RSA Validation Manager 3.1
Command Line Reference Manual

The Security Division of EMC

Contact Information

Go to the RSA corporate web site for regional Customer Support telephone and fax numbers**:** **www.rsa.com**

Trademarks

RSA and the RSA logo are registered trademarks of RSA Security Inc. in the United States and/or other countries. For the most up-to-date listing of RSA trademarks, go to **www.rsasecurity.com/legal/trademarks_list.pdf**. EMC is a registered trademark of EMC Corporation. All other goods and/or services mentioned are trademarks of their respective companies.

License agreement

This software and the associated documentation are proprietary and confidential to RSA, are furnished under license, and may be used and copied only in accordance with the terms of such license and with the inclusion of the copyright notice below. This software and the documentation, and any copies thereof, may not be provided or otherwise made available to any other person.

No title to or ownership of the software or documentation or any intellectual property rights thereto is hereby transferred. Any unauthorized use or reproduction of this software and the documentation may be subject to civil and/or criminal liability.

This software is subject to change without notice and should not be construed as a commitment by RSA.

Third-party licenses

This product may include software developed by parties other than RSA. The text of the license agreements applicable to third-party software in this product may be viewed in the **thirdpartylicenses.pdf** file.

This product includes software developed by The Apache Software Foundation (**www.apache.org/**).

Note on encryption technologies

This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulations should be followed when using, importing or exporting this product.

Distribution

Limit distribution of this document to trusted personnel.

RSA notice

Portions of this product use technologies patented under U.S. patent numbers 5,922,074 and 6,249,873.

First printing: October 2007

Contents

Preface 5
About This Guide 5
RSA Validation Manager Documentation 5
Related Documentation 5
Getting Support and Service 6
Before You Call Customer Support 6

Chapter 1: RSA Validation Manager and the Command Line 7
Introducing RSA Validation Manager 7
RSA Validation Manager Administration Utility 8
Command Form 8
General Options 9
Configuration File 10
Administration Script File 11
Starting and Stopping RSA Validation Manager 11
Passphrases 12
Clustering 13

Chapter 2: Configuring RSA Validation Manager for the First Time 15
Creating an OCSP Signer 15
Obtaining Certificates to Sign Responses 16
Creating a Status Source 18
Revocation List-Based Status Source 18
OCSP-Based Status Source 19
LDAP-Based Status Source 19
Adding CAs 20
Configuring Revocation Lists 20
Restarting Validation Manager 21

Chapter 3: System Commands 23
Commands 23

Chapter 4: CA Commands 35
Commands 35

Chapter 5: OCSP Signer Commands 57
Commands 57

Chapter 6: Status Source Commands 69
Commands 69

Glossary 89

Acronyms 101

Index 103

Preface

About This Guide

This guide describes the use of the RSA Validation Manager command line administration utility, vmadmin. It is intended for administrators and other trusted personnel. Do not make this guide available to the general user population.

RSA Validation Manager Documentation

For more information about Validation Manager, see the following documentation:

Readme. Provides information about what is new and changed in this release, as well as workarounds for known issues. The latest version of the *Readme* is available from RSA SecurCare Online: **https://knowledge.rsasecurity.com**.

Getting Started. Lists what the kit includes (all CDs, diskettes, licenses and documentation), specifies the location of documentation on the CD, and lists RSA Customer Support web sites.

Installation Guide. Describes detailed procedures on how to install Validation Manager.

Administrator's Guide. Provides information for your administrators about how to configure and administer Validation Manager.

Command Line Reference Guide. Provides information on using the command line utility available in Validation Manager.

RSA Validation Manager Help. Describes day-to-day administration tasks performed in the administration user interface. To view Help, click the **Help** tab on the administration user interface.

Related Documentation

For more information about products related to RSA Validation Manager, see the following:

RSA Certificate Manager documentation set. The full documentation set for RSA Certificate Manager is included in the **\Documentation** directory of the RSA Certificate Manager CD.

RSA Secured Partner Solutions directory. RSA has worked with a number of manufacturers to qualify products that work with RSA products. Qualified third-party products include virtual private network (VPN) and remote access servers (RAS), routers, web servers, and many more. To access the directory, including implementation guides and other information, go to **www.rsasecured.com**.

Getting Support and Service

RSA SecurCare Online	https://knowledge.rsasecurity.com
Customer Support Information	www.rsa.com/support
RSA Secured Partner Solutions Directory	www.rsasecured.com

RSA SecurCare Online offers a knowledgebase that contains answers to common questions and solutions to known problems. It also offers information on new releases, important technical news, and software downloads.

The RSA Secured Partner Solutions Directory provides information about third-party hardware and software products that have been certified to work with RSA products. The directory includes Implementation Guides with step-by-step instructions and other information about interoperation of RSA products with these third-party products.

Before You Call Customer Support

Make sure you have direct access to the computer running the RSA Validation Manager software.

Please have the following information available when you call:

- ❑ Your RSA Customer/License ID. You can find this number on the license certificate that shipped with the product or on the label of the license CD, as applicable.
- ❑ RSA Validation Manager software version number.
- ❑ The make and model of the machine on which the problem occurs.
- ❑ The name and version of the operating system under which the problem occurs.

1 RSA Validation Manager and the Command Line

This chapter describes how to control and configure RSA Validation Manager through the command line. For information on controlling and configuring Validation Manager using a web browser, see the *Administrator's Guide*.

This chapter includes the following topics:

- Introducing Validation Manager
- The Validation Manager administration utility
- General options
- The configuration and administration scripts file
- Starting and stopping Validation Manager
- Clustering

Introducing RSA Validation Manager

Validation Manager provides certificate status information to PKI applications. Certificate status information can be acquired for certificates issued by one or more certificate authorities (CAs). Validation Manager processes certificate status requests from Online Certificate Status Protocol (OCSP) clients, generates a response, and then returns it to the requestor. Validation Manager status responses are signed by a keypair that can be certified by a trusted CA. This ensures that OCSP clients are able to validate and trust status responses.

This reference manual describes the functionality of Validation Manager commands that are used to control and configure Validation Manager. These commands are divided into four sections:

- System commands
- CA commands
- OCSP signer commands
- Status source commands

This reference manual also describes the configuration file, administration script file, starting and stopping Validation Manager, and configuring Validation Manager for the first time after installation.

Validation Manager commands are accessible on both Windows and UNIX systems; however, the **startupVM** and **shutdownVM** commands apply only to UNIX.

RSA Validation Manager Administration Utility

When controlling and configuring Validation Manager using the command line, access to all functionality is through the administration utility, vmadmin.

To control Validation Manager at the command line, your working directory must be the ***installed-dir*/Util** directory.

Command Form

At the command line, an administrator types the command **vmadmin** followed by a Validation Manager command, general options, and parameters.

A vmadmin command consists of five command elements in the following format:

```
vmadmin [general options] command [[command options]
[command parameters]...]
```

These five command elements are:

vmadmin	The name of the administration utility
[*general options*]	Options that are independent of any Validation Manager command
command	The name of the administrative command
[*command options*]	Options specific to the named command
[*command parameters*]	Parameters specific to the named command

For example, to set the default signer for a CA, the prototype is:

```
SetDefaultSigner Nickname
```

which, for a CA with the nickname CA1, translates at the command line to:

```
vmadmin SetDefaultSigner CA1
```

Note: All data that the user provides (for example, *Nickname*) is case sensitive.

Commands are not case sensitive and are capitalized only to enhance readability. Spaces can be used in commands; however, command parameter values that contain spaces must be enclosed in quotation marks.

Important: The following special characters cannot be used for any option:
< > " &

For commands that output certificates, PKCS #10 certificate requests, or revocation lists, output can be redirected to a file.

When a command is entered, changes take effect immediately. You do not need to restart Validation Manager. However RSA recommends that you restart Validation Manager after you create a new OCSP signer.

When running a command, if you do not specify a general option, vmadmin uses general options (if any) set in the default configuration file, **Default.conf**.

When controlling Validation Manager by way of the command line, options entered at the command line override any settings in the configuration file.

General Options

All general options have a short form and a long form. You can use either form. The short form is preceded by a single dash (-), while the long form is preceded by two dashes (--). For example, -h and --help.

The [*general options*] element of a vmadmin command may contain one or more of the following:

-cf *ConfigFileName* --config-file *ConfigFileName*	Instructs vmadmin to read configuration data from the named file. Settings specified on the command line override any configuration file settings. (Settings specified in a configuration file override any default configuration settings.) If unspecified, Validation Manager uses the default configuration settings.
-h --help	Displays Help for vmadmin commands. If used before a command, detailed information for the specified command is displayed. If used without a command, a summary of all available commands is displayed. All other options are ignored.
-v --version	Displays the version number of the vmadmin utility. All other options are ignored.
-ho *HostName* --host *HostName*	Specifies the domain name service (DNS) hostname of the Validation Manager OCSP Server that the vmadmin utility controls. If unspecified, the default is **localhost**.
-p *Port* --port *Port*	Specifies the transmission control protocol (TCP) port number of the Validation Manager OCSP Server that the vmadmin utility controls. The default is **1221**.
-tkc *KeyCertFileName* --tls-keyCert *KeyCertFileName*	Specifies the file containing the private key and Transport Layer Security (TLS) client certificate that the vmadmin utility uses in its TLS session with the Validation Manager OCSP Server. If unspecified, the default is **./admin.p12**.

-tp *Passphrase* --tls-password *Passphrase*	Specifies the passphrase that the vmadmin utility uses to unlock the TLS private key file. If unspecified, and if the TLS private key file requires a passphrase, the vmadmin utility prompts for the passphrase.
-tca *CACertFileName* --tls-ca *CACertFileName*	Specifies the file containing the Validation Manager installation System CA certificate the vmadmin utility uses to authenticate the Validation Manager OCSP Server.
-s *ScriptFileName* --script *ScriptFileName*	Specifies a file containing an administration script. Any command specified on the command line is ignored.

Configuration File

The configuration file stores configuration settings for the vmadmin utility. This universal transformation format (UTF-8) file consists of Option/ Value pairs, one per line, of general options from the [*general options*] element of the vmadmin commands.

Each line consists of an option name (*OptionName*), an equal sign (=), and an option value (*Value*):

```
OptionName=Value
```

where

OptionName	The long name of the option, without the preceding two dashes. For example, `host`, not `--host` or `-ho`.
Value	The same as the command line option.

For example,

```
port=1221
```

Default configuration settings are specified in the **Default.conf** file (located in the ***installed-dir*\Util** directory).

If any setting in the configuration file results in an error, all settings in the file are ignored.

The number sign (#) signals a comment. Any text after the number sign, to the end of the line, is ignored.

Administration Script File

The administration script file is a list of administration commands that can be executed consecutively. The general option -s or --script directs vmadmin to run this file. Commands are listed in the following format:

```
command [[command options] [command parameters]...]
```

For example,

```
SetConfigurationRefreshTime 32
```

Each line contains a single vmadmin command with its related options and parameters. Either double quotation marks (“ ”) or single quotation marks (‘ ’) must be used to enclose a command parameter value that contains spaces. Command parameter values cannot contain quotation marks.

The number sign (#) designates a comment. Any text after the number sign, to the end of the line, is ignored.

Each command is executed in turn, as if it was individually invoked at the command line. If any command results in an error, the commands before the error are executed, while the remaining commands in the file are ignored.

Starting and Stopping RSA Validation Manager

Each startup and shutdown event is logged to the system log file, **syslog**, on UNIX platforms or to the Event Viewer on Windows platforms. You can also configure Validation Manager to log startup and shutdown of the OCSP Server to the audit log file.

On UNIX

To start and stop Validation Manager when installed on the UNIX platform, locate the command line shell scripts in the ***installed-dir*/Util** directory (the same directory where the vmadmin utility is located). The scripts to start and stop Validation Manager are:

- startupVM
- shutdownVM

To start Validation Manager:

1. Change the current directory to ***installed-dir*/Util**.
2. Type:

   ```
   startupVM
   ```

   If Validation Manager is running, it is restarted.

To stop Validation Manager:

1. Change the current directory to ***installed-dir*/Util**.
2. Type:

```
shutdownVM
```

On Windows

To start Validation Manager:

1. From the Windows Control Panel, click **Administrative Tools > Services**.
2. In the Services list, right-click the RSA Validation Manager GUI Server service, and click **Start** in the pop-up menu.
3. If prompted, enter the passphrases protecting the system services, smart cards, or OCSP signer private keys.

   For more information on passphrases, see the following section, "Passphrases."

   The Passphrase Prompting service is also started.
4. In the Services list, right-click the Validation Server service, and click **Start** in the pop-up menu.
5. Close the **Services** dialog box.

To stop Validation Manager:

1. From the Windows Control Panel, click **Administrative Tools > Services**.
2. In the Services list, right-click the Validation Server service, and click **Stop** in the pop-up menu.
3. In the Services list, right-click the GUI Server service, and click **Stop** in the pop-up menu.

   The Prompting service is also stopped.
4. Close the **Services** dialog box.

Passphrases

When Validation Manager starts, the system may prompt you for a number of passphrases:

System passphrase. The passphrase as set during installation.

nCipher smart card passphrase. If the host computer system has an operational nCipher unit with an inserted operator card, the system prompts you for the PIN of the inserted smart card.

OCSP signer key passphrase. If an OCSP signer is created with a passphrase-protected private key, the system prompts you for this passphrase.

Validation Manager does not prompt you for the same passphrase more than once. For example, if the nCipher smart card, system passphrase, and OCSP signer key passphrase are all abcd1234, you only need to provide the passphrase once.

Clustering

In a case where multiple Validation Manager installations are clustered, commands do not execute if attempted at a secondary node. If a command is attempted at a secondary node, the command fails and an error is returned. This error message provides you with the primary node hostname.

2 Configuring RSA Validation Manager for the First Time

This chapter describes how to configure RSA Validation Manager following installation, to bring it into a useful, functioning state. This chapter describes:

- Creating an OCSP signer
- Creating a status source
- Importing a CA
- Configuring revocation lists

Creating an OCSP Signer

You must create a default OCSP signer to sign OCSP certificate status responses returned to the OCSP client.

Important: To configure Validation Manager, it must be started and the OCSP Server state must be **true**.

To create an OCSP signer:

Run the **CreateSigner** command.

When you create the default OCSP signer, consider the following:

Key size. This is the size of the keypair that signs all certificate status responses. The larger the key size the more processing time it takes to sign OCSP certificate status responses. The default size is **1024** bits.

Cryptographic provider. This is the type of keypair used: **nCipher**, **pkcs11**, or **Software**. Using **nCipher** or **pkcs11** reduces the OCSP certificate status response time and is more secure than **Software**. **nCipher** can only be used if an nCipher hardware security module (HSM) has been installed. **pkcs11** must be configured in the configuration file. For more information about HSMs, see the *Installation Guide*. If **Software** is selected, Validation Manager generates an OCSP signer keypair.

Passphrase. If **Software** is selected for keypairs, you must specify a passphrase to protect the OCSP signer private key. **nCipher** uses its own PIN.

Nickname. The OCSP signer must have a unique nickname of unicode characters.

For a complete description of **CreateSigner** command, see "CreateSigner" on page 59.

The following vmadmin command creates a 1024-bit software-based OCSP signer named MySigner with a passphrase of abcd1234:

```
vmadmin CreateSigner -keysize 1024 -crypto Software
-passphrase abcd1234 MySigner
```

The following vmadmin command creates a 2048-bit nCipher-based OCSP signer named nCipherSigner:

```
vmadmin CreateSigner -keysize 2048 -crypto nCipher
nCipherSigner
```

You do not need to supply a passphrase to create an nCipher-based OCSP signer, as the nCipher token PIN is entered during the startup of Validation Manager.

Note: If defaults are used, you do not need to enter them at the command line. For example, to use the software cryptographic provider, you do not have to enter the parameter **-crypto Software** for **Software** to be selected as the cryptographic provider.

Obtaining Certificates to Sign Responses

All responses to certificate status queries by an OCSP client are signed by Validation Manager using a self-signed certificate or a CA-issued certificate. When an OCSP signer is created, Validation Manager creates a self-signed certificate to sign certificate status responses. To use a CA-issued certificate for signing OCSP responses to an OCSP client, you must make a request to the CA and import the CA-issued certificate into Validation Manager.

Using Self-Signed Certificates

When an OCSP signer is created, Validation Manager automatically creates a self-signed certificate to sign responses. You can obtain this self-signed certificate to make it available to OCSP clients who want it to verify responses from Validation Manager.

The following vmadmin command retrieves the Base64-encoded self-signed certificate for the OCSP signer named MySigner:

```
vmadmin GetCert -format pem "MySigner (Self)"
```

Important: If the nickname of the certificate contains spaces, it must be enclosed in quotation marks.

All OCSP signer self-signed certificates use the nickname ***Signer Nickname*** **(Self)**, where *Signer Nickname* is the nickname of the OCSP signer.

Using a CA-Issued OCSP Signer Certificate

To use a CA-issued certificate for signing OCSP responses to an OCSP client, you must make a request to the CA and import the CA-issued certificate into Validation Manager. There are several vmadmin commands that must be run to complete this task:

- CreateSignerCertRequest
- GetCertRequest
- ImportSignerCert

To use a CA-issued certificate:

1. Run the **CreateSignerCertRequest** command to create an OCSP signer certificate request.

   When creating a new certificate request, consider the following:

   Subject DN. This string can include Common Name, Organization, Organizational Unit, Locality, Province/State, and Country of the subject DN for the request. At least one part of the subject DN must be included in the request.

