panther • Mac OS Tiger Windows platforms return these names: • Windows Server 2003 Enterprise Edition • Windows Server 2003 Web Edition • Windows Server 2003 Standard Edition • Windows XP Home Edition • Windows XP Professional • Windows 2000 Advanced Server • Windows 2000 Server	Linux, Mac OS X, Windows
Cisco:PA:OS-Version	The operating system version number in the format *major*.*minor*.*sustaining*.*build*.	Linux, Mac OS X, Windows
Cisco:PA:PA-Name	Contains the name of the posture agent running on the client. In this case, the name would be Cisco Trust Agent.	Linux, Mac OS X, Windows
Cisco:PA:PA-Version	The Cisco Trust Agent version number in the format *major*.*minor*.*sustaining*.*build*.	Linux, Mac OS X, Windows

Condition name in Posture Validation Policies page of ACS	Attribute Description	Used by this operating system
Cisco:PA: System-Posture-Assessment	This indicates the overall posture token of a client on which CTA runs.	Linux, Mac OS X, Windows
Cisco:PA:User-Notification	Contains an informational message that the Cisco Trust Agent displays to the user on request from the ACS server.	Linux, Mac OS X, Windows

Machine Posture State

Machine posture state is provided by CTA to inform ACS about the status of the machine when it boots. One of the following states can be reported:

- Booting
- Running
- Logged in (Not supported on Linux)

Booting is the initial state when the machine is started. The state is set to running when all services are started. On Linux, running is the final state because there is no simple way to determine if a user has logged in. On Windows, when a user has logged in, the logged in state is available. The machine posture state is set once a user has logged into the machine. If the user logs out of the machine, the state is set back to the running state until another user logs in.

This state is used with ACS to determine the posture of a machine based on the rules set up for different policies. For example, the following policy could be set on ACS.

Rule	Posture Token
Antivirus Enabled = TRUE	Healthy
Antivirus Installed and Cisco Trust Agent Machine State = Booting	Transitional
Antivirus Installed = FALSE	Quarantine

CTA Scripting Posture Plugin

The ctascriptPP retrieves the posture credentials requested by a third party script.

A binary posture plugin consists of two parts, a dynamic link library (.dll) file for Windows systems, or a shared object (.so) file for Linux systems, and a .inf file. The .inf file points to the .dll (or .so) file that retrieves the posture credentials. Typically, these are pairs of files; one .inf file is associated with one .dll or .so file.

A script can be substituted for a binary posture plugin. However, the script still needs the .dll (or .so) file and the .inf file to retrieve the posture credentials. The .inf file supplied by the third party must always point to the ctascriptPP file. In the case of a script, many unique .inf files point to one .dll (or .so) and that is the ctascriptPP file.

See Chapter 9, “Using the Scripting Interface” for more information on the CTA scripting interface.

Plugin Installation and Upgrade

Each NAC-compliant application is responsible for installing its own posture plugin on end systems.

Plugins for Windows environments are installed in this directory:

\Program Files\Common Files\PostureAgent\Plugins\Install

Plugins for Linux and Mac OS X environments are installed in this directory:

/opt/PostureAgent/Plugins/install

When CTA receives a posture request, it scans the PostureAgnt\Plugins\Install directory for new or updated posture plugins. If there are new or updated posture plugins in the PostureAgnt\Plugins\Install directory, CTA performs one of the following actions:

- If the .dll (Windows) or the .so (Linux and Mac OS X) plugin **does not exist** in the PostureAgent\Plugins directory, CTA moves the plugin files from the PostureAgent\Plugins\Install directory to the PostureAgent\Plugins directory.

- If the .dll (Windows) or the .so (Linux and Mac OS X) plugins **does exist** in the PostureAgent\Plugins directory, then CTA checks to see if the plugin, in the PostureAgent\Plugins\Install directory, is newer than the one in the Plugins directory. CTA then moves the newer plugin to the PostureAgent\Plugins directory and overwrites the older one. If the plugin in the PostureAgent\Plugins\Install directory is older than the one in the Plugins directory, CTA deletes it, and continues to use the original plugin.
- If the plugin creates an error during registration, CTA moves the plugin to one of the following directories (if the logging is enabled, the error information is logged):

  Windows:

  ```
  \Program Files\Common Files\PostureAgent\Plugins\Quarantine
  ```

  Linux and Mac OS X:

  ```
  /opt/PostureAgent/Plugins/Quarantine
  ```

Note Quarantined plugins do not participate in posture validation.

