DigiCert® PKI Platform

Web Services Developer’s Guide

Version 8.22. 6

Legal Notice

DigiCert, Inc.
2801 North Thanksgiving Way, Suite 500
Lehi, UT 84043
https://www.digicert.com

Table of Contents

CHAPTER 1

1 DigiCert PKI Web Service Overview

This chapter includes the following topics:

- Introduction
- About DigiCert PKI Web Services

1.1 Introduction

DigiCert PKI is a public key infrastructure (PKI) platform. DigiCert PKI protects the confidentiality and integrity of electronic communications using digital certificates, public-key cryptography, and Certification Authorities (CA) to create an enterprise-wide network security architecture.

This guide is designed for developers who integrate DigiCert's® PKI certificate lifecycle tasks into their RA (Registration Authority) applications. DigiCert PKI Web Services provides the ability to use an RA application to obtain a certificate policy, enroll for a certificate, and renew a certificate, and suspend and resume a certificate. DigiCert PKI Web Services also includes solutions for escrowing private keys and requesting recovery of the keys in the event they are lost, stolen, or compromised.

NOTE: The sample code and utilities that are provided in this guide are meant as examples only. Use them to guide you in developing your own solution to integrate PKI Web Services into your RA application.

1.2 About DigiCert PKI Web Services

DigiCert PKI Web Services lets you integrate DigiCert's certificate issuance and administration tasks into your RA applications. DigiCert's PKI consists of the following protocols:

- Certificate Enrollment Policy Protocol. This protocol allows the RA application to obtain the certificate enrollment policy of the certificate profiles that are configured for the account.
- Certificate Enrollment Protocol. This protocol allows the RA application to enroll for a new certificate, or renew an expiring one.
- Certificate Management Protocol. This protocol allows the RA application to search for, revoke, and recover an existing certificate.
- User Management Protocol. This protocol allows the RA application to create and view users and manage enrollment codes.

- Health Check Protocol. This protocol allows the RA application to check the status of different operations.

These protocols support multiple operations and functions. See Table 1-1 for these operations.

Table 1-1 DigiCert PKI protocol descriptions

Protocol Name	Operations Supported	Functions Supported	HTTP endpoint
Certificate Enrollment Policy Protocol	RequestPolicy	Obtain policy	https://pki-ws.symauth.com/pki-ws/policyService?wsdl
Certificate Enrollment Protocol	RequestSecurityToken	• Enroll for certificate • Renew certificate	https://pki-ws.symauth.com/pki-ws/enrollmentService?wsdl (Optional) If using key escrow and recovery service, go to https://<KeyEscrowServer:port>/ digicert-escrow-recovery-service/ enrollmentService?wsdl (for local certificate enrollment or renewal only)
Certificate Management	updateCertificateStatus	Revoke, suspend, or resume certificate	https://pki-ws.symauth.com/pki-ws/certificateManagementService?wsdl (Optional) If using key escrow and recovery service, go to https://<KeyEscrowServer:port>/ digicert-escrow-recovery-service/ certificateManagementService?wsdl (for local certificate recovery only)
	recoverCertificate	Recover certificate	
	searchCertificate	Search for certificates	
	bulkUpdateCertificateStatus	Revoke multiple certificates	

Protocol Name	Operations Supported	Functions Supported	HTTP endpoint
User Management	createOrUpdateUser	• Create user • Modify user	https://pki-ws.symauth.com/pki-ws/userManagementService?wsdl
	createOrUpdatePasscode	• Generate enrollment code • Replace enrollment code	
	getUserInformation	Obtain user data and all valid certificates	
Health Check	getStatus	Obtain the health status for the different operations.	https://pki-ws.symauth.com/pki-ws/healthcheckService?wsdl

Table 1-2 DigiCert PKI WSDL names

HTTP endpoint	WSDL Name
https://pki-ws.symauth.com/pki-ws/policyService?wsdl	CertificateEnrollmentPolicy.wsdl
https://pki-ws.symauth.com/pki-ws/enrollmentService?wsdl	VS_WSTEP.wsdl
https://pki-ws.symauth.com/pki-ws/certificateManagementService?wsdl	CertificateManagementService.wsdl
https://pki-ws.symauth.com/pki-ws/userManagementService?wsdl	UserManagementService.wsdl
https://pki-ws.symauth.com/pki-ws/healthcheckService?wsdl	HealthcheckService.wsdl

Figure 1-1 illustrates the workflow for these DigiCert PKI Web Service protocols.

