Customization Guide

Sun™ ONE Certificate Server

Version 4.7

September 2002
816-5544-10
Second Edition

Contents

About This Guide 9
What's in This Guide 10
What You Should Already Know 10
Conventions Used in This Guide 11
Where to Go for Related Information 12

Chapter 1 Before You Begin 15
What You Need to Know to Change Forms 15
    HTTP, Query URLs, and HTML Forms 16
    JavaScript 16
How the Forms Work 16
    Requests Sent to the CMS server 17
    Responses and Output Templates 17
    Errors and the Error Template 19
JavaScript Used By All Interfaces 20

Part 1 Customizing End-Entity Services Interface 25

Chapter 2 Introduction to End-Entity Services Interface 27
End-Entity Services Interface 27
    How Client Type Determines the End-Entity Interface 29
    Certificate Request Formats Specific to End Entities 29
Accessing the End-Entity Services Interface 31
End-Entity Forms and Templates 31
    Locating End-Entity Forms and Templates 31
    Forms for Certificate Enrollment 32
    Forms for Certificate Renewal 33
    Forms for Certificate Revocation 34
    Forms for Certificate Retrieval 34

Forms for Key Recovery . . . 35
Other Forms . . . 35
Output Templates for End-Entity Interfaces . . . 36

Chapter 3 End-Entity Interface Reference . . . 39
Overview of End-Entity Interfaces . . . 40
Certificate Enrollment Protocol Interface . . . 41
Description . . . 41
Default Forms . . . 41
Request Parameters . . . 41
Challenge Revocation Interface . . . 42
Description . . . 42
Default Forms . . . 42
Request Parameters . . . 42
Response . . . 44
Display Certificate By Serial Number Interface . . . 44
Description . . . 44
Default Forms . . . 44
Request Parameters . . . 44
Response . . . 45
Display Certificate From Request Interface . . . 46
Description . . . 46
Default Forms . . . 46
Request Parameters . . . 47
Response . . . 47
Enrollment Interface . . . 49
Description . . . 49
Default Forms . . . 50
Request Parameters . . . 51
Response . . . 55
Get CA Chain Interface . . . 59
Description . . . 59
Default Forms . . . 60
Request Parameters . . . 60
Response . . . 61
Get Certificate By Serial Number Interface . . . 61
Description . . . 61
Default Forms . . . 62
Request Parameters . . . 62
Response . . . 63
Get Certificate From Request Interface . . . 65
Description . . . 65
Default Forms . . . 66

Request Parameters 66
Response 67
Get CRL Interface 69
Description 69
Default Forms 70
Request Parameters 70
Response 71
List Certificates Interface 72
Description 72
Default Forms 72
Request Parameters 73
Response 78
Renewal Interface 80
Description 80
Default Forms 81
Request Parameters 81
Response 81
Revocation Interface 82
Description 82
Default Forms 82
Request Parameters 82
Response 85

Chapter 4 Internationalization of End-Entity Interface 87
Displaying Forms in Non-English Languages 87

Part 2 Customizing Agent Services Interface 91

Chapter 5 Introduction to Agent Services Interface 93
Agent Services Interface 93
Certificate Manager Agent Services 94
Registration Manager Agent Services 95
Data Recovery Manager Agent Services 96
Accessing the Agent Services Interface 97
Agent Forms and Templates 97
Structure of the Agent Services Interface 97
Locating Agent Forms and Templates 98

Chapter 6 Agent Interface Reference 99
Overview of Agent Interfaces 100
Approve Revocation Interface 101

- Description 101
- Default Forms 102
- Request Parameters 102
- Response 104

Bulk Enrollment Interface 105
- Description 105
- Configuration Parameters 106
- Default Forms 107
- Request Parameters 107
- Response 112

Display Key By Serial Number Interface 115
- Description 115
- Default Forms 116
- Request Parameters 116
- Response 116

Display Key For Recovery Interface 118
- Description 118
- Default Forms 118
- Request Parameters 118
- Response 119

Examine Recovery Interface 120
- Description 120
- Default Forms 121
- Request Parameters 121
- Response 122

Get Approval Status Interface 122
- Description 122
- Default Forms 123
- Request Parameters 123
- Response 124

Get PKCS #12 Data Interface 125
- Description 125
- Default Forms 125
- Request Parameters 125
- Response 126

Grant Recovery Interface 126
- Description 126
- Default Forms 126
- Request Parameters 126
- Response 127

Key Query Interface 128
- Description 128
- Default Forms 128

Request Parameters 129
Response 131
Key Recovery Query Interface 133
Description 133
Default Forms 134
Request Parameters 134
Response 136
Process Certificate Request Interface 138
Description 138
Default Forms 139
Request Parameters 139
Response 142
Process DRM Request Interface 147
Description 147
Default Forms 147
Request Parameters 147
Response 148
Process Request Interface 150
Description 150
Default Forms 151
Request Parameters 151
Response 152
Recover Key By Serial Number Interface 153
Description 153
Default Forms 153
Request Parameters 153
Response 155
Remove Certificate Hold Interface 156
Description 156
Default Forms 157
Request Parameters 157
Response 157
Requests Query Interface 159
Description 159
Default Forms 159
Request Parameters 159
Response 160
Select for Revocation Interface 164
Description 164
Default Forms 164
Request Parameters 164
Response 165
Update CRL Interface 167

Description 167
Default Forms 167
Request Parameters 167
Response 168
Update Directory Interface 168
Description 168
Default Forms 169
Request Parameters 169
Response 171

Index 173

About This Guide

The *Customization Guide* provides reference information about all the plug-in modules provided with iPlanet Certificate Management Server (CMS). Plug-in modules help you configure and customize Certificate Management System, and use it for issuing and managing certificates to various end entities, such as web browsers (users), servers, Virtual Private Network (VPN) clients, and Cisco™ routers.

NOTE Sun™ ONE Certificate Server was previously known as iPlanet™ Certificate Management System. The product was renamed shortly before the launch of this 4.7 release.

The late renaming of this product has resulted in a situation where the new product name is not fully integrated into the shipping product. In particular, you will see the product referenced as iPlanet Certificate Management Server within the product GUI and within the product documentation. For this release, please consider iPlanet Certificate Management Server (CMS) and Sun ONE Certificate Server as interchangeable names for the same product.

This chapter has the following sections:

- What's in This Guide (page 10)
- What You Should Already Know (page 10)
- Conventions Used in This Guide (page 11)
- Where to Go for Related Information (page 12)

What's in This Guide

This guide covers topics that help you customize CMS agent and end-entity interfaces. You should use this guide in conjunction with the other CMS documentation, such as the one that explains how to install and configure Certificate Management System. Complete list of CMS documentation is provided later in this preface.

What You Should Already Know

This guide is intended for experienced system administrators who are planning to deploy Certificate Management System. CMS agents should refer to *CMS Agent's Guide* for information on how to perform agent tasks, such as handling certificate requests and revoking certificates.

This guide assumes that you

- Are familiar with the basic concepts of public-key cryptography and the Secure Sockets Layer (SSL) protocol.
  - SSL cipher suites
  - The purpose of and major steps in the SSL handshake
- Understand the concepts of intranet, extranet, and the Internet security and the role of digital certificates in a secure enterprise. These include the following topics:
  - Encryption and decryption
  - Public keys, private keys, and symmetric keys
  - Significance of key lengths
  - Digital signatures
  - Digital certificates, including various types of digital certificates
  - The role of digital certificates in a public-key infrastructure (PKI)
  - Certificate hierarchies

  If you are new to these concepts, we recommend you read the security-related documents available online at this URL:
  `http://docs.sun.com/db?p=coll/S1_nsCMS_42_Resources`

You may also refer to the security-related appendixes (Appendix D and Appendix E) of the accompanying manual, *Managing Servers with iPlanet Console.*

- Are familiar with the role of iPlanet Console in managing iPlanet version 4.x servers. Otherwise, see the accompanying manual, *Managing Servers with iPlanet Console.*
- Are reading this guide in conjunction with the documentation listed in “Where to Go for Related Information” on page 12.

Conventions Used in This Guide

The following conventions are used in this guide:

- `Monospaced font`—This typeface is used for any text that appears on the computer screen or text that you should type. It’s also used for filenames, functions, and examples.

  Example: `Server Root` is the directory where the CMS binaries are kept.

- *Italic*—Italic type is used for emphasis, book titles, and glossary terms.

  Example: This control depends on the access permissions the *superadministrator* has set up for you.

- Text within “quotation marks”—Cross-references to other topics within this guide.

  Example: For more information, see “Issuing a Certificate to a New User” on page 154.

- **Boldface—**Boldface type is used for various UI components such as captions and field names, and the terminology explained in the glossary, which can be found in iPlanet Certificate Management Server Installation and Setup Guide.

  Example:

  Rotation frequency. From the drop-down list, select the interval at which the server should rotate the active error log file. The available choices are Hourly, Daily, Weekly, Monthly, and Yearly. The default selection is Monthly.

- `Monospaced [ ]`—Square brackets enclose commands that are optional.

  Example:

  ```
  PrettyPrintCert <input_file> [<output_file>]
  ```

`<input_file>` specifies the path to the file that contains the base-64 encoded certificate.

`<output_file>` specifies the path to the file to write the certificate. This argument is optional; if you don't specify an output file, the certificate information is written to the standard output.

- `<>`—Angle brackets enclose variables or placeholders. When following examples, replace the angle brackets and their text with text that applies to your situation. For example, when path names appear in angle brackets, substitute the path names used on your computer.

  Example: Using Netscape Communicator 4.04 or later, enter the URL for the administration server: `http://<hostname>:<port_number>`

- `/`—A slash is used to separate directories in a path. If you use the Windows NT operating system, you should replace `/` with `\` in paths.

  Example: Except for the Security Module Database Tool, you can find all the other command-line utilities at this location: `<server_root>/bin/cert/tools`

- Sidebar text—Sidebar text marks important information. Make sure you read the information before continuing with a task.

  Examples:

NOTE You can use iPlanet Console only when Administration Server is up and running.

CAUTION A caution note documents a potential risk of losing data, damaging software or hardware, or otherwise disrupting system performance.

Where to Go for Related Information

This section summarizes the documentation that ships with Certificate Management System, using these conventions:

- `<server_root>` is the directory where the CMS binaries are kept (which you specify during installation).
- `<instance_id>` is the ID for this instance of iPlanet Certificate Management Server (specified during installation).

The documentation set for Certificate Management System includes the following:

- *Managing Servers with iPlanet Console*

  Provides background information on basic cryptography concepts and the role of iPlanet Console. To view the HTML version of this guide, open this file: `<server_root>/manual/en/admin/help/contents.htm`

- *CMS Installation and Setup Guide*

  Describes how to plan for, install, and administer Certificate Management System. To access the installation and configuration information from within the CMS Installation Wizard or from the CMS window (within iPlanet Console), click any help button.

  To view the HTML version of this guide, open this file: `<server_root>/manual/en/cert/setup_guide/contents.htm`

- *CMS Plug-Ins Guide*

  Provides detailed reference information on CMS plug-ins. To access this information from the CMS window within iPlanet Console, click any help button.

  To view the HTML version of this guide, open this file: `<server_root>/manual/en/cert/plugin_guide/contents.htm`

- *CMS Command-Line Tools Guide*

  Provides detailed reference information on CMS tools.

  To view the HTML version of this guide, open this file: `<server_root>/manual/en/cert/tools_guide/contents.htm`

- *CMS Customization Guide* (this guide)

  Provides detailed reference information on customizing the end-entity and agent interfaces.

  To view the HTML version of this guide, open this file: `<server_root>/manual/en/cert/custom_guide/contents.htm`

- *CMS Agent's Guide*

  Provides detailed reference information on CMS agent interfaces. To access this information from the Agent Services pages, click any help button.

  To view the HTML version of this guide, open this file: `<server_root>/cert-<instance_id>/web/agent/manual/agent_guide/contents.htm`

- End-entity help (online only, not printed)

  Provides detailed reference information on CMS end-entity interfaces. To access this information from the end-entity pages, click any help button.

  To view the HTML version of this guide, open this file: `<server_root>/cert-<instance_id>/web/ee/manual/ee_guide/contents.htm`

NOTE Do not change the default location of any of the HTML files; they are used for online help. You may move the PDF files to another location.

For a complete list of all documentation for Certificate Management System, including documentation for Directory Server, see Documentation Summary, located at: `<server_root>/manual/index.html`

For the latest information about Certificate Management System, including current release notes, technical notes, and deployment information, check this site: `http://docs.sun.com/?p=coll/S1_s1CertificateServer_47.`

Chapter 1

Before You Begin

The services interfaces that come with iPlanet Certificate Management Server (CMS) make it possible for end-entities and agents to interact with the server. Your end-entities and agents can use the interface's HTML-based forms to carry out various certificate and key-related operations, such as enrolling for, renewing, and revoking certificates.

You can use the default forms as they are, customize them, or develop your own forms to suit your organization's policies or terminology. This chapter explains how to customize the forms and templates used by the interfaces.

The chapter has the following sections:

- What You Need to Know to Change Forms (page 15)
- How the Forms Work (page 16)
- JavaScript Used By All Interfaces (page 20)

What You Need to Know to Change Forms

Changing the default forms' appearance requires only a basic knowledge of HTML and a text editor. However, more significant customizing efforts require a good understanding of the following:

- HTTP, Query URLs, and HTML Forms
- JavaScript
- The way the End-Entity Services Interface works

HTTP, Query URLs, and HTML Forms

Requests from the end-entity services interface to Certificate Management System are submitted using the HTTP `GET` and `POST` methods. Requests take the form of query URLs (in the case of the `GET` method) or data sent through standard output (in the case of the `POST` method).

For background on these topics, consult books on authoring for the World Wide Web, on HTML, and on the HTML specification.

JavaScript

JavaScript is a scripting language that most browser software, including Netscape Navigator and Communicator, used for dynamic forms. (JavaScript is not the same as the more sophisticated and powerful Java language that is also supported by Netscape clients.)

In the agent and end-entity services forms, JavaScript is used to check input values and to formulate requests to the CMS server. JavaScript is also used in the output templates to present and format responses from the CMS server.

To customize the forms and templates, you need to be familiar with JavaScript. For more information on JavaScript and its use in Netscape browsers, see the *Netscape JavaScript Authoring Guide* available at this URL:
`http://home.netscape.com/eng/mozilla/3.0/handbook/javascript/index.html`

There are also several books on this topic.

How the Forms Work

Administrators, end-entities, and agents request service operations using the HTTP or HTTPS (HTTP over SSL) protocol using either the `GET` method (by submitting query URLs) or the `POST` method (by submitting a URL-encoded form with the content-type `application/x-www-form-urlencoded`). The `GET` method and the `POST` method result in a set of name-value pairs; this set constitutes the request.

For certificate service operations, the URI portion should indicate

```
/<operation>
```

where `operation` designates the certificate (management) service portion, such as enrollment, retrieval, renewal, or revocation of the CMS server. Any HTTP operations with URIs that do not begin with the `/<operation>` prefix are treated as requests for other kinds of web service by the CMS server. See chapters Chapter 3, "End-Entity Interface Reference" and Chapter 6, "Agent Interface Reference" for details on all the available operations.

Requests Sent to the CMS server

The services interface handles a set of operations. Each operation has a name and expects a specific set of parameters.

(The examples in this chapter are what you would see if Certificate Management System were running on the host `certs.siroe.com` and listening on the standard HTTPS port.)

For example, the `displayBySerial` interface displays the certificate with the serial number matching the `serialNumber` parameter. To use the `displayBySerial` interface to retrieve the certificate with serial number 58 (0x3a) using HTTP GET, you would use the following URL:

```
https://certs.siroe.com/displayBySerial?serialNumber=58
```

You could embed this URL in a link on an HTML page to allow users to view this certificate.

If you used an HTML form (rather than a query URL) to invoke this operation, the HTML for the form might look like this:

```
<FORM ACTION="https://certs.siroe.com/displayBySerial"
METHOD="POST">
Serial Number: <INPUT TYPE="TEXT" NAME="serialNumber">
<INPUT TYPE="SUBMIT" VALUE="Display This Certificate">
</FORM>
```

In the resulting form, the user enters the serial number of the certificate to be displayed.

Responses and Output Templates

Certificate Management System responds to service requests by sending back an HTML page built from two parts: a fragment of JavaScript code containing the data resulting from the operation and a template defining how the data is processed and displayed.

The fragment of JavaScript code consists of a `result` object that contains data properties only (no methods). The properties of the object correspond to parts of the response.

The template generally contains a combination of HTML and JavaScript code that processes and displays data. The template is set up to make use of the data in the `result` object.

In responding to a request, Certificate Management System determines the data that needs to be returned, embeds the data in the definition of the `result` object, and inserts the `result` object in the template. When the browser receives the constructed HTML page from the CMS server, the JavaScript code in the template file looks at the values in the `result` object and uses the data to display the HTML page to the user.

Because the functions that manipulate and display the data are accessible to you in the plain-text template files (as opposed to being hardcoded in the CMS server's libraries), you can customize the way data is used and presented to the user by editing the JavaScript and HTML in the template files.

For example, the `displayBySerial` operation generates the following JavaScript code fragment:

```
<SCRIPT LANGUAGE="JavaScript">
var header = new Object();
var fixed = new Object();
var recordSet = new Array;
var result = new Object();
var httpParamsCount = 0;
var httpHeadersCount = 0;
var authTokenCount = 0;
var serverAttrsCount = 0;
header.HTTP_PARAMS = new Array;
header.HTTP_HEADERS = new Array;
header.AUTH_TOKEN = new Array;
header.SERVER_ATTRS = new Array;
header.certPrettyPrint = [long string containing pretty-printed
certificate]
header.noCertImport = false;
header.certFingerprint = [string containing certificate
fingerprints]
header.authorityid = "ca";
header.serialNumber = 5;
header.emailCert = true;
header.certChainBase64 = [string containing base-64 encoded
certificate]
```

```
result.header = header;
result.fixed = fixed;
result.recordSet = recordSet;
</SCRIPT>
```

Notice how this code fragment defines an object named `result` and puts the resulting data from the operation in the properties of that object. Each certificate service operation returns an object named `result`. The contents of the `result` object are specific to the operation.

When it responds to the request, the `displayBySerial` interface running on Certificate Management System inserts this JavaScript fragment into the template file it uses to return results to the requestor. the CMS server inserts the fragment in the template file where it finds the tag `<CMS_TEMPLATE>`. It then returns the template with the inserted fragment to the client. The client then processes the completed template and displays the resulting page. In the case of the `displayBySerial` operation, the template file uses JavaScript and HTML to display the contents of the `result` object to the user.

Because the data from the operation is available in the `result` object, you can customize the JavaScript in the template or write your own functions to use this data. For example, to access the certificate's serial number, you can write a JavaScript function that uses `result.header.serialNumber`.

Templates for each operation are stored in the `web` subdirectory of the CMS server instance. The `web` subdirectory contains the following subdirectories where forms and templates are located:

- `ee` for end-entity interfaces
- `agent/ca` for Certificate Manager agent interfaces
- `agent/kra` for Data Recovery Manager agent interfaces
- `agent/ra` for Registration Manager agent interfaces

the CMS server reads the templates dynamically; you do not have to restart the CMS server for it to read changes to the template files.

Errors and the Error Template

All certificate service errors in the end-entity interface are returned through a single template called `GenError.template`. The `error result` object contains the following data properties:

```
<SCRIPT LANGUAGE="JavaScript">
var header = new Object();
var result = new Object();
header.errorDetails = [a string describing the context of the
error]
header.errorDescription = [a string describing the error]
result.header = header;
</SCRIPT>
```

The default CMS error template prints the information in the `error result` object along with some explanatory text.

JavaScript Used By All Interfaces

This section describes the JavaScript variables that are common to all responses from end-entity and agent interfaces. The interface definitions in subsequent sections give details about additional JavaScript that may be added by specific interfaces.

The CMS server handling the interface request replaces the `<CMS_TEMPLATE>` tag in a template with the JavaScript variables described in this section. When you modify or create templates, make sure the `<CMS_TEMPLATE>` tag appears in the file if your response needs to use any of the variables returned by an interface.

The JavaScript included in a response allows you to customize how the response is displayed or processed. For example, the `queryCert.template` file is used to list certificates returned by the List Certificates interface. The template makes extensive use of the JavaScript in the response to display a summary of each certificate and also to create new HTTP forms that can show details about a certificate.

All responses will include the following JavaScript:

```
<SCRIPT LANGUAGE="JavaScript">
var header = new Object();
var fixed = new Object();
var recordSet = new Array;
var result = new Object();
var httpParamsCount = 0;
var httpHeadersCount = 0;
var authTokenCount = 0;
var serverAttrsCount = 0;
header.HTTP_PARAMS = new Array;
header.HTTP_HEADERS = new Array;
header.AUTH_TOKEN = new Array;
header.SERVER_ATTRS = new Array;
```

```
fixed.preserved = "foo";
var recordCount = 0;
var record;
record = new Object;
record.HTTP_PARAMS = new Array;
record.HTTP_HEADERS = new Array;
record.AUTH_TOKEN = new Array;
record.SERVER_ATTRS = new Array;
recordSet[recordCount++] = record;
result.header = header;
result.fixed = fixed;
result.recordSet = recordSet;
</SCRIPT>
```

On its own, the base JavaScript is not very useful. Data pertinent to the response is added to this framework by the interface that creates the response. For example, an interface that lists a certificate might add the base-64 encoding of the certificate to the `record` object.

The `result` object contains most of the useful data, including the `header` object, the `fixed` object, and the `recordSet` array. Responses will usually add relevant data to one of these components of the result object. The purpose of these components can be summarized as follows:

- The `header` object contains variables and data that apply generally to all parts of the response. For example, a response including several certificates would store the total number of certificates returned in the header, while individual certificate data would be stored in `recordSet` elements. Variables in the header are accessed as `result.header.`*`variableName`*.
- The `fixed` object contains information that is fixed and independent of the data being returned in the response. Such data includes the hostname and port of the CMS server that handled the request, which is useful for creating HTTP forms that use the server and port that generated the response. Variables in the `fixed` object are accessed as `result.fixed.`*`variableName`*.
- The `recordSet` array is available to any response that returns certificates. Data for each certificate is returned in variables in a `record` object. Each `record` object is then added to the `recordSet` array. Variables in a particular record object are accessed by an index into the `recordSet` array: `result.recordSet[i].`*`variableName`*.

The following table describes the format an purpose of all the data included in the base `<CMS_TEMPLATE>` JavaScript:

Table 1-1 Variables Returned by the Base JavaScript

Variable	Format/Type and Description
`AUTH_TOKEN`	array
	Each element in this array is a name-value pair. These pairs represent variables that were returned from an authentication plug-in used (internally) by the interface. For example, if the authentication plug-in returned variables `fooValid` and `barValid` indicating whether the `foo` and `bar` parameters of a request were valid, the JavaScript in the response might look like:
	`header.AUTH_TOKEN[0].name = "fooValid";`
	`header.AUTH_TOKEN[0].value = "false";`
	`header.AUTH_TOKEN[1].name = "barValid";`
	`header.AUTH_TOKEN[1].value = "true";`
	These values are created when an authentication plug-in invokes `set(nameSrting, valueString)` on an `AuthToken` object.
`authTokenCount`	number
	The number of `AUTH_TOKEN` objects returned in this response.
`fixed`	object
	An object for containing data that is constant, such as the hostname, port number, or ID number associated with a request.
`header`	object
	An object for containing data that applies to the entire response, such as the number of records or the LDAP query that was used to get the data.
`HTTP_HEADERS`	Array
	Each element in this array is a name-value pair. These pairs represent HTTP request headers sent from the client to the interface.
	Use the `CMS.cfg` parameter `saveHttpHeaders` to list HTTP header values that should be saved and returned in responses.
	For example, if `saveHttpHeaders` is set to `"accept-language, user-agent"`, the JavaScript in the response might look like:
	`header.HTTP_HEADERS[0].name = "accept-language"`
	`header.HTTP_HEADERS[0].value = "en";`
	`header.HTTP_HEADERS[1].name = "user-agent"`
	`header.HTTP_HEADERS[1].value = "Mozilla/4.51 (X11; U; SunOS 5.7 sun4u)";`
	The default value for `saveHttpHeaders` (if it is not explicitly set in `CMS.cfg`) is `"accept-language, user-agent"`.

Table 1-1 Variables Returned by the Base JavaScript *(Continued)*

Variable	Format/Type and Description
`httpHeadersCount`	number The number of `HTTP_HEADERS` objects returned in this response.

`HTTP_PARAMS`	Array
	
	Each element in this array is a name-value pair. These pairs represent variables and their values that were used in the HTTP request made to the interface.
	
	Some HTTP parameters will be discarded after the request has been authenticated so that sensitive data is not stored on the CMS server. By default, parameters named `pwd`, `password`, and `passwd` are discarded after authentication. All other parameters are passed back in the response. Use the `CMS.cfg` parameter `dontSaveHttpParams` to list parameters that should be discarded by the CMS server after authentication.
	
	For example, if `dontSaveHttpParams` is set to `"pwd"` and the request used parameters named `uid` and `pwd`, the JavaScript in the response might look like:
	
	`header.HTTP_PARAMS[0].name = "uid"`
	`header.HTTP_PARAMS[0].value = "userOne";`
	
	The default value for `dontSaveHttpParams` (if it is not explicitly set in `CMS.cfg`) is `"pwd, password, passwd".`

`httpParamsCount`	number The number of HTTP_PARAMS objects returned in this response.
`fixed.preserved`	any data This variable contains the value of the parameter named `preserved` passed to the interface. It allows any data to be passed through from a request to the response template by submitting a parameter named `preserved` in the request.
`record`	Object An object for containing data about a single certificate. Variables and values are added to a `record` object, then the `record` object is added as an element in the `recordSet` array.

Table 1-1 Variables Returned by the Base JavaScript *(Continued)*

Variable	Format/Type and Description

`recordCount`	number
	
	The number of `record` objects returned in this response. Usually this is incremented for each `record` added to the `recordSet` array. For example,
	
	`recordCount = 0;`
	`record.serialNumber = 1;`
	`recordSet[recordCount++] = record;`
	`record.serialNumber = 2;`
	`recordSet[recordCount++] = record;`

`recordSet`	Array This array contains any number of `record` objects. The base JavaScript will also add `recordSet` to the `result` object, so use `result.recordSet[i].`*`variableName`* to access individual fields of a given `recordSet` element.
`result`	Object The primary container for all of the results returned in this template. The `fixed`, `header`, and `recordSet` objects are added to the `result` object as the last statements in the base JavaScript.

`SERVER_ATTRS`	Array
	
	Each element in this array is a name-value pair. These pairs represent variables and their values that were added by modules internal to the CMS serever or policy plug-ins. For example, if a policy plug-in added a variable called `requestStatus` with a value of `pending`, the resulting JavaScript might be:
	
	`header.SERVER_ATTRS[0].name = "requestStatus";`
	`header.SERVER_ATTRS[0].value = "pending";`

`serverAttrsCount`	number The number of `SERVER_ATTRS` objects returned in this response.

Part 1

Customizing End-Entity Services Interface

Chapter 2, “Introduction to End-Entity Services Interface”

Chapter 3, “End-Entity Interface Reference”

Chapter 4, “Internationalization of End-Entity Interface

Chapter 2

Introduction to End-Entity Services Interface

The services interfaces that come with iPlanet Certificate Management Server (CMS) make it possible for end-entities to interact with the server. Your end-entities can use the interface's HTML-based forms to carry out various certificate and key-related operations, such as enrolling for, renewing, and revoking certificates.