You do not need to install CTA before other NAC-compliant applications in order for CTA to make use of their plugins. All plugins are stored in a common directory. When CTA receives a request for posture credentials, it checks the common directory for new plugins before it proceeds to retrieve the posture credentials.

CHAPTER 8

Cisco Trust Agent's Use of Certificates

CTA uses certificates to establish a PEAP and an EAP FAST session with Cisco Secure Access Control Server (ACS). You need to install the ACS root certificate on the client system for this session to be established.

Typically, this certificate is installed as part of a custom Cisco Trust Agent installation package. If it was not installed, CTA provides a the ctacert utility for installing and updating the posture validation server certificate on the client.

If you have installed a supplicant on the client such as the Cisco Secure Services Client, you may perform machine and user authentication using certificates. You can configure CTA for this authentication after the ACS root certificate has been installed.

This chapter contains the following sections:

About The ACS Server Root Certificate

For ACS to establish a secure PEAP or an EAP FAST session with Cisco Trust Agent, you must install the ACS root certificate on the network client. This certificate is either the CA certificate used to validate the server certificate, or a self-signed certificate generated by the ACS server. On Windows platforms, CTA supports PEM wrappered Base-64 or DER encoded binary X.509 certificates. On Linux platforms, CTA supports PEM wrappered Base-64 certificates only.

Note The ACS certificate must have "server authentication" as the certificate purpose for the PEAP session to be created.

Before you begin reviewing this chapter, obtain the ACS root certificate. If ACS uses self-signed certificates, obtain the certificate from the server. (Refer to the *User Guide for Cisco Secure ACS for Windows Server* for information about obtaining the certificate.) If you use a CA certificate, obtain the certificate from your certificate server.

Cisco Trust Agent installs a utility on the local client to help you add, delete, and manage certificates. See "About The ctacert Utility" section on page 8-3 for detailed procedures describing the use of this utility.

About The ctacert Utility

Use the ctacert utility to install, delete, and manage the root certificate used by Cisco Trust Agent for PEAP (EAPoUDP) sessions with ACS or any other certificates you want to install on the client.

The ctacert utility is installed on Linux, Mac OS X, and Windows platforms. The utility's executable file name on Linux and Mac OS X is ctacert. The utility's executable file name on Windows is ctaCert.exe. This chapter refers to the utility generically as "ctacert."

On Windows, the ctaCert.exe utility can accept PEM wrappered Base-64 or DER encoded binary X.509 certificates. On Linux platforms, the ctacert utility only accepts PEM wrappered Base-64 certificates. However, on Linux platforms, the certificates can be converted from DER to PEM formats. See, the "Converting DER Formatted Certificates to PEM Formatted Certificates" section on page 8-11 for the command to perform the conversion.

Installing or Updating Certificates Using the ctacert Utility

The ctacert utility can be used on Linux, Mac OS X, and Windows operating systems to install or update certificates.

Installing or Updating a Certificate on Linux Operating Systems

Step 1 Copy the certificate to the client.

Step 2 Open a terminal window on the network client.

Step 3 At the prompt type either of the following commands and press <**Enter**>:

- ctacert -a */path/cert_name.cer*
- ctacert --add */path/cert_name.cer*

In these examples, */path/cert_name.cer* represents the full path and file name of the certificate.

After the certificate has been installed, you receive the message, "Certificate successfully added to store with Hashed Name *Number*", where *Number* is the numeric Hashed Name of the certificate.

Installing or Updating a Certificate on Mac OS X Operating System

Step 1 Copy the certificate to the client.

Step 2 Open a terminal window.

Step 3 Change the directory to the **/opt/CiscoTrustAgent/bin** directory.

Step 4 At the prompt enter either of these commands and press **<Enter>**.