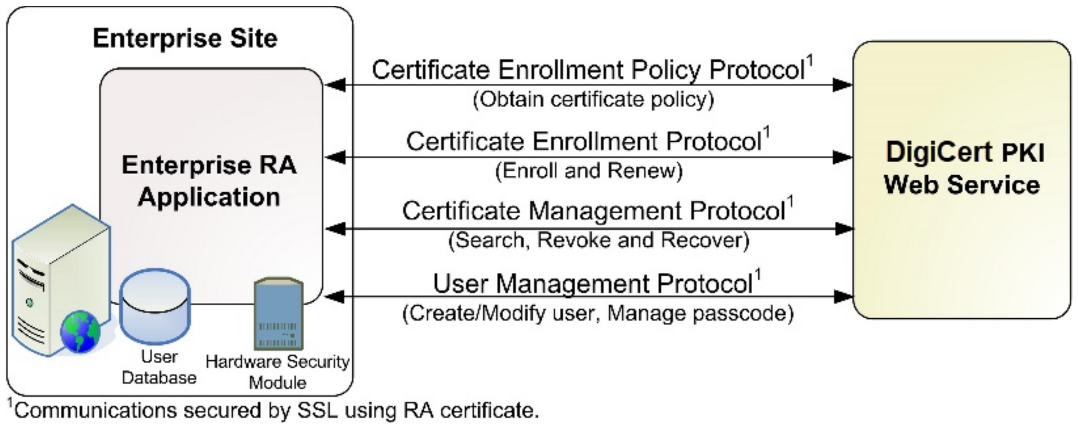

Enterprise Site
Enterprise RA Application
User Database
Hardware Security Module
Certificate Enrollment Policy Protocol[1]
(Obtain certificate policy)
Certificate Enrollment Protocol[1]
(Enroll and Renew)
Certificate Management Protocol[1]
(Search, Revoke and Recover)
User Management Protocol[1]
(Create/Modify user, Manage passcode)
DigiCert PKI Web Service
[1]Communications secured by SSL using RA certificate.

Figure 1-1 DigiCert PKI Web Service workflow

Figure 1-2 illustrates the workflow for these DigiCert PKI Web Service protocols, including the key escrow and recovery service.

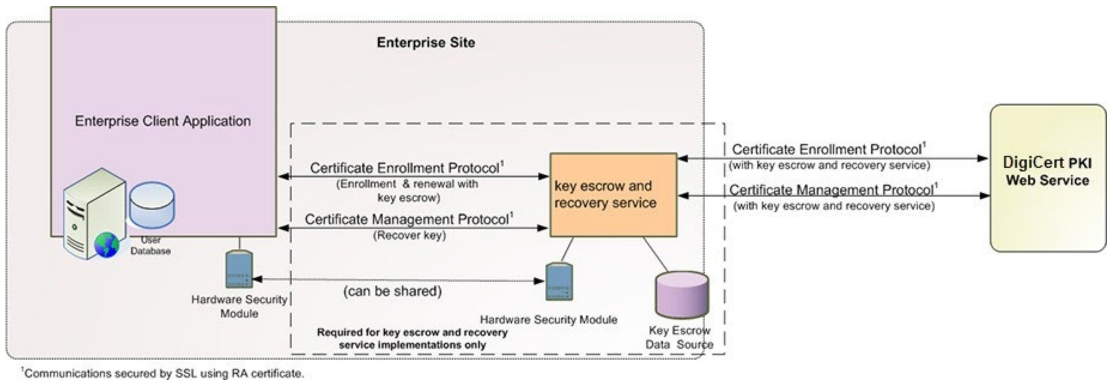

Enterprise Site
Enterprise Client Application
User Database
Hardware Security Module
Certificate Enrollment Protocol[1]
(Enrollment & renewal with key escrow)
Certificate Management Protocol[1]
(Recover key)
(can be shared)
key escrow and recovery service
Hardware Security Module
Key Escrow Data Source
Required for key escrow and recovery service implementations only
Certificate Enrollment Protocol[1]
(with key escrow and recovery service)
Certificate Management Protocol[1]
(with key escrow and recovery service)
DigiCert PKI Web Service
[1]Communications secured by SSL using RA certificate.