You can use the default forms as they are, customize them, or develop your own forms to suit your organization's policies or terminology. This chapter explains the default forms and templates used by the end-entity interface.

The chapter has the following sections:

- End-Entity Services Interface (page 27)
- Accessing the End-Entity Services Interface (page 31)
- End-Entity Forms and Templates (page 31)

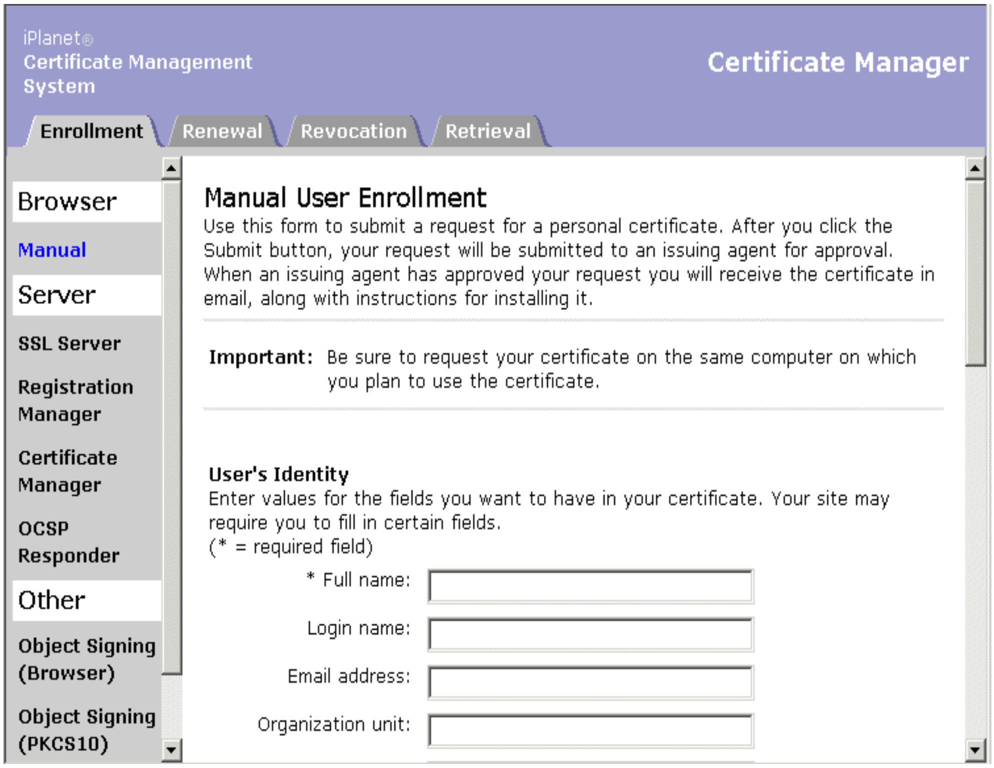
End-Entity Services Interface

Certificate Management System provides HTML forms for the various entities—people, routers, servers, and others—that use certificates to identify themselves and that need to be able to request certificate issuance and management operations. These forms, collectively called the *End-Entity Services* interface, use different protocols and life-cycle management procedures for different kinds of end entities. For example, the Certificate Manager provides separate certificate enrollment forms for clients such as Netscape Navigator 3.x, versions of Netscape Communicator later than 4.5, and Microsoft Internet Explorer. The reason for this is that end entities running Navigator 3.x and

Communicator versions earlier than 4.5 present an enrollment form based on the use of the HTML tag `KEYGEN` to generate keys; end entities running Internet Explorer present a form based on PKCS #10, the RSA standard for certificate request syntax.

Figure 2-1 shows the end-entity services interface hosted by a Certificate Manager.

Figure 2-1 End-entity services interface



Manual User Enrollment

Use this form to submit a request for a personal certificate. After you click the Submit button, your request will be submitted to an issuing agent for approval. When an issuing agent has approved your request you will receive the certificate in email, along with instructions for installing it.

Important: Be sure to request your certificate on the same computer on which you plan to use the certificate.

User's Identity

Enter values for the fields you want to have in your certificate. Your site may require you to fill in certain fields.
(* = required field)

* Full name:	
Login name:	
Email address:	
Organization unit:	

For a summary of the various end entities, protocols, cryptographic algorithms, and key pairs (single or dual) supported by Certificate Management System, see Table 2-1 on page 30.

For a complete list of the end-entity forms—for enrollment, renewal, retrieval, revocation, and key recovery—that come with Certificate Management System, see “End-Entity Forms and Templates” on page 31.

How Client Type Determines the End-Entity Interface

Each type of end-entity form provided by Certificate Management System is served by a servlet. This servlet determines which version of the form to present based on information about the end entity (the type, version, language, and so on), information in the form itself, and other factors.

Each form also specifies both an authentication manager and an output template:

- An authentication manager is a configured instance of an authentication plug-in module. When Certificate Management System receives a request from an end entity, it uses the authentication manager specified by the request to determine how to authenticate the end entity. For more information, see Chapter 15, "Setting Up End-User Authentication" in *CMS Installation and Setup Guide*.
- The output template is an HTML page with embedded JavaScript used to return information from the end entity to the servlet. For more information, see "Responses and Output Templates" on page 17.

Based on all the information, a form's servlet sends the end entity the version of the form (including the embedded JavaScript code) appropriate for that end entity. For example, in the case of end entities that support the `KEYGEN` tag, the Certificate Manager or Registration Manager sends a form that uses `KEYGEN` to generate keys and formulate a certificate request. In the case of end entities that support the Certificate Management Message Format (CMMF) protocol, the Certificate Manager or Registration Manager sends a form that uses a JavaScript API to fully automate both key generation and certificate issuance.

Certificate Request Formats Specific to End Entities

Table 2-1 lists the forms provided by the Certificate Manager and Registration Manager for certificate issuance and life-cycle management operations, and indicates supported authentication mechanisms and request formats. You can customize any of the default forms and their corresponding servlets and output templates. For details, see Chapter 3, "End-Entity Interface Reference."

Table 2-1 Summary of end-entity forms, authentication methods and certificate request formats

Form for end-entity operation	Authentication method	Supported certificate request formats
Certificate enrollment		
Client (end user) certificates	Manual, LDAP directory based, and NIS server based	• KEYGEN for Navigator/Communicator • PKCS #10 for Internet Explorer • Certificate Request Message Format (CRMF) for future versions of Communicator
Server certificates	Manual	PKCS #10
Cisco routers	Manual or automated	Certificate Enrollment protocol (CEP)
Certificate renewal		
Client (end user) certificates	SSL client authentication	• KEYGEN for Navigator/Communicator • PKCS #10 for Internet Explorer • CRMF for future versions of Communicator
Server certificates	Manual	PKCS #10
Cisco routers	Manual	CEP
Certificate revocation		
Client (end user) certificates	SSL client authentication and challenge-password based	• KEYGEN for Navigator/Communicator • PKCS #10 for IE • CRMF for future versions of Communicator
Server certificates	Manual	PKCS #10
Cisco routers	Manual	CEP
Encryption private key storage and recovery		
Client (end user) certificates	Not applicable	• Not supported for clients that can’t generate dual key pairs • CRMF for future versions of Communicator

Accessing the End-Entity Services Interface

By default, access to the end-entity services interface of a Certificate Manager or Registration Manager is open to all users. To access the Agent Services interface for a particular subsystem:

1. Open a web browser window.
2. Go to the page where the End-Entity Services interface for the Certificate Manager or Registration Manager is installed.

   The default URL for this page is:

   `http://<hostname>:<ee_port>` or `https://<host_name>:<ee_ssl_port>`

   `<hostname>` is in the form `<machine_name>.<your_domain>.<domain>`.

   The appropriate interface appears. (If you have disabled the unsecure end-entity port, you won't be able to access the interface on that port.)

End-Entity Forms and Templates

This section describes the end-entity interface and its default forms.

The end-entity services interface is divided into three parts or frames—top, menu, and content. The top frame includes tabs that are specific to end-entity operations, such as certificate enrollments and renewals. The menu lists all the operations supported by the selected tab. The content shows the form pertaining to the operation an end entity chooses in the menu; the form contains information to carry out the selected operation. Figure 2-1 on page 28 shows the end-entity interface of a Certificate Manager.

Locating End-Entity Forms and Templates

You can find the HTML forms and the corresponding output templates for the end-entity interface at this location:

```
<server_root>/cert-<instance_id>/web/ee
```

Forms for Certificate Enrollment

Table 2-2 lists the file names of forms that appear as menu options in the Enrollment tab of the end-entity interface. The forms are available on Certificate Manager instances and Registration Manager instances. The only exception is that the Certificate Manager enrollment form is available only on Certificate Manager instances.

Table 2-2 Forms for end-entity enrollment

Form Type: Menu Link and Filename	What form is used for...
User Enrollment (lists menu options for end-user enrollment)	
Manual (`ManUserEnroll.html`)	End users can use the User Enrollment forms to request SSL client and S/MIME certificates. Except for `Manual`, these links only appear when an appropriate authentication manager has been configured on the CMS server.
Directory Based (`DirUserEnroll.html`)	Enroll using directory user ID and password.
Directory and PIN Based (`DirPinUserEnroll.html`)	Enroll using directory user ID, password, and one time PIN.
NIS Server (`NISUserEnroll.html`)	Enroll using authentication against a NIS server.
Portal (`PortalEnrollment.html`)	Enroll using any unique user ID and a password.
Certificate (`CertBasedDualEnroll.html`)	Enroll for dual key certificates using a pre-issued certificate (on a hardware token) for authentication.
Certificate (`CertBasedSingleEnroll.html`)	Enroll for a single certificate using a pre-issued certificate (on a hardware token) for authentication. (This form is not used in the default interface.)
Certificate (`CertBasedEncryptionEnroll.html`)	Enroll for an encryption certificate only using a pre-issued certificate (on a hardware token) for authentication. (Thisform is not used in the default interface.)
Server Enrollment (lists menu options for server enrollment)	
SSL Server (`ManServerEnroll.html`)	Server administrators can use this form to request SSL server certificates for servers.
Directory Based Server (`DirServerEnroll.html`)	Server administrators can use this form to request SSL server certificates for servers.

Table 2-2 Forms for end-entity enrollment *(Continued)*

Form Type: Menu Link and Filename	What form is used for...
OCSP Responder (`OCSPResponder.html`)	Server administrators can use this form to request signing certificates for OCSP Responder servers.
Registration Manager Enrollment (lists menu options for Registration Manager enrollment)	
Registration Manager (`ManRAEnroll.html`)	Registration Manager administrators can use this form to request a *signing certificate* for a Registration Manager.
Certificate Manager Enrollment (lists menu options for Certificate Manager enrollment)	
Certificate Manager (`ManCAEnroll.html`)	Certificate Manager administrators can use this form to request *CA signing certificates* for Certificate Managers functioning as subordinate CAs.
Object Signing Enrollment (lists menu options for object signing enrollment)	
Object Signing (Browser) (`ManObjSignEnroll.html`) Object Signing (PKCS10) (`ObjSignPKCS10Enroll.html`)	End users and administrators can use this form to enroll for a certificate that allows them to sign objects, such as Java applets. Both the Certificate Manager and Registration Manager provide this form.

Forms for Certificate Renewal

Table 2-3 lists the forms that correspond to the menu options in the Renewal tab of the end-entity interface on Certificate Manager instances and Registration Manager instances.

Table 2-3 Forms for certificate renewal

Menu Link and Filename	What form is used for...
Server Certificate (`ServerRenewal.html`)	Server administrators can use this form to renew server certificates.
User Certificate (`UserRenewal.html`)	End users can use this form to renew their SSL client certificates and their S/MIME certificates if the S/MIME certificates were issued with the SSL client bit set.

Forms for Certificate Revocation

Table 2-4 lists the forms that correspond to the menu options in the Revocation tab of the end-entity services interface.

Table 2-4 Forms for certificate revocation

Menu Link and Filename	What form is used for...
Certificate (challenge phrase-based) (`ChallengeRevoke1.html`)	End users can use this form to revoke their SSL client certificates using a password created during enrollment.
Server Revocation (`ServerRevocation.html`)	Server administrators can use this form to revoke server certificates.
User Revocation (`UserRevocation.html`)	End users can use this form to revoke their SSL client certificates using SSL client authentication.

Forms for Certificate Retrieval

Table 2-5 lists the forms that correspond to the menu options in the Retrieval tab of the end-entity interface on Certificate Manager instances. Only the Import CA Certificate Chain interface is also available on Registration Manager instances.

Table 2-5 Forms provided for certificate retrieval

Menu Link and Filename	What form is used for...
List Certificates (`queryBySerial.html`)	End users and administrators can use this form to list certificates based on their serial numbers.
Search for Certificates (queryCert.html)	• End users and administrators can use this form to search for specific certificates. The search criteria can be a combination of the following: • Serial number of the certificate • Subject name of the certificate • Revocation status of the certificate • Issuing Information—when the certificate was issued • Validity period of the certificate • Type of certificate

Table 2-5 Forms provided for certificate retrieval *(Continued)*

Menu Link and Filename	What form is used for...
Import CA Certificate Chain (`GetCAChain.html`)	End users and administrators can use this form to import the certificate chain of a Certificate Manager (CA) into their browsers or servers. They can • Import the CA certificate chain into their browsers • Download the CA certificate chain in binary form • View the CA certificate chain for importing into a server • Display certificates in the CA certificate chain for importing individually into a server
Import Certificate Revocation List (`DisplayCRL.html`)	End users and administrators can use this form to: • Manually check the revocation status of a particular certificate (if they are not sure whether they have the latest version of the CRL) • Import the latest CRL to Netscape Navigator • Download the latest CRL in binary form • View the CRL header information

Forms for Key Recovery

Table 2-6 lists the form that corresponds to the menu option in the Recovery tab of the end-entity interface. This form is available on a Certificate Manager instance or a Registration Manager instance that is configured as a trusted manager for a Data Recovery Manager instance.

Table 2-6 Form for encryption private key recovery

Menu Link and Filename	What form is used for...
Key Recovery (`KeyRecovery.html`)	End users can use this form to retrieve their encryption private keys from the Data Recovery Manager.

Other Forms

Table 2-6 lists common forms that are used by the operation-specific forms in the end-entity interface.

Table 2-7 Files and forms used by other forms

Form filename	What form is used for...
`enrollMenu.html`	This file loads and highlights the Enrollment tab.
`renewalMenu.html`	This file loads and highlights the Renewal tab.
`recoveryMenu.html`	This file loads and highlights the Recovery tab.
`retrievalMenu.html`	This file loads and highlights the Retrieval tab.
`index.html`	This file contains the menu options. To change the name of an option, search for it in the file and then edit it.
`*.js`	Files with a `.js` file extension include JavaScript helper functions that are used by other forms.
`xenroll.dll`	This file enables the end-user enrollment forms to work with Microsoft Internet Explorer.

Output Templates for End-Entity Interfaces

Table 2-8 lists the default templates that are used by the end-enetity interfaces to return data to the requestor.

Table 2-8 Response templates used by the end-entity interface

Template filename	Description
`displayBySerial.template`	Used to display information pertaining to a certificate when users view an individual certificate (for example, when they click the Details button next to a certificate).
`EnrollSuccess.template`	Used to inform the CMS administrator that the agent certificate he or she requested has been successfully installed in the subsystem’s internal database.
`GenError.template`	Used to display error messages to the user.
`GenPending.template`	Used to inform a user requesting a certificate that the request has been queued for agent approval.
`GenRejected.template`	Used to inform a user requesting a certificate that the request has been rejected by the CMS server.
`GenSuccess.template`	Used to inform a user requesting a certificate that the request has been approved by the CMS server.

Table 2-8 Response templates used by the end-entity interface *(Continued)*

Template filename	Description
`GenSvcPending.template`	Used to inform a user requesting a certificate that the request has been queued for agent approval.
`GenUnauthorized.template`	Used to inform users when thay perform unauthorized operations.
`GenUnexpectedError.template`	Used to inform the user that the CMS server encountered an unexpected error while processing the request.
`ImportCert.template`	Used to display the CA certificate when users import the CA certificate.
`queryCert.template`	Used to display the list of certificates when users search for certificates.
`RenewalSuccess.template`	Used to inform a user requesting a certificate renewal that the request has been successfully renewed.
`RevocationSuccess.template`	Used to inform a user requesting a certificate revocation that the certificate has been revoked.

Chapter 3

End-Entity Interface Reference

This chapter provides a detailed reference of all the service interfaces available on an end-entity port of iPlanet Certificate Management Server. For each interface, there is a description including the URI used and the purpose, a list of forms that use the interface by default, a detailed description of valid input parameters and their values, and information about the response which lists the templates used and the additional JavaScript variables available.

The chapter has the following sections:

- Overview of End-Entity Interfaces (page 40)
- Certificate Enrollment Protocol Interface (page 41)
- Challenge Revocation Interface (page 42)
- Display Certificate By Serial Number Interface (page 44)
- Display Certificate From Request Interface (page 46)
- Enrollment Interface (page 49)
- Get CA Chain Interface (page 59)
- Get Certificate By Serial Number Interface (page 61)
- Get Certificate From Request Interface (page 65)
- Get CRL Interface (page 69)
- List Certificates Interface (page 72)
- Renewal Interface (page 80)
- Revocation Interface (page 82)

Overview of End-Entity Interfaces

The following table lists the end-entity interfaces and their functions. The sections that follow cover each interface in detail.

Table 3-1 Overview of End-Entity Interfaces

Interface	URI	Purpose
Certificate Enrollment Protocol Interface	`/pkiclient.exe`	Process Simple Certificate Enrollment Protocol (SCEP) certificate requests from routers and other VPN clients.
Challenge Revocation Interface	`/challenge_revocation1`	Revoke a certificate using a challenge phrase set during enrollment.
Display Certificate By Serial Number Interface	`/displayBySerial`	Retrieve a certificate with a given serial number in human-readable form (use Get Certificate By Serial Number to get a binary form).
Display Certificate From Request Interface	`/displayCertFromRequest`	Used on a Registration Manager to display the certificate issued for a given request identifier.
Enrollment Interface	`/enrollment`	Process manual or automated certificate requests.
Get CA Chain Interface	`/getCAChain`	Retrieve the certificate authority's certificate or certificate chain (if the CA is not self-signed).
Get Certificate By Serial Number Interface	`/getBySerial`	Get a certificate with a given serial number in a binary format (for example, PKCS #7 or a CMMF response).
Get Certificate From Request Interface	`/getCertFromRequest`	Use the id assigned to a pending request to retrieve the certificate once it has been issued.
Get CRL Interface	`/getCRL`	Retrieve the Certificate Revocation List.
List Certificates Interface	`/listCerts`	List certificates based on flexible query criteria.
Renewal Interface	`/renewal`	Process requests for renewing a certificate presented to the interface using SSL client authentication.

Table 3-1 Overview of End-Entity Interfaces *(Continued)*

Interface	URI	Purpose
Revocation Interface	`/revocation`	Process requests for manual revocation or for revocation of a certificate presented to the interface using SSL client authentication.

Certificate Enrollment Protocol Interface

Description

URI: `/cgi-bin/pkiclient.exe`

Available on: Certificate Manager and Registration Manager

Function: Handles Certificate Enrollment Protocol (CEP) requests from devices such as Virtual Private Network (VPN) routers.

VPN routers use CEP to enroll in and get information about their PKI. The Certificate Enrollment Protocol interface uses CEP to issue new certificates, distribute Certificate Revocation List (CRL) data, and distribute the CA certificate.

Default Forms

There are no forms that use the Certificate Enrollment Protocol. The interface is provided so that VPN clients, such as routers, can use CEP to interact with the PKI.

Request Parameters

You will not generally develop your own request forms or response templates for use with CEP. The Certificate Enrollment Protocol interface complies with the CEP protocol developed by Cisco, so if your application or device uses this protocol it will be able to use the Certificate Enrollment Protocol Interface.

To use the interface with a Cisco router, for example, you configure the router to point to the end-entity gateway port using the router's enrollment url command. You can then use `crypto ca enroll` to request a certificate:

```
> crypto ca identity Example
> enrollment url https://example:443/
```

```
> crypto ca enroll Example
```

The router uses the CEP protocol and expects to find the `/cgi-bin/pkiclient.exe` interface at the URL named by the `enrollment url` command. The details of interacting with the interface are handled by the protocol itself.

Challenge Revocation Interface

Description

URI: `/challenge_revocation1`

Available on: Certificate Manager and Registration Manager

Function: Allows an entity to revoke a certificate using a challenge password set during enrollment.

The Challenge Revocation interface is useful if an entity must revoke a certificate that is not available or not valid for SSL client authentication. (The Revocation Interface can be used to present a certificate using SSL client authentication for revocation.) To use this interface, the challenge password for revocation must be set during enrollment (see the `challengePassword` request parameter in "Enrollment Interface" on page 49).

Default Forms

The `ChallengeRevoke1.html` form is the only default form that uses the Challenge Revocation interface. It allows an end user to enter either the certificate's serial number or subject name and the challenge password to revoke the certificate.

Request Parameters

The following table lists the parameters accepted by the Challenge Revocation interface.

Table 3-2 Parameters Accepted by the Challenge Revocation Interface

Parameter	Format and Description
`certSerialToRevoke`	number (decimal or hexadecimal) The serial number of the certificate to revoke. Either this parameter or `subjectName` are required.
`challengePhrase`	string The challenge phrase, set during certificate enrollment, that allows the certificate to be revoked.
`reasonCode`	0-8 The code for the reason why the certificate is being revoked. This code is added to the Certificate Revocation List (CRL) entry for the revoked certificate. Valid `reasonCode` values are: • `0` - Reason not specified • `1` - Key compromised • `2` - CA key compromised • `3` - Affiliation changes • `4` - Certificate superseded • `5` - Cessation of operation • `6` - Certificate is on hold
`subjectName`	Distinguished Name (DN) string. See RFC 2253. The DN appearing in the certificate subject field. Either subjectName or certSerialToRevoke must be specified. The subject name should only be used when the subject DN is guaranteed to uniquely identify the certificate. Example DN: `CN=Alice Apple, UID=alice, OU=People, O=Example, C=US`
`templateName`	string Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.
`templateType`	`RevocationConfirmation` This parameter specifies which response template to use. At this time, the only valid value is `"Revocation Confirmation."` This value causes the `revocationResult.template` file to be used.

Response

The response from the Challenge Revocation interface will be identical to a response from the Revocation interface. See the Response section in “Revocation Interface” on page 82 for details on what JavaScript variables are returned in the response template.

Display Certificate By Serial Number Interface

Description

URI: `/displayBySerial`

Available on: Certificate Manager

Function: Displays a single certificate in human-readable form.

The Display Certificate By Serial Number interface is typically used within a form that lists certificates to display detailed information about a selected certificate. The response is an HTML page built from a template (not just raw certificate data), so this interface should not be used to retrieve certificates for processing (such as importing into a browser); use the Get Certificate By Serial Number Interface (`/getBySerial`) instead.

Default Forms

The Display Certificate By Serial Number interface is used in the `queryCert.template` file. Each certificate in the list of certificates satisfying the query has a button the user can press to see the certificate in detail. This button submits data to the Display Certificate By Serial Number interface.

Request Parameters

The following table lists the parameters accepted by the Display Certificate By Serial Number interface.

Table 3-3 Parameters Accepted by the Display Certificate By Serial Number Interface

Parameter	Format and Description
`op`	`displayBySerial`
	Specifies the operation to perform. The only valid value is `displayBySerial`.
`serialNumber`	number
	The serial number of the certificate to display.
`templateName`	string
	Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.

Response

The default response template is `displayBySerial.template.` The base JavaScript for responses is inserted in place of the `<CMS_TEMPLATE>` tag. In addition, the Display Certificate By Serial Number interface adds the JavaScript variables listed in the following table:

Table 3-4 Variables Returned by the Display Certificate By Serial Number Interface

Variable	Description
result.header variables	Variables added to the header object.
`authorityid`	ca
	Indicates the source of the certificate information. Only Certificate Managers can return certificates by serial number directly.
`certChainBase64`	base-64 encoded data
	Contains the certificate in PKCS #7 format.
`certFingerprint`	string
	A string of hexadecimal numbers separated by colons that represent the certificate fingerprints. There are three substrings: one each for the MD2, MD5, and SHA1 fingerprint. Each fingerprint begins with the hash algorithm name and a colon, and ends with a newline (\n).

Table 3-4 Variables Returned by the Display Certificate By Serial Number Interface

Variable	**Description**
`certPrettyPrint`	string
	Contains details about the certificate in a human-readable form. This is the field used to show the certificate to a user in a page.
`serialNumber`	number
	The serial number of the certificate in decimal.

Display Certificate From Request Interface

Description

URI: `/displayCertFromRequest`

Available on: Certificate Manager or Registration Manager

Function: Retrieves the certificate associated with an enrollment or renewal request to be displayed in a response template.

The Display Certificate From Request interface is typically used in JavaScript embedded in the response template of an enrollment or renewal request. This interface uses the `requestID` returned in the JavaScript of a response to fetch the associated certificate.

In the `requestStatus.template` file, there is JavaScript code to build a URL that fetches the certificate from the Display Certificate From Request Interface if the CMS server is a Registration Authority.

The `requestID` parameter from the response template is required: it identifies the request from which to extract the certificate.

Default Forms

By default, the Display Certificate From Request interface is used by the `requestStatus.template` file only.

Request Parameters

The following table lists the parameters accepted by the Display Certificate From Request interface.

Table 3-5 Parameters Accepted by the Display Certificate From Request Interface

Parameter	Format and Description
`requestId`	number The `requestID` returned in the JavaScript by the Enrollment or Renewal interface (`fixed.requestID`).
`templateName`	string Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.

Response

By default, the `displayCertFromRequest.template` file is used to create the response. The `<CMS_TEMPLATE>` tag is replaced with the the base JavaScript for responses. In addition, the Get Certificate From Request interface adds the JavaScript variables listed in the following table:

Table 3-6 Variables Returned by the Display Certificate From Request Interface

Variable	Description
result.fixed variables	Variables added to the `fixed` object.
`authorityName`	`Certificate Manager` | `Registration Manager` The name of the system that handled the request.
`errorDescription`	string A message providing more details about the error described in `errorDetails`. This variable is only present if an error occurred while processing the request.

Table 3-6 Variables Returned by the Display Certificate From Request Interface *(Continued)*

Variable	**Description**
`errorDetails`	string A message explaining the error that occurred while processing the enrollment request. This variable is only present if an error occurred while processing the request.
`host`	string The fully qualified domain name of the CMS server that processed the request. This allows the resulting template to construct forms that post data to the same interface using the same port.
`port`	number The port number that was used to service the request.
requestId	number The request identification number that was requested.
`scheme`	`http` | `https` The protocol that was used to make the request. Use this along with `host` and `port` to make sure any new requests to the end-entity port use the correct scheme.
result.header variables	Variables added to the header object.
`emailCert`	`true` | `false` If true, the certificate contained in the `recordSet` array or `cmmfResponse` is a valid S/MIME certificate.
noCertImport	`true` | `false` Indicates whether the certificate should not be imported.
requestId	number The request identification number that was requested.
result.recordSet[i] variables	Variables added to each `record` object. Each `record` object is added as an element of the `recordSet` array. Multiple records may be returned if more than one certificate was generated as a result of the request. Dual-key requests (for example, if the request parameter `requestFormat = crmf`) may return two certificates if the request is successfully processed and approved.
`base64Cert`	string The newly issued certificate in base-64 encoded format. This string includes the "`-----BEGIN CERTIFICATE-----`" header and "`-----END CERTIFICATE-----`" footer.