   OCSP signer nickname. The nickname of the default OCSP signer.

   Request nickname. The nickname for the request.

   For a complete description of the **CreateSignerCertRequest** command, see "CreateSignerCertRequest" on page 60.

   The following vmadmin command creates a new certificate request named Req1 with the subject DN CN=Mysigner:

   ```
   vmadmin CreateSignerCertRequest CN=MySigner MySigner Req1
   ```

2. Run the **GetCertRequest** command to retrieve the certificate request.

   For a complete description of the **GetCertRequest** command, see "GetCertRequest" on page 62.

   The following vmadmin command retrieves the certificate request and displays it in the PEM-encoded format:

   ```
   vmadmin GetCertRequest -format pem Req1
   ```

   The administrator then submits this certificate request to a CA. The CA uses this request to issue a CA-issued certificate for the OCSP signer.

3. Run the **ImportSignerCert** command to import the CA-issued certificate into Validation Manager.

   For a complete description of the **ImportSignerCert** command, see "ImportSignerCert" on page 65.

   The following vmadmin command imports the new certificate for MySigner from a file named certifcate1.cer and assigns the nickname Cert1 to the certificate:

   ```
   vmadmin ImportSignerCert MySigner Cert1 certifcate1.cer
   ```

You can import any number of certificates for the OCSP signer. Certificates issued by CAs for the OCSP signer can be imported for each known CA. Certificates can also be imported for CAs that are not known to Validation Manager. By default, Validation Manager automatically uses the CA-issued certificate for an OCSP signer (if there is one) when certificate status responses are made. If Validation Manager does not have an appropriate CA-issued certificate, the self-signed certificate or the OCSP signer's default certificate is used.

The following vmadmin command sets the certificate nicknamed Cert1 as the default certificate for the OCSP signer MySigner:

```
vmadmin SetSignerDefaultCert MySigner Cert1
```

Creating a Status Source

After a default OCSP signer is created, Validation Manager is able to respond to OCSP client certificate status queries. However, Validation Manager responds to all certificate status queries with `unknown`. Validation Manager responds with `unknown` because it is not yet configured to obtain status data for certificates. To obtain certificate status data, a status source must be created. Three different status source types can be configured for Validation Manager: revocation list-based, OCSP-based, and LDAP-based.

The first status source created is automatically set as the default status source.

Revocation List-Based Status Source

To create a revocation list-based status source:

Run the **CreateRLStatusSource** command.

When creating a new revocation list-based status source, consider the following:

Hostname. The name of the server Validation Manager retrieves revocation lists from.

Nickname. The unique nickname of the status source.

Refresh time type. The type of refresh time Validation Manager uses to retrieve updated certificate status.

Refresh time value. The time before or after the last update of certificate status when Validation Manager retrieves certificate status from the OCSP server (based on the refreshTimeType value).

For a complete description of the **CreateRLStatusSource** command, see "CreateRLStatusSource" on page 74.

The following vmadmin command creates a revocation list-based status source named ldap1 for the LDAP server at ldap.example.com:

```
vmadmin CreateRLStatusSource -Host ldap.example.com ldap1
```

OCSP-Based Status Source

To create an OCSP-based status source:

Run the **CreateOCSPStatusSource** command.

When creating a new OCSP-based status source, consider the following:

Hostname. The name of the OCSP server from which Validation Manager retrieves certificate status.

Nickname. The unique nickname of the status source.

Refresh time type. The type of refresh time Validation Manager uses to retrieve updated certificate status.

Refresh time value. The time before or after the last update of certificate status when Validation Manager retrieves certificate status from the OCSP server (based on the refreshTimeType value).

OCSP status check. When or if Validation Manager checks the status of the OCSP-based status source response-signing certificate.

For a complete description of the **CreateOCSPStatusSource** command, see "CreateOCSPStatusSource" on page 73.

The following vmadmin command creates an OCSP-based status source named ocsp1 for the OSCP server at ocsp.example.com:

```
vmadmin CreateOCSPStatusSource -Host ocsp.example.com ocsp1
```

LDAP-Based Status Source

To create an LDAP-based status source:

Run the **CreateLDAPStatusSource** command.

When creating a new LDAP-based status source, consider the following:

Hostname. The DNS hostname of the server from which Validation Manager retrieves certificate status.

Nickname. The unique nickname of the status source.

For a complete description of the **CreateLDAPStatusSource** command, see "CreateLDAPStatusSource" on page 70.

The following vmadmin command creates an LDAP-based status source named ldap1 for the LDAP server at ldap.example.com:

```
vmadmin CreateLDAPStatusSource -Host ldap.example.com ldap1
```

Adding CAs

Finally, you must add a CA to Validation Manager. At this point (before a CA is added), Validation Manager responds to certificate status queries with `unknown`. Validation Manager responds with `unknown` because it does not yet recognize (and trust) CAs. Adding a CA makes it known to Validation Manager and therefore trusted. You add a CA to Validation Manager by adding the CA certificate. To acquire a CA certificate, you must obtain it from the CA and save it to a file.

To add a CA to Validation Manager:

Run the **ImportCA** command.

When adding a CA, consider the following:

Nickname. The unique nickname by which the CA is known to Validation Manager.

Filename. The filename of the CA certificate that you have acquired and saved to a file.

For a complete description of the **ImportCA** command, see "ImportCA" on page 45.

The following vmadmin command adds a CA to Validation Manager from the certificate in the file ca1.cer and gives the CA the nickname CA1:

```
vmadmin ImportCA CA1 ca1.cer
```

Configuring Revocation Lists

Validation Manager is now ready to respond to OCSP client certificate status queries. After you have configured Validation Manager, you may also set the types of revocation lists Validation Manager retrieves: complete certificate revocation lists (CRLs), delta certificate revocation lists (delta CRLs), and authority revocation list (ARLs). By default, Validation Manager only imports complete CRLs.

To set the revocation list types to be retrieved:

Run the **SetCARLTypes** command.

When setting the revocation list types to retrieve, consider the following:

Nickname. The nickname of the CA whose revocation lists are retrieved.

Types. The type of revocation list retrieved: complete CRL, delta CRL, and ARL.

For a complete description of the **SetCARLTypes** command, see "SetCARLTypes" on page 53.

The following vmadmin command configures Validation Manager to retrieve both complete CRLs and ARLs (but not delta CRLs) for CA1:

```
vmadmin SetCARLTypes CA1 +crl+arl-drl
```

The **SetCARLTypes** command does not retrieve revocation lists.

To retrieve revocation lists:

Run the **RefreshCAStatusSource** command.

For a complete description of the RefreshCAStatusSource command, see "RefreshCAStatusSource" on page 47.

The following vmadmin command retrieves the revocation lists configured using **SetCARLTypes**:

```
vmadmin RefreshCAStatusSource CA1
```

Restarting Validation Manager

After installing and configuring Validation Manager for the first time, restart Validation Manager. For instructions, see "Starting and Stopping RSA Validation Manager" on page 11.

3 System Commands

This chapter describes the various system commands used to control and configure RSA Validation Manager system-wide settings.

Commands

The following commands are used to manage and configure Validation Manager system-wide settings:

- DeleteSyncClient
- GetAuditEventList
- GetAuditLogSettings
- GetClusterNodeList
- GetOCSPEnabledSetting
- GetSyncClientInfo
- GetSyncClientList
- GetSystemSettings
- SetAuditEvent
- SetAuditLogging
- SetAuditLogRolloverInterval
- SetAuditLogSigningInterval
- SetConfigurationRefreshTime
- SetDefaultOCSPValidation
- SetDefaultSigner
- SetDefaultStatusSource
- SetOCSPEnabled
- SetOCSPSettings
- SetSyncClientState
- UseFreshStatusData

DeleteSyncClient

Prototype	`DeleteSyncClient` ***`Host`***
Purpose	Deletes a synchronization client installation.
Input Options	None.
Input Parameters	***`Host`*** Required. The hostname of the synchronization client installation to be deleted.

Note: *Host* is case sensitive.

Use the **GetSyncClientList** command to retrieve the hostname of the client to delete.

Output	None.
Return Values	0 Success. 1 Failure.

GetAuditEventList

Prototype	`GetAuditEventList`
Purpose	Retrieves the current event logging settings.
Input Options	None.
Input Parameters	None.
Output	A VMAuditEventList XML object representing the logging events.

The VMAuditEventList object has the following schema:

```
<complexType name="VMAuditEventList">
   <element name="auditEvent" maxOccurs="*"/>
</complexType>
```

where the value for the auditEvent attribute is of the form:

```
eventID=<eventID>;logSuccess=<true|false>;
logFailure=<TRUE|FALSE>;logCritical=<TRUE|FALSE|NA>
```

Note: The VMAuditEventList object can be viewed on the console or redirected to a file.

Return Values

0 Success.

1 Failure.

GetAuditLogSettings

Prototype	`GetAuditLogSettings`
Purpose	Retrieves audit log settings.
Options	None.
Parameter	None.
Output	A VMAuditLogSettings XML object representing the log settings.

The VMAuditLogSettings object has the following schema:

```
<complexType name="VMAuditLogSettings">
   <element name="signingIntervalSeconds"/>
   <element name="signingIntervalEntries"/>
   <element name="logRolloverTime"/>
   <element name="rolloverIntervalTime"/>
   <element name="rolloverIntervalEntries"/>
</complexType>
```

Return Values

0 Success.

1 Failure.

GetClusterNodeList

Prototype	`GetClusterNodeList`
Purpose	Retrieves information for Validation Manager installations that are part of a cluster.
Input Options	None.
Input Parameters	None.
Output	A VMClusterNodeList XML object containing information for all of the cluster nodes. The VMClusterNodeList object has the following schema: `<complexType name="VMClusterNodeList"> <element name="node" minOccurs="0" maxOccurs="*"/> </complexType>` where the value for the node attribute is of the form: `hostname=`*`hostname`*`;state=`*`state`*
Return Values	0 Success. 1 Failure.

GetOCSPEnabledSetting

Prototype	`GetOCSPEnabledSetting`
Purpose	Returns the enabled state of the OCSP server.
Input Options	None.
Input Parameters	None.
Output	`true` or `false`.
Return Values	0 Success. 1 Failure.

GetSyncClientInfo

Prototype	`GetSyncClientInfo` *`Host`*
Purpose	Retrieves information about a specific synchronization client installation.
Input Options	None.
Input Parameters	*`Host`* Required. The hostname of the synchronization client.

Note: *Host* is case sensitive.

Use the **GetSyncClientList** command to retrieve the hostname of the client.

Output

A VMSyncClientInfo XML object representing a synchronization client.

The VMSyncClientInfo object has the following schema:

```
<complexType name="VMSyncClientInfo">
   <element name="hostname"/>
   <element name="caNicknames" maxOccurs="*"/>
   <element name="registrationState"/>
   <element name="lastSynchTime"/>
</complexType>
```

Return Values

0 Success.

1 Failure.

GetSyncClientList

Prototype	`GetSyncClientList`
Purpose	Retrieves a list of hostnames of all synchronization clients.
Input Options	None.
Input Parameters	None.
Output	A VMHostNameList XML object containing the hostnames of all synchronization client installations.

The VMHostNameList object has the following schema:

```
<complexType name="VMHostnameList">
   <element name="hostname" minOccurs="0" maxOccurs="*"/>
</complexType>
```

Return Values

0 Success.

1 Failure.

GetSystemSettings

Prototype	`GetSystemSettings`
Purpose	Retrieves and displays Validation Manager system default settings.
Input Options	None.
Input Parameters	None.
Output	A VMSystemSettings XML object specifying Validation Manager system default settings. The VMSystemSettings object has the following schema:

```
<complexType name="VMSystemSettings">
   <element name="defaultSignerNickname" minOccurs="0"/>
   <element name="defaultStatusSourceNickname"
   minOccurs="0">
   <element name="configurationRefreshTime"/>
   <element name="ocspValidationMode"/>
   <element name="ocspValidationLevel"/>
   <element name="ocspRespectServiceLocator"/>
   <element name="ocspResponseCacheLifetime"/>
   <element name="webProxyOption"/>
   <element name="webProxySvrURL/>
   <element name="auditLogging"/>
   <element name="useFreshStatusData"/>
 </complexType>
```

Note: The VMSystemSettings object can be viewed on the console or redirected to a file.

Return Values	0 Success. 1 Failure.

SetAuditEvent

Prototype

```
SetAuditEvent [-logSuccess {true | false}]
[-logFailure {true | false}] [-logCritical {true | false}]
EventId
```

Purpose

Sets the current logging events settings.

Input Options

At least one of the following must be specified:

`-logSuccess {true | false}`	Optional. Specifies whether the event is logged when it succeeds.
`-logFailure {true | false}`	Optional. Specifies whether the event is logged when it fails.
`-logCritical {true | false}`	Optional. Specifies whether the event is aborted if the logging itself fails to occur. A failure occurs if this is set for an event where logging is not transactional.

Input Parameters

`EventId` Required. The identifier for the event. The possible values can be retrieved using the **GetAuditEventList** command.

Note: *EventId* is case sensitive.

Output

None.

Return Values

0 Success.

1 Failure.

SetAuditLogging

Prototype

```
SetAuditLogging {true | false}
```

Purpose

Turns audit logging on or off.

Input Options

None.

Input Parameters

`true | false` Required. The desired state of the audit log. One of the following must be specified:

`true`	Turns audit logging on.
`false`	Turns audit logging off.

Output

None.

Return Values

0 Success.

1 Failure.

SetAuditLogRolloverInterval

Prototype
`SetAuditLogRolloverInterval [-time Time] [-numEntries NumEntries]`

Purpose
Specifies when the audit log is to be rolled over. A new file is created to hold logging data.

Input Options
At least one of the following must be specified:

`-time Time`	Optional. Specifies the time of day when a new log is to be started. The format must be *hhmm*, based on the 24 hour clock. A value of **-none** specifies that log rollover by time is turned off.
`-numEntries NumEntries`	Optional. Specifies the number of entries allowed before a new log is started. A value of **0** specifies that log rollover by number of entries is turned off.

Input Parameters
None.

Output
None.

Return Values
0 Success.

1 Failure.

SetAuditLogSigningInterval

Prototype
`SetAuditLogSigningInterval [-numSeconds NumSeconds] [-numEntries NumEntries]`

Purpose
Sets the interval after which the audit log is signed.

Input Options
At least one of the following must be specified:

`-numSeconds NumSeconds`	Optional. Specifies the number of seconds allowed before a log is to be signed. A value of **0** specifies that log signing by number of seconds is turned off.
`-numEntries NumEntries`	Optional. Specifies the number of entries allowed before a log is to be signed. A value of **0** specifies that log signing by number of entries is turned off.

Input Parameters
None.

Output
None.

Return Values
0 Success.

1 Failure.

SetConfigurationRefreshTime

Prototype	`SetConfigurationRefreshTime` ***`RefreshTime`***
Purpose	Sets the configuration refresh time.
Input Options	None.
Input Parameters	***`RefreshTime`*** Required. The time in seconds Validation Manager waits before retrieving fresh certificate status data.
Output	None.
Return Values	0 Success. 1 Failure.

SetDefaultOCSPValidation

Prototype	`SetDefaultOCSPValidation` ***`Mode Level`***
Purpose	Sets the default OCSP validation mode and level.
Input Options	None.
Input Parameters	***`Mode`*** Required. The default mode of OCSP validation. ***`Mode`*** can be one of the following case-insensitive values:

`None`	Accept both signed and unsigned requests.
`Optional`	Validate signed requests and accept unsigned requests.
`Required`	Validate signed requests and reject unsigned requests.

`Level` Required. The default level of OCSP validation. ***`Level`*** can be one of the following case-insensitive values:

`RequireSignature`	Requires that the request be signed and that the signature can be verified using the requestor's certificate.
`RequireKnownRequestor`	Same as RequireSignature, but also checks that the requestor certificate is issued by (or chained to) a known CA.
`RequireUnrevokedRequestor`	Same as RequireKnownRequestor, but also checks that the status of the requestor certificate is “good” (neither revoked, suspended, or unknown).

Output	None.
Return Values	0 Success. 1 Failure.

SetDefaultSigner

Prototype	`SetDefaultSigner` ***`Nickname`***
Purpose	Designates the named OCSP signer as the default OCSP signer.
Input Options	None.
Input Parameters	***`Nickname`*** Required. The nickname of the new default OCSP signer.
Output	None.
Return Values	0 Success. 1 Failure.

SetDefaultStatusSource

Prototype	`SetDefaultStatusSource` ***`Nickname`***
Purpose	Designates the named status source as the default status source.
Input Options	None.
Input Parameters	***`Nickname`*** Required. The nickname of the new default status source.
Output	None.
Return Values	0 Success. 1 Failure.

SetOCSPEnabled

Prototype	`SetOCSPEnabled {true | false}`
Purpose	Sets the enabled state of the OCSP server. This includes all OCSP servers in a cluster.
Input Options	None.
Input Parameters	`true | false` Required. The enabled state of the OCSP server. One of the following must be specified: `true` Enables (resumes) the OCSP server. `false` Disables (suspends) the OCSP server.
Output	None.
Return Values	0 Success. 1 Failure.