Figure 1-2 DigiCert PKI Web Service workflow with key escrow and recovery service

1.2.1 About the Certificate Enrollment Policy Protocol and the Certificate Enrollment Protocol

The Certificate Enrollment Policy Protocol allows your RA application to obtain the certificate policy that you configured when you set up your DigiCert PKI account. This policy determines the content and behavior of the certificates your account issues. The Certificate Enrollment Protocol can use the policy that is returned to create a valid certificate enrollment or renewal request.

1.2.1.1 Certificate Enrollment Policy Protocol and Certificate Enrollment Protocol Workflow

The Certificate Enrollment Policy Protocol and the Certificate Enrollment Protocol work together. They work together to allow an end user to enroll for a certificate that matches the certificate policy appropriate to the end user. Figure 1-3 illustrates this relationship.

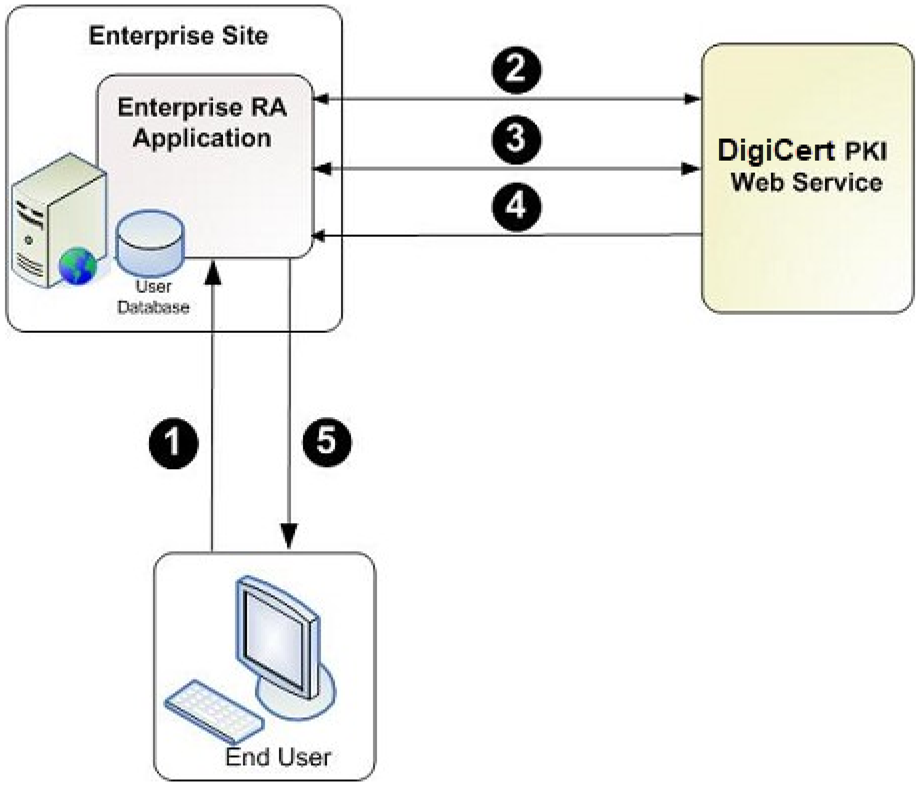


Figure 1-3 Certificate enrollment using the Certificate Enrollment Policy and Certificate Enrollment Protocols

1. The end user accesses the enterprise RA application to initiate a request for a new or a renewed certificate.
2. The RA application initiates a Certificate Enrollment Policy Protocol (getPolicies) request to DigiCert.
3. The RA application uses the certificate policy information to construct an enrollment request. The enrollment request includes the appropriate certificate enrollment data based on the policy for the end user.

   For example, the RA application may parse the certificate policy information and dynamically generate an enrollment page which the end user completes and submits. The end user's private key is generated at the end user's computer.

4. The RA application submits the request for a certificate, and then returns the certificate to the RA application. If the key escrow store is set to DigiCert, DigiCert PKI generates the private key. If the key escrow store is set to a local user store, the key escrow and recovery service generates the private key.
5. The RA application provides the certificate to the end user.

Once the certificate policy information is returned (Step 2), it can be cached. If so, future enrollments where the certificate policy or account features have not changed will not require Step 2.

1.2.2 About the Certificate Management Protocol

The Certificate Management Protocol provides the following operations that you need to perform the daily tasks of managing the certificate:

- Searching for certificate by specified search criteria, such as seat ID, account ID, profile ID, email address, and so on.
- Revoking the certificates that have been compromised or stolen.
- Recovering the certificates that have been lost or unusable (requires that the key escrow option be implemented).