- sudo ./ctacert -a */path/cert_name.cer*
- sudo ./ctacert --add */path/cert_name.cer*

In these examples, */path/cert_name.cer* represents the full path and file name of the certificate.

After the certificate has been installed, you receive the message, "Certificate successfully added to store with Hashed Name *Number",* where *Number* is the numeric Hashed Name of the certificate.

Installing or Updating a Certificate on Windows Operating Systems

On Windows operating systems, all certificates are stored in the Microsoft Certificate Store. The ctaCert.exe utility only allows you to add certificates to the Microsoft Certificate Store. All other management of certificates is done through Microsoft's Certificate Management interface.

This is the /add command syntax for ctaCert.exe:

```
ctaCert.exe /ui {2 | 3| 4 | 5} /add "cert_path" /store "cert_store"
```

Command Parameters

Table 8-1 describes the command parameters for the ctaCert utility.

Table 8-1 ctaCert Utility Command Parameters

Parameter	Description
/ui	Specifies silent or verbose install. Accepts the following values: • **2** or **3**—Silent installation. • **4** or **5**—Full user interaction installation. Any other value entered is treated as full user interaction.
/add	Specifies the full path to the certificate being added. You can also specify *.cer to all certificates in the specified directory, for example: `c:\My_Certs\*.cer`.
/store	Specifies the system certificate store. Specifying **Root** stores the certificate in the Trusted Root Certification Authorities store.

To install a certificate using the ctaCert.exe utility, follow this procedure:

Step 1 Copy the certificate to the network client.

Step 2 Open a command prompt on the network client.

Step 3 Change directory to the location of the ctaCert.exe utility. By default, the location is `C:\Program Files\Cisco Systems\CiscoTrustAgent\`.

Step 4 At the prompt, type the following and press <**Enter**>.

`ctaCert.exe /ui x /add C:\path\cert_name.cer /store Root`

Where **/ui x** specifies the level of user interaction and where ***C:\path\cert_name.cer*** is the full path and file name of the certificate.

The certificate is added to the Trusted Root Certification Authorities store on the network client.

Listing Certificates in the Certificate Store

The ctacert utility can be used on Linux and Mac OS X to list the certificates in the client certificate store. Use the Microsoft's Certificate Management interface to perform this task on Windows operating systems.

Listing Certificates in the Certificate Store on Linux Operating Systems

Step 1 Open a terminal window on the network client.

Step 2 From any prompt enter either of these commands and press **<Enter>**.

- ctacert -l
- ctacert --list

This command displays the hashed file name, certificate version, signature algorithm, subject/issuer name, validity period, and MD5 fingerprint information. Output pertaining to different certificates are separated by a string of dashes.

Example 8-1 ctacert --list command output on Linux

```
#ctacert --list
hashed file name: 814661db.0
Version: 3 (0x2)
Serial Number: 0 (0x0)
```

```
Signature Algorithm: md5WithRSAEncryption
Issuer: O=Cisco Systems, Inc., CN=Stress
Validity
Not Before: Aug  7 11:38:06 2002 GMT
Not After : Aug 20 05:09:50 2048 GMT
Subject: O=Cisco Systems, Inc., CN=Stress
MD5 Fingerprint=13:5A:A9:B5:98:DE:78:F5:1A:7E:27:FA:E0:8B:1D:D7
------------------------------------------------------------
```

Listing Certificates in the Certificate Store on Mac OS X Operating System

Step 1 Open a terminal window on the network client.

Step 2 Change the directory to /opt/CiscoTrustAgent/bin directory.

Step 3 At the prompt enter either of these commands and press **<Enter>**.

- sudo ./ctacert -l
- sudo ./ctacert --list

Step 4 When prompted, type the root user's password.

This command displays the hashed file name, certificate version, signature algorithm, subject/issuer name, validity period, and MD5 fingerprint information. Output pertaining to different certificates are separated by a string of dashes.