SetOCSPSettings

Prototype

```
SetOCSPSettings [-ocspRespectServiceLocator {true | false}]
[-ocspResponseCacheLifetime Lifetime]
[-ocspHttpProxyMode Mode] [-ocspHttpProxyUrl Url]
```

Purpose

Sets system-wide OCSP settings.

Input Options

At least one of the following must be specified:

`-ocspRespectServiceLocator {true | false}` Optional. Specifies whether the system should respect service locator extensions. The default is **true**.

`-ocspResponseCacheLifetime` ***`Lifetime`*** Optional. Specifies the number of seconds that the system reuses responses. The default is **0** (the response nextUpdate time). Positive values indicate the number of seconds after the time the response is first generated. Negative values indicate the number of seconds prior to the response nextUpdate time.

`-ocspHttpProxyMode` ***`Mode`*** Specifies whether the system uses an HTTP proxy server for requests forwarded or proxied to a remote responder. ***`Mode`*** must be one of the following:

- `none` Default. No HTTP proxy server is used.
- `url` An HTTP proxy server is used. If the -ocspHttpProxyUrl argument is not supplied, an error is returned.
- `MSIE` Settings are retrieved from the Windows registry settings for Internet Explorer (Windows only).

`-ocspHttpProxyUrl` ***`Url`*** The full URL of the HTTP proxy server, including the port number.

Input Parameters

None.

Output

None.

Return Values

0 Success.

1 Failure.

SetSyncClientState

Prototype `SetSyncClientState` *`Host State`*

Purpose Sets the activation state of a synchronization client.

Input Options None.

Input Parameters *`Host`* Required. The hostname of the synchronization client to be changed.

Note: *Host* is case sensitive.

Use the **GetSyncClientList** command to retrieve the hostname of the client.

`State` Required. The state of the synchronization client installation. *`State`* must be one of the following:

`allowed`	The installation is accepted as synchronized.
`pending`	The installation is not currently available.
`forbidden`	The installation is rejected as synchronized.

Output None.

Return Values 0 Success.

1 Failure.

UseFreshStatusData

Prototype `UseFreshStatusData {true | false}`

Purpose Designates whether the system waits for fresh status data or responds with available stale data.

Input Options None.

Input Parameters `true | false` Required. The state of the fresh status data. One of the following must be specified:

`true`	Waits for fresh data.
`false`	Responds with stale data.

Output None.

Return Values 0 Success.

1 Failure.

4 CA Commands

This chapter describes the various CA commands used to control and configure CAs.

Commands

The following commands are used to manage and configure CAs:

- DeleteCA
- GetCA
- GetCAARL
- GetCAARLInfo
- GetCACert
- GetCACRL
- GetCACRLInfo
- GetCADRL
- GetCADRLInfo
- GetCAList
- GetCAOCSPRequestCount
- GetCARLEntryInfo
- GetCASyncUpdates
- GetCertStatus
- GetLocallyRevokedCertList
- ImportARL
- ImportCA
- ImportCASyncUpdates
- ImportCRL
- ImportDRL
- RefreshCAStatusSource
- SetCAIndirectRLIssuer
- SetCALocalStatus
- SetCANickname
- SetCAOCSPSettings
- SetCAOCSPValidation
- SetCAPath
- SetCAPurposes
- SetCASigner
- SetCARLTypeAttributes
- SetCARLTypes
- SetCASignerCert
- SetCAStatusSource
- SetCASyncEnabled
- SetCertLocalStatus

DeleteCA

Prototype	`DeleteCA` ***`Nickname`***
Purpose	Deletes a known CA from RSA Validation Manager.
Input Options	None.
Input Parameters	***`Nickname`*** Required. The nickname of the CA to delete.
Output	None.
Return Values	0 Success. 1 Failure.

Note: When a CA is deleted, all revocation lists associated with the CA are also deleted.

GetCA

Prototype `GetCA` *`Nickname`*

Purpose Obtains and displays a known CA.

Input Options None.

Input Parameters *`Nickname`* Required. The nickname of the CA to be displayed.

Output A VMCA XML object representing the known CA.

The VMCA object has the following schema:

```
<complexType name="VMCA">
   <element name="sha1KeyHash"/>
   <element name="nickname"/>
   <element name="sha1NameHash"/>
   <element name="md5KeyHash"/>
   <element name="md5NameHash"/>
   <element name="purposeOCSPCA"/>
   <element name="purposeOCSPClientAuth"/>
   <element name="purposeTLSServerAuth"/>
   <element name="localStatus"/>
   <element name="signerNickname"/>
   <element name="statusSourceNickname"/>
   <element name="path" minOccurs="0"/>
   <element name="useCRL"/>
   <element name="useARL"/>
   <element name="useDRL"/>
   <element name="cRLAttribute"/>
   <element name="aRLAttribute"/>
   <element name="dRLAttribute"/>
   <element name="ocspValidationMode"/>
   <element name="ocspValidationLevel"/>
   <element name="ocspReuseResponses"/>
   <element name="certificateNickname"/>
   <element name="signerCertificateNickname"/>
   <element name="usingDefaultSigner"/>
   <element name="usingDefaultStatusSource"/>
   <element name="useDefaultSignerCertificate"/>
   <element name="lastSyncNumber"/>
   <element name="localStatusChanegTime"/>
</complexType
```

Return Values 0 Success.

1 Failure.

GetCAARL

Prototype `GetCAARL [-format {pem}] Nickname`

Purpose Retrieves the last ARL imported for the CA.

Input Options `-format` Optional. Specifies the format of the output. The option value **pem** specifies that the ARL is output in PEM-encoded format. If not specified, the ARL is output in binary.

Input Parameters *`Nickname`* Required. The nickname of the CA.

Output The X.509 ARL of the CA.

Return Values
0 Success.
1 Other failures.
2 ARL not found.
3 CA does not have a revocation list-based status source.

GetCAARLInfo

Prototype `GetCAARLInfo Nickname`

Purpose Retrieves general information for the last ARL imported for the CA.

Input Options None.

Input Parameters *`Nickname`* Required. The nickname of the CA.

Output A VMRevocationList XML object containing general information about an ARL.

The VMRevocationList object has the following schema:

```
<complexType name="VMRevocationList">
   <element name="importTime"/>
   <element name="thisUpdate"/>
   <element name="nextUpdate" minOccurs="0"/>
   <element name="crlNumber" minOccurs="0"/>
   <element name="extensions" minOccurs="0"/>
   <element name="expiryTime" minOccurs="0"/>
</complexType>
```

The extensions attribute is an ASCII formatted representation of the revocation list extensions data.

Return Values
0 Success.
1 Other failures.
2 ARL not found.
3 CA does not have a revocation list-based status source.

GetCACert

Prototype	`GetCACert [-format {pem}] Nickname`
Purpose	Retrieves the certificate of a known CA.
Input Options	`-format` Optional. Specifies the format of the output. The option value **pem** specifies that the certificate is output in PEM-encoded format. If not specified, the certificate is output in binary.
Input Parameters	*`Nickname`* Required. The nickname of the CA whose certificate is to be retrieved.
Output	The X.509 certificate of the CA.
Return Values	0 Success. 1 Failure.

GetCACRL

Prototype	`GetCACRL [-format {pem}] Nickname`
Purpose	Retrieves the last complete CRL imported for the CA.
Input Options	`-format` Optional. Specifies the format of the output. The option value **pem** specifies that the complete CRL is output in PEM-encoded format. If not specified, the complete CRL is output in binary.
Input Parameters	*`Nickname`* Required. The nickname of the CA.
Output	The X.509 complete CRL of the CA.
Return Values	0 Success. 1 Other failures. 2 Complete CRL not found. 3 CA does not have a revocation list-based status source.

GetCACRLInfo

Prototype `GetCACRLInfo` ***Nickname***

Purpose Retrieves the metainfo for the last complete CRL imported for the CA.

Input Options None.

Input Parameters ***Nickname*** Required. The nickname of the CA.

Output A VMRevocationList XML object containing metadata about a complete CRL.

The VMRevocationList object has the following schema:

```
<complexType name="VMRevocationList">
   <element name="importTime"/>
   <element name="thisUpdate"/>
   <element name="nextUpdate" minOccurs="0"/>
   <element name="crlNumber" minOccurs="0"/>
   <element name="extensions" minOccurs="0"/>
   <element name="expiryTime" minOccurs="0"/>
</complexType>
```

The extensions attribute is an ASCII-formatted representation of the revocation list extensions data.

Return Values

0 Success.

1 Other failures.

2 CRL not found.

3 CA does not have a revocation list-based status source.

GetCADRL

Prototype `GetCADRL [-format {pem}]` ***Nickname***

Purpose Retrieves the last delta CRL imported for the CA.

Input Options `-format` Optional. Specifies the format of the output. The option value **pem** specifies that the delta CRL is output in PEM-encoded format. If not specified, the delta CRL is output in binary.

Input Parameters ***Nickname*** Required. The nickname of the CA.

Output The X.509 delta CRL of the CA.

Return Values

0 Success.

1 Other failures.

2 Delta CRL not found.

3 CA does not have a revocation list-based status source.

GetCADRLInfo

Prototype	`GetCADRLInfo` *`Nickname`*
Purpose	Retrieves the metainfo for the last delta CRL imported for the CA.
Input Options	None.
Input Parameters	*`Nickname`* Required. The nickname of the CA.

Output	A VMRevocationList XML object containing metadata about a delta CRL.
	The VMRevocationList object has the following schema:
	`<complexType name="VMRevocationList">`
	`  <element name="importTime"/>`
	`  <element name="thisUpdate"/>`
	`  <element name="nextUpdate" minOccurs="0"/>`
	`  <element name="crlNumber" minOccurs="0"/>`
	`  <element name="extensions" minOccurs="0"/>`
	`  <element name="expiryTime" minOccurs="0"/>`
	`</complexType>`
	The extensions attribute is an ASCII formatted representation of the revocation list extensions data.

Return Values	0 Success. 1 Other failures. 2 Delta CRL not found. 3 CA does not have a revocation list-based status source.

GetCAList

Prototype	`GetCAList`
Purpose	Obtains and displays a list of saved CA nicknames.
Input Options	None.
Input Parameters	None.

Output	A VMNicknameList XML object containing the nicknames of all known CAs.
	The VMNicknameList object has the following schema:
	`<complexType name="VMNicknameList">`
	`  <element name="nicknames" minOccurs="0" maxOccurs="*"/>`
	`</complexType>`

Return Values	0 Success. 1 Failure.

GetCAOCSPRequestCount

Prototype	`GetCAOCSPRequestCount [CANickname]`
Purpose	Retrieves the number of OCSP requests in the previous hour for all CAs or a specified CA.
Input Options	*`CANickname`* Optional. The nickname of the CA. If not specified, the count for all CAs is returned.
Input Parameters	None.
Output	A VMCAOCSPRequestCount XML object representing the logging events. The VMOCSPRequestCount object has the following schema: `<complexType name="VMOCSPRequestCount"> <element name="caRequestCount" minOccurs="0" maxOccurs="*"/> <element name="unknownRequestCount" minOccurs="0"/> <element name="totalRequestCount" minOccurs="0"/> </complexType>` The value for the caRequestCount attribute is of the form: `nickname=`*`nickname`*`;count=`*`count`*
Return Values	0 Success. 1 Failure.

GetCARLEntryInfo

Prototype

`GetCARLEntryInfo` ***`Nickname RLType`***
`{-serialNumber` ***`SerialNumber`*** `|` ***`FileName`*** `| -}`

Purpose

Retrieves status information for a certificate entry in a revocation list for the CA.

Input Options

None.

Input Parameters

`Nickname` Required. The nickname of the CA.

`RLType` Required. The type of revocation list. ***`RLType`*** can be one of the following:

`crl`	Complete CRL
`arl`	ARL
`drl`	Delta CRL

`-serialNumber` ***`SerialNumber`*** `|` ***`FileName`*** `| -` Required. Identifies the certificate. One of the following must be specified:

`-serialNumber` ***`SerialNumber`***	The serial number of a certificate.
`FileName`	Read the certificate from a file.
`-` or blank	Read the certificate from standard input.

Output

A VMStatusEntry XML object containing information for a certificate entry in a revocation list.

The VMStatusEntry object has the following schema:

```
<complexType name="VMStatusEntry">
   <element name="revocationTime"/>
   <element name="reasonCode" minOccurs="0"/>
   <element name="holdInstructionCode" minOccurs="0"/>
   <element name="invalidityDate" minOccurs="0"/>
</complexType>
```

Return Values

0	Success.
1	Other failures.
2	Revocation list of specified type not found.
3	CA does not have a revocation list-based status source.
4	Entry not found.

GetCASyncUpdates

Prototype `GetCASyncUpdates` ***`Nickname`*** `{`***`FileName`*** `| -}`

Purpose Retrieves synchronization updates for a CA.

Input Options None.

Input Parameters ***`Nickname`*** Required. The nickname of a CA.

`FileName` `|` `-` Required. Specifies the location where the updates are to be written. One of the following must be specified:

`FileName`	Write the synchronization updates to a file.
`-` or blank	Write the synchronization updates to standard output.

Output The synchronization updates data in a format suitable for use by the **ImportCASyncUpdates** command.

Return Values 0 Success.

1 Failure.

GetCertStatus

Prototype `GetCertStatus Nickname`
`{-serialNumber` ***`SerialNumber`*** `|` ***`FileName`*** `| -}`

Purpose Retrieves status information for a certificate issued by a CA.

Input Options None.

Input Parameters ***`Nickname`*** Required. The nickname of the CA.

`-serialNumber` ***`SerialNumber`*** `|` ***`FileName`*** `|` `-` Required. Identifies the certificate. One of the following must be specified:

`-serialNumber` ***`SerialNumber`***	The serial number of the certificate.
`FileName`	Read the certificate from a file.
`-` or blank	Read the certificate from standard input.

Output A VMStatusEntry XML object containing the status information for a certificate.

The VMStatusEntry object has the following schema:

```
<complexType name="VMStatusEntry">
   <element name="revocationTime"/>
   <element name="reasonCode" minOccurs="0"/>
   <element name="holdInstructionCode" minOccurs="0"/>
   <element name="invalidityDate" minOccurs="0"/>
</complexType>
```

Return Values 0 Success.

1 Failure.

GetLocallyRevokedCertList

Prototype `GetLocallyRevokedCertList` ***Nickname***

Purpose Returns a list of locally revoked certificate serial numbers.

Input Options None.

Input Parameters ***Nickname*** Required. The nickname of the CA.

Output A VMSerialNumberList XML object containing the serial numbers of all locally revoked certificates for the CA.

The VMSerialNumberList object has the following schema:

```
<complexType name="VMLocallyRevokedCertList">
   <element name="serialNumbers" minOccurs="0"
   maxOccurs="*"/>
</complexType>
```

Return Values

0 Success.

1 Failure.

ImportARL

Prototype `ImportARL` ***Nickname*** {***FileName*** | `-`}

Purpose Imports a new ARL type revocation list.

Input Options None.

Input Parameters ***Nickname*** Required. The nickname of the CA. An error occurs if the CA is not configured for this type of revocation list, or if the CA is associated with a status source of type SYNC.

FileName | `-` Required. One of the following must be specified:

- ***FileName*** The name of a file containing the revocation list to import. The file contents may be PEM-encoded or the binary revocation list Distinguished Encoding Rules (DER).
- `-` or blank Read the revocation list from standard input, in any of the formats specified above.

Output None.

Return Values

0 Success.

1 Other failures.

2 CA does not have a revocation list-based status source.

3 ARL not found.

ImportCA

Prototype

```
ImportCA [-purposes Purposes]
[-statussource StatusSourceNickname]
[-signer SignerNickname] Nickname {FileName | -}
```

Purpose

Adds a new CA to Validation Manager.

Input Options

`-purposes` ***Purposes*** Optional. The purposes Validation Manager applies to the new CA. ***Purposes*** is a single string composed of a series of one or more flag (+ or -) and purpose name pairs:

{+ | -} ***PurposeName***

A + (plus) flag enables the CA for the named purpose, while a - (minus) flag disables the CA for the named purpose. If a ***PurposeName*** appears more than once in the ***Purposes***, the ***PurposeName*** is set according to the flag associated with its last appearance.

PurposeName can be one of the following:

`OCSPClientAuth`	Client OCSP request authentication
`OCSPCA`	The server serves OCSP for the CA
`TLSServerAuth`	TLS server authentication.
`IdentrusRoot`	Identifies the IdenTrust Root CA.

For example, the following enables the CA for both OCSP service and for authenticating OCSP client requests:

```
+OCSPCA+OCSPClientAuth
```

If no ***Purposes*** are specified, the CA is enabled for all purposes except `IdentrusRoot`.

`-statussource` ***StatusSourceNickname*** Optional. The nickname of the status source to use for the new CA. If not specified, the new CA uses the default status source.

`-signer` ***SignerNickname*** Optional. The nickname of the OCSP signer to use for the new CA. If not specified, the CA uses the default OCSP signer.

Input Parameters

Nickname Required. The nickname of the new CA.

FileName | - Required. One of the following must be specified:

FileName	The name of a file containing the CA certificate. The file contents may be PEM-encoded or the binary DER of either a single certificate or PKCS #7 message containing a chain of certificates (in which case the first certificate in the chain is used as the CA certificate).
- or blank	Read the CA certificate from standard input, in any of the same formats as ***FileName***.