Table 3-6 Variables Returned by the Display Certificate From Request Interface *(Continued)*

Variable	Description
`certFingerprint`	string A string of hexadecimal numbers separated by colons that represent the certificate fingerprints. There are three substrings: one each for the MD2, MD5, and SHA1 fingerprint. Each fingerprint begins with the hash algorithm name and a colon, and ends with a newline (\n).
`certPrettyPrint`	string A long text string that shows all of the certificate data in a human readable form.
`serialNo`	number The serial number (in decimal) of the certificate.

Enrollment Interface

Description

URI: `/enrollment`

Available on: Certificate Manager and Registration Manager.

Function: Enrolls an entity into the Public-Key Infrastructure (PKI).

This servlet uses data from an `HTTP POST` or `HTTP GET` to formulate a certificate request, hands the request off to a Certificate Manager, and returns a response (which may include the newly issued certificate) to the entity.

The certificate request may be based only on the data in the request, or it may get additional data from an authentication plug-in named in the `authenticator` parameter. If an authentication plug-in is used, the Certificate Manager may be able to automatically issue a certificate which is passed to the entity in the enrollment servlet’s response. If no authentication plug-in is used, the request is placed in an agent queue for manual approval and the enrollment servlet returns a “request pending” page to the entity.

NOTE The forms rely on a shared library called `xenroll.dll` (downloaded from the CMS server) to generate keys for Microsoft Internet Explorer browsers. By default, the keys generated by `xenroll.dll` have a "medium" security setting which means they will be stored unencrypted and that they can be used by the browser for signing without prompting the user for a password. A "high" security setting will store the keys in a separate, encrypted file and force the user to enter a password to use the keys for signing. There is no way to force a "high" setting for keys, but you can force a dialog to appear to allow the user to choose a security setting when the key is first generated. Edit the the VisualBasic script for `xenroll.dll` used in the enrollment forms (listed in the next section). Set the value of the `GenKeyFlags` parameter to 3 to prompt the user for a security setting when a key is generated using Microsoft Internet Explorer.

Default Forms

There are two types of default HTML forms that use the enrollment interface: manual or automated enrollment. Forms that use automated enrollment send an authentication plug-in name as a parameter in the request which the servlet can use to authenticate and process the request without manual intervention.

The default manual enrollment forms are:

- `ManUserEnroll.html` for requesting client certificates.
- `ManServerEnroll.html` for requesting server certificates.
- `ManObjSign.html` for requesting object signing certificates.
- `ManCAEnroll.html` for requesting subordinate Certificate Manager signing certificates.
- `ManRAEnroll.html` for requesting Registration Manager certificates.

The default automated enrollment forms are:

- `DirUserEnroll.html` uses a `UserDirEnrollment` instance of the `UidPwdDirAuth` plug-in class by default.
- `DirPinUserEnroll.html` uses a `PinDirEnrollment` instance of the `UidPwdPinDirAuth` plug-in class by default.

Request Parameters

The following table lists the parameters accepted by the enrollment interface.

Table 3-7 Parameters Accepted by the Enrollment Interface

Parameter	Format and Description
Subject Name	
`subject`	Distinguished Name (DN) string. See RFC 2253. DN to be used for the certificate subject. Example: `CN=Alice Apple, UID=alice, OU=People, O=Example, C=US`
Contact Information	
`csrRequestorName`	string Name of the entity making a request; helps identify the requestor during manual enrollment. Example: Alice Apple
`csrRequestorEmail`	string Email address of the entity making a request. May be used to send out notification when a certificate has been issued. Example: `alice@example.com`
`csrRequestorPhone`	string Phone number of the entity making a request. Example: `650.555.1212`
`csrRequestorComments`	string Additional comments provided by the requestor on the HTML form. This field can be used if there is additional information you want to collect to help the manual enrollment.
Netscape Certificate Type Extensions	Parameters for setting bits in the `netscape-cert-type` certificate extension. See `http://home.netscape.com/eng/security/comm4-cert-exts.html` for details. A `true` value sets the bit to 1; `false` sets the bit to 0.
`email`	`true` | `false` Sets the S/MIME client certificate bit (bit 2).
`email_ca`	`true` | `false` Sets the S/MIME certificate issuer bit (bit 6).

Table 3-7 Parameters Accepted by the Enrollment Interface *(Continued)*

Parameter	Format and Description
`object_signing`	`true` | `false`
	Sets the object signing certificate bit (bit 3).
`object_signing_ca`	`true` | `false`
	Sets the object signing certificate issuer bit (bit 7).
`ssl_ca`	`true` | `false`
	Sets the SSL certificate issuer bit (bit 5).
`ssl_client`	`true` | `false`
	Sets the SSL client authentication certificate bit (bit 0).
`ssl_server`	`true` | `false`
	Sets the SSL server authentication certificate bit (bit 1).
Key Usage	Parameters for setting bits in the `keyUsage` certificate extension. A `true` value sets the bit to 1; `false` sets the bit to 0.
`crl_sign`	`true` | `false`
	Sets the `keyUsage` extension bit (6) indicating that the key may be used to sign Certificate Revocation Lists (CRLs).
`data_encipherment`	`true` | `false`
	Sets the `keyUsage` extension bit (3) indicating that the key may be used to encipher application data (as opposed to key material).
`decipher_only`	`true` | `false`
	Sets the `keyUsage` extension bit (8) indicating that the key may only be used to decipher data and keys. If this parameter is `true`, `keyAgreement` should also be `true`.
`digital_signature`	`true` | `false`
	Sets the `keyUsage` extension bit (0) indicating that the key may be used to sign any data. This parameter should be `true` for SSL client certificates, S/MIME signing certificates, and object signing certificates.
`encipher_only`	`true` | `false`
	Sets the `keyUsage` extension bit (7) indicating that the key may only be used to encipher data and keys. If this parameter is `true`, keyAgreement should also be true.

Table 3-7 Parameters Accepted by the Enrollment Interface *(Continued)*

Parameter	Format and Description
`key_agreement`	`true` | `false`
	Sets the `keyUsage` extension bit (4) indicating that the key may be used to encipher and decipher keys during key agreement.
`key_certsign`	`true` | `false`
	Sets the `keyUsage` extension bit (5) indicating that the key may be used to sign other certificates. All CA signing certificates should set this parameter to `true`.
`key_encipherment`	`true` | `false`
	Sets the `keyUsage` extension bit (2) indicating that the key may be used to encipher symmetric session keys. This parameter should be `true` for SSL server and S/MIME encryption certificates.
`non_repudiation`	`true` | `false`
	Sets the `keyUsage` extension bit (1) indicating that the key may be used to create non-repudiable (by the signer) digital signatures. Non-repudiation service requires more infrastructure, planning, and policy than just setting this bit. Consider the ramifications before using this bit
Automated Enrollment	Parameters to configure automatic authentication for entity requests.
`authenticator`	string
	Specifies the name of the authentication plug-in instance to use to authenticate the entity.
`uid`	string
	Specifies a unique identifier passed to the authentication plug-in.
`pin`	string
	An optional identifying string that helps to authenticate an entity. Usually used when the Pin Generator tool has been used to populate a directory with unique identifiers for each user.
`pwd`	string
	Specifies the password passed to the authentication plug-in.
Other	
`certNickname`	string
	Specifies the nickname that should be associated with the certificate in the reply; used with Certificate Request Management Format (CRMF) requests.

Table 3-7 Parameters Accepted by the Enrollment Interface *(Continued)*

Parameter	Format and Description
`certType`	`ca` | `CEP-Request` | `client` | `objSignClient` | `ra` | `server` | `other`
	Specifies the type of certificate requested by the entity. The default is `client`. The `certType` is not associated with any certificate extensions. It may be used by policy modules to make decisions, and it may be used by a CMS server to determine how to decode the request or format the response.
`challengePassword`	string
	An optional challenge phrase or password that can be used later by the entity to revoke the certificate. This parameter is optional. If you use this, entities can use the "Challenge Revocation Interface" (`/challenge_revocation1`, page 42)with this challenge password to revoke a certificate without manual intervention and without SSL client authentication.
`CRMFRequest`	base-64 encoded data
	If `requestFormat = crmf`, this parameter should be used to send the base-64 encoded CRMF request.
`importCAChain`	`true` | `false`
	Used only when `importCert = true`. The default, if this parameter is not explicitly passed, is `true`. If set to `true`, a successful certificate request will return a PKCS #7 formatted certificate chain; if set to `false`, a single, DER-encoded certificate will be returned. The certificate chain includes the issued certificate and the CA (issuer) certificate.
`importCert`	`true` | `false`
	If `true`, and the certificate request is not deferred or rejected, the CMS server's response will be binary data with the MIME type determined by the `importCertMimeType` parameter. The data returned will be either a certificate or a certificate chain, based on the value of `importCAChain`.
`importCertMimeType`	MIME Type string
	Sets the MIME type the CMS server uses when a certificate is returned to the requestor. The default is `application/x-x509-user-cert`. The MIME type should be in the standard MIME type format of `<type>/<subtype>`.
`pkcs10Request`	base-64 encoded data
	If `requestFormat = pkcs10`, this parameter should be used to send the base-64 encoded certificate request.

Table 3-7 Parameters Accepted by the Enrollment Interface *(Continued)*

Parameter	Format and Description
`requestFormat`	`clientAuth` | `crmf` | `keygen` | `pkcs10` The value indicates the format used to submit the certificate request: • `clientAuth` - information for the new request is taken from the certificate presented by the client during SSL client authentication. • `crmf` - the certificate request is a base-64 encoded blob contained in the `CRMFRequest` parameter. • `keygen` - the certificate request is a base-64 encoded blob generated using the HTML `<KEYGEN>` tag. It is contained in the `subjectKeyGenInfo` parameter. • `pkcs10` - the certificate request is a base-64 encoded blob contained in the `pkcs10Request` parameter.
`subjectKeyGenInfo`	base-64 encoded data If `requestFormat=keygen`, this parameter should be used to send the base-64 encoded `keygen` request. To use the `<KEYGEN>` HTML tag to cause the browser to generate the request using this parameter, the format is `<KEYGEN name="subjectKeyGenInfo">`
`templateName`	string Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.

Response

The Registration Authority or Certificate Authority that process an enrollment request will perform some processing, determine the status of the request, then return a result using the appropriate template for the status.

The response templates are ASCII files that you can edit to create responses suited to your needs. The templates may include JavaScript that depends on the JavaScript inserted in place of the `<CMS_TEMPLATE>` tag when the response is sent. The status of the request determines which template will be used for the response. The following table describes the templates used by the enrollment interface (more details on the request status codes can be found in Table 3-9):

Table 3-8 Enrollment Interface Response Templates

Template File Name	Request Status	Description
`EnrollSuccess.template`	2 (Success)	Used only for requests that specify an authenticator. If authentication and subsequent policy processing are successful and `importCert` was "`true`" in the request, a certificate is generated (otherwise, see `GenRejected.template`). The issued certificate is included in base-64 in the response. The template includes JavaScript and VisualBasic code that attempts to import the certificate into a browser's certificate database.
`GenError.template`	6 (Error)	Used to display an error message.
`GenPending.template`	3 (Pending)	Used to inform the user that the certificate request he or she submitted has been queued for agent approval.
`GenRejected.template`	5 (Rejected)	Used to inform the user that the certificate request he or she submitted has been rejected by the CMS server.
`GenSuccess.template`	2 (Success)	Used to inform the user that the certificate request he or she submitted has been approved by the CMS server.
`GenSvcPending.template`	3 (Pending)	Used to inform the user that the certificate request he or she submitted has been queued for agent approval.
`GenUnauthorized.template`	1 (Unauthorized)	Used to inform the user that he or she performed an unauthorized operation.

The `<CMS_TEMPLATE>` tag in the selected template is replaced with the base JavaScript code. In addition, the Enrollment interface may add the JavaScript variables listed in the following table. Not all templates use all of the variables listed in the table; a variable is only included when it is used (for example, the `EnrollmentSuccess.template` does not include `result.fixed.unexpectedError`).

Table 3-9 Variables Returned by the Enrollment Interface

Variable	Description
result.fixed variables	Variables added to the `fixed` object.

Table 3-9 Variables Returned by the Enrollment Interface *(Continued)*

Variable	Description
`authorityName`	`Certificate Manager` | `Registration Manager` The name of the system that handled the request.
`certType`	`ca` | `CEP-Request` | `client` | `objSignClient` | `ra` | `server` | `other` The type of certificate returned. This value is the same as the `certType` value passed to the interface in the request.
`errorDescription`	string A message providing more details about the error described in `errorDetails`. This variable is only present if an error occurred while processing the request.
`errorDetails`	string A message explaining the error that occurred while processing the enrollment request. This variable is only present if an error occurred while processing the request.
`host`	string The fully qualified domain name of the CMS server that processed the request. This allows the resulting template to construct forms that post data to the same interface using the same port.
`port`	number The port number that was used to service the request.
`requestId`	number A unique number assigned by the CMS server to this request. This is especially useful for pending requests since there is no unique certificate serial number yet assigned.

Table 3-9 Variables Returned by the Enrollment Interface *(Continued)*

Variable	Description
`requestStatus`	number A code indicating the current status of the request: • 1 (Unauthorized): The request specified a value for an authenticator to perform an automated enrollment, and the authenticator did not authorize the request. • 2 (Success): Processing the request was successful and a certificate has been issued. If `importCert` was set to `true`, the response will include code (from the `EnrollSuccess.template`) to import the certificate into the browser making the request. Otherwise, the response is only a success message. • 3 (Pending): The request has been successfully processed by the CMS server and added to a queue for approval by an agent. If the request has been submitted to another Certificate Manager or Data Recovery Manager and is currently pending in the queue for that service, the response template will be `GenSvcPending.template` instead of `GenPending.template`. • 4 (Reserved): Not currently used. • 5 (Rejected): The request was rejected during policy processing. • 6 (Error): An error occurred when the CMS server processed the request. The error may be the result of missing or improperly formatted parameters. • 7 (Exception): An unknown or unexpected error occurred when the CMS server processed the request.
`scheme`	`http` | `https` The protocol that was used to make the request. Use this along with `host` and `port` to make sure any new requests to the end-entity port use the correct scheme.
`unexpectedError`	string A message explaining the exception or unexpected error that occurred.
result.recordSet[i] variables	Variables added to each `record` object. Each `record` object is added as an element of the `recordSet` array. Multiple records may be returned if more than one certificate was generated as a result of the request. Dual-key requests (for example, if the request parameter `requestFormat = crmf`) may return two certificates if the request is successfully processed and approved.

Table 3-9 Variables Returned by the Enrollment Interface *(Continued)*

Variable	Description
`base64Cert`	string The newly issued certificate in base-64 encoded format. This string includes the "`-----BEGIN CERTIFICATE-----`" header and "`-----END CERTIFICATE-----`" footer.
`certFingerprint`	string A string of hexadecimal numbers separated by colons that represent the certificate fingerprints. There are three substrings: one each for the MD2, MD5, and SHA1 fingerprint. Each fingerprint begins with the hash algorithm name and a colon, and ends with a newline (`\n`).
`certPrettyPrint`	string A long text string that shows all of the certificate data in a human readable form.
`policyMessage`	string If the request was rejected by policy processing on the CMS server, this variable will contain a message explaining why.
`serialNo`	number The serial number (in decimal) of the newly issued certificate.

Get CA Chain Interface

Description

URI: `/getCAChain`

Available on: Certificate Manager only.

Function: Retrieves the CA certificate or certificate chain for the Certificate Manager either in binary form for use by an application or in a format for display.

The Get CA Chain interface accepts an operation (for example, download) and a MIME type to be used for the response. The response is always the CA certificate or certificate chain (if the CA certificate is not self-signed) for the Certificate Manager handling the request. No templates are used; the response is either ASCII data that can be displayed or a binary blob (using the indicated MIME type) that can be used by the requesting application.

Using the Get CA Chain interface to display certificates is useful for creating data that can be imported into another application such as an HTTP or LDAP server.

Default Forms

The Get CA Chain interface uses one default form: `GetCAChain.html`. This form allows an entity to choose one of four operations:

- Import the CA certificate chain into a browser (`download`).
- Download the CA certificate chain in binary form (`downloadBIN`).
- Display the CA certificate chain in PKCS #7 for importing into a server (`display`).
- Display certificates in the CA certificate chain for importing individually into a server (`displayIND`).

Request Parameters

The following table lists the parameters accepted by the Get CA Chain interface.

Table 3-10 Parameters Accepted by the Get CA Chain Interface

Parameter	Format and Description
`mimeType`	`<type>/<subtype>` Indicates the MIME the CMS server should use for the response. The default form uses `application/x-x509-ca-cert` for downloads. For `op = downloadBIN`, `mimeType` is ignored and `application/octet-stream` is always used.

Table 3-10 Parameters Accepted by the Get CA Chain Interface

Parameter	Format and Description
`op`	`display` | `displayIND` | `download` | `downloadBIN` This required parameter specifies how the CA certificate chain should be returned: • `display` returns a base-64 encoded PKCS #7 certificate chain. • `displayIND` returns each certificate in the CA chain in a base-64 encoded DER blob and a human-readable format. • `download` returns the entire certificate chain in binary form using the MIME type specified. If the browser is not Internet Explorer, the chain is returned as a PKCS #7 blob. If the browser is Internet Explorer, only the CA signing certificate will be returned in a DER-encoded format. • `downloadBIN` is the same as download, except that the MIME type is always set to `application/octet-stream`. Use of `downloadBIN` is deprecated; use `download` and set `mimeType = application/octet-stream` in the request instead.
`templateName`	string Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.

Response

The Get CA Chain interface does not use any templates. The response is just the requested certificate or certificates in the format indicated by the request parameters.

Get Certificate By Serial Number Interface

Description

URI: `/getBySerial`

Available on: Certificate Manager only

Function: Retrieves the certificate with the given serial number in a specified format. The certificate can be imported into a browser.

This interface is used in the `EnrollSuccess.template` and `RenewalSuccess.template` to download and import the newly issued certificate. The `displayBySerial.template` also uses this interface to create "Import Certificate" and "Import S/MIME Certificate" buttons on a page that displays a certificate; this allows a user to retrieve and import a certificate that was issued manually.

Default Forms

There are no default forms that use the Get Certificate By Serial Number interface. This interface is usually used embedded in a response template to either embed a certificate in the response or provide a button on a form that downloads and imports the certificate.

Request Parameters

The following table lists the parameters accepted by the Get Certificate By Serial Number interface.

Table 3-11 Parameters Accepted by the Get Certificate By Serial Number Interface

Parameter	Format and Description
`cmmfResponse`	`true` | `false` Indicates whether the certificate should be returned using CMMF. The CMMF format can be used with browsers that understand it to allow the browser to import the certificate into its database. The CMMF data will be returned as a JavaScript variable in the response.
`emailCert`	`true` | `false` Indicates whether the certificate returned is expected to be valid for signing e-mail.
`importCert`	`true` | `false` Indicates how to format the certificate in the response if `cmmfResponse` is `false` or not set. By default, the response will be a page created from the ImportCert.template file. If `importCert = true`, the MIME type will be set to `application/x-x509-user-cert` and the data will be a binary blob in PKCS #7 format.

Table 3-11 Parameters Accepted by the Get Certificate By Serial Number Interface *(Continued)*

Parameter	Format and Description
`serialNumber`	number The serial number of the certificate to retrieve.
`templateName`	string Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.

Response

If `importCert = true` in the request, the response is a binary blob containing the certificate and no templates are used to construct the response. See the Request Parameters for details on how the response is formatted.

By default, the `ImportCert.template` file is used to create the response. The `<CMS_TEMPLATE>` tag is replaced with the base JavaScript for responses. In addition, the Get Certificate By Serial Number interface adds the JavaScript variables listed in the following table:

Table 3-12 Variables Returned by the Get Certificate By Serial Number Interface

Variable	Description
result.fixed variables	Variables added to the `fixed` object.
`authorityName`	`Certificate Manager` | `Registration Manager` The name of the system that handled the request.
`certNickname`	string The nickname for the certificate. By default, this is just the certificate subject DN.
`certType`	`ca` | `CEP-Request` | `client` | `objSignClient` | `ra` | `server` | `other` The type of certificate returned. This value is the same as the `certType` value passed to the interface in the request.

Table 3-12 Variables Returned by the Get Certificate By Serial Number Interface *(Continued)*

Variable	Description
`cmmfResponse`	base-64 encoded data The CMMF response data containing the certificate (if `cmmfResponse` was `true` in the request). If the browser supports the Personal Security Manager crypto API, you can use this response with a call to `importUserCertificates` to import the certificate into a local database: `importUserCertificates(result.fixed.nickname, result.fixed.cmmfResponse, true);` (The last parameter indicates whether the user should be prompted to back up the key.)
`errorDescription`	string A message providing more details about the error described in `errorDetails`. This variable is only present if an error occurred while processing the request.
`errorDetails`	string A message explaining the error that occurred while processing the enrollment request. This variable is only present if an error occurred while processing the request.
`host`	string The fully qualified domain name of the CMS server that processed the request. This allows the resulting template to construct forms that post data to the same interface using the same port.
`port`	number The port number that was used to service the request.
`scheme`	`http` | `https` The protocol that was used to make the request. Use this along with host and port to make sure any new requests to the end-entity port use the correct scheme.
	Variables added to each `record` object. Each `record` object is added as an element of the `recordSet` array. Multiple records may be returned if more than one certificate was generated as a result of the request. Dual-key requests (for example, if the request parameter `requestFormat = crmf`) may return two certificates if the request is successfully processed and approved.

Table 3-12 Variables Returned by the Get Certificate By Serial Number Interface *(Continued)*

Variable	Description
`base64Cert`	string The newly issued certificate in base-64 encoded format. This string includes the "`-----BEGIN CERTIFICATE-----`" header and "`-----END CERTIFICATE-----`" footer.
`certFingerprint`	string A string of hexadecimal numbers separated by colons that represent the certificate fingerprints. There are three substrings: one each for the MD2, MD5, and SHA1 fingerprint. Each fingerprint begins with the hash algorithm name and a colon, and ends with a newline (`\n`).
`certPrettyPrint`	string A long text string that shows all of the certificate data in a human readable form.
`serialNo`	number The serial number (in decimal) of the certificate.

Get Certificate From Request Interface

Description

URI: `/getCertFromRequest`

Available on: Certificate Manager or Registration Manager

Function: Retrieves the certificate associated with an enrollment or renewal request to be displayed in a response template or imported into a browser.

The Get Certificate From Request interface is typically used in JavaScript embedded in the response template of an enrollment or renewal request. This interface uses the `requestID` returned in the JavaScript of a response to fetch the associated certificate.

In the `EnrollSuccess.template` and `RenewalSuccess.template` files, there is JavaScript code to build a URL that fetches the certificate from the Get Certificate From Request interface. The URL is assigned to the `window.location` object, which causes the contents of the URL to be displayed in-line in the response.

The `requestID` parameter from the response template is required: it identifies the request from which to extract the certificate. A parameter can also be used to instruct the requesting browser to import the certificate into its database: `importCert` or `cmmfResponse` (for browsers that support CMMF).

Default Forms

The Get Certificate From Request interface is used by response templates, not forms that an entity submits. The interface is used in these templates to incorporate the certificate in the page itself, so that it can be displayed and possibly imported into the browser. The templates that use the interface by default are:

- `displayCertFromRequest.template` is a generic template that shows how to display a certificate from a request.
- `EnrollSuccess.template` is the template returned by a successful enrollment request (when a certificate has been issued, not when the request is successful but still pending).
- `RenewalSuccess.template` is the template returned by a successful automated renewal request (such as a renewal using SSL client authentication).

Request Parameters

The following table lists the parameters accepted by the Get Certificate From Request interface.

Table 3-13 Parameters Accepted by the Get Certificate From Request Interface

Parameter	Format and Description
`cmmfResponse`	`true` | `false` Indicates whether the returned certificate should be formatted using Certificate Management Message Format (CMMF). Entities that support CMMF can use the result to import the certificate into a local database. The CMMF data will be returned as a JavaScript variable in the response.
`importCert`	`true` | `false` Indicates whether the returned certificate should be imported into the local database. This causes the MIME type of the returned HTTP to be `application/x-x509-user-cert` and the certificate to be a PKCS #7 formatted binary blob.

Table 3-13 Parameters Accepted by the Get Certificate From Request Interface *(Continued)*

Parameter	Format and Description
`requestId`	number The `requestID` returned in the JavaScript by the Enrollment or Renewal interface (`fixed.requestID`).
`templateName`	string Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.

Response

If `importCert = true` in the request, the response is a binary blob containing the certificate and no templates are used to construct the response. See the Request Parameters for details on how the response is formatted.

By default, the `ImportCert.template` file is used to create the response. The `<CMS_TEMPLATE>` tag is replaced with the base JavaScript for responses. In addition, the Get Certificate From Request interface adds the JavaScript variables listed in the following table:

Table 3-14 Variables Returned by the Get Certificate From Request Interface

Variable	Description
result.fixed variables	Variables added to the `fixed` object.
`authorityName`	`Certificate Manager` | `Registration Manager` The name of the system that handled the request.
`certNickname`	string The nickname for the certificate. By default, this is just the certificate subject DN.

Table 3-14 Variables Returned by the Get Certificate From Request Interface *(Continued)*

Variable	Description
`cmmfResponse`	base-64 encoded data The CMMF response data containing the certificate (if `cmmfResponse` was `true` in the request). If the browser supports the Personal Security Manager crypto API, you can use this response with a call to `importUserCertificates` to import the certificate into a local database: `importUserCertificates(result.fixed.nickname, result.fixed.cmmfResponse, true);` (The last parameter indicates whether the user should be prompted to back up the key.)
`errorDescription`	string A message providing more details about the error described in `errorDetails`. This variable is only present if an error occurred while processing the request.
`errorDetails`	string A message explaining the error that occurred while processing the enrollment request. This variable is only present if an error occurred while processing the request.
`host`	string The fully qualified domain name of the CMS server that processed the request. This allows the resulting template to construct forms that post data to the same interface using the same port.
`port`	number The port number that was used to service the request.
`requestId`	number The request identification number that was requested.
`scheme`	`http` | `https` The protocol that was used to make the request. Use this along with host and port to make sure any new requests to the end-entity port use the correct scheme.
result.header variables	Variables added to the header object.
`emailCert`	`true` | `false` If true, the certificate contained in the `recordSet` array or `cmmfResponse` is a valid S/MIME certificate.

Table 3-14 Variables Returned by the Get Certificate From Request Interface *(Continued)*

Variable	Description
`noCertImport`	`true` | `false` Indicates whether the certificate should not be imported.
`requestId`	number The request identification number that was requested.
result.recordSet[i] variables	Variables added to each `record` object. Each `record` object is added as an element of the `recordSet` array. Multiple records may be returned if more than one certificate was generated as a result of the request. Dual-key requests (for example, if the request parameter `requestFormat = crmf`) may return two certificates if the request is successfully processed and approved.
`base64Cert`	string The newly issued certificate in base-64 encoded format. This string includes the "`-----BEGIN CERTIFICATE-----`" header and "`-----END CERTIFICATE-----`" footer.
`certFingerprint`	string A string of hexadecimal numbers separated by colons that represent the certificate fingerprints. There are three substrings: one each for the MD2, MD5, and SHA1 fingerprint. Each fingerprint begins with the hash algorithm name and a colon, and ends with a newline (\n).
`certPrettyPrint`	string A long text string that shows all of the certificate data in a human readable form.
`serialNo`	number The serial number (in decimal) of the certificate.