Example 8-2 ctacert --list command output on Mac OS X

```
Hashed Name: 5e8a8166.0
Certificate:
    Data:
      Version: 3 (0x2)
      Serial Number: 1 (0x1)
      Signature Algorithm: sha1WithRSAEncryption
      Issuer: CN=Cisco Systems
      Validity
        Not Before: Jul 19 15:12:24 2006 GMT
        Not After : Jul 19 15:12:24 2007 GMT
      Subject: CN=Cisco Systems
        X509v3 extensions:
          X509v3 Basic Constraints:
            CA:TRUE
```

```
X509v3 Key Usage:
  Digital Signature, Key Encipherment, Key Agreement, Certificate Sign
X509v3 Subject Key Identifier:
  B5:79:DE:6A:C7:42:47:25:42:BC:68:43:93:04:69:2E:9B:08:0E:64
X509v3 Extended Key Usage:
  TLS Web Server Authentication
Netscape Cert Type:
  SSL Server
```

Deleting Certificates from the Certificate Store

The ctacert utility can be used on Linux and Mac OS X to delete certificates in the client certificate store. Use the Microsoft's Certificate Management interface to perform this task on Windows operating systems.

Deleting a Certificate from the Certificate Store on Linux Operating Systems

Step 1 Open a terminal window on the network client.

Step 2 From any prompt, enter either of these commands and press **<Enter>**.

- `ctacert -d` *HASHED-CERT-FILENAME*
- `ctacert --delete` *HASHED-CERT-FILENAME*

The `hashed-cert-file-name` can be obtained from the `ctacert --list` output. In Example 8-1, the hashed certificate file name is `814661db.0`.

For example:

ctacert -d 814661db.0

Note The hashed file name for a certificate may change when other certificates are removed.

Step 3 When prompted, type **y** to confirm your desire to delete the certificate.

Deleting a Certificate from the Certificate Store on Mac OS X Operating System

Step 1 Open a terminal window on the network client.

Step 2 Change the directory to /opt/CiscoTrustAgent/bin directory.

Step 3 At the prompt enter either of these commands and press **<Enter>**.

- `sudo ./ctacert -d` *HASHED-CERT-FILENAME*
- `sudo ./ctacert --delete` *HASHED-CERT-FILENAME*

The `hashed-cert-file-name` can be obtained from the `ctacert --list` output. In Example 8-1, the hashed certificate file name is `814661db.0`.

For example:

```
# sudo ./ctacert -d 814661db.0
```

Step 4 When prompted, type the root user's password.

Step 5 When prompted, type **y** to confirm your desire to delete the certificate.

Note The hashed file name for a certificate may change when other certificates are removed.

Clearing Certificates from the Certificate Store

The ctacert utility can be used on Linux and Mac OS X to clear all the certificates in the client certificate store. Use the Microsoft's Certificate Management interface to perform this task on Windows operating systems.

Clearing All Certificates from the Certificate Store on Linux Operating Systems

Step 1 Open a terminal window on the network client.

Step 2 From any prompt enter either of these commands:

- `ctacert -c`
- `ctacert --clear`

Clearing All Certificates from the Certificate Store on Mac OS X Operating Systems

Step 1 Open a terminal window on the network client.

Step 2 Change the directory to /opt/CiscoTrustAgent/bin directory.

Step 3 At the prompt enter either of these commands and press **<Enter>**.

- `sudo ./ctacert -c`
- `sudo ./ctacert --clear`

Step 4 When prompted, type the root user's password.

Step 5 When prompted, type **y** to confirm your desire to clear the certificate store.

Distinguished Name Matching

When using CA certificates to validate your Cisco Secure ACS server certificate, you can implement additional security using distinguished name (DN) matching to validate the server certificate. This prevents other servers or processes that may be using the same root certificate from gaining a trust relationship with the network client.

DN matching occurs at the end of the TLS handshake, after the certificate chain is built. Invalid DN matching rules are ignored, but logged. Matched rules are logged. Failed rules are not logged.

DN matching rules are configured in the [ServerDNVerification] section of the ctad.ini configuration file. If the [ServerDNVerification] section does not exist, or if there are no rules configured, then the DN matching feature is disabled and the system accepts connections with any validated certificate chain. Otherwise, the server certificate must match one of the DN matching rules for the connection to continue.

If the configuration file does not exist, the default values for these settings are used. To change the value for any of these items, you need to create the configuration file and save it to the appropriate location.