1.2.3 About the User Management Protocol

Before you issue a certificate to a PKI Web Services user, the user must exist in your DigiCert PKI account. Additionally, users require an enrollment code to pick up their certificates. This enrollment code verifies that the users are authorized to pick up their certificates.

- A seat ID is a unique value assigned to identify the user in the DigiCert PKI account. Users are created with a specific seat ID under the DigiCert PKI account. The User Management Protocol allows your RA application to create or modify users.
- Enrollment codes are assigned to a specific user (by the user's seat ID) and certificate profile (by certificate profile OID). If your Authentication method is set to enrollment code in PKI Manager, the User Management Protocol allows your RA application to create or modify users. The User Management Protocol also allows management (generate or replace) of enrollment codes.

Once the user exists in your account, the User Management Protocol lets you obtain information about the user. The User Management Protocol also lets you obtain the valid certificates that are issued to the user.

1.2.4 About the Health Check Protocol

The Health Check Protocol provides you with a "getStatus" operation that takes an operation type as input and returns a consolidated status code as "UP" or "DOWN". Currently it supports only the enroll operation and checks all dependencies required for enroll operation.

CHAPTER 2

2 Getting Started with PKI Web Services

This chapter includes the following topics:

2.1 About PKI Web Services

This section provides information you need to get started with PKI Web Services, including:

- See “Sample Java Application and Java Utilities” on page 16. This section describes the sample RA applications and Java utilities you can use to test your DigiCert PKI account and gain an understanding of the main Web Service operations.
- See “Getting Started Workflow” on page 17. This section lists the general steps you follow to set up the DigiCert PKI to begin integrating DigiCert's PKI protocols into your RA application.
- See “Pre-requisites” on page 20. This section lists the pre-requisites you need to complete to begin working with DigiCert's PKI Web Services protocols.
- See “Recommended Skill Sets” on page 20. This section lists the recommended skill sets for integrators working with DigiCert's PKI Web Services protocols.
- See “Obtaining Your Registration Authority Certificate” on page 21. This section describes the steps you need to follow to get a Registration Authority (RA) certificate.

2.2 Sample Java Application and Java Utilities

The DigiCert PKI Web Service package includes the following resources. These resources are to help you test your DigiCert PKI account. They also help you gain an understanding of the main Web Service operations. The package is preparatory to developing your own RA application.

- DigiCert PKI Java utility: Use this utility to perform certificate lifecycle and user management tasks. Tasks include getpolicy, enroll, search, revoke, recover, and so on. These tasks are for testing and troubleshooting purposes. The utility is designed to require minimal configuration and set-up. However, it is designed only to demonstrate the functionality of PKI Web Service. It should not be used in place of your own, integrated RA application.

  See "About the PKI Web Service Java Utility" on page 24.

- DigiCert PKI Java sample code: The DigiCert PKI Java sample code that you can compile and run to perform Web Service calls against your DigiCert PKI account. Use this sample Java application to understand how the DigiCert PKI Web Service behaves. Understanding its behavior can help you can better integrate it with your specific PKI solution.

  See "About the PKI Web Service Sample Java Code" on page 31.

2.3 Getting Started Workflow

Figure 2-1 describes the general steps that are required to set up the DigiCert PKI. It also describes how to integrate DigiCert PKI into your RA application using the DigiCert PKI protocols.

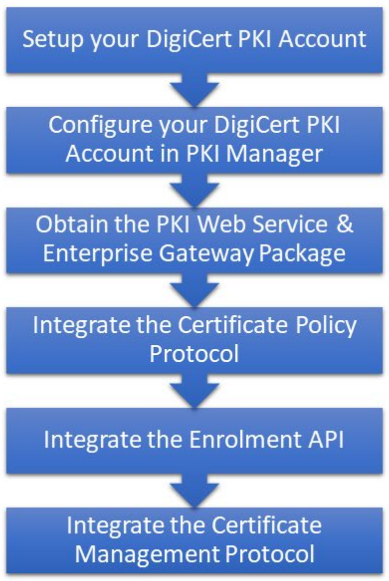

Setup your DigiCert PKI Account
Configure your DigiCert PKI Account in PKI Manager
Obtain the PKI Web Service & Enterprise Gateway Package
Integrate the Certificate Policy Protocol
Integrate the Enrolment API
Integrate the Certificate Management Protocol

Figure 2-1 DigiCert PKI getting started workflow

2.3.1 Task 1. Set up your DigiCert PKI account

Contact your DigiCert Sales representative to set up your DigiCert PKI account and get the RA application integration process started. Your representative can provide you with the necessary documents to begin defining your account and your certificate profile.