Get CRL Interface

Description

URI: `/getCRL`

Available on: Certificate Manager only

Function: Retrieves the current Certificate Revocation List (CRL) for this certificate authority.

This interface can be used to retrieve a CRL for display or importing into an application and it can be used simply to check whether a certificate appears on the current CRL.

Default Forms

The only default form that uses the Get CRL interface is `DisplayCRL.html`. This form allows the user to choose any of the possible options for the getCRL interface:

- Check whether a particular certificate is on the CRL.
- Import the CRL into a browser.
- Display the CRL in binary form.
- Display the CRL in human-readable form (pretty print).

Request Parameters

The following table lists the parameters accepted by the Get CRL interface.

Table 3-15 Parameters Accepted by the Get CRL Interface

Parameter	Format and Description
`certSerialNumber`	Number string in decimal (e.g., `330`) or hexadecimal (`0x14A`). If `op = checkCRL`, use this parameter to specify the serial number of the certificate to check.
`issuepoint`	`MasterCRL` The default value, `MasterCRL`, indicates that the complete master CRL should be checked. Other issue points may be configured for a Certificate Manager; in that case use the token that defines the issue point in the configuration file (`CMS.cfg`).

Table 3-15 Parameters Accepted by the Get CRL Interface *(Continued)*

Parameter	Format and Description
`op`	`checkCRL` | `displayCRL` | `getCRL` | `importCRL` This required parameter specifies the CRL operation to perform: • `checkCRL` instructs the Certificate Manager to look for the serial number specified in `certSerialNumber` on the CRL. The result is returned in the `result.header.isOnCRL` field. • `displayCRL` returns the entire CRL formatted in HTML for display in a browser. • `getCRL` returns the entire CRL as a PKCS #7 formatted blob; the MIME type of the response will be `application/octet-stream` for Communicator clients or `application/x-pkcs7-crl`. for Internet Explorer. • `importCRL` is the same as `getCRL` except the MIME type is always `application/x-pkcs7-crl.`
`templateName`	string Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.

Response

Responses to requests for `checkCRL` or `displayCRL` use the displayCRL.template template. The `<CMS_TEMPLATE>` tag is replaced with the base JavaScript for responses. In addition, the Get CRL interface adds the JavaScript variables listed in the following table:

Table 3-16 Variables Returned by the Get CRL Interface

Variable	Description
result.header variables	Variables added to the header object.
`certSerialNumber`	number If `toDo = checkCRL`, this field contains the serial number of the certificate that was checked (from `certSerialNumber` in the request).

Table 3-16 Variables Returned by the Get CRL Interface *(Continued)*

Variable	Description
`crlBase64`	base-64 encoded data
	The base-64 encoded CRL data in PKCS #7 format.
`crlPrettyPrint`	string
	Contains the CRL formatted for human-readable display if `op=displayCRL` in the request.
`isOnCRL`	`true` | `false`
	If `toDo = checkCRL`, this field indicates whether the named certificate serial number appears on the CRL.
`toDo`	`displayCRL` | `checkCRL`
	Indicates the type of result being returned.

List Certificates Interface

Description

URI: `/listCerts`

Available on: Certificate Manager

Function: Retrieves a list of certificates that match a query filter.

The query criteria are search filters that the interface uses to select certificates in the Certificate Manager's repository. The `queryCert.html` default form contains JavaScript code that can construct all possible filters. You should study this file for examples before you write code to construct your own forms. Valid query filter constructions are explained in the Request Parameters section.

The response is constructed using the `queryCert.template`. The listing that this template provides by default has code for displaying a listed certificate in more detail, revoking a listed certificate, or revoking all certificates listed.

Default Forms

The List Certificates interface uses two default forms:

- `queryBySerial.html` is a simple form that accepts a lower and upper bound for the range of serial numbers and the option to skip revoked or invalid certificates. This form constructs a simple query filter to select certificates that meet the user's preferences for certificate status (valid, invalid, and revoked).
- `queryCert.html` is a complex form that allows the user to specify all possible query criteria. This form makes extensive use of JavaScript to formulate a query filter for any criteria the user chooses.

Request Parameters

The following table lists the parameters accepted by the List Certificates interface.

The `queryCertFilter` parameter must be a valid query filter. The syntax and valid query parameters are too complex to describe in the parameter table. Details about valid parameters and values for query filters are in a separate table following the parameters.

Table 3-17 Parameters Accepted by the List Certificates Interface

Parameter	Format and Description
`maxCount`	number Specifies the maximum number of certificates to display on each page returned. If more than `maxCount` certificates match the search criteria, each page will have controls to see the next or previous page of results.
`op`	`listCerts` The only operation supported by the List Certificates interface is `listCerts`.

Table 3-17 Parameters Accepted by the List Certificates Interface *(Continued)*

Parameter	Format and Description

`queryCertFilter`	`([<OP>]<FILTER>[<FILTER>...])`
	
	Details about building query filters are provided in the next table.
	
	The `queryCertFilter` must be enclosed in parentheses.
	
	The `<OP>` argument, required if there is more than one `<FILTER>`, specifies how the filters that follow should be logically evaluated:
	
	• `&` (ampersand) means that the filters should be linked by a logical AND: all filters must evaluate to true for the expression to match a certificate.
	
	• `|` (pipe) means that the filters should be linked by a logical OR: if at least one filter evaluates to true, the expression matches a certificate.
	
	Any number of filters can be concatenated within any set of parentheses.
	
	An example filter is
	
	`(&(certStatus=VALID)(|(x509cert.nsExtension.SSLClient=on)( x509cert.nsExtension.SecureEmail=on)))`
	
	This filter matches any certificate that is valid and has either the SSL Client or S/MIME bit set in the netscape-cert-type extension.

`querySentinel`	`number` number The `querySentinel` indicates which record out of the total matching set should be the first displayed on the resulting output page. For example, if `totalRecordCount = 15` and `maxCount = 5`, set `querySentinel` to 6 to show the 6th through 10th element of the matching set.
`templateName`	string Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.

The following table describes the parameter names that are valid for constructing query filters, and the range of valid values that can be used with the parameter. The parameters can be combined using parentheses and logical operators (as described in the previous table) to construct query filters of arbitrary complexity.

In a filter, the parameter name is compared to the expression value using one of the relational operators = (matches), < (less than), <= (less than or equal to), > (greater than), or >= (greater than or equal to). Some expressions (such as `x509cert.subject`) accept the asterisk (*) as a wildcard to match 0 or more characters; for example. `"E=jdoe*"` matches `"E=jdoe@example.com"` and `"E=jdoe@example.com."`

Table 3-18 List Certificates queryCertFilter Parameters

Parameter	Expression Values

`certCreateTime`	Value: date (number of seconds since Jan 1, 1970)
	
	A date object can be created using the JavaScript `Date()` constructor.
	
	This parameter matches the date a certificate was issued. For example, to find certificates created during 1999:
	
	`Object lowDate = new Date(1999,00,01);`
	`Object highDate = new Date(1999,11,31);`
	`form.queryCertFilter.value =`
	`"(&(certCreateTime>=" + lowDate + ")" +`
	`"(certCreateTime<=" + highDate + ")";`

`certIssuedBy`	Value: user ID of an agent issuing a certificate Use the asterisk (*) wildcard to match partial names. For example, `(certIssuedBy=localAgent*)`.
`certRecordId`	Value: number in decimal or hexadecimal. This parameter matches the serial number on a certificate. Connect a lower and upper bound with a logical AND (&) to specify a bounded range of serial numbers. For example: `(&(certRecordId>=100)(certRecordId<=199))`
`certRevokedBy`	Value: user ID of an agent that revoked a certificate Use the asterisk (*) wildcard to match partial names. For example, `(certRevokedBy=localAgent*)`.
`certRevokedOn`	Value: date (number of seconds since Jan 1, 1970) A date object can be created using the JavaScript `Date()` constructor. This parameter matches the date when a certificate was revoked. See `certCreateTime` for an example of creating a date value in JavaScript

Table 3-18 List Certificates queryCertFilter Parameters *(Continued)*

Parameter	Expression Values
`certStatus`	Value: `*` | `EXPIRED` | `INVALID` | `REVOKED` | `VALID` | `REVOKED_EXPIRED` This parameter matches the current status of a certificate. The asterisk (*) matches any status.
`x509cert.certRevoInfo`	Value: `*` | number between 0 and 6 This parameter matches the reason for revocation code on a certificate. The revocation codes are: • `0` - Reason not specified • `1` - Key compromised • `2` - CA key compromised • `3` - Affiliation changes • `4` - Certificate superseded • `5` - Cessation of operation • `6` - Certificate is on hold To search for multiple values, construct a filter with multiple `x509cert.certRevoInfo` parameters connected with a logical OR. Do not connect these parameters with an AND, since a certificate cannot have more than one revocation reason. For example, to match certificates revoked due to key compromise or an unspecified reason: `(|(x509cert.certRevoInfo=0)(x509cert.certRevoInfo=1))`
`x509cert.duration`	Value: number of milliseconds This parameter matches the total number of milliseconds of a certificates validity period. Typically a range is specified using filters with >= and <= operators, rather than an exact match. The following list shows the number of milliseconds in some typical time intervals: • Day: 86,400,000 • Week: 604,800,000 • Month (30 Days): 2,592,000,000 • Year: 31,536,000,000

Table 3-18 List Certificates queryCertFilter Parameters *(Continued)*

Parameter	Expression Values
`x509cert.notAfter`	Value: date (number of seconds since Jan 1, 1970) A date object can be created using the JavaScript `Date()` constructor. This parameter matches the date when a certificate expires. See `certCreateTime` for an example of creating a date value in JavaScript
`x509cert.notBefore`	Value: date (number of seconds since Jan 1, 1970) A date object can be created using the JavaScript `Date()` constructor. This parameter matches the date when a certificate became valid. See `certCreateTime` for an example of creating a date value in JavaScript
`x509cert.nsExtension.<X>`	Value: `on` | `off` This parameter matches the bit in the ns-cert-type extension specified by the extension `<X>`. Substitute an extension identifier for `<X>`: • `SecureEmail` - for S/MIME certificates • `SSLClient` - for SSL client certificates • `SSLServer` - for SSL server certificates • `SubordinateEmailCA` - for certificates with the S/MIME CA bit set • `SubordinateSSLCA` - for certificates with the SSL CA bit set For example, to match only certificates with the SSL client bit (bit 0) set in the ns-cert-type extension: `(x509cert.nsExtension.SSLClient=on)`

Table 3-18 List Certificates queryCertFilter Parameters *(Continued)*

Parameter	Expression Values

`x509cert.subject`	Value: a pattern that may include the wildcard (*)
	
	This parameter matches the certificate subject DN. You can use a single filter or connect multiple filters to build more complex DN patterns.
	
	The value is typically a string in the form `*<TAG>=<VALUE>*`. The asterisks allow the name-value pair to be matched at any location in the DN. The tag is one of the subject DN attributes: `CN`, `E`, `UID`, `OU`, `O`, `L`, `ST`, `C`.
	
	To allow partial matches, use the wildcard in the attribute value. For example, to match email addresses containing `"jdoe,"`
	
	`(x509cert.subject=*E=*jdoe*)`
	
	To force an exact match of `"jdoe@example.com,"` you should still use the wildcard to allow the `E` attribute to occur anywhere in the DN:
	
	`(|(x509cert.subject=*E=jdoe@example.com,*)`
	`(x509cert.subject=*E=jdoe@example.com))`

Response

The default response template is `queryCert.template`. The base JavaScript for responses is inserted in place of the `<CMS_TEMPLATE>` tag. In addition, the Revocation interface adds the JavaScript variables listed in the following table:

Table 3-19 Variables Returned by the List Certificates Interface

Variable	Description
result.header Variables	Variables added to the `header` object.
`currentRecordCount`	The total number of certificates displayed on this page of output. This number may be less than totalRecordCount.
`issuerName`	The distinguished name (DN) of the certificate authority that processed the query. This DN appears in the issuer field of all of the certificates listed. Example: CN=Certificate Manager, O=Organization, C=US
`maxCount`	The maximum number of certificate records to display on any single page of output.
`op`	The operation parameter to send (to the `serviceURL`) when the user requests more certificates. This value will always be `listCerts` for the List Certificates interface.

Table 3-19 Variables Returned by the List Certificates Interface *(Continued)*

Variable	Description
`queryCertFilter`	The `queryCertFilter` parameter that was used to generate the current list of certificates, and will be used for subsequent pages if the user requests to see more certificates. For information on how the filter is constructed, see the Request Parameters section. An example `queryCertFilter` is `(&(certStatus=VALID)(certRecordId>=100))`
`querySentinel`	This field holds the number of the lowest certificate serial number to display on the current page of output.
`serviceURL`	The URI to use to post requests for more certificates matching the query criteria. This variable will always be `/listCerts` for the List Certificates interface.
`templateName`	The name of the response template the CMS server should use to construct a response from the `serviceURL`. By default, this field will always be `queryCert.template` for the List Certificates interface.
`totalRecordCount`	The total number of records that match the query criteria. This number may be larger than the currentRecordCount if not all of the matching certificates can be displayed on one page of output.
recordSet Variables	These fields are added to each `record` object in the `recordSet` array. They are accessed as fields of a `recordSet` element in the JavaScript; for example, `result.recordSet[1].error`.
`issuedBy`	The user ID of the agent that issued the certificate.
`issuedOn`	The date when the certificate was issued. Dates are represented as number of seconds since January 1, 1970. The default template provides a function, `renderDateFromSecs`, that converts these dates to a human-readable string.
`revocationReason`	If the certificate has been revoked, this field contains the code for the reason. The revocation codes are: • 0 - Reason not specified • 1 - Key compromised • 2 - CA key compromised • 3 - Affiliation changes • 4 - Certificate superseded • 5 - Cessation of operation • 6 - Certificate is on hold
`revokedBy`	The user ID of the agent who revoked the certificate.

Table 3-19 Variables Returned by the List Certificates Interface *(Continued)*

Variable	Description
`revokedOn`	The date when the certificate was revoked. See the description for issuedOn for details on date values.
`serialNumber`	The serial number of the certificate (in decimal).
`signatureAlgorithm`	The Object Identifier (OID) of the algorithm used to sign the certificate. For example, `"1.2.840.113549.1.1.4"` is the OID for an MD5 with RSA signature.
`subject`	The subject distinguished name of the certificate. For example, `"CN=Jane Doe, UID=jdoe, OU=Users, O=Organization, ST=California, C=US."`
`subjectPublicKeyAlgorithm`	The OID of the key algorithm used by the public key contained in the certificate. For example, `"1.2.840.113549.1.1.1"` represents an RSA key.
`subjectPublicKeyLength`	The number of bits of the public key contained in the certificate.
`validNotAfter`	The date when the certificate expires. See the description for `issuedOn` for details on date values.
`validNotBefore`	The date when the certificate became valid. See the description for `issuedOn` for details on date values.

Renewal Interface

Description

URI: `/renewal`

Available on: Certificate Manager or Registration Manager

Function: Processes requests for certificate renewal.

The Renewal interface allows an end entity to present a certificate and have it renewed. Only certificates that can be used for SSL client authentication can be renewed using the Renewal interface.

Default Forms

The only default form used by the Renewal interface is `UserRenewal.html`. This form allows a user to renew a certificate using SSL client authentication.

Request Parameters

The following table lists the parameters accepted by the Renewal interface.

Table 3-20 Parameters Accepted by the Renewal Interface

Parameter	Format and Description
`certType`	`client`
	Only client certificate renewals are supported through the Renewal interface.
`doSslAuth`	`on` | `off`
	Set to on to force the CMS server to request an SSL client authentication certificate. Since this is the only renewal method supported, `doSslAuth` should always be set to `on`.
`requestFormat`	`clientAuth`
	Only client certificate renewals are supported through the Renewal interface.
`templateName`	string
	Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.

Response

Responses from the Renewal interface are functionally equivalent to the Enrollment interface. Once the request has been submitted, the role of the Certificate Manager or Registration Manager is the same as if the request were for enrollment: the CMS server either rejects the request, queues it for manual processing, or issues a certificate.

The only difference in the response is for a successful request. The Renewal interface uses the `RenwalSuccess.template` file by default instead of `EnrollSuccess.template`. The difference between these two files (by default) is superficial: the word "Enrollment" is replaced with the word "Renewal." If you want to customize the renewal success message, customize the RenewalSuccess.template file.

Except for template for a successful request, the Renewal interface response is identical to the Enrollment interface response.

Refer to the Response section of the section "Enrollment Interface" on page 49 for complete details on the data returned and templates used.

Revocation Interface

Description

URI: `/revocation`

Available on: Certificate Manager or Registration Manager

Function: Allows automatic revocation of certificates by client authentication (an entity can revoke a certificate it presents).

The response is always a form that indicates the status of the revocation request (revoked, pending, or error), the result of updating the Certificate Revocation List, and the result of updating the certificate directory (if publishing is enabled).

Default Forms

The Revocation interface uses the UserRevocation.html form by default. This form posts requests that use SSL client authentication to present the certificate to be revoked. The certificate is automatically revoked.

Request Parameters

The following table lists the parameters accepted by the Revocation interface.

Table 3-21 Parameters Accepted by the Revocation Interface

Parameter	Format and Description
`certType`	`client` Specifies the type of certificate to revoke. For automatic revocation, the `certType` must be `client` and `doSslAuth` must be `on`.
`csrRequestorComments`	string Additional comments to assist the agent who will process the revocation request.
`csrRequestorEmail`	string Contact email address of someone responsible for the CMS server certificate. May be used to send out notification when a certificate has been revoked. Example: alice@example.com
`csrRequestorName`	string Name of someone responsible for the CMS server certificate; helps identify the requestor during manual revocation. Example: Alice Apple
`csrRequestorPhone`	string Phone number of someone responsible for the CMS server certificate. Example: 650.555.1212
`doSslAuth`	`on` | `off` Instructs the CMS server to request SSL client authentication. The certificate that the entity then presents will be the one that is automatically revoked. Only valid if `certType = client`.
`op`	`RevocationRequest` RevocationRequest is the only value currently supported for the `op` parameter. This parameter is required.

Table 3-21 Parameters Accepted by the Revocation Interface *(Continued)*

Parameter	Format and Description
`reasonCode`	`0-8` The `reasonCode` identifies the reason the certificate is being revoked. This information will be recorded on the Certificate Revocation List. The `reasonCode` is only valid for automatic revocation requests. Manual revocation requests can use the `csrRequestorComments` parameter to tell the processing agent why the certificate is being revoked. The meaning of the `reasonCode` values are: • `0` - Unspecified • `1` - Key Compromised • `2` - CA Compromise* • `3` - Affiliation Changed • `4` - Certificate Superseded • `5` - Cessation of Operation • `6` - Certificate Hold* • `7` - (Reserved for future use)* • `8` - Remove from CRL* Values marked with an asterisk (*) are valid `reasonCode` parameters that are not used in the default `UserRevocation.html` form. These values should generally not be used for client self-revocation.
`serialNumber`	string The serial number of the certificate to be revoked. This parameter is used for manual revocation: either a serial number or a subject name is used to identify the certificate to be revoked.
`subject`	string The subject distinguished name (DN) of the certificate to be revoked.
`templateName`	string Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.
`templateType`	`RevocationConfirmation` `RevocationConfirmation` is the only value currently supported. This parameter is required.

Response

The default response template is revocationResult.template. The base JavaScript for responses is inserted in place of the `<CMS_TEMPLATE>` tag. In addition, the Revocation interface adds the JavaScript variables listed in the following table:

Table 3-22 Variables Returned by the Revocation Interface

Variable	Description
Generic Variable	
`revokedCerts`	The number of certificates that were revoked as a result of the request.
result.header Variables	Variables added to the header object.
`certsToUpdate`	The number of certificates that need to be updated in the publishing directory as a result of the request. Used only when directory publishing is on, indicated by the `dirEnabled` field in the header object.
`certsUpdated`	A number, less than or equal to `certsToUpdate`, indicating how many certificates have already been successfully updated in the publishing directory. Used only when directory publishing is on, indicated by the dirEnabled field in the header object.
`dirEnabled`	`yes` | `no` Indicates whether directory publishing is enabled on the Certificate Manager where the revocation request was handled. May be `null` if directory publishing is not defined.
`error`	A text message indicating why the revocation request itself could not be processed. The `result.header.error` message will exist only when `result.header.revoked = no`.
`revoked`	`yes` | `pending` | `no` This field indicates the overall status of the revocation request. If the certificate could be revoked automatically, revoked will be `yes`. If the request was processed successfully, but requires manual processing by an agent, `revoked = pending`. If `revoked = no`, the request could not be processed. See `result.header.error` for the error message.
`totalRecordCount`	The total record count indicates the number of requests that were successfully processed. This number may be different from the number of certificates actually revoked. The `recordSet` array contains information for each processed revocation request (including requests that failed due to an error). The `recordSet.length` may be less than `totalRecordCount` if any certificates were already revoked.

Table 3-22 Variables Returned by the Revocation Interface *(Continued)*

Variable	Description
`updateCRL`	`yes` | `no`
	If present and equal to `yes`, this field indicates that the Certificate Manager has attempted to update the Certificate Revocation List (CRL). Check `updateCRLSuccess` to see if the update was successful. If this field is `null` or equal to `no`, the Certificate Manger will attempt to update the CRL at the next scheduled update interval.
`updateCRLError`	This text message indicates the reason why an attempt to update the CRL has failed. This will be present if `updateCRL = yes` and `updateCRLSuccess = no` (or is `null`).
`updateCRLSuccess`	`yes` | `no`
	If `updateCRL = yes`, this field indicates whether the attempt to update the CRL has succeeded and the certificate now appears as revoked on the CRL. If this field is null or equal to `no`, see `updateCRLError`.
recordSet Variables	These fields are added to each record object in the recordSet array. They are accessed as fields of a recordSet element in the JavaScript; for example, `result.recordSet[1].error`.
`error`	This text message indicates the reason that the certificate associated with this `recordSet` was processed, but could not be revoked.
`serialNumber`	Contains the serial number of the certificate represented by this `recordSet` object.

Chapter 4

Internationalization of End-Entity Interface

The services interfaces that come with iPlanet Certificate Management Server (CMS) make it possible for end-entities and agents to interact with the server. Your end-entities and agents can use the interface's HTML-based forms to carry out various certificate and key-related operations, such as enrolling for, renewing, and revoking certificates.

You can use the default forms as they are, customize them, or develop your own forms to suit your organization's policies or terminology. This chapter explains how to customize the forms and templates used by the interfaces.

The chapter has the following sections:

- Displaying Forms in Non-English Languages (page 87)

Displaying Forms in Non-English Languages

The forms and response templates that come with Certificate Management System are all in English. Certificate Management System supports forms and templates in other languages, and multiple languages can be supported on the same CMS server instance. Every aspect of the CMS server is designed to accomodate multiple languages, including all storage and certificate processing (it is possible to have certificate subject names with data in Chinese, for example). The CMS administration console windows support data in non-English languages, but the messages and menu items cannot be localized.

When an HTTP or HTTPS request arrives at the CMS server, the CMS server checks the HTTP `Accept-language` header to see what languages are preferred by the requestor. For example, a client that prefers content in Korean would have the value "ko" in the `Accept-language` header. The server looks in the directory where the default form would be stored to see if there is a directory matching the first value in the `Accept-language` header. If there is such a directory, the CMS server looks for the correct form or template in the language-specific directory; if the form or template is not found, the default is still used.

For example, the manual user enrollment form is `ManUserEnroll.html`. It is stored in the `web/ee/` directory below the CMS server root. If you wanted to provide a version of this form in French and German for your users, you would translate the form, create the directories `web/ee/fr` (French) and `web/ee/de` (German), and put the translated versions of the form in the appropriate, language-specific subdirectory. The appropriate form is sent to users automatically based on the language preferences set in their browsers.

Localized versions of the agent forms and templates are supported in the same way. Create language-specific subdirectories of `web/agent/ca`, `web/agent/kra`, and `web/agent/ra` to provide forms and templates for agents in non-English languages.

Note that if a browser sends more than one language, the CMS server will try to match one of the browser’s language preferences with the default locale of the system where the server is running. If no match is found, the default page in English will be returned to the browser. Users having trouble accessing your localized content should make sure they have only one language set in their browsers.

Certificate Management System uses a default character set for each language (see Table 4-1). If you want to use a different character set for a language, you must edit the CMS server configuration file CMS`.cfg` and add a line with the following format:

```
i18nCharset.<lang>=<charset>
```

Where `<lang>` is the two-letter code for the language (the same as the directory where the localized files are stored) and `<charset>` is the character set to use with files in that language. For example, to use a character set named `EUC_KR` for Korean-language (ko) content, add the following line to `CMS.cfg`:

```
i18nCharset.ko=EUC_KR
```

The following table lists the languages supported by Certificate Management System, the two-letter language code to use for language-specific directories, and the default character set Certificate Management System uses for the language:

Table 4-1 Languages and Default Character Sets

Language	Code	Character set	Language	Code	Character set
Albanian	`sq`	ISO-8859-2	Arabic	`ar`	ISO-8859-6
Bulgarian	`bg`	ISO-8859-5	Byelorussian	`be`	ISO-8859-5
Catalan (Spanish)	`ca`	ISO-8859-1	Chinese (Simplified / Mainland)	`zh`	GB2312
Chinese (Traditional / Taiwan)	`zh`	Big5	Croatian	`hr`	ISO-8859-2
Czech	`cs`	ISO-8859-2	Danish	`da`	ISO-8859-1
Dutch	`nl`	ISO-8859-1	English	`en`	ISO-8859-1
Estonian	`et`	ISO-8859-1	Finnish	`fi`	ISO-8859-1
French	`fr`	ISO-8859-1	German	`de`	ISO-8859-1
Greek	`el`	ISO-8859-7	Hebrew	`he`	ISO-8859-8
Hungarian	`hu`	ISO-8859-2	Icelandic	`is`	ISO-8859-1
Italian	`it`	ISO-8859-1	Japanese	`ja`	Shift_JIS
Korean	`ko`	KSC_5601	Latvian (Lettish)	`lv`	ISO-8859-2
Lithuanian	`lt`	ISO-8859-2	Macedonian	`mk`	ISO-8859-5
Norwegian	`no`	ISO-8859-1	Polish	`pl`	ISO-8859-2
Portuguese	`pt`	ISO-8859-1	Romanian	`ro`	ISO-8859-2
Russian	`ru`	ISO-8859-5	Serbian	`sr`	ISO-8859-5
Serbo-Croatian	`sh`	ISO-8859-5	Slovak	`sk`	ISO-8859-2
Slovenian	`sl`	ISO-8859-2	Spanish	`es`	ISO-8859-1
Swedish	`sv`	ISO-8859-1	Turkish	`tr`	ISO-8859-9
Ukranian	`uk`	ISO-8859-5			

Part 2

Customizing Agent Services Interface

Chapter 5, “Introduction to Agent Services Interface”

Chapter 6, “Agent Interface Reference”

Chapter 5

Introduction to Agent Services Interface

iPlanet Certificate Management Server (CMS) provides HTML forms-based interfaces for agents to use in performing certificate- and key-related operations. This chapter introduces these forms and explains how they work. You can use the forms as they are provided out of the box or customize them to meet your organization's requirements.