Output

None.

Return Values

0 Success.

1 Failure.

ImportCASyncUpdates

Prototype `ImportCASyncUpdates` ***Nickname*** {***FileName*** | `-`}

Purpose Imports status updates for a CA.

Input Options None.

Input Parameters ***Nickname*** Required. The nickname of the CA.

FileName | `-` Required. Specifies the location where the updates are to be read. One of the following must be specified:

FileName	Read the synchronization updates from a file.
`-` or blank	Read the synchronization updates from standard input.

Output None.

Return Values 0 Success.

1 Failure.

ImportCRL

Prototype `ImportCRL` ***Nickname*** {***FileName*** | `-`}

Purpose Imports a new complete CRL type revocation list.

Input Options None.

Input Parameters ***Nickname*** Required. The nickname of the CA. An error occurs if the CA is not configured for this type of revocation list, or if the CA is associated with a status source of type SYNC.

FileName | `-` Required. One of the following must be specified:

`FileName`	The name of a file containing the revocation list to import. The file contents may be PEM-encoded or the binary revocation list DER.
`-` or blank	Read the revocation list from standard input, in any of the formats specified above.

Output None.

Return Values 0 Success.

1 Other failures.

2 CA does not have a revocation list-based status source.

3 CRL not found.

ImportDRL

Prototype `ImportDRL` ***`Nickname`*** `{`***`FileName`*** `| -}`

Purpose Imports a new delta CRL type revocation list.

Input Options None.

Input Parameters ***`Nickname`*** Required. The nickname of the CA. An error occurs if the CA is not configured for this type of revocation list, or if the CA is associated with a status source of type SYNC.

`FileName` | - Required. One of the following must be specified:

`FileName`	The name of a file containing the revocation list to import. The file contents may be PEM-encoded or the binary revocation list DER.
- or blank	Read the revocation list from standard input, in any of the same formats as ***`FileName`***.

Output None.

Return Values

0	Success.
1	Other failures.
2	CA does not have a revocation list-based status source.
3	DRL not found.

RefreshCAStatusSource

Prototype `RefreshCAStatusSource` ***`Nickname`***

Purpose Manually forces Validation Manager to retrieve new status information for the CA. This forces Validation Manager to immediately retrieve new revocation lists. This command does not affect on CAs that use OCSP and LDAP-based status sources.

Input Options None.

Input Parameters ***`Nickname`*** Required. The nickname of the CA whose status source is refreshed.

Output None.

Return Values

0	Success.
1	Failure.

SetCAIndirectRLIssuer

Prototype `SetCAIndirectRLIssuer` ***`Nickname`*** `{`***`rlIssuerNickname`*** `|` `-none}`

Purpose Sets the indirect revocation list issuer of a CA.

Input Options None.

Input Parameters ***`Nickname`*** Required. The nickname of the CA.

`rlIssuerNickname` `|` `-none` Required. The nickname of the indirect revocation list issuer. The current nickname can be deleted by using the keyword `-none`.

Output None.

Return Values 0 Success.

1 Failure.

SetCALocalStatus

Prototype `SetCALocalStatus` ***`Nickname`*** `{revoked | unrevoked}`

Purpose Sets the local revocation status of a known CA.

Input Options None.

Input Parameters ***`Nickname`*** Required. The nickname of the CA.

`revoked | unrevoked` Required. The desired local revocation status of the CA. One of the following must be specified:

`revoked`	Locally revokes the CA.
`unrevoked`	Locally unrevokes the CA.

Output None.

Return Values 0 Success.

1 Failure.

SetCANickname

Prototype `SetCANickname` ***`OldNickname NewNickname`***

Purpose Changes the nickname of a CA.

Input Options None.

Input Parameters ***`OldNickname`*** Required. The current nickname of the CA.

`NewNickname` Required. The new nickname for the CA.

Output None.

Return Values 0 Success.

1 Failure.

SetCAOCSPSettings

Prototype	`SetCAOCSPSettings [-ocspReuseResponses {true | false}] Nickname`
Purpose	Sets the OCSP settings for a CA.
Input Options	`-ocspReuseResponses {true | false}` Optional. Specifies whether Validation Manager reuses OCSP responses for the lifetime specified by the ocspResponseCacheLifetime option in the SetOCSPSettings command. The default is **true**.
Input Parameters	*`Nickname`* Required. The nickname of the CA.
Output	None.
Return Values	0 Success. 1 Failure.

SetCAOCSPValidation

Prototype `SetCAOCSPValidation` ***Nickname*** {`-default` | ***Mode Level***}

Purpose Sets the OCSP validation for a CA.

Input Options None.

Input Parameters

Nickname Required. The nickname of the CA.

`-default` | ***Mode Level*** Required. One of the following must be specified:

`-default` Default values created using SetDefaultOCSPValidation. When `-default` is used, ***Mode*** and ***Level*** values are ignored.

Mode Level

Mode The default mode of OCSP validation. ***Mode*** can be one of the following case-insensitive values:

`None`	Accept both signed and unsigned requests
`Optional`	Validate signed requests and accept unsigned requests
`Required`	Validate signed requests and reject unsigned requests

Level The default level of OCSP validation. ***Level*** can be one of the following case-insensitive values:

`RequireSignature`	Requires that the request be signed, and that the signature can be verified using the requestor certificate.
`RequireKnownRequestor`	Same as RequireSignature, but also verifies that the requestor certificate is issued by (or chained to) a known CA.
`RequireUnrevokedRequestor`	Same as RequireKnownRequestor, but also verifies that the status of the requestor certificate is good (neither revoked, suspended, or unknown).

Output None.

Return Values 0 Success.

1 Failure.

SetCAPath

Prototype `SetCAPath` ***`Nickname`*** `{-none | ` ***`Path`***`}`

Purpose Changes the value of the CA's path component.

Input Options None.

Input Parameters ***`Nickname`*** Required. The nickname of the CA.

`-none | ` ***`Path`*** Required. The new path value, or the keyword. One of the following must be specified:

`-none`	Erases the current value.
`Path`	The path of the CA. It may be an arbitrary string, but is typically a DN or part of the path element of a URL.

Output None.

Return Values 0 Success.

1 Failure.

SetCAPurposes

Prototype `SetCAPurposes` ***`Nickname Purposes`***

Purpose Sets the purposes Validation Manager applies to a CA.

Input Options None.

Input Parameters ***`Nickname`*** Required. The nickname of the CA.

`Purposes` Required. The purposes to apply to the CA. ***`Purposes`*** is a single string composed of a series of one or more flags (+ or -) and purpose name pairs:

`{+ | -}` ***`PurposeName`***

A + (plus) flag enables the CA for the named purpose, while a - (minus) flag disables the CA for the named purpose. If a ***`PurposeName`*** appears more than once in the ***`Purposes`***, the ***`PurposeName`*** is set according to the flag associated with its last appearance.

`PurposeName` can be one of the following:

`OCSPClientAuth`	OCSP client request authentication.
`OCSPCA`	The server serves OCSP status responses for the CA.
`TLSServerAuth`	TLS server authentication.
`IdentrusRoot`	Identifies the IdenTrust Root CA.

For example, the following enables the CA for both OCSP service and for authenticating OCSP client requests:

```
+OCSPCA+OCSPClientAuth
```

Output None.

Return Values 0 Success.

1 Failure.

SetCARLTypeAttributes

Prototype

`SetCARLTypeAttributes` ***`Nickname`***
`{-arl` ***`ArlAttribute`*** `|` `-crl` ***`CrlAttribute`*** `|` `-drl` ***`DrlAttribute`***`}`

Purpose

Defines the LDAP attribute names used to retrieve different types of revocation lists from an LDAP server.

Input Options

None.

Input Parameters

`Nickname` Required. The nickname of the CA.

`-arl` ***`ArlAttribute`*** `|` `-crl` ***`CrlAttribute`*** `|` `-drl` ***`DrlAttribute`***
Required. One of the following must be specified:

`-arl` ***`ArlAttribute`***	The location the CA publishes its ARLs to, or the keyword `-default` to use the standard value.
`-crl` ***`CrlAttribute`***	The location the CA publishes its complete CRLs to, or the keyword `-default` to use the standard value.
`-drl` ***`DrlAttribute`***	The location the CA publishes its delta CRLs to, or the keyword `-default` to use the standard value.

Output

None.

Return Values

0 Success.

1 Failure.

SetCARLTypes

Prototype `SetCARLTypes` *`Nickname Types`*

Purpose Specifies the types of revocation lists the CA publishes.

Input Options None.

Input Parameters *`Nickname`* Required. The nickname of the CA.

`Types` Required. The types of revocation lists the CA publishes. `Types` is a single string composed of a series of one or more flags (+ or -) and revocation list type name pairs:

`{+ | -}` *`RLTypeName`*

A + (plus) flag tells Validation Manager that the CA publishes revocation lists of the named type, while a - (minus) flag tells Validation Manager that the CA does not publish revocation lists of the named type. If a revocation list type name appears more than once in the *`Types`*, the type is set according to the flag associated with its last appearance.

`RLTypeName` can be one of the following values:

`crl`	The CA publishes complete CRLs
`arl`	The CA publishes ARLs
`drl`	The CA publishes delta CRLs

For example, the following indicates that the CA publishes complete CRLs, but does not publish ARLs or delta CRLs:

```
+crl-arl-drl
```

Note: By default, Validation Manager is configured to only import complete CRLs.

Output None.

Return Values 0 Success.

1 Failure.

SetCASigner

Prototype `SetCASigner` *`CANickname`* {`-default` | *`SignerNickname`*}

Purpose Associates an OCSP signer with a known CA.

Input Options None.

Input Parameters *`CANickname`* Required. The nickname of the CA.

`-default` | *`SignerNickname`* Required. One of the following must be specified:

`-default`	Specifies that the default certificate of the OCSP signer is used.
`SignerNickname`	Specifies the nickname of the OCSP signer certificate to use.

Output None.

Return Values 0 Success.

1 Failure.

SetCASignerCert

Prototype `SetCASignerCert` *`CANickname`* {`-default` | `-auto` | *`CertNickname`*}

Purpose Specifies which certificate a CA's OCSP signer uses when returning the status of a certificate issued by the CA.

Input Options None.

Input Parameters *`CANickname`* Required. The nickname of the CA.

`-default` | `-auto` | *`CertNickname`* Required. One of the following must be specified:

`-default`	Specifies that the OCSP signer default certificate is used.
`-auto`	Specifies that the OCSP signer uses its automatic certificate selection behavior.
`CertNickname`	Specifies the nickname of the OCSP signer certificate to use.

Output None.

Return Values 0 Success.

1 Failure.

SetCAStatusSource

Prototype `SetCAStatusSource` ***CANickname*** `{-default | StatusSourceNickname}`

Purpose Changes the status source of a known CA.

Input Options None.

Input Parameters ***CANickname*** Required. The nickname of the CA.

`-default | StatusSourceNickname` Required. One of the following must be specified:

`-default`	The CA uses the default status source.
StatusSourceNickname	The nickname of the new CA status source.

Output None.

Return Values 0 Success.

1 Failure.

Note: If a CA status source is deleted, the CA status source is set to the default status source.

SetCASyncEnabled

Prototype `SetCASyncEnabled` ***Nickname*** `{True | False}`

Purpose Sets whether or not a CA can be synchronized.

Input Options None.

Input Parameters ***Nickname*** Required. The nickname of a CA.

Output None.

Return Values 0 Success.

1 Failure.

SetCertLocalStatus

Prototype

`SetCertLocalStatus` ***`Nickname`*** `{revoked | unrevoked} {-serial` ***`SerialNumber`*** `|` ***`FileName`*** `| -}`

Purpose

Sets the local revocation status of a certificate for a CA.

Input Options

None.

Input Parameters

`Nickname` Required. The nickname of the CA that issued the certificate.

`revoked | unrevoked` Required. The desired local status of the certificate. One of the following must be specified:

`revoked`	Locally revokes the certificate.
`unrevoked`	Locally unrevokes the certificate.

`-serial` ***`SerialNumber`*** `|` ***`FileName`*** `| -` Required. One of the following must be specified:

`-serial` ***`SerialNumber`***	The serial number of the certificate. The format is assumed to be hexadecimal without punctuation.
`FileName`	The name of a file containing the certificate. The file contents may be PEM-encoded or the binary DER of either a single certificate or PKCS #7 message containing a chain of certificates (in which case the first certificate in the chain is used as the certificate).
`-` or blank	Read the certificate from standard input, in any of the formats specified above.

Output

None.

Return Values

0 Success.

1 Failure.

5 OCSP Signer Commands

This chapter describes the various OCSP signer commands used to control and configure OCSP signers.

Commands

The following commands are used to manage and configure OCSP signers:

- CreateCertRequestFromCert
- CreateDefaultSigner
- CreateSigner
- CreateSignerCertRequest
- DeleteSigner
- DeleteSignerCert
- DeleteSignerCertRequest
- GetCert
- GetCertRequest
- GetDefaultSigner
- GetSigner
- GetSignerCertList
- GetSignerCertRequestList
- GetSignerList
- ImportSignerCert
- RenewSignerSelfCert
- SetSignerCertNickname
- SetSignerDefaultCert
- SetSignerNickname
- SetSignerPassphrase
- ViewCertInfo

CreateCertRequestFromCert

Prototype	`CreateCertRequestFromCert [-format {pem}] [-CertRequestNickname` ***`CertRequestNickname`*** `]` ***`SignerCertNickname`***
Purpose	Creates a new PKCS #10 certificate based on an existing certificate.
Input Options	`-format` Optional. Specifies the format of the output. The option value **pem** specifies that the certificate request is output in PEM-encoded format. If not specified, the certificate request is output in binary. `-CertRequestNickname` ***`CertRequestNickname`*** Optional. The nickname of the certificate request. If not specified, the request is generated, but not stored in Validation Manager.
Input Parameters	***`SignerCertNickname`*** Required. The nickname of the existing certificate. An error occurs if the certificate does not belong to an OCSP signer.
Output	The generated PKCS #10 certificate request.
Return Values	0 Success. 1 Failure.

CreateDefaultSigner

Prototype

`CreateDefaultSigner [-keySize` ***`Size`***`] [-crypto` ***`Provider`***`] [-passphrase` ***`Passphrase`***`]` ***`Nickname`***

Purpose

Creates the default OCSP signer. If a default already exists, this new OCSP signer is added and set as the default OCSP signer.

Input Options

`-keySize` ***`Size`*** Optional. The size in bits of the keypair of the default OCSP signer. Valid values are **1024**, **2048**, and **4096**. The default is **1024**.

Note: If this command results in an error when a keysize of 4096 is selected and the cryptographic provider is nCipher, restart Validation Manager.

`-crypto` ***`Provider`*** Optional. The name of the cryptography library provider that is used to generate and store the OCSP signer keypair. ***`Provider`*** can be one of the following case-insensitive values:

`nCipher`	An nCipher hardware security module (HSM) generates the keypair.
`pkcs11`	A generic PKCS #11 HSM generates the keypair.
`Software`	Validation Manager generates the keypair. The default is **Software**.

`-passphrase` ***`Passphrase`*** Optional. The passphrase to protect the default OCSP signer private key. The passphrase must be at least eight characters in length and contain at least one numerical and one alphabetical character. The passphrase must be specified if the cryptographic library provider is of type Software. If the cryptographic library provider is of any other type, the passphrase is ignored.

Input Parameters

`Nickname` Required. The nickname of the default OCSP signer.

Output

None.

Return Values

0 Success.

1 Failure.

CreateSigner

Prototype

`CreateSigner [-keySize Size] [-crypto Provider] [-passphrase Passphrase] Nickname`

Purpose

Creates an OCSP signer. If it is the first OCSP signer, Validation Manager automatically sets it as the default OCSP signer.

Input Options

`-keySize Size` Optional. The size, in bits, of the keypair of the OCSP signer. Valid values are **1024**, **2048**, and **4096**. The default is **1024**.

Note: If this command results in an error when a keysize of 4096 is selected and the cryptographic provider is nCipher, restart Validation Manager.

`-crypto Provider` Optional. The name of the cryptography library provider that is used to generate and store the OCSP signer keypair. `Provider` can be one of the following case-insensitive values:

`nCipher`	An nCipher hardware security module (HSM) generates the keypair.
`pkcs11`	A generic PKCS #11 HSM generates the keypair.
`Software`	Validation Manager generates the keypair. The default is **Software**.

`-passphrase Passphrase` Optional. The passphrase to protect the OCSP signer private key. The passphrase must be specified if the provider is of type Software. If the provider is of any other type, the passphrase is ignored. The passphrase must be at least eight characters long, and contain at least one numerical and one alphabetical character.

Input Parameters

`Nickname` Required. The nickname of the OCSP signer.

Output

None.

Return Values

0 Success.

1 Failure.

CreateSignerCertRequest

Prototype

`CreateSignerCertRequest [Extensions] SubjectDN SignerNickname RequestNickname`

Purpose

Creates a new PKCS #10 request for an OCSP signer.

Input Options

Extensions Optional. The extensions to include in the certificate request. *Extensions* is a single string composed of a series of one or more flags (+ or -) and extension name pairs:

{+ | -} *ExtensionName*

A + (plus) flag includes the named extension in the request, while a - (minus) flag omits it. If an *ExtensionName* appears more than once in the *Extensions*, the extension is included or not included according to the flag associated with its last appearance.