Any changes made to the [ServerDNVerification] section of the ctad.ini configuration file are detected and are implemented by Cisco Trust Agent the next time DN matching occurs.

To learn more about configuring Domain Name matching in the ctad.ini file, see "Certificate Distinguished Name Matching" section on page 5-25.

Converting DER Formatted Certificates to PEM Formatted Certificates

On Linux and Mac OS X platforms, CTA supports PEM wrappered Base-64 certificates but not DER encoded binary X.509 certificates. However DER certificates can be converted to PEM certificates using the following procedure. (For the sake of this procedure, assume that the name of the DER formatted certificate is **ca.der**.)

Note This procedure requires that OpenSSL is installed on the computer.

Step 1 Log in to the Linux or Mac computer as the root user.

Step 2 Open a terminal window.

Step 3 At the prompt, type the following:

```
openssl x509 -inform DER -outform PEM -in ca.der -out ca.pem
```

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.

APPENDIX A

ctastat Diagnostic Tool

CTA provides a utility for administrators to retrieve diagnostic information from CTA. This information includes a snapshot of the communication taking place between ACS and CTA. You can view posture information using this utility, including the last time a posture check was performed.

If you are troubleshooting an issue on a system, Cisco support (TAC) may ask you to run ctastat to retrieve information. You must be working locally on the system in order to run ctastat.

This appendix contains these sections:

Running the ctastat Utility

The ctastat utility runs using a command line interface (CLI). The utility is available for both Windows and Linux operating systems.

Running ctastat on a Linux Operating System

Step 1 Open a terminal window.

Step 2 Change the directory to the /opt/CiscoTrustAgent/sbin directory.

Step 3 At the prompt type **`./ctastat`** and press <**Enter**>. ctastat displays its output in the terminal window.

Running ctastat on a Mac OS X Operating System

Step 1 Open a terminal window.

Step 2 Change the directory to the /opt/CiscoTrustAgent/sbin directory.

Step 3 At the prompt type **`./ctastat`** and press <**Enter**>. ctastat displays its output in the terminal window.

Running ctastat on a Windows Operating System

Step 1 Open a command prompt window.

Step 2 Change the directory to the Program Files\CiscoSystems\CiscoTrustAgent directory.

Step 3 At the prompt type, **`ctastat.exe`** and press <**Enter**>. ctastat displays its output in the terminal window.

ctastat Utility Output

The output from the ctastat command provides general information about CTA, session information which describes the communication between CTA and ACS, and plugin information which summarizes the status of the posture plugins running on the system.

For an example of a ctastat output see, Example A-1 on page A-4 and Example A-2 on page A-5.

General CTA Information

The general CTA information provided in ctastat output is the local time the statistics were collected and the CTA version running on the system. Example A-1 on page A-4 shows that the local time ctastat run was Friday, September 16, 2005 at 15:49:06. The CTA version is 2.0.0.26.

Session Information

The session information describes the communication between CTA and ACS.Table A-1describes the fields in the Session Information area of the output.

Table A-1 Session Information fields from ctastat output

Field name	Description
Session Number (Hex)	Session identification number
Session Type	Indicates communication using 802.1x or EAP over UDP (EoU) protocol.
IP Address	IP address and port id of Router/Switch when using EAP over UDP protocol.
Local MAC Address	MAC addresses of local network card when using 802.1x protocol.
Remote MAC Address:	MAC addresses of Router (or Switch) when using 802.1x protocol.
System Posture Token Value	Last reported posture token of overall system.

Table A-1 Session Information fields from ctastat output (continued)

Field name	Description
Received on	Time last system posture token was received.
Total Postures Received	Number of posture requests received.
Last SQ Response	Value of last status query response.
Plugin Vendor/Application:	Identifying number of plugin vendor and application. Value correlates with the information in the Plug-ins section of output.
Application Posture Token Value	Posture of the application
Received	The time the application posture token was received.
Posture Request last received	The last time the application posture credentials were requested.
Length of last response to posture request	Length of response to posture credential request measured in bytes.
Sent	The time the posture credentials were received.