This chapter has the following sections:

- Agent Services Interface (page 93)
- Accessing the Agent Services Interface (page 97)
- Agent Forms and Templates (page 97)

Agent Services Interface

As an administrator, you can designate privileged users, called agents, for each subsystem. Agents are responsible for the day-to-day operation of requests from end entities. To enable agents to accomplish their duties, Certificate Management System provides a set of HTML forms for Certificate Manager, Registration Manager, and Data Recovery Manager agents. Collectively, these forms are called the *Agent Services* interface.

Depending on the choices you made during installation, a combination of the following agent services will be installed:

- Certificate Manager Agent Services
- Registration Manager Agent Services
- Data Recovery Manager Agent Services

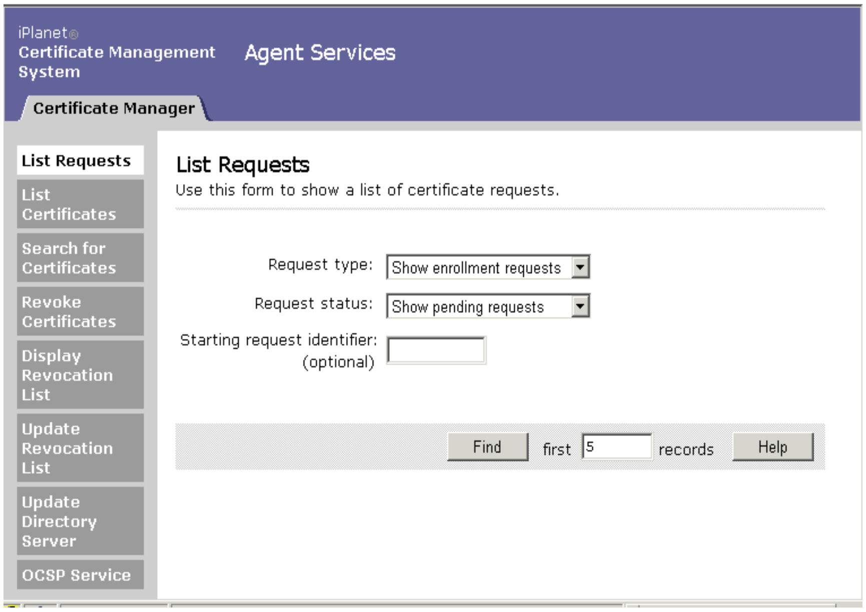
This section gives an overview of these forms and explains how to access them. For a complete list of the agent forms and output templates that come with Certificate Management System, see “Agent Forms and Templates” on page 97. For step-by-step instructions on using the agent forms, see *CMS Agent’s Guide*. For information on locating this guide, see “Where to Go for Related Information” on page 12.

Note that accessing the Agent Services interface is a privileged operation, requiring certificate-based (or *strong*) authentication. It can be done only by users belonging to authorized agent groups maintained by Certificate Management System in its internal database. For details, see section “Agents” in Chapter 13, “Managing Privileged Users and Groups” of *CMS Installation and Setup Guide*.

Certificate Manager Agent Services

The Certificate Manager Agent Services interface enables a Certificate Manager agent to interact with the Certificate Manager (the server). Figure 5-1 shows the Certificate Manager Agent Services interface.

Figure 5-1 Certificate Manager Agent Services interface



List Requests

Use this form to show a list of certificate requests.

Request type: Show enrollment requests

Request status: Show pending requests

Starting request identifier: (optional)

first 5 records

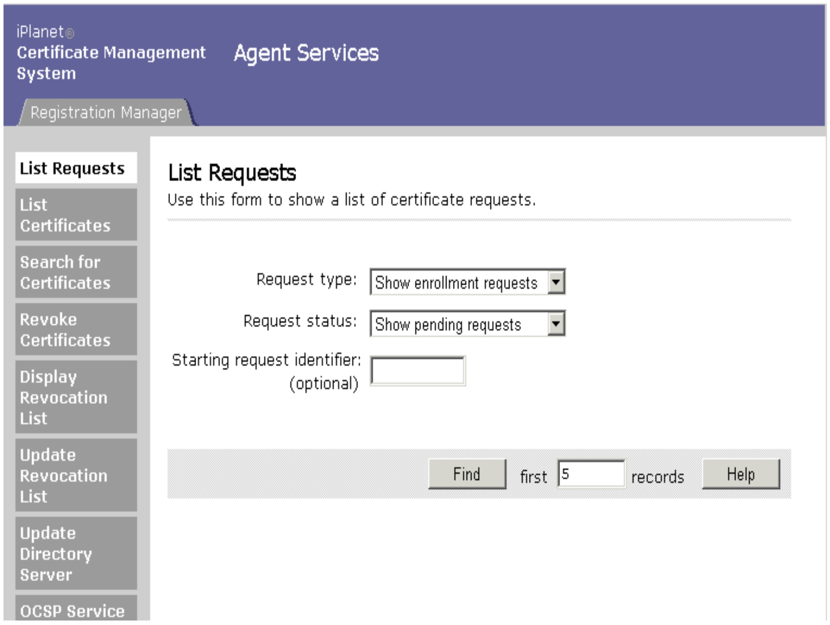
Using the default forms, a Certificate Manager agent can accomplish tasks such as these:

- Listing *deferred* certificate requests from end entities and process them
- Listing certificates issued by the server
- Searching for certificates issued by the server
- Revoking certificates issued by the server
- Updating certificates and certificate revocation lists (CRLs) maintained in the publishing directory

Registration Manager Agent Services

The Registration Manager Agent Services interface enables a Registration Manager agent to interact with the Registration Manager (the server). Figure 5-2 shows the Registration Manager Agent Services interface.

Figure 5-2 Registration Manager Agent Services interface



List Requests

Use this form to show a list of certificate requests.

Request type: Show enrollment requests

Request status: Show pending requests

Starting request identifier: (optional)

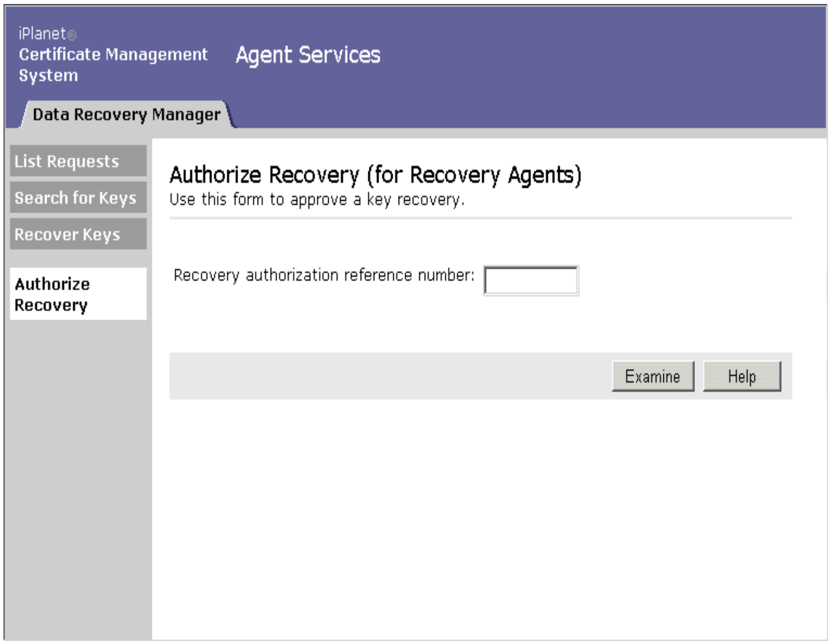
first 5 records

Using the default forms, a Registration Manager agent can list *deferred* certificate requests from end entities and process them.

Data Recovery Manager Agent Services

The Data Recovery Manager Agent Services interface enables a Data Recovery Manager agent to interact with the Data Recovery Manager (the server). Figure 5-3 shows the Data Recovery Manager Agent Services interface.

Figure 5-3 Data Recovery Manager Agent Services interface



Authorize Recovery (for Recovery Agents)
Use this form to approve a key recovery.

Recovery authorization reference number:

Using the default forms, a Data Recovery Manager agent can search for and recover end users' encryption private keys from the key archive. Key recovery requires authorization from key recovery agents; see section "Key Recovery Process" in Chapter 13, "Managing Privileged Users and Groups" of *CMS Installation and Setup Guide.*

Accessing the Agent Services Interface

Access to the Agent Services interface is restricted to authorized agents only. To access the Agent Services interface for a particular subsystem:

1. Open a web browser.
2. Go to the page where the Agent Services interface for Certificate Management System is installed.

   The default URL for this page is: `https://<hostname>:<agent_port>`

   `<hostname>` is in the form: `<machine_name>.<your_domain>.<domain>`

   If you have customized Certificate Management System, go to the page containing the agent forms that you would use to submit a request.
3. In the Agent Services menu, choose the agent services you require:
   - To access the agent services for the Certificate Manager, click the Certificate Manager Agent Services link.
   - To access the agent services for the Registration Manager, click the Registration Manager Agent Services link.
   - To access the agent services for the Data Recovery Manager, click the Data Recovery Manager Agent Services link.

   The appropriate interface appears.

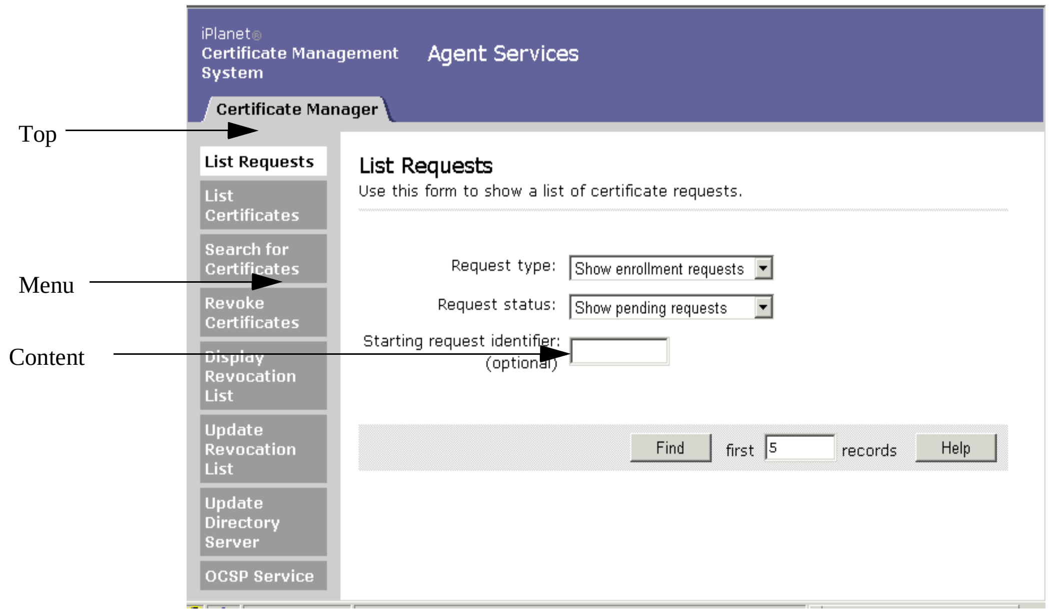
Agent Forms and Templates

This section describes the Agent Services interface, gives the location of the agent forms and output templates, and lists all of the default forms and templates.

Structure of the Agent Services Interface

As shown in Figure 5-4, the Agent Services interface is divided into three parts or frames—top, menu, and content. The top frame includes the tabs that allow you to select a subsystem. The menu lists all the operations supported by the selected subsystem. The content shows the form pertaining to the operation an agent chooses in the menu; the form contains information to carry out the selected operation.

Figure 5-4 Various parts of the Agent Services interface



List Requests

Use this form to show a list of certificate requests.

Request type: Show enrollment requests

Request status: Show pending requests

Starting request identifier: (optional)

first 5 records

Locating Agent Forms and Templates

You can find the HTML forms specific to agent operations and the corresponding output templates at this location:

`<server_root>/cert-<instance_id>/web/agent/<subsystem>`

> `<server_root>` is the directory where the CMS binaries are kept, as specified during installation.
>
> `<instance_id>` is the ID for this instance of Certificate Management System. You specified this ID during installation.
>
> `<subsystem>` refers to the forms directory pertaining to a subsystem, the Certificate Manager (`ca`), Registration Manager (`ra`), or Data Recovery Manager (`kra`).

Chapter 6

Agent Interface Reference

This chapter provides a detailed reference of all the service interfaces available on an agent port of iPlanet Certificate Management Server. For each interface, there is a description including the URI used, the purpose, and which agents can use it, a list of forms that use the interface by default, a detailed description of valid input parameters and their values, and information about the response which lists the templates used and the additional JavaScript variables available.

The chapter has the following sections:

- Overview of Agent Interfaces (page 100)
- Approve Revocation Interface (page 101)
- Bulk Enrollment Interface (page 105)
- Display Key By Serial Number Interface (page 115)
- Display Key For Recovery Interface (page 118)
- Examine Recovery Interface (page 120)
- Get Approval Status Interface (page 122)
- Get PKCS #12 Data Interface (page 125)
- Grant Recovery Interface (page 126)
- Key Query Interface (page 128)
- Key Recovery Query Interface (page 133)
- Process Certificate Request Interface (page 138)
- Process DRM Request Interface (page 147)
- Process Request Interface (page 150)
- Recover Key By Serial Number Interface (page 153)

- Remove Certificate Hold Interface (page 156)
- Requests Query Interface (page 159)
- Select for Revocation Interface (page 164)
- Update CRL Interface (page 167)
- Update Directory Interface (page 168)

Overview of Agent Interfaces

The following table lists the agent interfaces and their functions:

Table 6-1 Agent Interfaces

Interface	**URI**	**Purpose**
Approve Revocation Interface	`/ca/doRevoke` `/ra/doRevoke`	Use the id assigned to a pending request to retrieve the certificate once it has been issued.
Bulk Enrollment Interface	`/ca/bulkissuance`	Allows a process to programatically connect to the interface using SSL client authentication (with a CMS agent certificate) to submit a request and receive a certificate in response.
Display Key By Serial Number Interface	`/kra/displayBySerial`	Display information about an archived key.
Display Key For Recovery Interface	`/kra/displayBySerialForRecovery`	Display a form for recovering a key.
Examine Recovery Interface	`/kra/examineRecovery`	Check to see if a recovery request id is valid.
Get Approval Status Interface	`/kra/getApprovalStatus`	Display the status of a key recovery operation.
Get PKCS #12 Data Interface	`/kra/getPk12`	Retrieve the PKCS #12 data containing a recovered key.
Grant Recovery Interface	`/kra/grantRecovery`	Submit an agent password to approve a key recovery.
Key Query Interface	`/kra/queryKey`	View archived keys that meet query criteria.

Table 6-1 Agent Interfaces *(Continued)*

Interface	URI	Purpose
Key Recovery Query Interface	`/kra/queryKeyForRecovery`	Display archived keys that meet query criteria and get links to initiate recovery of these keys.
Process Certificate Request Interface	`/ca/processCertReq` `/ra/processCertReq`	Allows agents to accept or reject requests for enrollment, renewal, or revocation.
Process DRM Request Interface	`/kra/processReq`	Allows key recovery agents to view requests and change request assignments.
Process Request Interface	`/ca/processReq` `/ra/processReq`	Allows agents to view pending requests and assign them to themselves.
Recover Key By Serial Number Interface	`/kra/recoverBySerial`	Given an archive serial number, display a form for recovering the archived key.
Remove Certificate Hold Interface	`/ca/doUnrevoke` `/ra/doUnrevoke`	Remove the "on hold" revocation status of a certificate.
Requests Query Interface	`/ca/queryReq` `/ra/queryReq` `/kra/queryReq`	View requests that match certain criteria (such as request type or status).
Select for Revocation Interface	`/ca/reasonToRevoke` `/ra/reasonToRevoke`	Revoke a set of certificates for a given reason.
Update CRL Interface	`/ca/updateCRL` `/ra/updateCRL`	Force the Certificate Revocation List (CRL) to be updated before the scheduled update.
Update Directory Interface	`/ca/updateDir` `/ra/updateDir`	Force the publishing directory to be updated before the next scheduled update.

Approve Revocation Interface

Description

URI: `/ca/doRevoke` or `/ra/doRevoke`

Available on: Certificate Manager or Registration Manager

Function: Actually revokes a certificate or group of certificates for a given reason.

The Select for Revocation Interface is used to select a certificate or group of certificates for revocation based on some criteria. That interface returns a list of certificates using the `reasonToRevoke.template` file. The `reasonToRevoke.template` response contains a form that posts data to the Approve Revocation interface with the serial numbers and reasons to finally revoke the certificates.

Default Forms

The Approve Revocation interface is accessed through the `reasonToRevoke.template` file by default. No forms directly post data to this interface. Other forms, such as `revokeCert.html`, are used to select certificates based on query criteria; the selected certificates can then be marked for revocation by creating buttons or links that post data to the Approve Revocation interface.

Request Parameters

The following table lists the parameters that are used to revoke certificates through the Approve Revocation interface. This is an agent interface, so the HTTP POST or GET request must use SSL client authentication with a valid agent certificate.

Table 6-2 Parameters Accepted by the Approve Revocation Interface

Parameter	Format and Description
`b64eCertificate`	base-64 encoded certificate data Allows you to specify the certificate to revoke by posting its base-64 encoding to the interface.
`csrRequestorComments`	string A comment field to provide more details about why the certificates are being revoked.
`invalidityDate`	number of seconds since 1 January 1970 The time when the certificates became invalid.
`op`	`doRevoke` The only operation supported is `doRevoke`.
`requestId`	number Specifies the enrollment request id corresponding to the certificate to revoke. This allows you to post revocations to a Registration Authority.

Table 6-2 Parameters Accepted by the Approve Revocation Interface *(Continued)*

Parameter	Format and Description
`revocationReason`	`0`-`6` The code for the reason the certificates are being revoked. The revocation codes are: • `0` - Reason not specified • `1` - Key compromised • `2` - CA key compromised • `3` - Affiliation changes • `4` - Certificate superseded • `5` - Cessation of operation • `6` - Certificate is on hold
`revokeAll`	QUERY_FILTER For information on constructing a query filter, see Table 3-17 in the section for “List Certificates Interface” on page 72. To ensure accuracy when revoking certificates, you should use a query filter that selects each certificate by its serial number. An example value for `revokeAll` to revoke certificates with serial numbers 10 and 14 is: `(|(certRecordId=10)(certRecordId=14))`
`serialNumber`	number Specifies the serial number of a certificate to revoke.
`templateName`	string Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.
`templateType`	string The name of the template to use for the response. The `reasonToRevoke.template` file sets this to `RevocationSuccess`.
`totalRecordCount`	number The total number of certificates selected by the `revokeAll` filter. This number can be determined from the response variables in a template used to select certificates for revocation.

Table 6-2 Parameters Accepted by the Approve Revocation Interface *(Continued)*

Parameter	Format and Description
`verifiedRecordCount`	number Not presently used by the interface.

Response

The default response template is `revocationResult.template`. The base JavaScript for responses is inserted in place of the `<CMS_TEMPLATE>` tag. In addition, the Approve Revocation interface adds the JavaScript variables listed in the following table.

Table 6-3 Variables Returned by the Approve Revocation Interface

Variable	Description
result.header variables	Variables added to the header object.
`certsUpdated`	number Contains the number of certificates that were revoked from the publishing directory, if publishing is enabled (`dirEnabled = yes`).
`dirEnabled`	`yes` | `no` Indicates whether LDAP publishing is enabled on the Certificate Manager that handled the request.
`error`	message If there was an error while processing the revocation request, the error message is stored in this variable. Otherwise, the value is `null`.
`revoked`	`yes` | `pending` Indicates whether or not all certificates were successfully revoked.
`totalRecordCount`	number Indicates the total number of revocation requests that were processed as a result of the request.
`updateCRL`	`yes` | `no` Indicates whether or not the CMS server attempted to update the Certificate Revocation List (CRL). If `no` or `null`, the CRL will be updated at the next scheduled update interval. If `yes`, check `udpateCRLSuccess` to determine if the update was successful.

Table 6-3 Variables Returned by the Approve Revocation Interface *(Continued)*

Variable	Description
updateCRLError	message
	If the CMS server attempted to update the CRL and encountered an error, this variable contains the text of the error message.
updateCRLSuccess	yes | no
	If the CMS server attempted to update the CRL, this variable will indicate whether the update was successful.
result.recordSet[i] variables	Variables added to record objects in the response.
error	message
	If a particular certificate could not be revoked, the error field in its record object will contain an error message. If this field is null, the certificate was revoked successfully.
serialNumber	number
	The decimal serial number of the certificate.

Bulk Enrollment Interface

Description

URI: /ca/bulkissuance

Available on: Certificate Manager only

Function: The Bulk Enrollment interface allows a connection using SSL client authentication with a valid agent certificate to have a certificate issued on behalf of another entity. The entire process is automated so that a device or application with an agent certificate, the ability to do SSL client authentication, and the ability to parse and store the certificate in the response can programmatically request and receive certificates.

An application or hardware device that can also generate keys could use the Bulk Enrollment interface to generate keys, request a certificate for the public key, receive the certificate and store it (for example on a smart card to be distributed to a user).

The reply from the Bulk Enrollment interface can be just the certificate chain (in PKCS #7format), or it can be an HTML page.

Configuration Parameters

The Bulk Enrollment interface can be configured with parameters in the `CMS.cfg` configuration file.

The interface can be enabled or disabled using the `agentGateway.enableBulkInterface` parameter. The rest of the parameters configure which template file to use when a response has a given `requestStatus` code. The template configuration parameter names should be prefixed with `agentGateway.bulkissuance`. For example, `agentGateway.bulkissusance.successTemplate=/ca/bulkissuance.template`.

The following table lists the valid configuration file parameters and what they control. The file names for the template parameters are relative to the `<server_root>/cert-<instance_id>/web/agent` directory.

Table 6-4 Bulk Enrollment Interface Configuration File Parameters

Parameter	Format and Description
`agentGateway.enableBulkInterface`	`true` | `false` Enables or disables the servlet handling bulk enrollment at `/ca/bulkissuance`.
`errorTemplate`	filename The template file to use when the response `requestStatus = 6`, meaning an error occurred while processing the request.
`pendingTemplate`	filename The template file to use when the response `requestStatus = 3`, meaning the request has been deferred for manual approval.
`rejectedTemplate`	filename The template file to use when the response `requestStatus = 5`, meaning the request was rejected by a policy on the CMS server.
`successTemplate`	filename The template file to use when the response `requestStatus = 2`, meaning the certificate has been issued.

Table 6-4 Bulk Enrollment Interface Configuration File Parameters *(Continued)*

Parameter	Format and Description
`svcpendingTemplate`	filename
	The template file to use when the response `requestStatus = 4`, meaning the request is pending a response from a Data Recovery Manager.
`unauthorizedTemplate`	filename
	The template file to use when the response `requestStatus = 1`, meaning the SSL client authentication certificate presented to authenticate the request is not a valid agent certificate.
`unexpectedErrorTemplate`	filename
	The template file to use when the response `requestStatus = 7`, meaning an unexpected error prevented the CMS server from processing the response.

Default Forms

No default forms use the Bulk Enrollment interface. The intent of the interface is to provide a programmatic, rather than interactive, method for enrolling entities into the PKI.

Request Parameters

The following table lists the parameters accepted by the Bulk Enrollment interface.

Note that the Bulk Enrollment interface requires SSL Client authentication with an agent certificate authorized to approve certificate requests.

Table 6-5 Parameters Accepted by the Bulk Enrollment Interface

Parameter	Format and Description
Subject Name	
`subject`	Distinguished Name (DN) string. See RFC 2253.
	DN to be used for the certificate subject. Example: `CN=Alice Apple, UID=alice, OU=People, O=Example, C=US`
Contact Information	

Table 6-5 Parameters Accepted by the Bulk Enrollment Interface *(Continued)*

Parameter	Format and Description
`csrRequestorName`	string Name of the entity making a request; helps identify the requestor during manual enrollment. Example: Alice Apple
`csrRequestorEmail`	string Email address of the entity making a request. May be used to send out notification when a certificate has been issued. Example: alice@example.com
`csrRequestorPhone`	string Phone number of the entity making a request. Example: 650.555.1212
`csrRequestorComments`	string Additional comments provide by the requestor on the HTML form. This field can be used if there is additional information you want to collect to help the manual enrollment.
Netscape Certificate Type Extensions	Parameters for setting bits in the netscape-cert-type certificate extension. See http://home.netscape.com/eng/security/comm4-cert-exts.html for details. A true value sets the bit to 1; false sets the bit to 0.
`email`	`true` | `false` Sets the S/MIME client certificate bit (bit 2).
`email_ca`	`true` | `false` Sets the S/MIME certificate issuer bit (bit 6).
`object_signing`	`true` | `false` Sets the object signing certificate bit (bit 3).
`object_signing_ca`	`true` | `false` Sets the object signing certificate issuer bit (bit 7).
`ssl_ca`	`true` | `false` Sets the SSL certificate issuer bit (bit 5).
`ssl_client`	`true` | `false` Sets the SSL client authentication certificate bit (bit 0).

Table 6-5 Parameters Accepted by the Bulk Enrollment Interface *(Continued)*

Parameter	Format and Description
`ssl_server`	`true` | `false` Sets the SSL server authentication certificate bit (bit 1).
Key Usage	Parameters for setting bits in the `keyUsage` certificate extension. A true value sets the bit to 1; false sets the bit to 0.
`crl_sign`	`true` | `false` Sets the `keyUsage` extension bit (6) indicating that the key may be used to sign Certificate Revocation Lists (CRLs).
`data_encipherment`	`true` | `false` Sets the `keyUsage` extension bit (3) indicating that the key may be used to encipher application data (as opposed to key material).
`decipher_only`	`true` | `false` Sets the `keyUsage` extension bit (8) indicating that the key may **only** be used to decipher data and keys. If this is true, keyAgreement should also be true.
`digital_signature`	`true` | `false` Sets the `keyUsage` extension bit (0) indicating that the key may be used to sign any data. This should be true for SSL client certificates, S/MIME signing certificates, and object signing certificates.
`encipher_only`	`true` | `false` Sets the `keyUsage` extension bit (7) indicating that the key may **only** be used to encipher data and keys. If this is true, keyAgreement should also be true.
`key_agreement`	`true` | `false` Sets the `keyUsage` extension bit (4) indicating that the key may be used to encipher and decipher keys during key agreement.
`key_certsign`	`true` | `false` Sets the `keyUsage` extension bit (5) indicating that the key may be used to sign other certificates. All CA signing certificates should set this to true.
`key_encipherment`	`true` | `false` Sets the `keyUsage` extension bit (2) indicating that the key may be used to encipher symmetric session keys. This should be true for SSL server and S/MIME encryption certificates.