ExtensionName can be one of the following values:

OCSPSigning	The Extended Key Usage extension with the id-kp-OCSPSigning value.
NoCheck	The id-pkix-ocsp-nocheck extension.
KeyUsage	The Key Usage extension with the digitalSignature bit asserted.

For example, the following includes both the Extended Key Usage and id-pkix-ocsp-nocheck extensions, and omits the Key Usage extension:

```
+NoCheck+OCSPSigning
```

If no *Extensions* are specified, all three of the preceding extensions are included in the request.

Input Parameters

SubjectDN Required. The Subject DN for the request. *SubjectDN* must be an LDAP string DN. Acceptable DN values are: **CN** (CommonName), **OU** (OrganizationalUnit), **O** (Organization), **L** (Locality), and **C** (Country). If a SubjectDN value contains spaces, the value must be enclosed by quotation marks.

SignerNickname Required. The nickname of the OCSP signer.

RequestNickname Required. The nickname of the new request.

Output

None.

Return Values

0 Success.

1 Failure.

DeleteSigner

Prototype	`DeleteSigner` *`Nickname`*
Purpose	Deletes an OCSP signer from Validation Manager. Attempting to delete the default OCSP signer returns an error. To delete the default OCSP signer, first use **SetDefaultSigner** to specify a new default OCSP signer.
Input Options	None.
Input Parameters	*`Nickname`* Required. The nickname of the OCSP signer to delete.
Output	None.
Return Values	0 Success. 1 Other failures. 2 Attempted to delete the default OCSP signer.

Note: When an OCSP signer is deleted, all associated OCSP signer certificates and requests are also deleted. A known CA, using a deleted OCSP signer, reverts to using the default OCSP signer. If the known CA is configured to use an associated OCSP signer certificate, the known CA is reset to use the automatic selection function.

DeleteSignerCert

Prototype	`DeleteSignerCert` *`SignerNickname CertNickname`*
Purpose	Deletes a certificate of an OCSP signer.
Input Options	None.
Input Parameters	*`SignerNickname`* Required. The nickname of the OCSP signer whose certificate is to be deleted. *`CertNickname`* Required. The nickname of the OCSP signer certificate.
Output	None.
Return Values	0 Success. 1 Failure.

Important: A self-signed certificate cannot be deleted. If you attempt to delete a self-signed certificate an error occurs.

DeleteSignerCertRequest

Prototype	`DeleteSignerCertRequest` *`SignerNickname CertRequestNickname`*
Purpose	Deletes a certificate request object from Validation Manager.
Input Options	None.
Input Parameters	*`SignerNickname`* Required. The nickname of the OCSP signer. *`CertRequestNickname`* Required. The nickname of the certificate request to delete.
Output	None.
Return Values	0 Success. 1 Failure.

GetCert

Prototype	`GetCert [-format {pem}]` *`Nickname`*
Purpose	Retrieves a certificate.
Input Options	`-format` Optional. Specifies the format of the output. The option value **pem** specifies that the certificate is output in PEM-encoded format. If not specified, the certificate is output in binary.
Input Parameters	*`Nickname`* Required. The nickname of the certificate.
Output	The OCSP signer X.509 certificate.
Return Values	0 Success. 1 Failure.

GetCertRequest

Prototype	`GetCertRequest [-format {pem}]` *`Nickname`*
Purpose	Retrieves a PKCS #10 certificate request.
Options	`-format` Optional. Specifies the format of the output. The option value **pem** specifies that the certificate request is output in PEM-encoded format. If not specified, the certificate request is output in binary.
Input Parameters	*`Nickname`* Required. The nickname of the request to return.
Output	The named PKCS #10 certificate request.
Return Values	0 Success. 1 Failure.

GetDefaultSigner

Prototype	`GetDefaultSigner`
Purpose	Retrieves and displays the default OCSP signer.
Input Options	None.
Input Parameters	None.
Output	A VMSigner XML object representing the OCSP signer. The VMSigner object has the following schema:

```
<complexType name="VMSigner">
   <element name="nickname"/>
   <element name="creationDate"/>
   <element name="keysize"/>
   <element name="cryptoProvider"/>
   <element name="selfIssuedCertificateNickname"/
   <element name="defaultCertificateNickname"/>
</complexType>
```

Return Values	0 Success. 1 Failure.

GetSigner

Prototype	`GetSigner` ***`Nickname`***
Purpose	Obtains an OCSP signer object.
Input Options	None.
Input Parameters	***`Nickname`*** Required. The nickname of the OCSP signer.
Output	A VMSigner XML object representing the OCSP signer. The VMSigner object has the following schema:

```
<complexType name="VMSigner">
   <element name="nickname"/>
   <element name="creationDate"/>
   <element name="keysize"/>
   <element name="cryptoProvider"/>
   <element name="selfIssuedCertificateNickname"/>
   <element name="defaultCertificateNickname"/>
</complexType>
```

Return Values	0 Success. 1 Failure.

GetSignerCertList

Prototype	`GetSignerCertList` ***`SignerNickname`***
Purpose	Retrieves and displays a list of certificate nicknames associated with an OCSP signer.
Input Options	None.
Input Parameters	***`SignerNickname`*** Required. The nickname of the OCSP signer whose certificates are to be retrieved.

Output	A VMNicknameList XML object containing the nicknames of the certificates for the specific OCSP signer.
	The VMNicknameList object has the following schema:
	`<complexType name="VMNicknameList">`
	`  <element name="vmObjectClass"/>`
	`  <element name="nicknames" minOccurs="0" maxOccurs="*"/>`
	`</complexType>`

Return Values	0 Success. 1 Failure.

GetSignerCertRequestList

Prototype	`GetSignerCertRequestList` ***`Nickname`***
Purpose	Retrieves and displays a list of certificate request nicknames associated with an OCSP signer.
Input Options	None.
Input Parameters	***`Nickname`*** Required. The nickname of the OCSP signer whose request certificates are to be retrieved.

Output	A VMNicknameList XML object containing the nicknames of the certificate requests for the specific OCSP signer.
	The VMNicknameList object has the following schema:
	`<complexType name="VMNicknameList">`
	`  <element name="vmObjectClass"/>`
	`  <element name="nicknames" minOccurs="0" maxOccurs="*"/>`
	`</complexType>`

Return Values	0 Success. 1 Failure.

GetSignerList

Prototype `GetSignerList`

Purpose Obtains a list of OCSP signer nicknames.

Input Options None.

Input Parameters None.

Output A VMNicknameList XML object containing the nicknames of all OCSP signers in Validation Manager.

The VMNicknameList object has the following schema:

```
<complexType name="VMNicknameList">
   <element name="objectClass"/>
   <element name="nicknames" minOccurs="0" maxOccurs="*"/>
</complexType>
```

Return Values

0 Success.

1 Failure.

ImportSignerCert

Prototype `ImportSignerCert [-overwrite] SignerNickname CertNickname {FileName | -}`

Purpose Imports a certificate for an OCSP signer. The public key of the certificate must match the OCSP signer public key, or an error is returned. An error is also returned if the OCSP signer already has a certificate issued by the same CA. (In this case, the -overwrite option may be used to replace the certificate.)

Input Options

`-overwrite` Optional. Specifies that if a certificate has already been imported for the OCSP signer with the same issuing CA, the old certificate is replaced with the new one.

Input Parameters

`SignerNickname` Required. The nickname of the OCSP signer.

`CertNickname` Required. The nickname of the new certificate of the OCSP signer.

`FileName` | `-` Required. One of the following must be specified:

- *`FileName`* The name of the file containing the certificate. The file contents may be PEM-encoded or the raw DER of either a single certificate or PKCS #7 message containing a chain of certificates.
- `-` or blank Read the certificate from standard input, in any of the same formats as *`FileName`*.

Output None.

Return Values

0 Success.

1 Other failures.

2 Public key mismatch.

3 Existing certificate with same CA issuer.

RenewSignerSelfCert

Prototype	`RenewSignerSelfCert` ***`Nickname`***
Purpose	Renews the self-signed certificate of an OCSP signer. Allows an administrator to manually renew the certificate.
Input Options	None.
Input Parameters	***`Nickname`*** Required. The nickname of the OCSP signer whose self-signed certificate should be renewed.
Output	None.
Return Values	0 Success. 1 Failure.

SetSignerCertNickname

Prototype	`SetSignerCertNickname` ***`OldNickname NewNickname`***
Purpose	Changes the nickname of an OCSP signer certificate.
Input Options	None.
Input Parameters	***`OldNickname`*** Required. The current nickname of the OCSP signer certificate. ***`NewNickname`*** Required. The new nickname for the OCSP signer certificate.
Output	None.
Return Values	0 Success. 1 Failure.

SetSignerDefaultCert

Prototype	`SetSignerDefaultCert` ***`SignerNickname CertNickname`***
Purpose	Sets the default certificate of an OCSP signer.
Input Options	None.
Input Parameters	***`SignerNickname`*** Required. The nickname of the OCSP signer whose default certificate is to be set. ***`CertNickname`*** Required. The nickname of the OCSP signer default certificate. The certificate must be one of the current certificates of the OCSP signer or an error is returned.
Output	None.
Return Values	0 Success. 1 Failure.

SetSignerNickname

Prototype	`SetSignerNickname` ***`OldNickname NewNickname`***
Purpose	Changes the nickname of an OCSP signer.
Input Options	None.
Input Parameters	***`OldNickname`*** Required. The current nickname of the OCSP signer.
	`NewNickname` Required. The new nickname of the OCSP signer.
Output	None.
Return Values	0 Success.
	1 Failure.

SetSignerPassphrase

Prototype	`SetSignerPassphrase` ***`Nickname OldPassphrase NewPassphrase`***
Purpose	Changes the passphrase of a software-based OCSP signer.
Input Options	None.
Input Parameters	***`Nickname`*** Required. The nickname of the OCSP signer. An error is returned if the OCSP signer is hardware-based.
	`OldPassphrase` Required. The old passphrase of the software-based OCSP signer.
	`NewPassphrase` Required. The new passphrase of the software-based OCSP signer.
Output	None.
Return Values	0 Success.
	1 Other failures.
	2 Signer is hardware-based.

ViewCertInfo

Prototype	`ViewCertInfo` ***`Nickname`***
Purpose	Displays general information about a certificate.
Input Options	None.
Input Parameters	***`Nickname`*** Required. The nickname of the certificate.

Output	A VMCertificate XML object containing general information about the certificate.
	The VMCertificate object has the following schema:
	`<complexType name="VMCertificate">`
	`  <element name="nickname"/>`
	`  <element name="issuerNickname"/>`
	`  <element name="signerNickname"/>`
	`  <element name="issuerDNString"/>`
	`  <element name="subjectDNString"/>`
	`  <element name="notBefore"/>`
	`  <element name="notAfter"/>`
	`</complexType>`

Return Values	0 Success. 1 Failure.

6 Status Source Commands

This chapter describes the various status source commands used to control and configure status sources.

Commands

The following commands are used to manage and configure status sources:

- CreateLDAPStatusSource
- CreateOCSPStatusSource
- CreateRLStatusSource
- DeleteStatusSource
- GetStatusSource
- GetStatusSourceList
- SetStatusSourceGracePeriod
- SetStatusSourceLDAPAttributes
- SetStatusSourceLDAPMissingObjectStatus
- SetStatusSourceLDAPReasonCodes
- SetStatusSourceLDAPStatusCodes
- SetStatusSourceLDAPStatusInfoObjectClass
- SetStatusSourceLDAPUseCertificatePresence
- SetStatusSourceNickname
- SetStatusSourceOCSPProxyMode
- SetStatusSourceOCSPServerAuthMode
- SetStatusSourceOCSPServerCert
- SetStatusSourceOCSPSigner
- SetStatusSourceOCSPSignRequests
- SetStatusSourceOCSPStatusCheck
- SetStatusSourceOCSPUseNonces
- SetStatusSourceRefreshTime
- SetStatusSourceRLRefreshMode
- SetStatusSourceTLSServerAuthMode
- SetStatusSourceTLSServerCert
- SetStatusSourceTLSStatusCheck
- SetStatusSourceURL

CreateLDAPStatusSource

Prototype

```
CreateLDAPStatusSource [-host Host] [-port Port][-path Path]
[-gracePeriod GracePeriod][-refreshTimeValue Value]
[-ldapStatusInfoObjectClass ObjectClass]
[-ladpMissingObjectStatus {good | revoked | unknown}]
[-ldapUseCertificatePresence {true | false}]
[-ldapSerialNumberAttribute Attribute]
[-ldapCertStatusAttribute Attribute]
[-ldapReasonCodeAttribute Attribute]
[-ladpDateAttribute Attribute]
[-ldapTimeAttribute Attribute]
[-ldapDateTimeAttribute Attribute]
[-ldapUserCertAttribute Attribute]
[-ldapGoodStatusCodes Codes] [-ldapRevokedStatusCodes Codes]
[-ldapKeyCompromiseReasonCode Code]
[-ldapCACompromiseReasonCode Code]
[-ldapAffiliationChangedReasonCode Code]
[-ldapSupersededReasonCode Code]
[-ldapOperationCessationReasonCode Code]
[-ldapPrivilegeWithdrawnReasonCode Code]
[-ldapCertHoldReasonCode Code] Nickname
```

Purpose

Creates a new LDAP-based status source. If none exist, the status source is automatically assigned as the default status source.

Input Options

`-host Host` Optional. The DNS hostname, FQDN, or IP address of the status source's server. If not specified, the default is **localhost**.

`-port Port` Optional. The TCP port number of the status source. If not specified, the default is **389**.

`-path Path` Optional. The path prefix of the LDAP-based status source.

`-gracePeriod GracePeriod` Optional. The grace period in seconds.

`-refreshTimeValue Value` Optional. The refresh time value. *Value* is the number of seconds after the status data has last been retrieved.

`-ldapStatusInfoObjectClass ObjectClass` Optional. The name of the LDAP object class that contains the LDAP status information.

`-ldapMissingObjectStatus {good | revoked | unknown}` Optional. The status value returned when an LDAP object cannot be found using a certificate's serial number.

`-ldapUseCertificatePresence {true | false}` Optional. Specifies whether or not status is determined by certificates being present in Validation Manager. If not specified, the default is **false**.

`-ldapSerialNumberAttribute Attribute` Optional. The name of the LDAP attribute employed for searching by a certificate serial number.

`-ldapCertStatusAttribute Attribute` Optional. The name of the LDAP attribute that contains a status value for the certificate. An error occurs if this option is specified when -ldapUseCertificatePresence is set to true.

`-ldapReasonCodeAttribute` ***`Attribute`*** Optional. The name of the LDAP attribute that contains a reason code. An error occurs if this option is specified when -ldapUseCertificatePresence is set to true.

`-ldapDateAttribute` ***`Attribute`*** Optional. The name of the LDAP attribute that contains the date when the certificate's status was last changed. An error occurs if this option is specified when -ldapUseCertificatePresence is set to true.

`-ldapTimeAttribute` ***`Attribute`*** Optional. The name of the LDAP attribute that contains the time when the certificate's status was last changed. An error occurs if this option is specified when -ldapUseCertificatePresence is set to true.

`-ldapDateTimeAttribute` ***`Attribute`*** Optional. The name of the LDAP attribute that contains the date and time when the certificate's status was last changed. This value takes precedence over values set using -ldapDateAttribute and -ldapTimeAttribute options. An error occurs if this option is specified when -ldapUseCertificatePresence is set to true.

`-ldapUserCertAttribute` ***`Attribute`*** Optional. The name of the LDAP attribute that contains the user certificate. An error occurs if this option is specified when -ldapUseCertificatePresence is set to false.

`-ldapGoodStatusCodes` ***`Codes`*** Optional. Reason code maps to `good`, delimited by “,”. An error occurs if this option is specified when -ldapUseCertificatePresence is set to true.

`-ldapRevokedStatusCodes` ***`Codes`*** Optional. Reason code maps to `revoked`, delimited by “,”. An error occurs if this option is specified when -ldapUseCertificatePresence is set to true.

`-ldapKeyCompromiseReasonCode` ***`Code`*** Optional. Reason code maps to `key compromise`. An error occurs if this option is specified when -ldapUseCertificatePresence is set to true.

`-ldapCACompromiseReasonCode` ***`Code`*** Optional. Reason code maps to `ca compromise`. An error occurs if this option is specified when -ldapUseCertificatePresence is set to true.

`-ldapAffiliationChangedReasonCode` ***`Code`*** Optional. Reason code maps to `affiliation changed`. An error occurs if this option is specified when -ldapUseCertificatePresence is set to true.

`-ldapSupersededReasonCode` ***`Code`*** Optional. The reason code which maps to `superseded`. An error occurs if this option is specified when -ldapUseCertificatePresence is set to true.

`-ldapOperationCessationReasonCode` ***`Code`*** Optional. Reason code maps to `operation cessation`. An error occurs if this option is specified when -ldapUseCertificatePresence is set to true.

`-ldapPrivilegeWithdrawnReasonCode` ***`Code`*** Optional. Reason code maps to `privilege withdrawn`. An error occurs if this option is specified when -ldapUseCertificatePresence is set to true.