Plugins Information

In the Plugins section of the ctastat output, the product vendor and application ID are listed. These numbers correlate with the information in the PluginVendor/Application field in the Session Information output.

ctastat Utility Sample Output

Example A-1 ctastat output for 802.1x connection between CTA and Cisco Secure ACS

CTA Statistics Reporting Tool

```
Cisco Trust Agent Statistics
Current Time: Fri Sep 16 15:49:06 2005
CTA Version: 2.0.0.26
```

```
Session Information
    Session Number (Hex): 02000000
        Session Type: 802.1X
            Local MAC Address: 0050DA2C7EBD
            Remote MAC Address: 00115DBE2BFF
        System Posture Token Value: Healthy
            Received on: Fri Sep 16 15:48:14 2005
            Total Postures Received: 2
        Plugin Vendor/Application: 9/1
            Application Posture Token Value: Healthy
                Received: Fri Sep 16 15:48:14 2005
            Posture Request last received: Fri Sep 16 15:48:14 2005
                Length of last response to Posture Req: 20
                Sent: Fri Sep 16 15:48:14 2005

Plug-ins:
    Vendor: Cisco Systems
        Application ID: 1
            Status: Operational
        Application ID: 2
            Status: Operational
```

Example A-2 ctastat output for EAP over UDP connection between CTA and Cisco Secure ACS

```
CTA Statistics Reporting Tool

Cisco Trust Agent Statistics
Current Time: Tue Sep 27 19:11:18 2005
CTA Version: 2.0.0.26

Session Information
    Session Number (Hex): 01000000
        Session Type: EOU
            IP Address: 8.8.0.1:21862
        System Posture Token Value: Healthy
            Received on: Mon Sep 26 11:42:14 2005
            Total Postures Received: 12
        Last SQ Response was "No Status Change"
        Plugin Vendor/Application: 9/1
            Application Posture Token Value: Healthy
                Received: Mon Sep 26 11:42:14 2005
            Posture Request last received: Mon Sep 26 11:42:14 2005
                Length of last response to Posture Req: 49
                Sent: Mon Sep 26 11:42:14 2005
        Plugin Vendor/Application: 9/2
            Posture Request last received: Mon Sep 26 11:42:14 2005
```

```
                    Length of last response to Posture Req: 39
                    Sent: Mon Sep 26 11:42:14 2005

Plug-ins:
    Vendor: Cisco Systems
        Application ID: 1
            Status: Operational
        Application ID: 2
            Status: Operational
```

APPENDIX B

Alternate Methods of Installing CTA

Cisco Trust Agent (CTA) can be installed on its own or it can be bundled with other Cisco or third party products. This appendix describes installing CTA using CSA MC 5.2.

Installing CTA 2.1 Using CSA MC 5.2

Your enterprise can use Cisco Security Agent (CSA) 5.2 to distribute and install CTA 2.1.

Note CSA MC 5.2 can only distribute CTA 2.1 on Windows operating systems. See "System Requirements for Installation" section on page 4-2 for the versions of Windows operating systems on which CTA 2.1 runs.

To distribute and install CTA by using CSA, follow these procedures:

Step 1 Perform the "Copying Cisco Trust Agent Installer Files" procedure in the *Installing Management Center for Cisco Security Agents* guide.

Step 2 Perform the "Creating CSA Agent Kits" procedure in the *Using Management Center for Cisco Security Agents Guide*.

Note There are no installation files bundled by default with CSA distributions.

APPENDIX C

Open Source License Acknowledgement

Cisco Trust Agent uses third-party, open-source software. The following acknowledgements pertain to this software license:

OpenSSL/Open SSL Project

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/).

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com).

This product includes software written by Tim Hudson (tjh@cryptsoft.com).

License Issues

The OpenSSL toolkit stays under a dual license, i.e. both the conditions of the OpenSSL License and the original SSLeay license apply to the toolkit. See below for the actual license texts. Actually both licenses are BSD-style Open Source licenses. In case of any license issues related to OpenSSL please contact openssl-core@openssl.org.

OpenSSL License:

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgment: “This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/)”.
4. The names “OpenSSL Toolkit” and “OpenSSL Project” must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact openssl-core@openssl.org.
5. Products derived from this software may not be called “OpenSSL” nor may “OpenSSL” appear in their names without prior written permission of the OpenSSL Project.
6. Redistributions of any form whatsoever must retain the following acknowledgment:

   “This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/)”.