Table 6-5 Parameters Accepted by the Bulk Enrollment Interface *(Continued)*

Parameter	Format and Description
`non_repudiation`	`true` | `false`
	Sets the keyUsage extension bit (1) indicating that the key may be used to create non-repudiable (by the signer) digital signatures. Non-repudiation service requires more infrastructure, planning, and policy than just setting this bit. Consider the ramifications before using this bit
Automated Enrollment	Parameters to configure automatic authentication for entity requests.
`authenticator`	string
	Specifies the name of the authentication plug-in instance to use to authenticate the entity.
`uid`	string
	Specifies a unique identifier passed to the authentication plug-in.
`pin`	string
	An optional identifying string that helps to authenticate an entity. Usually used when the Pin Generator tool has been used to populate a directory with unique identifiers for each user.
`pwd`	string
	Specifies the password passed to the authentication plug-in.
Other	
`certNickname`	string
	Specifies the nickname that should be associated with the certificate in the reply; used with Certificate Request Management Format (CRMF) requests.
`certType`	`ca` | `CEP-Request` | `client` | `objSignClient` | `ra` | `server` | `other`
	Specifies the type of certificate requested by the entity. The default is `client`. The `certType` is not associated with any certificate extensions. It may be used by policy modules to make decisions, and it may be used by a CMS server to determine how to decode the request or format the response.
`challengePassword`	string
	An optional challenge phrase or password that can be used later by the entity to revoke the certificate. This parameter is optional. If you use this, entities can use the Challenge Revocation Interface (`/challenge_revocation1`, page 42) with this challenge password to revoke a certificate without manual intervention and without SSL client authentication.

Table 6-5 Parameters Accepted by the Bulk Enrollment Interface *(Continued)*

Parameter	Format and Description
`CRMFRequest`	base-64 encoded data If `requestFormat = crmf`, this parameter should be used to send the base-64 encoded CRMF request.
`importCAChain`	`true` | `false` Used only when `importCert = true`. The default, if this parameter is not explicitly passed, is `true`. If set to `true`, a successful certificate request will return a PKCS #7 formatted certificate chain; if set to false, a single, DER-encoded certificate will be returned. The certificate chain includes the issued certificate and the CA (issuer) certificate.
`importCert`	`true` | `false` If `true`, and the certificate request is not deferred or rejected, the CMS server's response will be binary data with the MIME type determined by the `importCertMimeType` parameter. The data returned will be either a certificate or a certificate chain, based on the value of `importCAChain`.
`importCertMimeType`	MIME Type string Sets the MIME type the CMS server uses when a certificate is returned to the requestor. The default is `application/x-x509-user-cert`. The MIME type should be in the standard MIME type format of `<type>/<subtype>`.
`pkcs10Request`	base-64 encoded data If `requestFormat = pkcs10`, this parameter should be used to send the base-64 encoded certificate request.
`requestFormat`	`clientAuth` | `crmf` | `keygen` | `pkcs10` The value indicates the format used to submit the certificate request: • `clientAuth` - information for the new request is taken from the certificate presented by the client during SSL client authentication. • `crmf` - the certificate request is a base-64 encoded blob contained in the CRMFRequest parameter. • `keygen` - the certificate request is a base-64 encoded blob generated using the HTML `<KEYGEN>` tag. It is contained in the `subjectKeyGenInfo` parameter. • `pkcs10` - the certificate request is a base-64 encoded blob contained in the `pkcs10Request` parameter.

Table 6-5 Parameters Accepted by the Bulk Enrollment Interface *(Continued)*

Parameter	Format and Description
`subjectKeyGenInfo`	base-64 encoded data
	If `requestFormat=keygen`, this parameter should be used to send the base-64 encoded keygen request. To use the keygen HTML tag to cause the browser to generate the request using this parameter, the format is
	`<KEYGEN name="subjectKeyGenInfo">`
`templateName`	string
	Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.

Response

If the request parameter `importCert` is set to `true` and the certificate request is successful, the Certificate Manager will return the binary PKCS #7 certificate chain using the MIME type `application/x-x509-user-cert`. This is the most useful application of the Bulk Enrollment interface.

If `importCert` is not set to `true`, or if there is an error, the default response template is `bulkissuance.template`. Applications using the Bulk Enrollment interface should be prepared to handle the HTML output created using the template when errors occur.

The base JavaScript for responses is inserted in place of the `<CMS_TEMPLATE>` tag. In addition, the Bulk Enrollment interface adds the JavaScript variables listed in the following table.

Table 6-6 Variables Returned by the Bulk Enrollment Interface

Variable	Description
result.fixed variables	Variables added to the `fixed` object.
`authorityName`	`Certificate Manager` | `Registration Manager`
	The name of the system that handled the request.
`certType`	`ca` | `CEP-Request` |`client` | `objSignClient` |`ra` | `server` | `other`
	The type of certificate returned. This value is the same as the `certType` value passed to the interface in the request.

Table 6-6 Variables Returned by the Bulk Enrollment Interface *(Continued)*

Variable	Description
`errorDescription`	string A message providing more details about the error described in `errorDetails`. This variable is only present if an error occurred while processing the request.
`errorDetails`	string A message explaining the error that occurred while processing the enrollment request. This variable is only present if an error occurred while processing the request.
`host`	string The fully qualified domain name of the CMS server that processed the request. This allows the resulting template to construct forms that post data to the same interface using the same port.
`port`	number The port number that was used to service the request.
`requestId`	number A unique number assigned by the CMS server to this request. This is especially useful for pending requests since there is no unique certificate serial number yet assigned.

Table 6-6 Variables Returned by the Bulk Enrollment Interface *(Continued)*

Variable	Description
`requestStatus`	number A code indicating the current status of the request: • 1 (Unauthorized): The request specified a value for an authenticator to perform an automated enrollment, and the authenticator did not authorize the request. • 2 (Success): Processing the request was successful and a certificate has been issued. If `importCert` was set to `true`, the response will include code (from the `EnrollSuccess.template`) to import the certificate into the browser making the request. Otherwise, the response is only a success message. • 3 (Pending): The request has been successfully processed by the CMS server and added to a queue for approval by an agent. If the request has been submitted to another Certificate Manager or Data Recovery Manager and is currently pending in the queue for that service, the response template will be `GenSvcPending.template` instead of `GenPending.template`. • 4 (Reserved): Not currently used. • 5 (Rejected): The request was rejected during policy processing. • 6 (Error): An error occurred when the CMS server processed the request. The error may be the result of missing or improperly formatted parameters. • 7 (Exception): An unknown or unexpected error occurred when the CMS server processed the request.
`scheme`	`http` | `https` The protocol that was used to make the request. Use this along with host and port to make sure any new requests to the end-entity port use the correct scheme.
`unexpectedError`	string A message explaining the exception or unexpected error that occurred.
result.recordSet[i] variables	Variables added to each `record` object. Each `record` object is added as an element of the `recordSet` array. Multiple records may be returned if more than one certificate was generated as a result of the request. Dual-key requests (for example, if the request parameter `requestFormat = crmf`) may return two certificates if the request is successfully processed and approved.

Table 6-6 Variables Returned by the Bulk Enrollment Interface *(Continued)*

Variable	Description
`base64Cert`	string The newly issued certificate in base-64 encoded format. This string includes the "`-----BEGIN CERTIFICATE-----`" header and "`-----END CERTIFICATE-----`" footer.
`certFingerprint`	string A string of hexadecimal numbers separated by colons that represent the certificate fingerprints. There are three substrings: one each for the MD2, MD5, and SHA1 fingerprint. Each fingerprint begins with the hash algorithm name and a colon, and ends with a newline (\n).
`certPrettyPrint`	string A long text string that shows all of the certificate data in a human readable form.
`policyMessage`	string If the request was rejected by policy processing on the CMS server, this variable will contain a message explaining why.
`serialNo`	number The serial number (in decimal) of the newly issued certificate.

Display Key By Serial Number Interface

Description

URI: `/kra/displayBySerial`

Available on: Data Recovery Manager

Function: Displays information in human-readable form about a single archived key.

The Display Key By Serial Number interface is typically used within a form that lists keys to display detailed information about a selected key.

Default Forms

The Display Key By Serial Number interface is used in the `queryKey.template` file. Each key in the list of keys satisfying the query has a button the user can press to see the key in detail. This button submits data to the Display Key By Serial Number interface.

Request Parameters

The following table lists the parameters accepted by the Display Key By Serial Number interface.

Table 6-7 Parameters Accepted by the Display Key By Serial Number Interface

Parameter	Format and Description
`op`	`displayBySerial`
	Specifies the operation to perform. The only valid value is `displayBySerial`.
`serialNumber`	number
	The serial number of the key to display. Note that this is the DRM serial number, not the serial number from a certificate.
`templateName`	string
	Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.

Response

The default response template is `displayBySerial.template` (note that the template in the `web/agents/kra` directory differs significantly from the template for Certificate Managers, Registration Managers, or end entities; those forms are used to display certificates, not keys).

The base JavaScript for responses is inserted in place of the `<CMS_TEMPLATE>` tag. In addition, the Display Key By Serial Number interface adds the JavaScript variables listed in the following table:

Table 6-8 Variables Returned by the Display Key By Serial Number Interface

Variable	Description
result.header variables	Variables added to the header object.
`archivedBy`	user ID
	The user ID of the agent that processed the key archival request.
`archivedOn`	number of seconds since 1 January 1970
	The time when the key was stored in the archive (for completed Data Recovery Manager requests).
`keyAlgorithm`	OID string
	The object identifier (OID) used by the archived key corresponding to this request (Data Recovery Manager requests). For example, the OID for an RSA encryption key is `"1.2.840.113549.1.1.1."`
`keyLength`	number
	The number of bits in the archived key (Data Recovery Manager requests).
`op`	`displayBySerial`
	Indicates the operation that was requested.
`ownerName`	Distinguished Name (DN) string. See RFC 2253.
	The subject entry on the certificate corresponding to an archived encryption key (Data Recovery Manager requests only). Example: `CN=Alice Apple, UID=alice, OU=People, O=Example, C=US`
`publicKey`	string
	A string of two-digit hexadecimal numbers separated by colon. Each number represents a byte in the public key corresponding to the private key in the archive.
`serviceURL`	string
	Indicates the URI that was used to request this form. By default, this will always be `"/kra/displayBySerial"`.
`state`	`VALID` | `INVALID`
	The current status of the key corresponding to the request.

Display Key For Recovery Interface

Description

URI: `/kra/displayBySerialForRecovery`

Available on: Data Recovery Manager

Function: Displays a form for recovering a key.

The Display Key For Recovery interface is typically used in the list returned by the Key Recovery Query Interface. The purpose of the interface is to retrieve information about a specific key (based on its DRM serial number) and present a form to collect the rest of the information required to start the recovery process. The response template, `displayBySerialForRecovery.template`, will render a form that uses the Recover Key By Serial Number Interface.

Default Forms

The Display Key For Recovery interface is used in the `queryKeyForRecovery.template` file. Each key in the list of keys satisfying the query has a button the user can press to start the recovery process. This button submits data to the Display Key For Recovery interface.

Request Parameters

The following table lists the parameters accepted by the Display Key For Recovery interface.

Table 6-9 Parameters Accepted by the Display Key For Recovery Interface

Parameter	Format and Description
`op`	`displayBySerial` Specifies the operation to perform. The only valid value is `displayBySerial`.

Table 6-9 Parameters Accepted by the Display Key For Recovery Interface *(Continued)*

Parameter	Format and Description
`publicKeyData`	base-64 encoded certificate data This optional parameter allows you to pass the certificate corresponding to the key to revoke to the interface. The certificate will be required to recover the key, and passing it here allows the certificate to be automatically filled in on the resulting form.
`serialNumber`	number The serial number of the key to display. Note that this is the DRM serial number, not the serial number from a certificate.
`templateName`	string Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.

Response

The default response template is `displayBySerialForRecovery.template`. The base JavaScript for responses is inserted in place of the `<CMS_TEMPLATE>` tag. In addition, the Display Key For Recovery interface adds the JavaScript variables listed in the following table:

Table 6-10 Variables Returned by the Display Key For Recovery Interface

Variable	Description
result.header variables	Variables added to the header object.
`archivedBy`	user ID The user ID of the agent that processed the key archival request.
`archivedOn`	number of seconds since 1 January 1970 The time when the key was stored in the archive (for completed Data Recovery Manager requests).
`keyAlgorithm`	OID string The object identifier (OID) used by the archived key corresponding to this request (Data Recovery Manager requests). For example, the OID for an RSA encryption key is `"1.2.840.113549.1.1.1."`

Table 6-10 Variables Returned by the Display Key For Recovery Interface *(Continued)*

Variable	Description
`keyLength`	number
	The number of bits in the archived key.
`noOfRequiredAgents`	number
	Indicates the number of authorized agents who must approve the request before the key can be recovered.
`op`	`displayBySerial`
	Indicates the operation that was requested.
`ownerName`	Distinguished Name (DN) string. See RFC 2253.
	The subject entry on the certificate corresponding to an archived encryption key (Data Recovery Manager requests only). Example: `CN=Alice Apple, UID=alice, OU=People, O=Example, C=US`
`publicKey`	string
	A string of two-digit hexadecimal numbers separated by colon. Each number represents a byte in the public key corresponding to the private key to be archived.
`recoveryID`	number
	A unique identification number assigned to each recovery request when it gets created.
`serviceURL`	string
	Indicates the URI that was used to request this form. By default, this will always be `"/kra/displayBySerial"`.
`state`	`VALID` | `INVALID`
	The current status of the key corresponding to the request.

Examine Recovery Interface

Description

URI: `/kra/examineRecovery`

Available on: Data Recovery Manager

Function: Checks to see if a recovery request identification number is valid.

The Examine Recovery interface is an intermediate interface that validates a request identification number. The interface takes a request identification number and makes sure that it is associated with a valid recovery request on the CMS server. If the request number is not valid, an error is returned (using the `GenError.template` file). If the request number is valid, the interface returns a template that by default creates a form for posting a user ID and password to the Grant Recovery Interface.

Default Forms

The Examine Recovery interface is used by the `grantRecovery.html` form in the Data Recovery Manager web directory (`web/agents/kra`). The form accepts a recovery id and posts it to the Examine Recovery interface, which returns either a form for granting the recovery or an error message (if the request id is invalid).

Request Parameters

The following table lists the parameters accepted by the Examine Recovery interface.

Table 6-11 Parameters Accepted by the Examine Recovery Interface

Parameter	Format and Description
`op`	`examineRecovery`
	The only operation supported by the Examine Recovery interface is `examineRecovery`.
`recoveryID`	number
	The unique identification number assigned to the recovery request.
`templateName`	string
	Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.

Response

The `GenError.template` file is used to return any error messages. Successful requests use the `examineRecovery.template` file. Since the interface is used to validate a request id, the `examineRecovery.template` is used by default to create a form to submit a user ID and password to the Grant Recovery Interface to approve that request.

The base JavaScript for responses is inserted in place of the `<CMS_TEMPLATE>` tag. In addition, the Examine Recovery interface adds the JavaScript variables listed in the following table:

Table 6-12 Variables Returned by the Examine Recovery Interface

Variable	Description
result.header variables	Variables added to the header object.
`op`	`examineRecovery`
	Indicates the operation requested. Only `examineRecovery` is supported.
`recoveryID`	number
	The unique request identification number that was passed to the interface in the request. Since the `examineRecovery.template` has been returned, the `recoveryID` has been verified to correspond to a valid request.
`serialNumber`	number
	The serial number of the key in the archive.
`serviceURL`	`/kra/examineRecovery`
	The URL that was used to access the Examine Recovery interface.

Get Approval Status Interface

Description

URI: `/kra/getApprovalStatus`

Available on: Data Recovery Manager

Function: Displays the status of a recovery operation.

The Get Approval Status interface accepts a recovery request number and returns the status of the request. The response includes the number of agents required to approve the recovery and the number that have already granted approval. While a request is pending, agents can use the Grant Recovery Interface to submit user IDs and passwords. The Examine Recovery Interface can be used as an intermediate step to make sure that there is a valid request corresponding to a request identification number (this is the default behavior of the Authorize Recovery link from the Data Recovery Manager gateway).

Default Forms

The Get Approval Status interface requires a valid request id number, so it is used by default only in the response template for the Recover Key By Serial Number Interface. The Recover Key By Serial Number Interface returns a request id number for pending (non-local) requests. In the default response, this request id number is used in a form that polls the Get Approval Status interface using a `<META HTTP-EQUIV="Refresh">` tag to make continuous requests until the recovery is completed. The page looks at the `result.recordSet` array to see how many agents have approved the request so far.

Request Parameters

The following table lists the parameters accepted by the Get Approval Status interface.

Table 6-13 Parameters Accepted by the Get Approval Status Interface

Parameter	Format and Description
`recoveryID`	number The unique identification number assigned to the recovery request.
`templateName`	string Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.

Response

The default response template depends on the status of the request. While the request is pending, the `getApprovalStatus.template` file is used. Once the request is completed the `finishRecovery.template` file is used. By default, these templates are identical except for the `<META HTTP-EQUIV="Refresh" CONTENT="5">` tag in `getApprovalStatus.template`, which causes the page to refresh itself every 5 seconds.

The base JavaScript for responses is inserted in place of the `<CMS_TEMPLATE>` tag. In addition, the Get Approval Status interface adds the JavaScript variables listed in the following table:

Table 6-14 Variables Returned by the Get Approval Status Interface

Variable	Description
result.header variables	Variables added to the header object.
`errorDetails`	string If an error occurred while processing the request, this variable contains an error message.
`noOfRequiredAgents`	number The number of agents required to supply passwords before the key can be recovered. Compare to `result.recordSet.length`, which indicates how many agents have supplied valid passwords so far.
`recoveryID`	number The unique number assigned to this recovery request.
`serialNumber`	number The serial number of the key in the archive.
`status`	`complete` | `null` Once the required number of agents have supplied valid passwords, the status becomes `complete` and the `finishRecovery.template` is used for the response. Until the `status` is `complete`, `result.header.status` is not included in the response.
result.recordSet[i] variables	Variables added to `record` objects in the response.
`agentName`	string The user ID of an agent that has supplied a valid password to approve the recovery.

Get PKCS #12 Data Interface

Description

URI: `/kra/getPk12`

Available on: Data Recovery Manager

Function: Retrieves the PKCS #12 data containing a recovered key and certificate.

The Get PKCS #12 Data interface is used to retrieve the PKCS #12 blob containing a recovered key and its associated certificate. The PKCS #12 data is encrypted using the password supplied to the Recover Key By Serial Number Interface.

Default Forms

No default forms use the Get PKCS #12 Data interface. The `finishRecovery.template`, `getApprovalStatus.template`, and `recoverBySerial.template` files all embed links to the Get PKCS #12 Data interface that are displayed if the recovery has been granted (if `result.header.status = "complete"`).

Request Parameters

The Get PKCS #12 Data interface requires only a recovery request ID. This parameter is set in the HTML form and passed to the interface using HTTP.

Table 6-15 Parameters Accepted by the Get PKCS #12 Data Interface

Parameter	Format and Description
`recoveryID`	number The unique identification number assigned to the recovery request.
`templateName`	string Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.

Response

If the recovery has been granted, the response is to return a PKCS #12 blob with the MIME type `application/x-pkcs12`. This blob is encrypted with a password (supplied when the recovery request was initiated) and contains the key and corresponding certificate.

If the recovery has not yet been granted, the default response template is `finishRecovery.template`. See the Response section for the “Get Approval Status Interface” on page 122 for details on the JavaScript returned.

Grant Recovery Interface

Description

URI: `/kra/grantRecovery`

Available on: Data Recovery Manager

Function: Submits a password to approve a key recovery.

The Grant Recovery interface is used by agents to submit their passwords to authorize a key recovery. Key recovery requires a certain number of authorized agents submit passwords before the key can be recovered. Each agent uses the Grant Recovery interface to submit a password until the minimum number of passwords has been collected.

Default Forms

The Grant Recovery interface is used by the `examineRecovery.template` result template by default. This form is returned when a request ID has been verified as valid, so it ensures that the Grant Recovery interface will be called with a valid request id.

Request Parameters

The following table lists the parameters accepted by the Grant Recovery interface.

Table 6-16 Parameters Accepted by the Grant Recovery Interface

Parameter	Format and Description
`agentID`	string The key recovery agent id used by the agent making the request. Agent ids for key recovery are configured independently of agent user IDs, and so may be different.
`agentPWD`	string The password corresponding to the key recovery agent user ID submitted with `agentID`.
`op`	`grantRecovery` The only operation supported by the Grant Recovery interface is `grantRecovery`.
`recoveryID`	number The unique identification number assigned to the recovery request.
`templateName`	string Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.

Response

The default response template is `grantRecovery.template`. The base JavaScript for responses is inserted in place of the `<CMS_TEMPLATE>` tag. In addition, the Grant Recovery interface adds the JavaScript variables listed in the following table:

Table 6-17 Variables Returned by the Grant Recovery Interface

Variable	Description
result.header variables	Variables added to the header object.
`agentID`	string The key recovery agent id whose password was submitted.
`op`	`grantRecovery` Indicates the operation requested. Only `grantRecovery` is supported.

Table 6-17 Variables Returned by the Grant Recovery Interface *(Continued)*

Variable	Description
`recoveryID`	number The unique request identification number that was passed to the interface in the request.
`serialNumber`	number The serial number of the key in the archive.
`serviceURL`	`/kra/grantRecovery` The URL that was used to access the Grant Recovery interface.

Key Query Interface

Description

URI: `/kra/queryKey`

Available On: Data Recovery Manager only.

Function: Retrieves a set of archived keys based on a flexible query specification.

The Key Query interface allows you to build query criteria much like an LDAP query. Criteria can be combined using logical AND or OR for flexibility. You can match keys in the archive based on the DRM serial number, the DN of the key owner, the certificate containing the corresponding public key, or the agent that archived the key.

The interface returns the public keys corresponding to the archived keys that match the query criteria.

Default Forms

The Data Recovery Manager form `searchKey.html` uses the Key Query interface. The form allows the agent to specify any of the valid query parameters: serial number range, key owner DN, public key certificate, or user ID of the archiving agent.

Request Parameters

The following table lists the parameters accepted by the Key Query interface.

The `queryFilter` parameter must be a valid query filter. The syntax and valid query parameters are too complex to describe in the parameter table. Details about valid parameters and values for query filters are in a separate table following the parameters.

Table 6-18 Parameters Accepted by the Key Query Interface

Parameter	Format and Description
`maxCount`	number Specifies the maximum number of keys to display on each page returned. If more than `maxCount` keys match the search criteria, each page will have controls to see the next or previous page of results.
`op`	`queryKey` The only operation supported by the List Certificates interface is `queryKey`.

`queryFilter`	`([<OP>]<FILTER>[<FILTER>...])`
	
	Details about building query filters are provided in the next table.
	
	The `queryFilter` must be enclosed in parentheses.
	
	The `<OP>` argument, required if there is more than one `<FILTER>`, specifies how the filters that follow should be logically evaluated:
	
	• `&` (ampersand) means that the filters should be linked by a logical AND: all filters must evaluate to true for the expression to match a key.
	
	• `|` (pipe) means that the filters should be linked by a logical OR: if at least one filter evaluates to true, the expression matches a key.
	
	Any number of filters can be concatenated within any set of parentheses.
	
	An example filter is
	
	`(|(keyArchivedBy=drmagent1)(&(keySerialNumber>=5)(keySerialNumber<=35)))`
	
	This filter matches any key that was archived by `drmagent1` or keys with serial numbers between 5 and 35 in the archive.

`querySentinel`	number The `querySentinel` indicates which record out of the total matching set should be the first displayed on the resulting output page. For example, if `totalRecordCount = 15` and `maxCount = 5`, set `querySentinel` to 6 to show the 6th through 10th element of the matching set.

Table 6-18 Parameters Accepted by the Key Query Interface *(Continued)*

Parameter	Format and Description
`templateName`	string Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.
`totalRecordCount`	number The total number of keys in the archive that match the `queryFilter`. This number is returned by the interface in the initial response. This can be posted in subsequent calls to prevent the CMS server from calculating a number. In this way every page can tell the user the absolute size of the matching set, rather than have the size of the set appear to decrease on each next page of data.

The following table describes the parameter names that are valid for constructing query filters, and the range of valid values that can be used with the parameter. The parameters can be combined using parentheses and logical operators (as described in the previous table) to construct query filters of arbitrary complexity.

In a filter, the parameter name is compared to the expression value using one of the relational operators = (matches), < (less than), <= (less than or equal to), > (greater than), or >= (greater than or equal to). Some expressions (such as `keySerialNumber`) accept the asterisk (*) as a wildcard to match 0 or more characters; for example, `"keyOwnerName=CN=*,*OU=Engineering*"` matches `"CN=jsmith, OU=Engineering, O=Organization1, C=US"` and `"CN=jdoe, OU=Engineering, O=Organization2, C=CA."`

Table 6-19 Key Query queryFilter Parameters

Parameter	Expression Values
`keyArchivedBy`	Value: user ID of an agent This matches the user ID of the agent that approved archival. For example: `(keyArchivedBy=drmagent1)`

Table 6-19 Key Query queryFilter Parameters *(Continued)*

Parameter	**Expression Values**
`keySerialNumber`	Value: serial number of a key This matches the serial number assigned by the Data Recovery Manager to a key when it is archived. Note that this differs from the serial number of the certificate corresponding to the key. Use the asterisk (*) wildcard to match any or partial serial numbers. For example: `(&(keySerialNumber>=1)(keySerialNumber<=10))`

`keyOwnerName`	Value: distinguished name appearing as the subject of the certificate corresponding to the key.
	
	This parameter matches the DN (taken from the certificate) stored with the key during archival. Use the wildcard (*) to match parts of a DN. If users are asked to enter a DN and it is likely that they will not remember all of the data, use the wildcard to allow partial matches.
	
	For example:
	
	`<INPUT TYPE=TEXT NAME=partialDN>`
	`form.queryFilter.value = "(keyOwnerName=*" + partialDN + "*)";`

`publicKey`	Value: `x509cert#`<base-64 encoded certificate> The query will find the archived key corresponding to the public key in the certificate. See `searchKey.html` for an example. The certificate in the proper format can be retrieved from a Certificate Manager using an interface such as the Display Certificate By Serial Number Interface. For example: `(publicKey=x509cert#"-----BEGIN CERTIFICATE-----\nMII\"BFK364J978nmnJK89yjha\"asFDSJ973-----END CERTIFICATE-----")`

Response

The default response template is `queryKey.template`. The base JavaScript for responses is inserted in place of the `<CMS_TEMPLATE>` tag. In addition, the Key Query interface adds the JavaScript variables listed in the following table.

Table 6-20 Variables Returned by the Key Query Interface

Variable	Description
result.header variables	Variables added to the header object.
`archiverName`	Distinguished Name (DN) string. See RFC 2253.
	The subject name on the Data Recovery Manager's signing certificate.
`op`	`queryKey`
	Indicates the operation that was requested.
`queryFilter`	query string
	Contains the query string that was used to select keys from the archive. See the previous section, "Request Parameters" on page 129, for details on how `queryFilter` is constructed. The queryFilter should be reused to query for more keys when there are more records than displayed on the current page.
`querySentinel`	number
	The number of the first record shown on this page of output. This is an index relative to the `totalRecordCount`. For example, if `querySentinel = 6`, `totalRecordCount = 8`, and `maxCount = 5`, then the 6th through 8th records that matched the query will be displayed.
`templateName`	string
	Indicates the name of the template that was used to display the response. By default it is `queryKey.template`, but it can be changed with the `templateName` request parameter.
`totalRecordCount`	number
	The total number of records that match the query criteria (`queryFilter`). This may be more than the number of records currently displayed, which is limited by `fixed.maxCount`.
result.fixed variable	Variable added to the `fixed` object.
`maxCount`	number
	The maximum number of records that can be displayed on a single page of output (taken from the request parameter `maxCount`).
result.recordSet[i] variables	Variables added to `record` objects in the response.
`archivedBy`	user ID
	The user ID of the agent that processed the key archival request.