	`-ldapCertHoldReasonCode` *`Code`* Optional. Reason code maps to `certificate hold`. An error occurs if this option is specified when -ldapUseCertificatePresence is set to true.
Input Parameters	*`Nickname`* Required. The nickname of the new LDAP-based status source.
Output	None.
Return Values	0 Success. 1 Failure.

CreateOCSPStatusSource

Prototype

```
CreateOCSPStatusSource [-scheme Scheme] [-host Host]
[-port Port] [-path Path] [-refreshTimeType Type]
[-refreshTimeValue Value] [-gracePeriod GracePeriod]
[-ocspProxyMode ProxyMode] [-ocspStatusCheck StatusCheck]
[-ocspNonces {true | false}]
[-ocspSignRequests {true | false}] Nickname
```

Purpose

Creates a new OCSP-based status source. If none exist, this new status source is automatically assigned as the default status source.

Input Options

`-scheme` *`Scheme`* Optional. The scheme for the OCSP-based status source. *`Scheme`* can be `HTTP` or `HTTPS`. If not specified, the default is **HTTP**.

`-host` *`Host`* Optional. The DNS hostname, FQDN, or IP address of the server of the OCSP-based status source. If not specified, the default is **localhost**.

`-port` *`Port`* Optional. The TCP port number of the OCSP-based status source. If not specified, the default is **80**.

`-path` *`Path`* Optional. The path prefix of the OCSP-based status source.

`-refreshTimeType` *`Type`* Optional. The refresh time type, either `SinceFetch` or `NextUpdate`. If not specified, the default is **NextUpdate**.

`-refreshTimeValue` *`Value`* Optional. The refresh time value. If refreshTimeType is NextUpdate, the value is the number of seconds before the next update. If refreshTimeType is SinceFetch, the value is the number of seconds after the revocation list has been fetched.

`-gracePeriod` *`GracePeriod`* Optional. The grace period in seconds (starting from when the refresh time elapses).

`-ocspProxyMode` *`ProxyMode`* Optional. The method, either `Proxy` or `Forward`, by which certificate status is retrieved from the OCSP-based status source. If not specified, the default is **Forward**.

`-ocspStatusCheck` *`StatusCheck`* Optional. Specifies when, either `Always`, `Never`, or `NoCheck`, to check the status of the OCSP-based status source's response signing certificate. `NoCheck` specifies that status is only checked if id-pkix-ocsp-nocheck is absent from the OCSP-based status source's certificate. If not specified, the default is **Never**.

`-ocspNonces {true | false}` Optional. Specifies whether Validation Manager uses nonces in its OCSP responses. The default is **true**.

`-OCSPSignRequests {true | false}` Optional. Specifies whether Validation Manager signs OCSP requests. The default is **false**.

Input Parameters

`Nickname` Required. The nickname of the new OCSP-based status source.

Output

None.

Return Values

0 Success.

1 Failure.

CreateRLStatusSource

Prototype

```
CreateRLStatusSource [-scheme Scheme] [-host Host]
[-port Port] [-path Path]
[gracePeriod GracePeriod] [-refreshTimeType Type]
[-refreshTimeValue Value] [-refreshMode Mode] Nickname
```

Purpose

Creates a new revocation list-based status source. If none exist, this new status source is automatically assigned as the default status source.

Input Options

`-scheme` ***Scheme*** Optional. The scheme for the revocation list-based status source. ***Scheme*** can be `LDAP`, `Manual`, `HTTP`, `HTTPS`, `LDAPT`, or `SYNC`. If not specified, the default is **LDAP**.

`-host` ***Host*** Optional. The DNS hostname, FQDN, or IP address of the server of the revocation list-based status source. This option is required for all schemes except Manual. If not specified, the default is **localhost**.

`-port` ***Port*** Optional. The TCP port number of the revocation list-based status source. If not specified, the default is **389**.

`-path` ***Path*** Optional. The path prefix of the revocation list-based status source. ***Path*** is typically an LDAP DN. This option is ignored for status sources with a scheme of Manual.

`-gracePeriod` ***GracePeriod*** Optional. The grace period in seconds. This option is ignored for status sources with a scheme of Manual.

`-refreshTimeType` ***Type*** Optional. The refresh time type. ***Type*** can be either `SinceFetch` or `NextUpdate`. If not specified, the default is **NextUpdate**. This option is ignored for status sources with a scheme of Manual.

`-refreshTimeValue` ***Value*** Optional. The refresh time value. If refreshTimeType is NextUpdate, the value is the number of seconds before the next update. If refreshTimeType is SinceFetch, the value is the number of seconds after the revocation list has been imported. This option is ignored for status sources with a scheme of Manual.

`-RefreshMode` ***Mode*** Optional. The refresh mode, either `proactive` or `reactive`. If not specified, the default is **reactive**.

Input Parameters

Nickname Required. The nickname of the new revocation list-based status source.

Output

None.

Return Values

0 Success.

1 Failure.

DeleteStatusSource

Prototype	`DeleteStatusSource` *`Nickname`*
Purpose	Deletes a status source from Validation Manager. Attempting to delete the default status source returns an error. To delete the default status source, first use **SetDefaultStatusSource** to specify a different status source as the new default status source.
Input Options	None.
Input Parameters	*`Nickname`* Required. The nickname of the status source to delete.
Output	None.
Return Values	0 Success. 1 Other failures. 2 Attempted to delete the default status source.

Note: If a CA status source is deleted, the CA status source is set to the default status source.

GetStatusSource

Prototype	`GetStatusSource` ***`Nickname`***
Purpose	Retrieves and displays a status source.
Input Options	None.
Input Parameters	***`Nickname`*** Required. The nickname of the status source object returned.
Output	A VMStatusSource XML object representing the status source.

The VMStatusSource object has the following schema:

```
<complexType name="VMStatusSource">
   <element name="nickname"/>
   <element name="type"/>
   <element name="scheme"/>
   <element name="authority"/>
   <element name="pathPrefix"/>
   <element name="gracePeriod"/>
   <element name="refreshTimeType" minOccurs="0"/>
   <element name="refreshTimeValue" minOccurs="0"/>
   <element name="tlsServerAuthMode" minOccurs="0"/>
   <element name="tlsServerCertNickname" minOccurs="0"/>
   <element name="tlsStatusCheck" minOccurs="0"/>
```

Revocation list specific properties:

```
   <element name="rlRefreshMode" minOccurs="0"/>
```

OCSP specific properties:

```
   <element name="ocspProxyMode" minOccurs="0"/>
   <element name="ocspUseNonces" minOccurs="0"/>
   <element name="ocspSignRequests" minOccurs="0"/>
   <element name="ocspServerAuthMode" minOccurs="0"/>
   <element name="ocspServerCertNickname" minOccurs="0"/>
   <element name="ocspStatusCheck" minOccurs="0"/>
   <element name="ocspForwardSignerNickname" minOccurs="0"/>
   <element name="ocspForwardSignerCertNickname"
   minOccurs="0"/
```

LDAP specific properties:

```
   <element name="ldapStatusInfoObjectClass" minOccurs="0"/>
   <element name="ldapMissingObjectStatus" minOccurs="0"/>
   <element name="ldapUserCertificatePresence"
   minOccurs="0"/>
   <element name="ldapSerialNumberAttribute" minOccurs="0"/>
   <element name="ldapCertStatusAttribute" minOccurs="0"/>
   <element name="ldapReasonCodeAttribute" minOccurs="0"/>
   <element name="ldapDateAttribute" minOccurs="0"/>
   <element name="ldapTimeAttribute" minOccurs="0"/>
   <element name="ldapDateTimeAttribute" minOccurs="0"/>
   <element name="ldapUserCertAttribute" minOccurs="0"/>
   <element name="ldapGoodStatusCodes" minOccurs="0"/>
   <element name="ldapRevokedStatusCodes" minOccurs="0"/>
   <element name="ldapKeyCompromiseReasonCode"
   minOccurs="0"/>
   <element name="ldapCACompromiseReasonCode"
```

```
   minOccurs="0"/>
   <element name="ldapAffiliationChangedReasonCode"
   minOccurs="0"/>
   <element name="ldapSupersededReasonCode" minOccurs="0"/>
   <element name="ldapOperationCessationReasonCode"
   minOccurs="0"/>
   <element name="ldapPrivilegeWithdrawnReasonCode"
   minOccurs="0"/>
   <element name="ldapCertHoldReasonCode" minOccurs="0"/>
   < minOccurs="0"/>
</complexType>
```

Return Values

0 Success.

1 Failure.

GetStatusSourceList

Prototype `GetStatusSourceList`

Purpose Obtains a list of status source nicknames.

Input Options None.

Input Parameters None.

Output A VMNicknameList XML object containing the nicknames of all the status sources in Validation Manager.

The VMNicknameList object has the following schema:

```
<complexType name="VMNicknameList">
   <element name="nicknames" minOccurs="0" maxOccurs="*"/>
</complexType>
```

Return Values

0 Success.

1 Failure.

SetStatusSourceNickname

Prototype `SetStatusSourceNickname` ***`OldNickname NewNickname`***

Purpose Changes the nickname of a status source.

Input Options None.

Input Parameters

`OldNickname` Required. The current nickname of the status source.

`NewNickname` Required. The new nickname of the status source.

Output None.

Return Values

0 Success.

1 Failure.

SetStatusSourceGracePeriod

Prototype	`SetStatusSourceGracePeriod` ***`GracePeriod Nickname`***
Purpose	Sets the grace period of a status source.
Input Options	None.
Input Parameters	***`GracePeriod`*** Required. The grace period in seconds.
	`Nickname` Required. The status source nickname.
Output	None.
Return Values	0 Success.
	1 Failure.

SetStatusSourceLDAPAttributes

Prototype

```
SetStatusSourceLDAPAttributes
[-ldapSerialNumberAttribute Attribute]
[-ldapCertStatusAttribute Attribute]
[-ldapReasonCodeAttribute Attribute]
[-ladpDateAttribute Attribute]
[-ldapTimeAttribute Attribute]
[-ldapDateTimeAttribute Attribute]
[-ldapUserCertAttribute Attribute] Nickname
```

Purpose

Sets the LDAP attribute names for status source properties.

Input Options

At least one of the following must be specified:

`-ldapSerialNumberAttribute` *`Attribute`* Optional. The name of the LDAP attribute employed for searching by a certificate's serial number.

`-ldapCertStatusAttribute` *`Attribute`* Optional. The name of the LDAP attribute that contains a status value for the certificate. An error occurs if this option is specified when the -ldapUseCertificatePresence option is set to true in the **CreateLDAPStatusSource** command.

`-ldapReasonCodeAttribute` *`Attribute`* Optional. The name of the LDAP attribute that contains a reason code. An error occurs if this option is specified when the -ldapUseCertificatePresence option is set to true in the **CreateLDAPStatusSource** command.

`-ldapDateAttribute` *`Attribute`* Optional. The name of the LDAP attribute that contains the date when the certificate status was last changed. An error occurs if this option is specified when the -ldapUseCertificatePresence option is set to true in the **CreateLDAPStatusSource** command.

`-ldapTimeAttribute` *`Attribute`* Optional. The name of the LDAP attribute that contains the time when the certificate status was last changed. An error occurs if this option is specified when the -ldapUseCertificatePresence option is set to true in the **CreateLDAPStatusSource** command.

`-ldapDateTimeAttribute` *`Attribute`* Optional. The name of the LDAP attribute that contains the date and time when the certificate status was last changed. This value takes precedence over values set using the -ldapDateAttribute and -ldapTimeAttribute options. An error occurs if this option is specified when the -ldapUseCertificatePresence option is set to true in the **CreateLDAPStatusSource** command.

`-ldapUserCertAttribute` *`Attribute`* Optional. The name of the LDAP attribute that contains the user certificate. An error occurs if this option is specified when the -ldapUseCertificatePresence option is set to true in the **CreateLDAPStatusSource** command.

Input Parameters

`Nickname` Required. The status source nickname. An error occurs if the status source is not LDAP-based.

Output

None.

Return Values

0 Success.

1 Other failures.

2 Status source not LDAP-based.

SetStatusSourceLDAPMissingObjectStatus

Prototype	`SetStatusSourceLDAPMissingObjectStatus` *`Nickname`* `{good | revoked | unknown}`
Purpose	Sets the status value returned when an LDAP object cannot be found using a certificate serial number.
Input Options	None.
Input Parameters	*`Nickname`* Required. The status source nickname. An error occurs if the status source is not LDAP-based. `good | revoked | unknown` Required. The status to be returned when an LDAP object cannot be found using a certificate serial number.
Output	None.
Return Values	0 Success. 1 Other failures. 2 Status source not LDAP-based.

SetStatusSourceLDAPReasonCodes

Prototype

```
SetStatusSourceLDAPReasonCodes
[-ldapKeyCompromiseReasonCode Code]
[-ldapCACompromiseReasonCode Code]
[-ldapAffiliationChangedReasonCode Code]
[-ldapSupersededReasonCode Code]
[-ldapOperationCessationReasonCode Code]
[-ldapPrivilegeWithdrawnReasonCode Code]
[-ldapCertHoldReasonCode Code] Nickname
```

Purpose

Sets the LDAP reason codes. An error occurs if this option is specified when the -ldapUseCertificatePresence option is set to true in the **CreateLDAPStatusSource** command.

Input Options

At least one of the following must be specified:

`-ldapKeyCompromiseReasonCode` *`Code`* Optional. Reason code maps to `key compromise`.

`-ldapCACompromiseReasonCode` *`Code`* Optional. Reason code maps to `ca compromise`.

`-ldapAffiliationChangedReasonCode` *`Code`* Optional. Reason code maps to `affiliation changed`.

`-ldapSupersededReasonCode` *`Code`* Optional. Reason code maps to `superseded`.

`-ldapOperationCessationReasonCode` *`Code`* Optional. Reason code maps to `operation cessation`.

`-ldapPrivilegeWithdrawnReasonCode` *`Code`* Optional. Reason code maps to `privilege withdrawn`.

`-ldapCertHoldReasonCode` *`Code`* Optional. Reason code maps to `certificate hold`.

Input Parameters

`Nickname` Required. The status source nickname. An error occurs if the status source is not LDAP-based.

Output

None.

Return Values

0 Success.

1 Other failures.

2 Status source not LDAP-based.

SetStatusSourceLDAPStatusCodes

Prototype `SetStatusSourceLDAPStatusCodes [-ldapGoodStatusCodes Codes] [-ldapRevokedStatusCodes Codes] Nickname`

Purpose Sets the LDAP status codes. An error occurs if this option is specified when the -ldapUseCertificatePresence option is set to true in the **CreateLDAPStatusSource** command.

Input Options At least one of the following must be specified:

`-ldapGoodStatusCodes Codes` Optional. The status codes which map to `good`, delimited by ",".

`-ldapRevokedStatusCodes Codes` Optional. The status codes which map to `revoked`, delimited by ",".

Input Parameters `Nickname` Required. The status source nickname. An error occurs if the status source is not LDAP-based.

Output None.

Return Values 0 Success.

1 Other failures.

2 Status source not LDAP-based.

SetStatusSourceLDAPStatusInfoObjectClass

Prototype `SetStatusSourceLDAPStatusInfoObjectClass Nickname Class`

Purpose Sets the name of the LDAP object class that contains the LDAP status information.

Input Options None.

Input Parameters `Nickname` Required. The status source nickname. An error occurs if the status source is not LDAP-based.

`Class` Required. The name of the LDAP object class.

Output None.

Return Values 0 Success.

1 Failure.

SetStatusSourceOCSPProxyMode

Prototype `SetStatusSourceOCSPProxyMode Nickname Mode`

Purpose Sets an OCSP-based status source to retrieve certificate status using proxying or forwarding. An error is returned if the status source is not OCSP.

Input Options None.

Input Parameters `Nickname` Required. The status source nickname.

`Mode` Required. The method, either `Proxy` or `Forward`, by which certificate status is retrieved from the OCSP server.

Output None.

Return Values 0 Success.

1 Other failures.

2 Status source not OCSP-based.

SetStatusSourceOCSPServerAuthMode

Prototype `SetStatusSourceOCSPServerAuthMode` ***`Nickname AuthMode`***

Purpose Sets the server authentication mode for a status source that uses an OCSP-based scheme for status retrieval.

Input Options None.

Input Parameters ***`Nickname`*** Required. The status source nickname. An error occurs if the status source is not using a TLS-based scheme for status retrieval.

`AuthMode` Required. ***`AuthMode`*** can be one of the following:

`TrustedCert`	Uses the certificate of the remote server. An error occurs if the authentication mode is set to `TrustedCert`, but the remote server certificate does not yet exist.
`TargetCA`	Uses a CA with the TLSServerAuth purpose.

Output None.

Return Values 0 Success.

1 Other failures.

2 Status source not using an OCSP-based scheme for status retrieval.

SetStatusSourceOCSPServerCert

Prototype `SetStatusSourceOCSPServerCert` ***`Nickname`***
`[{`***`FileName`*** `| - | -none}]`

Purpose Specifies the certificate used to verify the signatures of the OCSP-based status source responses. An error occurs if the status source is not OCSP-based.