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT “AS IS”' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS 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 THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com).

Original SSLeay License:

Copyright © 1995-1998 Eric Young (eay@cryptsoft.com). All rights reserved.

This package is an SSL implementation written by Eric Young (eay@cryptsoft.com).

The implementation was written so as to conform with Netscapes SSL.

This library is free for commercial and non-commercial use as long as the following conditions are adhered to. The following conditions apply to all code found in this distribution, be it the RC4, RSA, lhash, DES, etc., code; not just the SSL code. The SSL documentation included with this distribution is covered by the same copyright terms except that the holder is Tim Hudson (tjh@cryptsoft.com).

Copyright remains Eric Young's, and as such any Copyright notices in the code are not to be removed. If this package is used in a product, Eric Young should be given attribution as the author of the parts of the library used. This can be in the form of a textual message at program startup or in documentation (online or textual) provided with the package.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement:

   "This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)".

   The word 'cryptographic' can be left out if the routines from the library being used are not cryptography-related.

4. If you include any Windows specific code (or a derivative thereof) from the apps directory (application code) you must include an acknowledgement: "This product includes software written by Tim Hudson (tjh@cryptsoft.com)".

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS 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 THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The license and distribution terms for any publicly available version or derivative of this code cannot be changed. i.e. this code cannot simply be copied and put under another distribution license [including the GNU Public License].

Info-ZIP

This is version 2004-May-22 of the Info-ZIP copyright and license. The definitive version of this document should be available at ftp://ftp.info-zip.org/pub/infozip/license.html indefinitely.

```
Copyright (c) 1990-2004 Info-ZIP. All rights reserved.

For the purposes of this copyright and license, "Info-ZIP" is defined as the following set
of individuals:

Mark Adler, John Bush, Karl Davis, Harald Denker, Jean-Michel Dubois, Jean-loup Gailly,
Hunter Goatley, Ian Gorman, Chris Herborth, Dirk Haase, Greg Hartwig, Robert Heath,
Jonathan Hudson, Paul Kienitz, David Kirschbaum, Johnny Lee, Onno van der Linden, Igor
Mandrichenko, Steve P. Miller, Sergio Monesi, Keith Owens, George Petrov, Greg Roelofs,
Kai Uwe Rommel, Steve Salisbury, Dave Smith, Christian Spieler, Antoine Verheijen, Paul
von Behren, Rich Wales, Mike White
```

This software is provided "as is," without warranty of any kind, express or implied. In no event shall Info-ZIP or its contributors be held liable for any direct, indirect, incidental, special or consequential damages arising out of the use of or inability to use this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. Redistributions of source code must retain the above copyright notice, definition, disclaimer, and this list of conditions.

2. Redistributions in binary form (compiled executables) must reproduce the above copyright notice, definition, disclaimer, and this list of conditions in documentation and/or other materials provided with the distribution. The sole exception to this condition is redistribution of a standard UnZipSFX binary (including SFXWiz) as part of a self-extracting archive; that is permitted without inclusion of this license, as long as the normal SFX banner has not been removed from the binary or disabled.

3. Altered versions--including, but not limited to, ports to new operating systems, existing ports with new graphical interfaces, and dynamic, shared, or static library versions--must be plainly marked as such and must not be misrepresented as being the original source. Such altered versions also must not be misrepresented as being Info-ZIP releases--including, but not limited to, labeling of the altered versions with the names "Info-ZIP" (or any variation thereof, including, but not limited to, different capitalizations), "Pocket UnZip," "WiZ" or "MacZip" without the explicit permission of Info-ZIP. Such altered versions are further prohibited from misrepresentative use of the Zip-Bugs or Info-ZIP e-mail addresses or of the Info-ZIP URL(s).

4. Info-ZIP retains the right to use the names "Info-ZIP," "Zip," "UnZip," "UnZipSFX," "WiZ," "Pocket UnZip," "Pocket Zip," and "MacZip" for its own source and binary releases.

INDEX

D

E

H

I

L

N

O

P

R

S

T

U

W