See "Pre-requisites" on page 20.

2.3.2 Task 2. Configure your DigiCert PKI account in PKI Manager

Configure your certificate profiles using PKI Manager. When defining your certificate profile, consider the following:

- PKI Manager only lets you configure the certificate profiles that support Web Services.
- By default, email notifications are not enabled for PKI Web Services certificate profiles. However, you can enable and customize email notifications under **Customize certificate notifications** on the **Manage profiles** page.

Refer to PKI Manager and the online Help for details on configuring DigiCert PKI for PKI Web Services.

2.3.3 Task 3. Obtain the PKI Web Service and PKI Enterprise Gateway Package

Obtain the PKI Web Service package from the **Resources** page of PKI Manager. If you configure the key escrow and recovery service, obtain the PKI Enterprise Gateway package from the **Resources** page of PKI Manager.

Obtain the PKI Web Service package and the PKI Enterprise Gateway package from the **Resources** page of PKI Manager.

2.3.4 Task 4. Integrate the Certificate Enrollment Policy Protocol

The Certificate Enrollment Policy Protocol obtains all of the certificate profiles that are configured for your account and CA. You can set your enterprise RA application to call the protocol to get the policy information each time an end user requests a certificate. Or, configure your RA application to call the protocol initially and cache the results for later enrollments.

See "About the Certificate Enrollment Policy Protocol" on page 91.

2.3.5 Task 5. Integrate the Certificate Enrollment Protocol

The Certificate Enrollment Protocol uses the certificate profile information. The Certificate Enrollment Policy Protocol returns the certificate profile information so you can craft an enrollment request for your users. For example, the enrollment request may take the form of an enrollment page. The enrollment page prompts the user for the correct enrollment data. It also requests that the certificate type for the end user is based on the certificate profile.

Once a valid request is submitted, DigiCert PKI automatically approves the request. DigiCert PKI then issues the certificate to your enterprise RA application.

See “About the Certificate Enrollment Protocol” on page 134.

2.3.6 Task 6. Integrate the Certificate Management Protocol

Use the Certificate Management Protocol to perform the day-to-day tasks of managing the end-user certificates. Tasks include searching for certificates, revoking certificates, suspending certificates, resuming certificates, and recovering private keys.

See “About the Certificate Management Protocol” on page 159.

2.3.7 Task 7. Integrate the User Management Protocol

Use the User Management Protocol to perform the day-to-day tasks of creating and modifying end users, managing (generating and replacing) enrollment codes, and obtaining information (and valid certificates, if any) for end users.

See “About the User Management Protocol” on page 178.

2.3.8 Task 8. Integrate the Health Check Protocol

Use the Health Check Protocol to check status details for different operations.

See “About the Health Check Protocol” on page 198.

2.4 Pre-requisites

To begin using DigiCert PKI, you need to complete the following:

- You need to complete and return the following documents. Your DigiCert representative can assist you with obtaining and completing these forms.
    - Master Service Agreement
    - Issuing Authority Naming Application (also known as the CA Naming Document)
    - DigiCert Services Order Form
    - Purchase Order, credit card, or reference number
- You need to obtain your initial DigiCert PKI administrator ID, which is your credential to access your DigiCert PKI account. Your DigiCert representative can assist you with obtaining your DigiCert PKI administrator ID.

  Use your DigiCert PKI administrator ID to log into PKI Manager, configure your DigiCert PKI account, and obtain your RA certificate. Refer to PKI Manager and its online Help for details on configuring DigiCert PKI.
- You need to obtain and install a Registration Authority (RA) certificate to communicate to the DigiCert PKI. See "Obtaining Your Registration Authority Certificate" on page 21.
- (Optional) If you configure the key escrow and recovery service, you need to install and configure PKI Enterprise Gateway. Refer to *DigiCert PKI Platform Enterprise Gateway Deployment Guide* for instructions.