Input Options ***`FileName`*** `| - | -none` Optional. One of the following must be specified:

`FileName`	The name of a file containing the certificate. The file contents may be PEM-encoded or the raw DER of either a single certificate or PKCS #7 message containing a chain of certificates (in which case the first certificate in the chain is used as the CA certificate).
`-` or blank	Read the CA certificate from standard input, in any of the same formats as ***`FileName`***.
`-none`	A keyword specifying that a known CA certificate is used to verify the OCSP-based status source responses.

Input Parameters ***`Nickname`*** Required. The status source nickname.

Output None.

Return Values 0 Success.

1 Other failures.

2 Status source not OCSP-based.

SetStatusSourceOCSPSigner

Prototype `SetStatusSourceOCSPSigner [{`*`SignerCertNickname`* `| -default}] `*`Nickname`* `{`*`SignerNickname`* `| -default}`

Purpose Sets an OCSP signer and associated certificate to sign an OCSP-based status source's forwarded requests. SetStatusSourceOCSPSigner can be set for OCSP-based status sources that are not configured for forwarding; however, the command is not be executed.

Input Options *`SignerCertNickname`* | `-default` Optional. *`SignerCertNickname`* specifies the nickname of the certificate to use. `-default` specifies that the OCSP signer default certificate is used. If unspecified, the default certificate is used. If *`SignerNickname`* is `-default`, *`SignerCertNickname`* is not allowed and an error occurs.

Input Parameters *`Nickname`* Required. The status source nickname.

`SignerNickname` | `-default` Required. The OCSP signer nickname. `-default` specifies that the default OCSP signer is used.

Note: If -default is used instead of *SignerNickname*, do not use the *SignerCertNickname* | -default parameter.

Output None.

Return Values 0 Success.

1 Other failures.

2 Status source not OCSP-based.

SetStatusSourceOCSPSignRequests

Prototype `SetStatusSourceOCSPSignRequests `*`Nickname`* `{true | false}`

Purpose Sets whether requests are signed. An error occurs if the status source is not OCSP-based.

Input Options None.

Input Parameters *`Nickname`* Required. The status source nickname.

`true | false` Required. The desired state of the request signing. One of the following must be specified:

`true` Used.

`false` Not used.

Output None.

Return Values 0 Success.

1 Other failures.

2 Status source not OCSP-based.

SetStatusSourceOCSPStatusCheck

Prototype `SetStatusSourceOCSPStatusCheck` ***`Nickname StatusCheck`***

Purpose Sets when to check the status of the response-signing certificate of an OCSP-based status source. An error occurs if the status source is not OCSP-based.

Input Options None.

Input Parameters ***`Nickname`*** Required. The status source nickname.

`StatusCheck` Required. Must be either `Always`, `Never`, or `NoCheck`.

Output None.

Return Values 0 Success.

1 Other failures.

2 Status source not OCSP-based.

SetStatusSourceOCSPUseNonces

Prototype `SetStatusSourceOCSPUseNonces` ***`Nickname`*** `{true | false}`

Purpose Sets whether nonces should be used in the OCSP requests. An error occurs if the status source is not OCSP-based.

Input Options None.

Input Parameters ***`Nickname`*** Required. The status source nickname.

`true | false` Required. The desired state of the nonces. One of the following must be specified:

`true` Used.

`false` Not used.

Output None.

Return Values 0 Success.

1 Other failures.

2 Status source not of type OCSP forwarding.

SetStatusSourceRLRefreshMode

Prototype `SetStatusSourceRLRefreshMode` ***`Mode Nickname`***

Purpose Sets revocation list-based status refresh mode.

Input Options None.

Input Parameters ***`Mode`*** Required. The refresh mode. ***`Mode`*** can be either `proactive` or `reactive`. If not specified, the default is **reactive**.

`Nickname` Required. The revocation list-based status source nickname.

Output None.

Return Values 0 Success.

1 Other failures.

2 Status source not revocation list-based.

SetStatusSourceRefreshTime

Prototype `SetStatusSourceRefreshTime` ***`Type`*** `[`***`Value`***`]` ***`Nickname`***

Purpose Sets the refresh time of a status source.

Input Options None.

Input Parameters ***`Type`*** `[`***`Value`***`]` Required. The method of status source refresh, and optionally a value for the refresh time, if relevant. ***`Type`*** can be one of the following:

`NextUpdate`	***`Value`*** is the number of seconds before the next update. If not specified, ***`Value`*** default is **0**.
`SinceFetch`	***`Value`*** is the time in seconds before the next update, and is required. If not specified, ***`Value`*** default is **0**.

`Nickname` Required. The status source nickname.

Output None.

Return Values 0 Success.

1 Failure.

SetStatusSourceTLSServerAuthMode

Prototype `SetStatusSourceTLSServerAuthMode` ***`Nickname AuthMode`***

Purpose Sets the server authentication mode for a status source that uses a TLS-based scheme for status retrieval.

Input Options None.

Input Parameters ***`Nickname`*** Required. The status source nickname. An error occurs if the status source is not using a TLS-based scheme for status retrieval.

`AuthMode` Required. ***`AuthMode`*** must be one of the following:

`TrustedCert`	Uses the remote server certificate. An error occurs if the authentication mode is set to `TrustedCert`, but the server certificate does not yet exist.
`KnownCA`	Uses a known CA with the TLSServerAuth purpose.

Output None.

Return Values 0 Success.

1 Other failures.

2 Status source is not using a TLS-based scheme for status retrieval.

SetStatusSourceTLSServerCert

Prototype `SetStatusSourceTLSServerCert` ***Nickname*** {***FileName*** | - | -none}

Purpose Sets the status source server certificate to be trusted for a status source that uses a TLS-based scheme for certificate status retrieval.

Input Options None.

Input Parameters ***Nickname*** Required. The status source nickname.

FileName | - | -none Required. One of the following must be specified:

FileName	The name of the file containing the certificate. The file contents may be PEM-encoded or the raw DER of either a single certificate or PKCS #7 message containing a chain of certificates (in which case the first certificate in the chain is used as the certificate).
- or blank	Read the certificate from standard input, in any of the same formats as ***FileName***.
-none	A keyword specifying that the certificate is deleted and the TLS server authentication mode changed to KnownCA.

Output None.

Return Values 0 Success.

1 Other failures.

2 Status source not using a TLS-based scheme for status retrieval.

SetStatusSourceTLSStatusCheck

Prototype `SetStatusSourceTLSStatusCheck` ***Nickname StatusCheck***

Purpose Sets when to check the status of response-signing certificate of a status source. The status source must use a TLS-based scheme for status retrieval.

Input Options None.

Input Parameters ***Nickname*** Required. The status source nickname. An error occurs if the status source is not using a TLS-based scheme for status retrieval.

StatusCheck Required. ***StatusCheck*** must be one of the following:

Always	Status is always checked.
Never	Status is never checked.

Output None.

Return Values 0 Success.

1 Failure.

SetStatusSourceURL

Prototype

```
SetStatusSourceURL [-scheme Scheme] [-host Host]
[-port Port] [-path Path] [-url Url] Nickname
```

Purpose

Sets the URL settings of a status source.

Input Options

`-scheme` ***Scheme*** Optional. The scheme for the status source. ***Scheme*** can be either `Manual`, `HTTPS`, `LDAP`, `LDAPT`, or `SYNC` for revocation list-based status sources, or `HTTP` or `HTTPS` for OCSP-based status sources. If not specified, the default is **LDAP** for revocation list-based status sources and **HTTP** for OCSP-based status sources.

`-host` ***Host*** Optional. The DNS hostname, FQDN, or IP address of the status source server. ***Host*** can be omitted for revocation list-based status sources with a scheme of Manual.

`-port` ***Port*** Optional. The TCP port number of the status source.

`-path` ***Path*** Optional. The path prefix of the status source. In the case of revocation list-based status sources, ***Path*** is typically an LDAP DN and ignored for status sources with a scheme of Manual. The current path value can be erased by using the keyword `-none`.

`-url` ***Url*** Optional. The full URL address of the status source. If specified, all other input options are ignored.

Input Parameters

Nickname Required. The status source nickname.

Output

None.

Return Values

0 Success.

1 Failure.

SetStatusSourceLDAPUseCertificatePresence

Prototype

```
SetStatusSourceLDAPUseCertificatePresence Nickname
{true | false}
```

Purpose

Sets whether or not LDAP determines status based upon certificate presence.

Input Options

None.

Input Parameters

Nickname Required. The status source nickname. An error occurs if the status source is not LDAP-based.

Output

None.

Return Values

0 Success.

1 Other failures.

2 Status source not LDAP-based.

Glossary

Abstract Syntax Notation One (ASN.1)

An International Standards Organization (ISO) standard notation for defining the syntax of information data. It defines a number of simple data types and specifies a notation for referencing these types and for specifying the values of these types.

Administrator

A person, possibly with an end-entity certificate, who has access to the administration interface of Validation Manager. Administrator tasks may include installing RSA Validation Manager, setting up the OCSP Signers, status sources, and CAs for which Validation Manager serves status.

ARL

See **Authority Revocation List**.

ASN.1

See **Abstract Syntax Notation One**.

Audit Log

A tamper resistant log Validation Manager uses to record operational and configuration changing events.

Authentication

A process by which people or applications who receive a certificate can verify the identity of the certificate owner and the validity of the certificate. Certificates identify the author of a message or entity, such as a web server or client.

Authority Revocation List (ARL)

A list of CA certificates that a CA has revoked or suspended. You can use ARLs to check the status of CA certificates offline.

Backend

A collection of functions within the OCSP server that are invoked when an OCSP request is made.

Base64

See **Privacy Enhanced Mail (PEM) Format**.

CA

See **Certificate Authority**.

CA Certificate

A certificate that identifies a CA. When a CA issues a certificate to a client, a server, or other entity, the CA private key signs the certificate. You can verify the signature using the public key in the CA certificate. See also **Root CA**.

Cache Lifetime

A period of time during which Validation Manager can reuse a previously generated and signed response.

CA Purposes

CA purposes define the use of the CA within Validation Manager. For example, if the purpose of a CA is to provide certificate status, Validation Manager processes status requests for certificates issued by that CA. By default, Validation Manager assigns the following purposes to a CA: provide certificate status, verify OCSP clients, and verify remote secure servers.

Certificate

Certificates verify the identity of an individual, organization, web server, or hardware device. They also ensure non-repudiation in business transactions, as well as enable confidentiality through the use of public-key encryption. PKI uses three main kinds of certificates: CA certificates, server certificates (also referred to as SSL certificates), and end-entity certificates.

Certificate Authority (CA)

An entity that issues and manages certificates within a PKI. You create and manage CAs using a CA software application, such as RSA Certificate Manager.

Certificate Extension

See **X.509 v3 Certificate Extension**.

Certificate Policy (CP)

A policy that explains the conditions and limitations of use for a digital certificate.

Certificate Revocation List (CRL)

A list of revoked and suspended certificates (CA or end-entity) for a particular CA. You can use CRLs to check the status of certificates offline. See also **Complete CRL** and **Delta CRL**.

Certification Practice Statement (CPS)

A statement of an organization's security policies for the issuance and management of certificates.

Client Certificate

See **End-Entity Certificate**.

Clustering

The use of multiple Validation Manager installations and separate machines to form what is a highly available, fault tolerant system. To an OCSP client, the cluster appears to be one system. You can configure Validation Manager to support clustering with the addition of a load balancer.

Within the cluster, there is one primary installation, where Validation Manager is administered, and any number of secondary installations. Both primary and secondary installations can receive OCSP requests from the load balancer.

Complete CRL

A list that contains the serial numbers of certificates that a CA has revoked or suspended.

CRL

See **Certificate Revocation List**.

Cryptographic Provider

The library Validation Manager uses for private-key cryptographic operations (such as key pair generation and digital signatures). The method is either software-based or hardware-based (using nCipher).

Delta CRL

A list that contains the serial numbers of certificates that a CA has suspended, reinstated, or revoked since the last complete CRL.

Digital Signature Algorithm (DSA)

A digital signature algorithm used in the Digital Signature Standard (DSS) created by the U.S. government. For more information, see the standard designation **FIPS 186-2+ChangeNotice** at http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf.

Distinguished Encoding Rules (DER)

An ASN.1 encoding standard used for signature calculation for end-entity certificates and revocation lists (that is, CRLs, delta CRLs and ARLs). Also known as **Binary**.

Distinguished Name (DN)

The combination of attributes in a certificate forms the certificate DN. The following attributes are the most commonly used attributes:

- Common Name
- User ID
- E-mail Address
- Organizational Unit
- Organization
- Locality
- State or Province
- Country
- Domain Component

To avoid potential problems, all CAs in the PKI, including trusted CAs, must have a unique DN.

End-Entity Certificate

A certificate issued to an entity that cannot itself issue certificates (that is, the entity is not a CA). Because the entity that requests such a certificate is sometimes referred to as the client, end-entity certificates are sometimes called client certificates.

End User (or End-Entity)

An individual, group, or organization that either requests or holds an end-entity certificate. An end user can also be an individual who requests an end-entity certificate for a hardware device (such as a router), a server, a software application, or a piece of code. An end user that requests a certificate is sometimes called a requestor. An end user that is issued a certificate is sometimes called a certificate owner, certificate subject, or end-entity. An end user that relies upon someone else's certificate to verify that person's identity is sometimes called an end user, certificate user, or relying party.

Enterprise

An organization that uses computers and applications. In general use, this term applies to businesses or organizations that operate on a large scale. These organization's applications are often referred to as enterprise applications.

Entity

A person, organization, or device (such as a router). In a PKI, an entity is anyone or anything you can issue a certificate to.

Expired Status Data

The freshness of a revocation list or status value in a status source in Validation Manager. A list or status value is expired once the refresh time plus the grace period elapse.

Extension

See **X.509 v3 Certificate Extension**.

FIPS 140-1 Level 2 & 3
FIPS 140-2 Level 2 & 3

A standard developed by the National Institute of Standards and Technology (NIST) for implementation of cryptographic modules. Level 3 provides greater security than Level 2.

Firewall

A system designed to prevent unauthorized access to or from a private network.

Fresh Status Data

The freshness of a revocation list or status value. A list or status value is fresh if the refresh time of its status source has not elapsed. For example, if the refresh time for a status source in Validation Manager to retrieve a new list or status value has not arrived, the list or status value within the Validation Manager database for that status source is considered fresh.

Forwarding

An OCSP client request triggers Validation Manager to send a second OCSP request to a remote OCSP server and use the remote server's response to construct its own response.

Fully Qualified Domain Name (FQDN)

The full name of a system, consisting of its local host name and its domain name. For example, "venera" is a host name and "venera.isi.edu" is the FQDN.

Grace Period

A period of time during which Validation Manager can reuse a stale status value, but must also attempt to obtain a newer status value. For example, when a remote OCSP response is in its grace period, and Validation Manager cannot fetch a new response for the same certificate, Validation Manager uses the status value from the previous OCSP response. The grace period specifies how long after the refresh time that the previous status value is valid. A status value expires once its refresh time and grace period elapse.

Hardware Security Module (HSM)

The module that performs cryptographic functions and stores cryptographic keys in a secure fashion.

Hypertext Transfer Protocol (HTTP)

A set of rules for exchanging files (text, graphic images, sound, video, and other multimedia files) on the World Wide Web.

Web browsers are HTTP clients that send requests to server machines. Users enter page requests by either typing a URL or clicking a hypertext link. The browser builds an HTTP request for the user and sends it to the Internet Protocol (IP) address indicated in the URL. The HTTP daemon in the destination server receives the request and, after any necessary processing, returns the requested page. See also **HTTPS**.

HTTPS

HTTP over an SSL/TLS connection.

Identity Certificate

A certificate that links a public key value to a real-world entity such as a person, a computer, or a web server. Server certificates, CA certificates, and most end-entity certificates are all examples of identity certificates.

Key Pair

A public key and a private key associated with an entity that needs to authenticate its identity electronically or to sign or encrypt data. The public key is published, and the corresponding private key is kept secret. You can only decrypt data encrypted with the public key with the private key.

Key Size

The size (in bits) of the key pair used to sign status responses. A larger key size provides greater security. Validation Manager supports 1024, 2048, and 4096 bit keys.

Known CA

A CA that is known to the system. A CA becomes known to the system when you import the CA certificate into the system.

LDAP Directory

An LDAP-based directory is a database. You can search for and retrieve attribute-value pairs. You can configure directories to use (or support) authentication and access control protection. The schema of a directory describes the objects in the directory.

LDAPS

LDAP over SSL/TLS connection. See also **StartTLS**.

Lightweight Directory Access Protocol (LDAP)

The standard Internet protocol for accessing directory servers over a network. LDAP is a “lightweight” (smaller amount of overhead) version of Directory Access Protocol (DAP), which is part of X.500, a standard for directory services in a network. There are two currently supported versions, LDAP versions 2.0 and 3.0. See also **LDAPS** and **StartTLS**.

Locally Revoked Certificate

A certificate that is revoked within Validation Manager, but not revoked by a CA. A locally revoked certificate is not listed on a revocation list. Validation Manager returns a status of revoked for all enquires on the status of this certificate.

Load Balancer

A software or hardware product that routes incoming data to one of a number of possible resources or applications.

Nickname

A user-friendly character string that uniquely identifies a CA, OCSP signer, status source, or certificate recipient.

Nonces

Random numbers used in security protocols to prove that a message is part of a current message exchange.

Non-repudiation

A concept that prevents the author of a message from denying having created that message at a later date (that is, repudiation cannot occur). Digital signatures help ensure the non-repudiation of transactions.