Table 6-20 Variables Returned by the Key Query Interface *(Continued)*

Variable	Description
`archivedOn`	number of seconds since 1 January 1970 The time when the key was stored in the archive (for completed Data Recovery Manager requests).
`keyAlgorithm`	OID string The object identifier (OID) used by the archived key corresponding to this request (Data Recovery Manager requests). For example, the OID for an RSA encryption key is `"1.2.840.113549.1.1.1."`
`keyLength`	number The number of bits in the archived key (Data Recovery Manager requests).
`ownerName`	Distinguished Name (DN) string. See RFC 2253. The subject entry on the certificate corresponding to an archived encryption key (Data Recovery Manager requests only). Example: `CN=Alice Apple, UID=alice, OU=People, O=Example, C=US`
`publicKey`	string A string of two-digit hexadecimal numbers separated by colon. Each number represents a byte in the public key corresponding to the private key to be archived.
`serialNumber`	number A unique identification number that identifies a key in the archive. This differs from the certificate serial number.
`state`	`VALID | INVALID` The current status of the key corresponding to the request.

Key Recovery Query Interface

Description

URI: `/kra/queryKeyForRecovery`

Available On: Data Recovery Manager only.

Function: Retrieves a set of archived keys, for the purpose of recovering them, based on a flexible query specification.

The Key Recovery Query interface allows you to build query criteria much like an LDAP query. Criteria can be combined using logical AND or OR for flexibility. You can match keys in the archive based on the DRM serial number, the DN of the key owner, the certificate containing the corresponding public key, or the agent that archived the key.

The interface returns the public keys corresponding to the archived keys that match the query criteria. The list of keys in the response will each have a "Recover" button that allows the key to be recovered.

Default Forms

The Data Recovery Manager form `recoverKey.html` uses the Key Recovery Query interface. The form allows the agent to specify any of the valid query parameters: serial number range, key owner DN, public key certificate, or user ID of the archiving agent.

Request Parameters

The following table lists the parameters accepted by the Key Recovery Query interface.

The `queryFilter` parameter must be a valid query filter. The syntax and valid query parameters are too complex to describe in the parameter table. The parameter is identical to the `queryFilter` request parameter for the Key Query Interface; refer to “Key Query Interface” on page 128 for complete details on building queries.

Table 6-21 Parameters Accepted by the Key Recovery Query Interface

Parameter	Format and Description
`maxCount`	number Specifies the maximum number of keys to display on each page returned. If more than `maxCount` keys match the search criteria, each page will have controls to see the next or previous page of results.
`op`	`queryKey` The only operation supported by the List Certificates interface is `queryKey`.

Table 6-21 Parameters Accepted by the Key Recovery Query Interface *(Continued)*

Parameter	Format and Description
`publicKeyData`	base-64 encoded certificate The certificate containing the public key associated with a key to be matched in the archive. The `publicKeyData` should include the "`-----BEGIN CERTIFICATE-----`" header and "`-----END CERTIFICATE-----`" footer.

`queryFilter`	`([<OP>]<FILTER>[<FILTER>...])`
	
	Details about building query filters are provided in the next table.
	
	The `queryFilter` must be enclosed in parentheses.
	
	The `<OP>` argument, required if there is more than one `<FILTER>`, specifies how the filters that follow should be logically evaluated:
	
	• `&` (ampersand) means that the filters should be linked by a logical AND: all filters must evaluate to true for the expression to match a key.
	
	• `|` (pipe) means that the filters should be linked by a logical OR: if at least one filter evaluates to true, the expression matches a key.
	
	Any number of filters can be concatenated within any set of parentheses.
	
	An example filter is
	
	`(|(keyArchivedBy=drmagent1)(&(keySerialNumber>=5)(keySerialNumber<=35)))`
	
	This filter matches any key that was archived by `drmagent1` or keys with serial numbers between 5 and 35 in the archive.

`querySentinel`	number The `querySentinel` indicates which record out of the total matching set should be the first displayed on the resulting output page. For example, if `totalRecordCount = 15` and `maxCount = 5`, set `querySentinel` to 6 to show the 6th through 10th element of the matching set.
`templateName`	string Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.

Table 6-21 Parameters Accepted by the Key Recovery Query Interface *(Continued)*

Parameter	Format and Description
`totalRecordCount`	number The total number of keys in the archive that match the queryFilter. This number is returned by the interface in the initial response. This can be posted in subsequent calls to prevent the CMS server from calculating a number. In this way every page can tell the user the absolute size of the matching set, rather than have the size of the set appear to decrease on each next page of data.

Response

The default response template is `queryKeyForRecovery.template`. The base JavaScript for responses is inserted in place of the `<CMS_TEMPLATE>` tag. In addition, the Key Recovery Query interface adds the JavaScript variables listed in the following table.

Table 6-22 Variables Returned by the Key Recovery Query Interface

Variable	Description
result.header variables	Variables added to the header object.
`archiverName`	Distinguished Name (DN) string. See RFC 2253. The subject name on the Data Recovery Manager's signing certificate.
`op`	`queryKey` Indicates the operation that was requested.
`queryFilter`	query string Contains the query string that was used to select keys from the archive. See the previous section, "Request Parameters" on page 129, for details on how `queryFilter` is constructed. The queryFilter should be reused to query for more keys when there are more records than displayed on the current page.
`querySentinel`	number The number of the first record shown on this page of output. This is an index relative to the `totalRecordCount`. For example, if `querySentinel = 6`, `totalRecordCount = 8`, and `maxCount = 5`, then the 6th through 8th records that matched the query will be displayed.

Table 6-22 Variables Returned by the Key Recovery Query Interface *(Continued)*

Variable	Description
`templateName`	string
	Indicates the name of the template that was used to display the response. By default it is `queryKey.template`, but it can be changed with the `templateName` request parameter.
`totalRecordCount`	number
	The total number of records that match the query criteria (`queryFilter`). This may be more than the number of records currently displayed, which is limited by `fixed.maxCount`.
result.fixed variable	Variable added to the `fixed` object.
`maxCount`	number
	The maximum number of records that can be displayed on a single page of output (taken from the request parameter `maxCount`).
result.recordSet[i] variables	Variables added to `record` objects in the response.
`archivedBy`	user ID
	The user ID of the agent that processed the key archival request.
`archivedOn`	number of seconds since 1 January 1970
	The time when the key was stored in the archive (for completed Data Recovery Manager requests).
`keyAlgorithm`	OID string
	The object identifier (OID) used by the archived key corresponding to this request (Data Recovery Manager requests). For example, the OID for an RSA encryption key is `"1.2.840.113549.1.1.1."`
`keyLength`	number
	The number of bits in the archived key (Data Recovery Manager requests).
`ownerName`	Distinguished Name (DN) string. See RFC 2253.
	The subject entry on the certificate corresponding to an archived encryption key (Data Recovery Manager requests only). Example: `CN=Alice Apple, UID=alice, OU=People, O=Example, C=US`

Table 6-22 Variables Returned by the Key Recovery Query Interface *(Continued)*

Variable	Description
`publicKey`	string A string of two-digit hexadecimal numbers separated by colon. Each number represents a byte in the public key corresponding to the private key to be archived.
`serialNumber`	number A unique identification number that identifies a key in the archive. This differs from the certificate serial number.
`state`	`VALID` | `INVALID` The current status of the key corresponding to the request.

Process Certificate Request Interface

Description

URI: `/ca/processCertReq` or `/ra/processCertReq`

Available on: Certificate Manager or Registration Manager

Function: Agents can use the Process Certificate Request interface to accept, reject, or cancel requests to sign, renew, or revoke requests.

The Process Request Interface is used to assign pending requests to agents. The Process Certificate Request interface is used by an agent to act on requests owned by the agent (or not yet owned).

The interface is called once the sequence number of a pending request is known, so forms that use the interface are usually embedded in a response template from an interface that can list pending requests (for example, the Requests Query Interface). The list of pending requests will also include the `requestType` variable to distinguish between signing, revocation, and renewal requests.

Default Forms

No default forms use the Process Certificate Request Interface directly. The `listRequests.html` form calls the Requests Query Interface which returns data using the `processReq.template` when pending requests are selected.

The `processReq.template` will create a link to the Process Certificate Request Interface for all requests with `status == pending`. Additional JavaScript in that template is used to add parameters to the request to the Process Certificate Request interface (such as the validity period for enrollments and renewals).

Request Parameters

The following table lists the parameters accepted by the Process Request Interface.

The agent interface requires SSL client authentication, so information about the agent can be gleaned from the certificate used to authenticate and does not need to be passed in parameters.

Table 6-23 Parameters Accepted by the Process Certificate Request Interface

Parameter	Format and Description
`addExts`	base-64 encoded certificate extensions Specifies any additional certificate extensions that should be added. There is sample code in the `cms_sdk/samples/exttools` directory that can be used to generate the base-64 encoding of some extensions; the code can be modified to generate other extensions.
`certTypeEmail`	`yes` | `no` Specifies whether to set the netscape certificate extension S/MIME client certificate bit (bit 2).
`certTypeEmailCA`	`yes` | `no` Specifies whether to set the netscape certificate extension S/MIME certificate issuer bit (bit 6).
`certTypeObjSigning`	`yes` | `no` Specifies whether to set the netscape certificate extension object signing certificate bit (bit 3).
`certTypeObjSigningCA`	`yes` | `no` Specifies whether to set the netscape certificate extension object signing certificate issuer bit (bit 7).

Table 6-23 Parameters Accepted by the Process Certificate Request Interface *(Continued)*

Parameter	Format and Description
`certTypeSSLCA`	`yes` | `no` Specifies whether to set the netscape certificate extension SSL certificate issuer bit (bit 5).
`certTypeSSLClient`	`yes` | `no` Specifies whether to set the netscape certificate extension SSL client authentication certificate bit (bit 0).
`certTypeSSLServer`	`yes` | `no` Specifies whether to set the netscape certificate extension SSL server authentication certificate bit (bit 1).
`checkPubKeyUniqueness`	`yes` | `no` Specifies whether the CMS server should ensure that the new certificate's public key is unique.
`checkValidityNesting`	`yes` | `no` Specifies whether the CMS server should check to make sure that the certificate does not expire later than the CA's signing certificate.
`csrRequestorEmail`	string The email address of the party making the original signing request.
`csrRequestorName`	string The real name of the party making the original signing request.
`csrRequestorPhone`	string The phone number of the party making the original signing request.
`grantCMAgentPrivilege`	`yes` | `no` Specifies whether the new certificate will be trusted as a Certificate Manager agent.
`grantDRMAgentPrivilege`	`yes` | `no` Specifies whether the new certificate will be trusted as a Data Recovery Manager agent.
`grantRMAgentPrivilege`	`yes` | `no` Specifies whether the new certificate will be trusted as a Registration Manager agent.

Table 6-23 Parameters Accepted by the Process Certificate Request Interface *(Continued)*

Parameter	Format and Description
`grantTrustedManagerPrivilege`	`yes` | `no` Specifies whether the new certificate will be an SSL certificate used by a server that is trusted. For example, set this to `yes` when you issue the SSL server certificate for a new Registration Manager.
`grantUID`	user ID The user ID that will be associated with the agent certificate.
`seqNum`	number The sequence number of the request that is being acted upon. When requests are listed from the Requests Query Interface, their sequence numbers are returned from the CMS server in the `result.header.seqNum` field.
`signatureAlgorithm`	`MD5withRSA` | `SHA1withDSA` | `SHA1withRSA` Specifies the signing algorithm that should be used to sign a newly issued certificate. The CA signing key must match the key type (RSA or DSA) of the selected algorithm.
`subject`	Distinguished Name (DN) string. See RFC 2253. DN to be used for the certificate subject. Example: CN=Alice Apple, UID=alice, OU=People, O=Example, C=US
`templateName`	string Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.
`toDo`	`accept` | `cancel` | `clone` | `reject` Specifies the action to take on the request.

Table 6-23 Parameters Accepted by the Process Certificate Request Interface *(Continued)*

Parameter	Format and Description
`validityLength`	number of seconds The length of time, in seconds, for which the newly issued certificate will be valid. The following list shows the approximate number of seconds in some common time intervals: • 1 day: 86,400 • 7 days: 604,800 • 14 days: 1,209,600 • 30 days: 2,592,000 • 180 days: 15,552,000 • 365 days: 31,536,000 • 540 days: 46,665,000 • 730 days: 63,072,000

Response

The default response template is `processCertReq.template`. The base JavaScript for responses is inserted in place of the `<CMS_TEMPLATE>` tag. In addition, the Process Certificate Request interface adds the JavaScript variables listed in the following table.

Table 6-24 Variables Returned by the Process Certificate Request Interface

Variable	Description
result.header variables	Variables added to the header object.
`assignedTo`	The user ID of the agent to whom the request is currently assigned. Compare to `callerName` to see if the agent viewing the request is the current owner.
`authorityid`	`ca` | `ra` The type of authority that handled the request: `ca` for Certificate Manager or `ra` for Registration Manager.
`callerName`	The user ID of the agent who is viewing the request. This data is determined from the SSL client certificate presented to the agent interface.

Table 6-24 Variables Returned by the Process Certificate Request Interface *(Continued)*

Variable	Description
`certsUpdated`	number
	The number of certificates that were updated in the publishing directory if publishing is enabled (if `result.header.dirEnabled = yes`).
`certType`	`ca` | `CEP-Request` | `client` | `objSignClient` | `ra` | `server` | `other`
	Specifies the type of certificate request that was acted upon. The `certType` is not associated with any certificate extensions. It may be used by policy modules to make decisions, and it may be used by a CMS server to determine how to decode the request or format the response.
`clonedRequestId`	number
	The sequence number (or request ID) of the newly cloned request, if the action `toDo = clone`.
`createdOn`	number of seconds since 1 January 1970
	The time when the certificate request was created on the CMS server.
`dirEnabled`	`yes` | `no`
	Indicates whether or not LDAP publishing is enabled on the CMS server. If `yes`, then certsUpdated will indicate how many certificates were updated in the directory as a result of the certificate processing action.
`errorDetails`	message
	A more detailed description of any processing errors.
`errors`	message
	A message explaining any errors that may have occurred.
`ext_email`	`true` | `false`
	Indicates whether or not the Netscape certificate extension S/MIME bit (bit 2) is set in the certificate or request that was processed.
`ext_email_ca`	`true` | `false`
	Indicates whether or not the Netscape certificate extension S/MIME CA bit (bit 6) is set in the certificate or request that was processed.
`ext_object_signing`	`true` | `false`
	Indicates whether or not the Netscape certificate extension object signing bit (bit 3) is set in the certificate or request that was processed.

Table 6-24 Variables Returned by the Process Certificate Request Interface *(Continued)*

Variable	Description
`ext_object_signing_ca`	`true` | `false` Indicates whether or not the Netscape certificate extension object signing CA bit (bit 7) is set in the certificate or request that was processed.
`ext_ssl_ca`	`true` | `false` Indicates whether or not the Netscape certificate extension SSL CA bit (bit 5) is set in the certificate or request that was processed.
`ext_ssl_client`	`true` | `false` Indicates whether or not the Netscape certificate extension SSL client bit (bit 0) is set in the certificate or request that was processed.
`ext_ssl_server`	`true` | `false` Indicates whether or not the Netscape certificate extension SSL server bit (bit 1) is set in the certificate or request that was processed.
`grantError`	`SUCCESS` | error message If there were any errors processing a request for an agent certificate, this field will have an error message. If `grantError = SUCCESS`, the agent was created successfully.
`grantPrivilege`	string Indicates the groups to which the new agent or certificate has been added. If there is more than one group, the group names will be separated by the text `" and "` in the string. Valid group names are: • `Certificate Manager Agents` • `Data Recovery Manager Agents` • `Registration Manager Agents` • `Trusted Managers` (for SSL certificates used by trusted servers)
`grantUID`	user ID The user name assigned to the agent certificate that was processed.
`localca`	`yes` | `no` Indicates whether the CMS server that processed the request is a Certificate Manager (the CMS server is not a Registration Manager).
`localkra`	`yes` | `no` Indicates whether the CMS server that processed the request also hosts a Data Recovery Manager.

Table 6-24 Variables Returned by the Process Certificate Request Interface *(Continued)*

Variable	Description
`requestType`	`enrollment` | `getCAChain` | `getCertificates` | `getRevocationInfo` | `renewal` | `revocation` | `unrevocation` The `requestType` returns the type of request that was made to the interface returning this template.
`serialNumber`	number The serial number of the certificate that was processed. This might be a newly issued certificate or a newly revoked certificate.
`seqNum`	number The unique sequence number assigned to the request by the Certificate Manager or Registration Manager.
`signatureAlgorithm`	OID The object identifier (OID) string corresponding to the algorithm used to sign a newly issued certificate. For example, `"1.2.840.113549.1.1.4"` is the OID for MD5 with RSA signing.
`signatureAlgorithmName`	`MD5withRSA` | `SHA1withDSA` | `SHA1withRSA` The name token associated with the signature algorithm whose OID is stored in `signatureAlgorithm`.
`status`	`pending` | `complete` Indicates whether the request is complete or if further action is required. Responses from the Process Certificate Request interface will return a `status` of `complete` unless an error occurred.
`subject`	Distinguished Name (DN) string. See RFC 2RFC 2253. The DN appearing in the certificate subject field. Example DN: CN=Alice Apple, UID=alice, OU=People, O=Example, C=US
`subjectPublicKey`	string of hexadecimal numbers The actual public key in the certificate that was processed. The string is a sequence of two-digit hexadecimal numbers separated by colons: `"A1:3F:23:90:D7\n"` Each pair of digits represents a byte or octet in the public key.
`subjectPublicKeyInfo`	string A string indicating the public key algorithm and certificate signing algorithm. For example, `"RSA - 1.2.840.113549.1.1.1"` for a certificate with an RSA key signed using MD5 with RSA.

Table 6-24 Variables Returned by the Process Certificate Request Interface *(Continued)*

Variable	Description
`toDo`	`accept` | `cancel` | `clone` | `reject` Indicates the action that was taken to produce this response (that is, this is the same as the value of the `toDo` request parameter).
`updatedBy`	user ID The user ID of the agent that updated the request. In the case of a response from the Process Certificate Request interface, this is the user ID of the agent that made the request.
`updatedOn`	number of seconds since 1 January 1970 The time that the request was made to the Process Certificate Request interface.
`validityLength`	number of seconds The length of time, in seconds, for which the newly issued certificate will be valid. The following list shows the approximate number of seconds in some common time intervals: • 1 day: 86,400 • 7 days: 604,800 • 14 days: 1,209,600 • 30 days: 2,592,000 • 180 days: 15,552,000 • 365 days: 31,536,000 • 540 days: 46,665,000 • 730 days: 63,072,000
result.recordSet[i] variables	Variables added to record objects in the response.
`ext_prettyprint`	string A representation of any extensions present in the certificate in human-readable form.
`serialNumber`	number The decimal serial number of the certificate.

Process DRM Request Interface

Description

URI: `/kra/processReq`

Available on: Data Recovery Manager

Function: This interface allows an agent to view a request or assign the request to himself.

The Process DRM Request interface is slightly different from the Process Request Interface used by Certificate Managers and Registration Managers.

Default Forms

The Process DRM Request interface is not used in any default forms. The interface requires the sequence number of an archival request, so it is used in templates that list requests (with their sequence numbers) to render buttons that allow an agent to view or change the assignment of a request.

Request Parameters

The following table lists the parameters accepted by the Process DRM Request interface.

The agent interface requires SSL client authentication, so information about the agent can be gleaned from the certificate used to authenticate and does not need to be passed in parameters.

Table 6-25 Parameters Accepted by the Process DRM Request Interface

Parameter	Format and Description
`doAssign`	`yes` | `no` Specifies whether to assign the request to the agent using the interface. If this parameter is `no`, `null`, or not supplied, the request will be displayed, but not assigned to the agent.

Table 6-25 Parameters Accepted by the Process DRM Request Interface *(Continued)*

Parameter	Format and Description
moreComments	string
	Specifies additional comments to be stored with the request. Comments may be useful for future reference or to provide data to another agent that needs to process the request.
op	processReq
	The only operation supported is processReq.
overrideAssignment	yes | no
	Specifies whether or not to override an existing assignment. If a request is already assigned to another agent, an agent must use doAssign = yes and overrideAssignment = yes to assume assignment of the request. If the request is not assigned, overrideAssignment is not required.
requestorEmail	email address
	The email address of the entity requesting archival or recovery, if the information is available.
requestorName	string
	The name of the entity requesting archival or recovery, if the information is available.
requestorPhone	string
	The phone number of the entity requesting archival or recovery, if the information is available.
seqNum	number
	The sequence number of a pending request to assign or redisplay.
templateName	string
	Filename relative to the template directory (web/ee, web/agent/ca, web/agent/kra, or web/agent/ra) of a file to use as the response template. This template will be used for any response, overriding default template settings.

Response

The default response template is processReq.template. The base JavaScript for responses is inserted in place of the <CMS_TEMPLATE> tag. In addition, the Process DRM Request interface adds the JavaScript variables listed in the following table.

Table 6-26 Variables Returned by the Process DRM Request Interface

Variable	Description
result.header variables	Variables added to the header object.
`archivedBy`	user ID
	The user ID of the Data Recovery Agent that authorized the archival. This is sent only for completed archival requests.
`archivedOn`	seconds since 1 January 1970
	The time when the key was stored in the archive. Sent only for completed archival requests.
`callerName`	user ID
	The user ID of the agent who is viewing the request. This data is determined from the SSL client certificate presented to the agent interface.
`createdOn`	seconds since 1 January 1970
	The time when the request was created.
`keyAlgorithm`	OID
	The object identifier (OID) of the algorithm used by the key in the archive request.
`keyLength`	number
	The number of bits in the key in the archive request.
`localca`	`yes` | `no`
	Indicates whether the Certificate Manager associated with the Data Recovery Manager is located on the same host.
`localkra`	`yes` | `no`
	Indicates whether the Data Recovery Manager is located on the same host as the Process DRM Request interface. For this interface, the value is always `yes`.
`ownerName`	Distinguished Name (DN) string. See RFC 2253.
	The subject name on the certificate corresponding to the key to be archived. This appears only for archival requests.
`publicKey`	string
	A string of two-digit hexadecimal numbers separated by colon. Each number represents a byte in the public key corresponding to the private key to be archived.

Table 6-26 Variables Returned by the Process DRM Request Interface *(Continued)*

Variable	Description
`requestType`	`enrollment` | `recovery` Indicates whether the request was made to archive (`enrollment`) or recover a key.
`serialNumber`	number A unique identification number that identifies a key in the archive. This differs from the certificate serial number and also from the request identifier (`seqNum`).
`seqNum`	number The sequence number assigned to the request by the Data Recovery Manager.
`state`	`VALID` | `INVALID` The current status of the key corresponding to the request.
`status`	`pending` | `complete` Only requests that have `status == pending` need to use the Process DRM Request interface to assign the request to an agent. Requests that are `complete` can only be displayed.
`updatedBy`	user ID Indicates the user ID of the agent that last changed the status of the request.
`updatedOn`	seconds since 1 January 1970 The time when the request status was last changed.

Process Request Interface

Description

URI: `/ca/processReq` or `/ra/processReq`

Available on: Certificate Manager or Registration Manager

Function: Changes the assignment of a pending certificate request to an agent.

This is an agent interface and requires SSL client authentication with a valid agent certificate. The Process Request Interface can be used to assign a certificate request (identified by a sequence number) to the agent user ID associated with the certificate presented for authentication or to assign the request to nobody (remove any existing assignment).

Default Forms

The Process Request Interface is not used in any default forms. The interface requires the sequence number of a certificate request, so it is used in templates that list pending requests (with their sequence numbers) to render buttons that allow an agent to change the assignment of a request.

Request Parameters

The following table lists the parameters accepted by the Process Request interface.

The agent interface requires SSL client authentication, so information about the agent can be gleaned from the certificate used for authentication and does not need to be passed in parameters.

Table 6-27 Parameters Accepted by the Process Request Interface

Parameter	Format and Description
`doAssign`	`reassignToMe` | `reassignToNobody` | `assignToMe` | `null` Specifies how to change the agent assigned to a pending request. Use `reassignToMe` if the request has already been assigned to another agent or `assignToMe` if the request has not been assigned (or has just been assigned with `reassignToNobody`). If the interface is used with no `doAssign` parameter, the request in question is redisplayed using the `processReq.template`.
`seqNum`	number The sequence number of a pending request to assign or redisplay.
`templateName`	string Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.

Response

The default response template is processReq.template. The base JavaScript for responses is inserted in place of the `<CMS_TEMPLATE>` tag.

In addition, the Process Request interface adds the JavaScript variables listed in the following table. The table lists only the variables in the template related to the Process Request interface. The template includes more variables related to the status and origin of the request; these are documented in other interfaces that use the template.

Table 6-28 Variables Returned by the Process Request Interface

Variable	Description
result.header variables	Variables added to the header object.
`assignedTo`	The user ID of the agent to whom the request is currently assigned. Compare to `callerName` to see if the agent viewing the request is the current owner.
`callerName`	The user ID of the agent who is viewing the request. This data is determined from the SSL client certificate presented to the agent interface.
`requestType`	`enrollment` | `getCAChain` | `getCertificates` | `getRevocationInfo` | `renewal` | `revocation` | `unrevocation`
	The `requestType` returns the type of request that was made to the interface returning this template. The Process Request Interface is used to assign pending requests, so only requests for `enrollment`, `renewal`, `revocation`, and `unrevocation` would need a link to the Process Request Interface.
`seqNum`	number
	The sequence number assigned to the request by the Certificate Manager or Registration Manager.
`status`	`pending` | `complete`
	Only requests that have `status == pending` need to use the Process Request interface to assign the request to an agent.

Recover Key By Serial Number Interface

Description

URI: `/kra/recoverBySerial`

Available on: Data Recovery Manager

Function: Displays a form for recovering a key.

Once a key has been selected from the archive, the Recover Key By Serial Number Interface can be used to start the recovery operation. This interface accepts the serial number of an archived key and optionally the user ID and passwords of recovery agents.

The recovery operation can be completed by one call to the Recover Key By Serial Number Interface if the proper number of recovery agent user IDs and passwords are provided. In this case, a PKCS #12 blob is returned with the recovered key and an associated public key certificate.

The recovery operation can also be deferred if the agent passwords cannot be immediately provided. In this case, the response from the interface is a new form that polls the Get Approval Status Interface until the passwords are entered by agents (from other locations) and the PKCS #12 blob can be delivered.

Default Forms

The Recover Key By Serial Number Interface typically follows a request to the Display Key For Recovery Interface. The result of the Display Key For Recovery Interface request is to display a key from the archive using the `displayBySerialForRecovery.template` file. This template provides a form for making a request to the Recover Key By Serial Number Interface to initiate the recovery operation for the selected key.

Request Parameters

The following table lists the parameters accepted by the Recover Key By Serial Number interface.