2.5 Recommended Skill Sets

DigiCert recommends that developers who integrate DigiCert's certificate lifecycle DigiCert PKIs have the following skill sets:

- Strong knowledge of SOAP, Web Services, and Web Services client stacks
- Strong knowledge of public key infrastructure (PKI)
- Strong knowledge of cryptographic API tools such as Microsoft CAPI, BouncyCastle, and OpenSSL. Specifically, developers need to know how to use these tools to generate and use keys if storing the RA certificate in a software keystore.

  If storing the RA certificate in a hardware security module (HSM), developers must know how to use the HSM tools to generate and use keys.
- Fundamental knowledge of XML

2.6 Obtaining Your Registration Authority Certificate

You need a Registration Authority (RA) certificate to secure communications and identify yourself to DigiCert PKI. In communications with DigiCert PKI, the RA certificate is used as a TLS/SSL client authentication certificate.

You have the option of storing your RA certificate in a software keystore or on an HSM. The method you choose in storing your RA certificate has implications on how you obtain your RA certificate.

NOTE: DigiCert recommends the use of a hardware security module to ensure the security of the RA certificate and its corresponding private key. Securing your RA certificate and private key are important because anyone who has access to them can act on your organization's behalf.

If you test your deployment or DigiCert PKI profile configuration, you can use the Microsoft certificate store rather than an HSM. However, you must use an HSM if you deploy DigiCert PKI to issue production certificates.

If you use the sample Java code, you must store your RA certificate in a software keystore.

See “About the PKI Web Service Sample Java Code” on page 31.

- See “Obtaining an RA Certificate to Store in a Java Keystore File” on page 21. This section describes how to obtain an RA certificate if you store your RA certificate in a software-based Java keystore.
- See “Obtaining an RA Certificate to Store in an HSM” on page 23. This section describes how to obtain an RA certificate if you store your RA certificate in an HSM.

2.6.1 Obtaining an RA Certificate to Store in a Java Keystore File

Complete the following steps if you store your RA certificate in a software-based Java keystore. These procedures require the Java keytool to generate the keys and import them into your keystore.

DigiCert recommends that you use strong passwords and store them in a secure location. A strong password is comprised of six or more characters with a numbers and upper- and lower-case letters.

1. Generate a key pair as follows:

   ```
   keytool -genkey -alias pki_ra -keyalg RSA -keysize 2048 -sigalg
   SHA256withRSA -dname "CN=<common name>" -keypass <password> -keystore
   <keystore name> -storepass <password>
   ```

2. Generate a Certificate Signing Request (CSR) as follows:

   ```
   keytool -certreq -alias pki_ra -sigalg SHA256withRSA -file
   pki_raCSR.req -keypass <password> -keystore <keystore name>
   -storepass <password>
   ```

3. Copy the pki_raCSR.req file to the computer that has access to the PKI Manager.
   - Open pki_raCSR.req in a text editor and copy the contents of the file into your clipboard.
   - Access PKI Manager and select **Get RA certificate** from the **Tasks** icon at the bottom of the screen.
   - Paste the contents of your clipboard in to the request field. Click **Submit**.
   - When you are prompted, download the resulting cert.p7b certificate file.
4. Copy the cert.p7b certificate file to a temporary directory on the computer where the key pair was generated.
5. Import the certificate into your keystore by using the following command:

   ```
   keytool -import -alias pki_ra -file cert.p7b -noprompt -keypass
   <password> -keystore <keystore name> -storepass <password>
   ```

6. The root and issuing CAs for the RA certificate can be found in the web page at https://knowledge.digicert.com/solution/ca-hierarchies-for-production-and-test-drive-ra-certificates.html. You need to import the CAs as trusted root CAs into the keystore using the appropriate command. The appropriate command ensures that the RA certificate you install is correctly trusted.
   - For intermediate CAs:

     ```
     keytool -import -trustcacerts -alias pki_ca -file
     RAintermediateCA.cer -keystore <keystore name>
      -storepass <password>
     ```

   - For root CAs:

     ```
     keytool -import -trustcacerts -alias root -file RAroot.cer
      -keystore <keystore name> -storepass <password>
     ```

2.6.2 Obtaining an RA Certificate to Store in an HSM

Complete the following steps if you store your RA certificate in an HSM.