OCSP

See **Online Certificate Status Protocol**.

OCSP Client

The entity that issues a certificate status request to an OCSP Responder. The OCSP client suspends acceptance of the certificate until the responder returns the certificate status.

OCSP Forwarding

One of two ways Validation Manager queries a remote OCSP server. During OCSP forwarding, a client request triggers Validation Manager to send a second OCSP request to a remote OCSP server, then use the remote server response to construct a second response to send to the client. Validation Manager can also use OCSP proxying to query a remote OCSP server.

OCSP Performance

The number of OCSP responses per second an OCSP server can process.

OCSP Proxying

One of two ways Validation Manager queries a remote OCSP server. During OCSP proxying, Validation Manager passes client requests unchanged to the remote OCSP server and returns the remote server response unchanged to the client. Validation Manager can also use OCSP forwarding to query a remote OCSP server.

OCSP Request

A client issues an OCSP request to obtain the status of a certificate. The client suspends acceptance of the certificate until it receives an OCSP response.

OCSP Responder

The OCSP Responder accepts certificate status requests from OCSP-enabled clients, looks up a certificate status, and responds with the certificate's current status.

OCSP Response

Validation Manager obtains the status of a certificate and returns an OCSP response to the client who issued the certificate status request.

OCSP Signer

An entity that signs OCSP responses.

Online Certificate Status Protocol (OCSP)

A protocol, defined in RFC 2560, that enables applications to check the status of a certificate every time the certificate is used. If you configure your PKI to use OCSP, CRLs are unnecessary for end users.

Online Validation

Online validation occurs when a CA can be queried directly about a certificate's validity every time the certificate is used.

Operator Card Set (OCS)

A card set within the nCipher security world that is used to generate, protect, and access the private keys created within it.

PKCS #7

The Cryptographic Message Syntax Standard. For more information on the standard, go to **www.rsasecurity.com/rsalabs/pkcs/pkcs-7/**.

PKCS #10

The Certification Request Syntax Standard. For more information on the standard, go to **www.rsasecurity.com/rsalabs/pkcs/pkcs-10/**.

PKCS #11

The Cryptographic Token Interface Standard. For more information on the standard, go to **www.rsasecurity.com/rsalabs/pkcs/pkcs-11/**.

PKI Performance

The number of revocation list per hour that an OCPS server can import.

PKIX (Public Key Infrastructure X.509)

The evolving Internet Engineering Task Force (IETF) standard for PKI using X.509 certificates. For more information on the standard, go to **www.ietf.org/html.charters/pkix-charter.html**.

Privacy Enhanced Mail (PEM) format

PEM was originally created to provide secure e-mail services on the Internet, but it became too unwieldy for widespread use. Now, "PEM format" usually refers to the Base64 encoding algorithm that was part of the PEM proposal.

PEM encoding is useful for presenting binary data in a text-readable form. (For example, to allow you to copy and paste data between applications.) Also known as **Base64**.

Private Key

The private part of a public-key key pair. With Validation Manager, private keys are generated on the OCSP server whenever an OCSP signer is created. Private keys must be securely stored to prevent unauthorized access and accidental deletion.

A digital signature involves encrypting a message digest with a private key and allows anyone with the corresponding public key to decrypt the message digest to be certain of who sent the message and that it has not been tampered with.

You can decrypt information encrypted with a public key with the corresponding private key.

Proxying

See **OCSP Proxying**.

Public Key

The public and widely distributed part of a public-key key pair. For example, a certificate contains information about the certificate subject, the certificate signer, and a public key value. In general, you can only decrypt information encrypted with a public key with the corresponding private key.

Public-Key Cryptography Standards (PKCS)

A set of standard protocols developed by RSA for making secure information exchange possible. The standards include RSA encryption, password-based encryption, and cryptographic message syntax. For more information on standards, go to **www.rsasecurity.com/rsalabs/pkcs/**.

Public Key Infrastructure (PKI)

A system for publishing, distributing, and managing the public key values used in public key cryptography. All PKIs involve issuing public key certificates to individuals, organizations, and other entities and verifying that these certificates are valid.

Refresh Time

The time after which Validation Manager attempts to retrieve a fresh status value. A status value is considered stale after its status source's refresh time elapses.

Response Caching

The process of reusing a previously generated and signed response.

Revocation

Revoking a certificate invalidates it and removes all of its privileges in the PKI. Revocation is necessary if the CA administrator wants to invalidate the certificate before it expires. Administrators revoke certificates by marking them as invalid in the Secure Directory. Users of the PKI are notified of the revoked status of a certificate during online validation or with revocation lists.

Revoking a CA invalidates the CA certificate and removes all PKI privileges of the CA. Revoke a CA only if you have organizational-based security concerns and only as a last resort.

Rivest-Shamir-Adleman (RSA)

A highly secure cryptography method created by the three founders of RSA: Professors Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman.

RSA uses a two-part key. The private key is kept by the owner; the public key is published. Data that is encrypted using the recipient's public key can only be decrypted by the recipient's private key, and vice-versa.

The RSA algorithm is computation intensive. Therefore, it is often used to create a digital envelope, which holds an RSA-encrypted symmetric key (often 3-DES or AES) and symmetric key-encrypted data. This method encrypts the secret symmetric key so that it can be transmitted over the network, but encrypts and decrypts the actual message using the much faster symmetric key algorithm.

The RSA algorithm is also used for authentication using digital signatures. In this case, the sender's private key is used for signing, and the sender's public key is used for verification. The RSA algorithm is also implemented in hardware. As RSA chips get faster, RSA encoding and decoding will add less overhead to the operation.

Root CA

A CA whose certificate is self-signed (that is, the issuer and the subject are the same). A root CA is at the top of a hierarchy.

Secure Hash Algorithm (SHA-1)

An algorithm developed by the U.S. National Institute of Standards & Technology (NIST). SHA-1 is used to create a cryptographic hash (or "fingerprint") of a message or data. SHA-1 is considered to be somewhat stronger than MD5. SHA-1 is defined in FIPS Publication 180-2, the Secure Hash Standard (SHS).

Secure Sockets Layer (SSL)

A protocol layer created by Netscape to manage the security of message transmissions in a network. Security is achieved through encryption. "Sockets" refers to the sockets method of passing data back and forth between client and server programs in a network or between program layers in the same computer.

Security World

A security world consists of at least one hardware module, a set of smart cards, and encrypted data stored on a computer.

Server Certificate

An end-entity certificate issued to a server. Servers present their certificates to web browsers so browsers can verify (authenticate) the identity of the server. Server certificates are sometimes called SSL or TLS certificates.

Signer

See **OCSP Signer**.

Signing

A process by which a digital signature is affixed to a file, document, or certificate as proof that it has not been tampered with and that the author is who claims to be the author.

Signer Certificate

A certificate, signed by a known CA, that the signer includes in signed status responses. To create a signer certificate, the CA signs a signer certificate request that you send to the CA. You can use the same request to obtain signer certificates from different CAs.

S/MIME

Microsoft and Netscape include S/MIME in the latest versions of their e-mail clients. Other vendors of message products also endorse S/MIME.

MIME itself, described in the IETF standard RFC 1521, defines the structure of an electronic message. S/MIME allows the message body to include encryption information and a digital certificate. S/MIME has extended the syntax provided in PKCS #7. For more information on the standard, go to **www.ietf.org/html.charters/smime-charter.html**.

SSL Client Authentication

The process whereby a server authenticates a client by verifying the end-entity certificate presented by the client during SSL operations.

SSL-LDAP

See **LDAPS**.

SSL Server Authentication

The process whereby a client application authenticates a server by verifying the certificate chain presented by the server during SSL operations, starting with a CA trusted by the client.

Stale Status Data

The freshness of a revocation list or status value. A list or status value is considered stale once the refresh time of its status source elapses. For example, if the refresh time for Validation Manager to retrieve a new list or status value has passed, the list or status value within the Validation Manager database is considered stale up until the time when the grace period elapses.

Status

The validity of a certificate: active, reinstated, revoked, or suspended.

Status Data Caching

The process of reusing previously obtained status data.

Status Source

A location and method for obtaining the status of certificates.

StartTLS

A method for opening a non-TLS connection, and then changing it into a TLS-protected connection. It is the standard way to use TLS for LDAP v3.

Suspension

The process of marking a certificate as temporary invalid. The end-user presenting the suspended certificate is denied access where the certificate previously allowed access. Reinstating a certificate returns all removed PKI privileges.

Suspending a CA certificate marks it as temporarily invalid and removes all of the CA's PKI privileges. Reinstating a CA certificate returns all removed PKI privileges.

System CA

The CA created during installation of Validation Manager to issue the server certificates.

Synchronization

The use of multiple Validation Manager installations to provide support for the synchronization of revocation data in a low bandwidth environment.

The synchronization server is a Validation Manager installation that has the most current revocation data. A synchronization client is a Validation Manager installation that requests revocation data from a synchronization server. The synchronization server and clients exchange certificates to authenticate each other.

Synchronization Performance

The number of status values per second that can be updated between two servers.

System Log

An operating system specific file that Validation Manager uses to record systemic events not related to regular operations or configuration changes.

Trace Log

A file containing information suitable for debugging purposes.

TLS Client Authentication

The process whereby a server authenticates a client by verifying the end-entity certificate presented by the client during TLS operations.

TLS Server Authentication

The process whereby a client application authenticates a server by verifying the certificate chain presented by the server during TLS operations, starting with a CA trusted by the client.

Transport Layer Security (TLS)

Internet protocol that provides privacy between server and client. TLS ensures that no third party may eavesdrop or tamper with any message. TLS is the successor to SSL; however, they are not interoperable.

UTF-8 Encoding

An ASCII compatible multibyte Unicode and UCS encoding, used by current browsers, Java and Plan 9.

Validation

The process of verifying that a certificate is valid. Validation can occur online or through the use of revocation lists.

Validation Manager

A server that accepts requests from clients to check the validity of certificates. Validation Manager supports the Online Certificate Status Protocol (OCSP).

Validation Manager Installation

An instance of Validation Manager. This may comprise a single machine hosting single instances of the various Validation Manager servers, or a farm of servers residing behind a Network Address Translator machine such as a load balancer. The servers within a Validation Manager installation are generally under a single administrative domain.

Validity

Whether a certificate is valid or invalid. A certificate is valid if it has not expired and a CA has not suspended or revoked it.

Web Server

An Apache-based server that is the primary interface to Validation Manager.

X.509

An International Standards Organization (ISO) standard that describes a basic electronic format for digital certificates.

X.509 v3 Certificate Extension

Certificate extensions, including extensions for PKIX, SET, and SSL. The RSA Validation Solution supports X.509 v3 that conform to version 3 of the X.509 standard and specify additional constraints or capabilities on the certificate subject.

Acronyms

API	application programming interface
ARL	authority revocation list
ASN.1	Abstract Syntax Notation One
CA	certificate authority
CP	certificate policy
CPS	certification practice statement
CRL	certificate revocation list
DER	Distinguished Encoding Rules
FQDN	fully qualified domain name
GUI	graphical user interface
HSM	hardware security module
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol (over an SSL connection)
I18N	Internationalization
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IPSec	IP Security Protocol
ISO	International Standards Organization
ITU/CCITT	International Telecommunication Union
LDAP	Lightweight Directory Access Protocol
LDAPS	Lightweight Directory Access Protocol (over an SSL/TLS connection)
MD5	Message Digest 5

MSIE	Microsoft Internet Explorer
OCSP	Online Certificate Status Protocol
PEM	Privacy Enhanced Mail format
PIN	personal identification number
PKCS	Public-Key Cryptography Standards
PKI	public key infrastructure
PKIX	Public Key Infrastructure (X.509)
RAM	random access memory
RSA	Rivest-Shamir-Adleman
S/MIME	Secure Multi-Purpose Internet Mail Extensions
SHA-1	Secure Hash Algorithm
SSL	Secure Sockets Layer
SSL-LDAP	Lightweight Directory Access Protocol over a Secure Sockets Layer connection
TLS	Transport Layer Security
UCS	Universal Character Set (the superset of all other character sets)
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTF-8	UCS Transformation Format
VPN	virtual private network
XML	Extensible Markup Language

Index

A

administration script file, 11
    quotation marks, 11
administration utility, 8

C

CA commands, 35
certificate authority
    adding, 20
command elements, 8
command form, 8
configuration file, 10
    Default.conf, 9, 10
controlling Validation Manager
    by way of a web browser, 7
    restarting, 21
    starting, 11
    stopping, 11
CreateCertRequestFromCert, 57
CreateDefaultSigner, 58
CreateLDAPStatusSource, 70
CreateOCSPStatusSource, 19, 73
CreateRLStatusSource, 18, 19, 74
CreateSigner, 15, 59
CreateSignerCertRequest, 17, 60

D

Default.conf, 9, 10
DeleteCA, 35
DeleteSigner, 61
DeleteSignerCert, 61
DeleteSignerCertRequest, 62
DeleteStatusSource, 75
DeleteSyncClient, 23

G

general options, 9
    administration script, 10
    configuration file, 9
    help, 9
    hostname, 9
    port number, 9
    private key and TLS certificate, 9
    System CA certificate, 10
    TLS passphrase, 10
    version number, 9
GetAuditEventList, 24
GetAuditLogSettings, 24
GetCA, 36
GetCAARL, 37
GetCAARLInfo, 37
GetCACert, 38
GetCACRL, 38
GetCACRLInfo, 39
GetCADRL, 39
GetCADRLInfo, 40
GetCAList, 40
GetCAOCSPRequestCount, 41
GetCARLEntryInfo, 42
GetCASyncUpdates, 43
GetCert, 16, 62
GetCertRequest, 17, 62
GetCertStatus, 43
GetClusterNodeList, 25
GetDefaultSigner, 63
GetLocallyRevokedCertList, 44
GetOCSPEnabledSetting, 25
GetSigner, 63
GetSignerCertList, 64
GetSignerCertRequestList, 64
GetSignerList, 65
GetStatusSource, 76
GetStatusSourceList, 77
GetSyncClientInfo, 26
GetSyncClientList, 26
GetSystemSettings, 27

H

Help, 9

I

ImportARL, 44
ImportCA, 20, 45
ImportCASyncUpdates, 46
ImportCRL, 46
ImportDRL, 47
ImportSignerCert, 17, 65

O

OCSP signer
    creating, 15
    default, 15
OCSP signer certificate
    CA-issued, 17
    self-signed, 16
OCSP signer commands, 57

P

passphrase
    nCipher smart card, 12
    OCSP signer key, 12
    system passphrase, 12

R

RefreshCAStatusSource, 21, 47
RenewSignerSelfCert, 66
revocation lists
    configuring, 20
RSA Validation Manager
    CA commands, 35
    OCSP signer commands, 57
    status source commands, 69
    system commands, 23

S

SetAuditEvent, 28
SetAuditLogging, 28
SetAuditLogRolloverInterval, 29
SetAuditLogSigningInterval, 29
SetCAIndirectRLIssuer, 48
SetCALocalStatus, 48
SetCANickname, 48
SetCAOCSPSettings, 49
SetCAOCSPValidation, 50
SetCAPath, 51
SetCAPurposes, 51
SetCARLTypeAttributes, 52
SetCARLTypes, 20, 53
SetCASigner, 54
SetCASignerCert, 54
SetCAStatusSource, 55
SetCASyncEnabled, 55
SetCertLocalStatus, 56
SetConfigurationRefreshTime, 30
SetDeafaultOCSPValidation, 30
SetDefaultSigner, 31
SetDefaultStatusSource, 31
SetOCSPEnabled, 31
SetOCSPSettings, 32
SetSignerCertNickname, 66
SetSignerDefaultCert, 18, 66
SetSignerNickname, 67
SetSignerPassphrase, 67
SetStatusSourceGracePeriod, 78
SetStatusSourceLDAPAttributes, 79
SetStatusSourceLDAPMissingObjectStatus, 80
SetStatusSourceLDAPStatusCodes, 82
SetStatusSourceLDAPStatusInfoObjectClass, 82
SetStatusSourceLDAPUseCertificatePresence, 88
SetStatusSourceNickname, 77
SetStatusSourceOCSPProxyMode, 82
SetStatusSourceOCSPServerAuthMode, 83
SetStatusSourceOCSPServerCert, 83
SetStatusSourceOCSPSigner, 84
SetStatusSourceOCSPSignRequests, 84
SetStatusSourceOCSPStatusCheck, 85
SetStatusSourceOCSPUseNonces, 85
SetStatusSourceRefreshTime, 86
SetStatusSourceRLRefreshMode, 85
SetStatusSourceTLSServerAuthMode, 86
SetStatusSourceTLSServerCert, 87
SetStatusSourceTLSStatusCheck, 87
SetStatusSourceURL, 88
SetSyncClientState, 33
shutdownVM script, 11
signing OCSP responses, 16
starting Validation Manager, 11
startupVM script, 11
status source
    creating, 18
    default, 18
    LDAP-based, 19
    OCSP-based, 19
    revocation list-based, 18
status source commands, 69
stopping Validation Manager, 11
system commands, 23

U

UseFreshStatusData, 33

V

Validation Manager
    restarting, 21
    starting, 11
    stopping, 11
ViewCertInfo, 68
vmadmin
    administration utility, 8
    command form, 8
    controlling Validation Manager, 8