Table 6-29 Parameters Accepted by the Recover Key By Serial Number Interface

Parameter	Format and Description
`cert`	base-64 encoded PKCS #7 certificate You must supply the certificate containing the public key corresponding to the key in the archive. Be sure to include the "-----BEGIN CERTIFICATE-----" header and "-----END CERTIFICATE-----" footer.
`localAgents`	`yes` | `no` Specifies whether agent user IDs and passwords have been entered locally (that is, submitted with this request), or will be entered remotely. If `localAgents = no`, agents will have to access the Examine Recovery interface with the request id and enter their passwords.
`nickname`	string Specifies an optional nickname for the certificate that will be returned in the PKCS #12 blob when the key is recovered.
`op`	`recoverBySerial` Specifies the operation to perform. The only valid value is `recoverBySerial`.
`p12password`	string Specifies a password used to protect the recovered key. When the PKCS #12 blob containing the key is returned, this password will be required to decrypt the data.
`p12passwordAgain`	string This parameter serves as a quality check; the string must match the value of `p12password`.
`pwd<n>`	string Specifies the password for agent `<n>`, where `<n>` is replaced by a sequence number (beginning with 0). If this is a local recovery operation, user IDs (`uid<n>`) and passwords must be supplied for the number of agents required to authorize a recovery. For example, if two agents are required, a local recovery operation requires `uid0`, `pwd0`, `uid1`, and `pwd1` parameters.
`serialNumber`	number The serial number of the key to recover. Note that this is the DRM serial number, not the serial number from a certificate.

Table 6-29 Parameters Accepted by the Recover Key By Serial Number Interface *(Continued)*

Parameter	Format and Description
`templateName`	string Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.
`uid<n>`	string Specifies the user ID for agent number `<n>`, if this is a local operation. See the description for `pwd<n>`.

Response

If the request was for a local recovery (`localAgents = yes` in the request), and the recovery is successful, the response will be the binary PKCS #12 blob containing the key and certificate. The MIME type of the response will be `application/x-pkcs12`.

If the request was not local or if there was an error, the default response template is `recoverBySerial.template`. The base JavaScript for responses is inserted in place of the `<CMS_TEMPLATE>` tag. If there are no errors, the default template includes JavaScript code that uses `window.location` to immediately replace the returned page with a call to the Get Approval Status Interface.

In addition, the Recover Key By Serial Number interface adds the JavaScript variables listed in the following table:

Table 6-30 Variables Returned by the Recover Key By Serial Number Interface

Variable	Description
result.fixed variables	Variables added to the fixed object
`host`	string The fully qualified domain name of the CMS server that processed the request. This allows the resulting template to construct forms that post data to the same interface using the same host and port.
`port`	number The port number that was used to service the request.

Table 6-30 Variables Returned by the Recover Key By Serial Number Interface *(Continued)*

Variable	Description
`scheme`	`http` | `https`
	The protocol that was used to make the request.
result.header variables	Variables added to the header object.
`errorDetails`	string
	If an error occurred while processing the request, this variable contains an error message.
`recoveryID`	number
	The unique number assigned to this recovery request. If agents need to approve the request, they will supply this number to the Examine Recovery Interface.

Remove Certificate Hold Interface

Description

URI: `/ca/doUnrevoke` or `/ra/doUnrevoke`

Available on: Certificate Manager or Registration Manager agent ports.

Function: Changes the status of a certificate that has been put on hold so that it is no longer considered revoked.

A certificate can be temporarily rendered invalid --or "put on hold"-- by revoking it with a revocation reason code of 6. If subsequent analysis reveals that the certificate does not need to be permanently revoked, it can be made valid again using the Remove Certificate Hold Interface.

Only agents with valid SSL client certificates can use the Remove Certificate Hold interface.

Default Forms

There are no default forms that use this interface. The Remove Certificate Hold Interface is usually accessed after a list of certificates currently on hold has been generated. In the response template that displays the list, certificates on hold can be rendered with a "Remove Certificate Hold" button next to their listing. The `displayByCert.template` and `queryCert.template` create buttons that use the Remove Certificate Hold Interface.

Request Parameters

The following table lists the parameters accepted by the Remove Certificate Hold interface.

Table 6-31 Parameters Accepted by the Remove Certificate Hold Interface

Parameter	Format and Description
`op`	`doUnrevoke`
	Specifies the operation to perform. The valid value for the Remove Certificate Hold interface is "`doUnrevoke`."
`serialNumber`	number
	The serial number (in decimal or hexadecimal) of the certificate to revoke.
`templateName`	string
	Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.

Response

The default response template is `unrevocationResult.template`. The base JavaScript for responses is inserted in place of the `<CMS_TEMPLATE>` tag. In addition, the `Remove Certificate Hold` Interface adds the JavaScript variables listed in the following table:

Table 6-32 Variables Returned by the Remove Certificate Hold Interface

Variable	Description
result.header variables	Variables added to the header object.
`dirEnabled`	`yes` | `no`
	Indicates whether LDAP publishing is enabled on the Certificate Manager that handled the request.
`dirUpdated`	`yes` | `no`
	If directory publishing is enabled, this indicates whether or not the directory was updated with the new certificate status.
`error`	message
	If there was an error while processing the unrevocation request, the error message is stored in this variable. Otherwise, the value is `null`.
`publishCRLError`	message
	If the CMS server attempted to publish the CRL and encountered an error, this variable contains the text of the error message.
`publishCRLSuccess`	`yes` | `no`
	If the CMS server attempted to publish the CRL to a directory, this variable will indicate whether the update was successful.
`serialNumber`	number
	The decimal serial number of the certificate.
`unrevoked`	`yes` | `no` | `pending`
	Indicates whether or not all certificates were successfully unrevoked.
`updateCRL`	`yes` | `no`
	Indicates whether or not the CMS server attempted to update the Certificate Revocation List (CRL). If `no` or `null`, the CRL will be updated at the next scheduled update interval. If `yes`, check `udpateCRLSuccess` to determine if the update was successful.
`updateCRLError`	message
	If the CMS server attempted to update the CRL and encountered an error, this variable contains the text of the error message.
`updateCRLSuccess`	`yes` | `no`
	If the CMS server attempted to update the CRL, this variable will indicate whether the update was successful.

Requests Query Interface

Description

URI: `/ca/queryReq` or `/ra/queryReq` or `/kra/queryReq`

Available on: Certificate Manager, Registration Manager, or Data Recovery Manager

Function: Lists requests that have been made to a given server.

The interface can return all requests, or subsets based on request status (pending, complete, etc.) and request type (enrollment, renewal, etc.).

Default Forms

The `listRequests.html` form uses the Requests Query Interface. This form can be found in the Certificate Manager, Registration Manager, and Data Recovery Manager web directories. The `listRequests.html` form presents menus for choosing the request type and status as well as a field for setting the lowest request id to return.

Since the number or records returned by the interface may be more than the user wants to see on one page, the response template (`queryReq.template`) may have a button to retrieve more records that also uses the Requests Query Interface.

Request Parameters

The following table lists the parameters that are used to view requests through the Requests Query Interface. This is an agent interface, so the HTTP POST or GET request must use SSL client authentication with a valid agent certificate.

Table 6-33 Parameters Accepted by the Requests Query Interface

Parameter	Format and Description
`maxCount`	number Specifies the maximum number of requests to display on a page. If more than `maxCount` requests match the criteria, the response template can include code to request more records from the Requests Query interface.

Table 6-33 Parameters Accepted by the Requests Query Interface *(Continued)*

Parameter	Format and Description
`op`	`queryReq` Specifies the operation to perform. For the Requests Query interface, this should be `queryReq`.
`reqState`	`showAll` | `showCancelled` | `showCompleted` | `showInService` | `showRejected` | `showWaiting` Specifies the status of requests to show.
`reqType`	`archival` | `enrollment` | `getCAChain` | `getCertificates` | `getCRL` | `getRevocationInfo` | `recovery` | `renewal` | `revocation` | `showAll` | `unrevocation` Specifies the type of request to show.
`seqNumFrom`	number Specifies the lowest request identification number to retrieve. This parameter is useful when the number of requests is more than `maxCount` and another page of data can be requested: set `seqNumFrom` to one more than the last request displayed on the current page and repost the request.
`templateName`	string Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.
`totalRecordCount`	number The total number of requests that match the criteria. This is, of course, not known until at least one page of requests have been retrieved. Requests to see more data can pass this number along so that the total number of matching requests can be displayed on every page (otherwise the total would decrease as subsequent requests used higher values for `seqNumFrom`). If this parameter is not passed, the total will be calculated on the CMS server, and the response table will include the total in the `result.header.totalRecordCount` variable.

Response

The default response template is `queryReq.template`. The base JavaScript for responses is inserted in place of the `<CMS_TEMPLATE>` tag.

For each request returned, a record object is created. Some of the record object fields listed below may not apply to some requests; for example, a pending or rejected enrollment request will have no certificate subject.

The Requests Query interface adds the JavaScript variables listed in the following table.

Table 6-34 Variables Returned by the Requests Query Interface

Variable	Description
result.header variables	Variables added to the header object.
`authorityId`	ca | kra | ra
	The type of server that generated the list of requests: ca for Certificate Manager, kra for Data Recovery Manager, or ra for Registration Manager.
`currentRecordCount`	number
	The number of request records displayed on the current page of output.
`error`	message
	If there was an error while processing the revocation request, the error message is stored in this variable. Otherwise, the value is `null`.
`querySentinel`	number
	A tracking number that indicates the default number of records to retrieve on the next page of output. This number is the lesser of the maxCount requested and the total number of records left in the result set.
`requestingUser`	user ID
	The user ID of the agent that requested the list of requests.
`totalRecordCount`	number
	The total number of request records that matched the criteria.
result.fixed variables	Variables added to the fixed object.
`maxCount`	number
	The maximum number of records to display on a page. This is taken from the `maxCount` request parameter.
`reqState`	`showAll` | `showCancelled` | `showCompleted` | `showInService` | `showRejected` | `showWaiting`
	Indicates the request state parameter that was used to generate the list.

Table 6-34 Variables Returned by the Requests Query Interface *(Continued)*

Variable	Description
`reqType`	`archival` | `enrollment` | `getCAChain` | `getCertificates` | `getCRL` | `getRevocationInfo` | `recovery` | `renewal` | `revocation` | `showAll` | `unrevocation`
	Indicates the request type parameter that was used to generate the response.
`seqNumFrom`	number
	The lowest request id number displayed in the current list of requests. This value is taken from the `seqNumFrom` request parameter.
result.recordSet[i] variables	Variables added to record objects in the response.
`archivedBy`	user ID
	The user ID of the agent that processed the key archival request.
`archivedOn`	number of seconds since 1 January 1970
	The time when the key was stored in the archive (for completed Data Recovery Manager requests).
`assignedTo`	user ID
	The user ID of the agent currently assigned to the request. If no agent is assigned, `assignedTo` will be `null`.
`callerName`	user ID
	The user ID of the agent that requested this list of requests.
`certType`	`ca` | `CEP-Request` | `client` | `objSignClient` | `ra` | `server` | `other`
	Indicates the type of certificate.
`createdOn`	seconds since 1 January 1970
	Indicates the time when the request was created.
`keyAlgorithm`	OID string
	The object identifier (OID) used by the archived key corresponding to this request (Data Recovery Manager requests). For example, the OID for an RSA encryption key is `"1.2.840.113549.1.1.1."`
`keyLength`	number
	The number of bits in the archived key (Data Recovery Manager requests).

Table 6-34 Variables Returned by the Requests Query Interface *(Continued)*

Variable	Description
`ownerName`	Distinguished Name (DN) string. See RFC 2253. The subject entry on the certificate corresponding to an archived encryption key (Data Recovery Manager requests only). Example: `CN=Alice Apple, UID=alice, OU=People, O=Example, C=US`
`requestType`	`enrollment` | `getCAChain` | `getCertificates` | `getRevocationInfo` | `renewal` | `revocation` | `unrevocation` Indicates the type of request for this record.
`serialNumber`	number The unique identification number for the request on a Data Recovery Manager. Registration Managers and Certificate Managers use `seqNum`.
`seqNum`	number The request identification number for this request. The request ID is unique on any instance of a server. The seqNum is used for Registration Manager and Certificate Manager Requests. Data Recovery Manager requests are indexed by the `serialNumber` field.
`state`	`VALID` | `INVALID` For Data Recovery Manager requests, the current state of the archived key.
`status`	`cancelled` | `complete` | `rejected` | `waiting` The current status of the request.
`subject`	Distinguished Name (DN) string. See RFC 2253. The subject entry on the certificate corresponding to this request (if there is one).
`updatedBy`	user ID The user ID of the agent who last updated this request.
`updatedOn`	seconds since 1 January 1970 The time when this request was last updated.

Select for Revocation Interface

Description

URI: `/ca/reasonToRevoke` or `/ra/reasonToRevoke`

Available on: Certificate Manager or Registration Manager

Function: Displays a set of certificates matching a query filter to be revoked.

The Select for Revocation Interface uses a query filter to select certificates. The response template lists these certificates in a form that allows them to be revoked using the Approve Revocation Interface.

Default Forms

By default forms that use the Select for Revocation Interface are embedded as buttons on certificate lists returned from the List Certificates Interface accessed through an agent port (`/ca/listCerts` or `/ra/listCerts`). These response are rendered using the `queryCert.template` discussed in the List Certificates Interface section.

Request Parameters

The following table lists the parameters that are used to select certificates through the Select for Revocation Interface. This is an agent interface, so the HTTP POST or GET request must use SSL client authentication with a valid agent certificate.

Table 6-35 Parameters Accepted by the Select For Revocation Interface

Parameter	Format and Description
`revokeAll`	QUERY_FILTER For information on constructing a query filter, see Table 3-17 in the End-Entity Interfaces “List Certificates Interface” section. To ensure accuracy when revoking certificates, you should use a query filter that selects each certificate by its serial number. An example value for `revokeAll` to revoke certificates with serial numbers 10 and 14 is: `(|(certRecordId=10)(certRecordId=14))`
`templateName`	string Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.
`totalRecordCount`	number The total number of certificates to select for revocation. This value is simply passed through and shows up in the response template as `result.header.totalRecordCount`.

Response

The default response template is `reasonToRevoke.template`. The base JavaScript for responses is inserted in place of the `<CMS_TEMPLATE>` tag. In addition, the Select for Revocation interface adds the JavaScript variables listed in the following table.

Table 6-36 Variables Returned by the Select For Revocation Interface

Variable	Description
result.header variables	Variables added to the header object.
`caSerialNumber`	number The decimal serial number of the Certificate Authority’s signing certificate.

Table 6-36 Variables Returned by the Select For Revocation Interface *(Continued)*

Variable	Description
`revokeAll`	QUERY_FILTER The query filter that was used in the request to select the certificates that appear in this response. An example value for `revokeAll` to revoke certificates with serial numbers 10 and 14 is: `(|(certRecordId=10)(certRecordId=14))`
`totalRecordCount`	number The value of totalRecordCount specified in the request.
`verifiedRecordCount`	number The number of revocable certificate that were actually selected based on the query criteria.
result.recordSet[i] variables	Variables added to record objects in the response.
`error`	message If a particular certificate could not be revoked, the `error` field in its `record` object will contain an error message. If this field is `null`, the certificate was revoked successfully.
`serialNumber`	number The decimal serial number of the certificate.
`subject`	string The subject distinguished name of the certificate. For example, `"CN=Jane Doe, UID=jdoe, OU=Users, O=Organization, ST=California, C=US."`
`validNotAfter`	number of seconds since 1 January, 1970 The date when the certificate expires. See the description for `issuedOn` for details on date values.
`validNotBefore`	number of seconds since 1 January, 1970 The date when the certificate became valid. See the description for `issuedOn` for details on date values.

Update CRL Interface

Description

URI: `/ca/updateCRL` or `/ra/updateCRL`

Available on: Certificate Manager and Registration Manager agent ports.

Function: Certificate Revocation Lists (CRLs) are automatically updated on a regular basis. If necessary, this interface can be used to force an update to the CRL.

Default Forms

The form `updateCRL.html`, available in the CA agent and RA agent directories, uses the Update CRL Interface. The form allows the user to select a signing algorithm for the CRL.

Request Parameters

The following table lists the parameters accepted by the Update CRL interface.

Table 6-37 Parameters Accepted by the Update CRL Interface

Parameter	Format and Description
`crlIssuingPoint`	`MasterCRL` Specifies the issuing point maintained by the CMS server handling the CRL update. In the default case, the only issuing point for all CRL information is the master CRL. If other issuing points have been configured in the Certificate Manager's configuration file, you can use the token used to define the issuing point for this parameter.
`signatureAlgorithm`	`MD5withRSA` | `SHA1withDSA` | `SHA1withRSA` Specifies the signing algorithm that should be used to sign the updated CRL. The CA signing key must match the key type (RSA or DSA) of the selected algorithm.

Table 6-37 Parameters Accepted by the Update CRL Interface *(Continued)*

Parameter	Format and Description
`templateName`	string
	Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.

Response

The default response template is `updateCRL.template`. The base JavaScript for responses is inserted in place of the `<CMS_TEMPLATE>` tag. This simple template uses the crlPublished variable to display either a success or failure message.

In addition, the Update CRL interface adds the JavaScript variables listed in the following table:

Table 6-38 Variables Returned by the Update CRL Interface

Variable	Description
result.header variables	Variables added to the header object.
`crlPublished`	`Success` | `Failure`
	Indicates whether or not the CRL was successfully updated and signed.
`error`	message
	A message explaining why the CRL update failed.

Update Directory Interface

Description

URI: `/ca/updateDir` or `/ra/updateDir`

Available on: Certificate Manager and Registration Manager.

Function: If enabled, the publishing directory is automatically updated on a regular basis. If necessary, this interface can be used to force new information to be published to the directory.

The interface allows all new information or just selected subsets (for example, only updated expired certificate information) to be published.

Default Forms

The form `updateDir.html`, available in the CA agent and RA agent directories, uses the Update Directory Interface.

Request Parameters

The following table lists the parameters accepted by the Update Directory interface.

Table 6-39 Parameters Accepted by the Update Directory Interface

Parameter	Format and Description
`expiredFrom`	number The low end of the range of serial numbers of expired certificates to be updated in the directory. For no lower bound, set this to null or omit the parameter.
`expiredTo`	number The high end of the range of serial numbers of expired certificates to be updated in the directory. For no upper bound, set this to null or omit the parameter.
`revokedFrom`	number The low end of the range of serial numbers of revoked certificates to be updated in the directory. For no lower bound, set this to null or omit the parameter.
`revokedTo`	number The high end of the range of serial numbers of revoked certificates to be updated in the directory. For no upper bound, set this to null or omit the parameter.

Table 6-39 Parameters Accepted by the Update Directory Interface *(Continued)*

Parameter	Format and Description
`templateName`	string Filename relative to the template directory (`web/ee`, `web/agent/ca`, `web/agent/kra`, or `web/agent/ra`) of a file to use as the response template. This template will be used for any response, overriding default template settings.
`updateAll`	`yes` | `no` Whether to update all information in the directory ("`yes`") or only to update selected categories ("`no`"). If "`no`," be sure to set one of `updateCA`, `updateCRL`, `updateExpired`, `updateRevoked`, or `updateValid` to "`yes`."
`updateCA`	`yes` | `no` Whether or not to update the Certificate Manager's signing certificate in the directory.
`updateCRL`	`yes` | `no` Whether or not to update the certificate revocation list (CRL) in the directory. Any new updates since the last automatic or manual update will be published.
`updateExpired`	`yes` | `no` Whether or not to remove certificates from the directory that have expired since the last update. If you want to restrict the range of certificates (by serial number), specify values for `expiredFrom`, `expiredTo`, or both.
`updateRevoked`	`yes` | `no` Whether or not to remove certificates from the directory that have been revoked since the last update. If you want to restrict the range of certificates (by serial number), specify values for `revokedFrom`, `revokedTo`, or both.
`updateValid`	`yes` | `no` Whether or not to publish certificates that have been issued (or otherwise become valid) since the last update to the directory. If you want to restrict the range of certificates (by serial number), specify values for `validFrom`, `validTo`, or both.
`validFrom`	number The low end of the range of serial numbers of valid certificates to be updated in the directory. For no lower bound, set this to null or omit the parameter.
`validTo`	number The high end of the range of serial numbers of valid certificates to be updated in the directory. For no upper bound, set this to null or omit the parameter.

Response

The default response template is `updateDir.template`. The base JavaScript for responses is inserted in place of the `<CMS_TEMPLATE>` tag. In addition, the Update Directory interface adds the JavaScript variables listed in the following table.

A variable will not be added (it will have a `null` value) if it does not apply; for example, if updating the CRL was not requested, the `crlPublished` variable will not be present (`crlPublished == null` will evaluate to `true`).

Table 6-40 Variables Returned by the Update Directory Interface

Variable	Description
result.header variables	Variables added to the header object.
`caCertError`	string A message explaining why the CA certificate could not be published to the directory, if there was an error.
`caCertPublished`	`Success` | `Failure` If updating the CA certificate was requested, this variable will indicate whether the update was successful or not. See `caCertError` for an error message in case of `Failure`.
`crlError`	string A message explaining why the CRL could not be published to the directory, if there was an error.
`crlPublished`	`Success` | `Failure` If updating the CRL was requested, this variable will indicate whether the update was successful or not. See `crlError` for an error message in case of `Failure`.
`expiredCertsError`	string A message explaining why the expired certificates could not be removed from the directory, if there was an error.
`expiredCertsUnpublished`	`Success` | `Failure` If removing expired certificates was requested, this variable will indicate whether the update was successful or not. See `expiredCertsError` for an error message in case of `Failure`.
`revokedCertsError`	string A message explaining why the revoked certificates could not be removed from the directory, if there was an error.

Table 6-40 Variables Returned by the Update Directory Interface *(Continued)*

Variable	Description
`revokedCertsUnpublished`	`Success` | `Failure` If removing revoked certificates was requested, this variable will indicate whether the update was successful or not. See `revokedCertsError` for an error message in case of `Failure`.
`validCertsError`	A message explaining why new certificates could not be published to the directory, if there was an error.
`validCertsPublished`	`Success` | `Failure` If publishing new certificates was requested, this variable will indicate whether the update was successful or not. See `validCertsError` for an error message in case of `Failure`.

Index

A

Agent Services interface, 93
  Approve Revocation, 101
  Bulk Enrollment, 105
  Display Key By Serial Number, 115
  Display Key For Recovery, 118
  Examine Recovery, 120
  for Certificate Manager agents, 94
  for Data Recovery Manager agents, 96
  for Registration Manager agents, 95
  Get Approval Status, 122
  Get PKCS#12 Data, 125
  Grant Recovery, 126
  how to access, 97
  Key Query, 128
  KeyRecovery Query, 133
  Process Certificate Request, 138
  Process DRM Request, 147
  Process Request, 150
  Recover Key By Serial Number, 153
  Remove Certificate Hold, 156
  Requests Query, 159
  Select for Revocation, 164
  Update CRL, 167
  Update Directory, 168
  who can access, 94
agents
  forms for, 97
  locating forms and templates for, 98
Approve Revocation, 101

B

bulkissuance, 105

C

CEP, 41
Certificate Chain
  get from CA
certificate chains
  getting, 35
certificate enrollment
  supported authentication mechanisms, 30
  supported request formats, 30
Certificate Enrollment Protocol Interface, 41
Certificate Manager
  enrollment forms for, 33
  interface for agents, 94
certificate renewal
  supported authentication mechanisms, 30
  supported request formats, 30
certificate request formats, 30
  for enrollments, 30
  for key archival and recovery, 30
  for renewals, 30
  for revocations, 30
certificate revocation
  supported authentication mechanisms, 30
  supported request formats, 30
certificate revocation list

manual update, 167
to retrieve, 69
Challenge Revocation Interface, 42
challenge_revocation1, 42
CMS_TEMPLATE tag, 20
conventions used in this book, 11

D

Data Recovery Manager
interface for agents, 96
display
<Italic>See retrieve
Display Certificate By Serial Number, 44
Display Certificate From Request, 46
Display Key For Recovery, 118
displayBySerial
key for recovery, 115
displayBySerialForRecovery, 118
displayCertFromRequest
documentation
conventions followed, 11
where to find, 12
doRevoke, 101
doUnrevoke, 156

E

end entities
forms provided for, 27
locating forms and templates, 31
supported request formats, 30
end-entity forms, 31
for enrollment, 32
for renewal, 33
for retrieval, 34
for revocation, 34
End-entity Interface
Certificate Enrollment Protocol, 41
Challenge Revocation, 42
Display Certificate By Serial Number, 44
Display Certificate From Request, 46
Enrollment, 49
Get CA Chain, 59
Get Certificate By Serial Number, 61
Get Certificate From Request, 65
Get CRL, 69
List Certificates, 72
Renewal, 80
Revocation, 82
end-entity templates, 36
Enrollment, 49
enrollment
approval, 138
bulk issuance, 105
list queued requests, 159
enrollment forms
for Certificate Managers, 33
for end users, 32
for object signing certificates, 33
for Registration Managers, 33
for servers, 32
Examine Recovery, 120
examineRecovery, 120

F

fonts used in this book, 11
forms
<Italic>See HTML forms

G

Get CA Chain, 59
Get Certificate By Serial Number, 61
Get Certificate From Request, 65
Get CRL, 69
Get PKCS#12 Data, 125
getApprovalStatus, 122
getBySerial
getCAChain

getCertFromRequest
getCRL, 69
getPk12, 125
Grant Recovery (DRM interface), 126
grantRecovery, 126

H

HTML forms
  for agents, 93, 97, 98
  for end entities, 27, 31
    for enrollment, 32
    for renewal, 33
    for retrieval, 34
    for revocation, 34

K

key archival
  required format for requests, 30
Key Query (DRM Interface), 128
key recovery
  archive request approval, 147
  by serial number, 115, 118
  check request, 120
  find by serial number, 153
  grant approval, 126
  list keys, 128, 133
  PKCS #12 data, 125
  request status, 122
Key Recovery Query (DRM Interface), 133

L

List Certificates, 72
list of
  agent forms and templates
  end-entity forms and templates
listCerts, 72
location of
  agent forms, 98
  CMS documentation, 12
  end-entity forms, 31

O

object signing certificates
  how to enroll for, 33
output templates
  for end-entity operations, 36

P

pending requests
  list, 159
PKCS #12
  key recovery, 125
pkiclient.exe, 41
Process Certificate Request, 138
Process DRM Request, 147
Process Request, 150
processCertReq, 138
processReq, 150
processReq (DRM), 147
publishing
  manual directory update, 168

Q

queryKey, 128
queryKeyForRecovery, 133
queryReq, 159

R

reasonToRevoke, 164

Recover Key By Serial Number, 153

recoverBySerial (DRM), 153

Registration Manager
- enrollment forms for, 33
- interface for agents, 95

Remove Certificate Hold, 156

renew certificates, 80

Renewal (interface), 80

request formats for certificates, 30

Requests Query, 159

retrieve certificate
- by list, 72
- by request number, 46, 65
- by serial number, 44, 61
- CA certificate chain, 59

retrieve certificate revocation list, 69

revocation
- agent approval, 164
- agent approval interface, 101
- challenge-phrase based, 42
- remove certificate hold, 156
- using SSL client authentication, 82

Revocation (interface), 82

S

Select for Revocation, 164

server enrollment forms, 32

subordinate CA
- enrollment forms for, 33

Sun ONE, 9

T

templates
- for agents
  - location, 98
- for end entities
  - location, 31
- for end-entity operations, 36

type styles used in this book, 11

U

unrevocation, 156

Update CRL, 167

Update Directory (interface), 168

updateCRL, 167

updateDir, 168

user enrollment forms, 32