1. Generate a CSR file on the HSM according to the vendor's instructions. The CSR file must meet the following additional requirements:
   - The key algorithm must be RSA
   - The key size must be 2048-bit
2. Copy the CSR file to the computer that has access to the PKI Manager.
   - Open pki_raCSR.req in a text editor and copy the contents of the file into your clipboard.
   - Access PKI Manager and select **Get RA certificate** from the **Tasks** icon at the bottom of the screen.
   - Paste the contents of your clipboard in to the request field. Click **Submit**.
   - When you are prompted, download the resulting cert.p7b certificate file.
3. Copy the cert.p7b certificate file to a temporary directory on the computer where the key pair was generated.
4. Import the certificate into your HSM according to your vendor's instructions.

   NOTE: For SafeNet HSMs, contact SafeNet Customer Support and request document ID 17271 to obtain these instructions.

5. You may need to also import the root CA certificate. Refer to the vendor's documentation to determine if it is necessary and how to import the certificate. The root and issuing CAs for the RA certificate can be found in the web page at https://knowledge.digicert.com/solution/ca-hierarchies-for-production-and-test-drive-ra-certificates.html.

CHAPTER 3

3 PKI Web Service Java Utility

This chapter includes the following topics:

3.1 About the PKI Web Service Java Utility

DigiCert PKI includes a command-line Java utility that you can use to perform transaction signing and operations certificate lifecycle and user management tasks for testing and troubleshooting purposes. This utility is designed to require minimal configuration and set-up.

3.2 Securing Communications to DigiCert

Communications between this utility and DigiCert are secured using an SSL connection. An RA certificate authenticates the connection. You must already have obtained and installed an RA certificate to use this utility. See "Obtaining Your Registration Authority Certificate" on page 21.

3.3 Installing and Running the Utility

1. Navigate to the sampleClient/tools/clientApp folder of the PKI Web Service package (available from the **Resources** page of PKI Manager). This folder contains the following files:

   NOTE: The folder contains the additional files that are required for PKI Web Services only.

   - pkiwebserviceclient.jar
   - sample input.txt
   - sample client configuration files (see Table 3-3)
   - sample log4j.properties
   - sample cacerts truststore

- lib directory containing crypto-api.jar and the third-party subdirectory.
- lib or third-party subdirectory containing appropriate third-party jar and license files.

2. Enter the following command to run the DigiCert PKI Java utility:

```
java -jar pkiwebserviceclient.jar -config <config file>
-operation <operation> [-input <input file>]
```

Where:

- <operation> is the certificate task to perform. Refer to Table 3-2 for a list of the supported operations.
- <config file> is the path to the DigiCert PKI utility configuration file. The utility requires the information in the file to perform the operation. See "DigiCert PKI Java Utility Configuration File" on page 28.
- <input file> is the path to the DigiCert PKI utility input file. This file is required only for the enroll and key recovery operations.
  This file contains the name-value pairs that are required for enrollment. Any name-value pairs that are part of the policy in the response to the getpolicy or enrollWithPolicy operation can be defined in this file.

The utility also supports the following additional options:

Table 3-1 Additional DigiCert PKI utility options

Flag	Value	Description
-dumprequest	true|false	Displays the content of the request being sent.
-help	N/A	Displays the available command line arguments.
- kms	N/A	This option has been deprecated.
-version	N/A	Displays the version of the utility.

3.4 Supported Operations

Table 3-2 DigiCert PKI Java utility supported operations

Operation	Description
getpolicy	Uses the Certificate Enrollment Policy Protocol to obtain the policy and prints it out to the console.
enroll	Uses the Certificate Enrollment Protocol to enroll for a certificate. Requires an input file listing all of the name-value pairs to put into the enrollment request.
searchCertificate	Uses the Certificate Management Protocol to obtain certificate information. If your search criteria find a large number of matches, only the maximum configured number of results are returned. The Java utility has a 10-minute timeout period for search operations. You should use make yur searches as exact as possible to avoid having the operation timeout.
certmgmt	Uses the Certificate Management Protocol to perform certificate management tasks based on the unique user identifier (seat ID) or the certificate serial number: • revoke. Permanently invalidates the certificate. This operation is not reversible. • recover. Recovers a certificate. • suspend. Temporary revocation of certificates. • resume. Resumes the suspended certificates. • revokeBulk. Permanently invalidates a list of certificates. This operation is not reversible.

Operation	Description
createUser	Uses the User Management Protocol to upload data for a user or users into DigiCert PKI. You can upload any or all of the following values. However, a user must have a unique identifier (Seat ID). • Seat ID (Required) • First name • Last name • Email address • Work or mobile telephone numbers • The other attributes that you define and should appear in the certificate If the user already exists in the system, this operation adds to or updates the user information using the data that was provided in the configuration file.
getUserInformation	Uses the User Management Protocol to obtain information about a user specified (by the seatID). Obtains any valid certificates that are assigned to the user. This operation returns all of the user data previously uploaded using the createOrUpdateUser (or createOrUpdatePasscode) call, or uploaded directly using PKI Manager.
createPasscode	Uses the User Management Protocol to assign a passcode to a specified user or users. At minimum, the configuration file for this operation must contain the Seat ID and profile OID for all users to which a passcode is assigned. If you provide a passcode value along with the Seat ID, the passcode is assigned to the specified user. If you leave the passcode value blank, the system generates a passcode, assigns it to the specified user, and prints the passcode to the console.

Operation	Description
	If a passcode already exists for a user, the existing passcode is overwritten by the new passcode.
getPasscodeInformation	Uses the User Management Protocol to obtain information about the passcode for the specified user. The configuration file for this operation must include the Seat ID and profile OID for each user for which passcode information should be returned.
getStatus	Uses the Health Check Protocol to get the health information for different operations. At present, only Enroll operation is supported.

3.5 DigiCert PKI Java Utility Configuration File

The Java utility requires a configuration file to specify the behavior of the utility. The file you use depends upon the operation you perform:

Table 3-3 Java utility configuration files

Configuration File	Description
clientConfig.txt.keyEscrow.enroll	Enroll for a certificate when the private key has been escrowed.
clientConfig.txt.nonKeyEscrow.enroll	Enroll for a certificate when the private key has not been escrowed.
clientConfig.txt.keyEscrow.renew	Renew a certificate when the private key has been escrowed.
clientConfig.txt.nonKeyEscrow.renew	Renew a certificate when the private key has not been escrowed.
clientConfig.txt.search	Return information for a specified certificate or certificates.
clientConfig.txt.allPolicies	Get all policies for this account.
clientConfig.txt.onePolicy	Get a specific policy for this account.

Configuration File	Description
clientConfig.txt.recover	Recover an escrowed private key.
clientConfig.txt.revoke	Revoke a certificate
clientConfig.txt.revokeBulk	Revoke certificates in bulk by Seat IDs or certificate serial numbers.
clientConfig.txt.suspend	Suspend a certificate.
clientConfig.txt.resume	Resume a suspended certificate.
clientConfig.txt.createPasscode	Assign an enrollment code to a user or users.
clientConfig.txt.createUser	Create or modify a user.
clientConfig.txt.getPasscodeInformation	Get information about the enrollment code assigned to a user.
clientConfig.txt.getUserInformation	Get information about a user previously added to the system.
clientConfig.txt.getStatus	Get health status of specified operation.

You need to either modify one of the configuration files that are provided with this utility, or create a configuration file manually.

- If you modify the configuration file, edit the values that are marked as TO_BE_REPLACED to match your environment. Refer to the comments in the sample file for additional information.
- By default, the non-key escrow enrollment configuration file sends your certificate enrollment request in PKCS#10 format. If your request is in SPKAC format, uncomment the line that reads:

  ```
  #enroll.csr_value_type=spkac
  ```

- Do not use the percentage character (%) or URL encoding for values in name and value pairs. For example, use Acme Bank rather than Acme%20Bank.

- The DigiCert PKI Java utility does not support generating certificates in x509 format by default. To configure the utility to generate certificates in x509 format:
  - In a standard text editor, comment out the following line in clientConfig.txt.nonKeyEscrow.enroll file:

    ```
    #enroll.token_type=http://docs.oasis-open.org/wss/2004/01/oasis-
    200401-wss-x509-token-profile-1.0#PKCS7
    ```

  - Add following line:

    ```
    enroll.token_type=http://docs.oasis-open.org/wss/2004/01/oasis-
    200401-wss-x509-token-profile-1.0#X509v3
    ```

  - Change the certificate output location to:

    ```
    enroll.certificate_output_file=./outputcertificate.cer
    ```

3.6 Sample DigiCert PKI Input File

The following input file lists the minimum name-value pairs that are required to enroll for a certificate against a policy with key escrow enabled.

```
mail_email=jramirez@acmebank.com
cert_org_unit=Finance
publish_flag=yes
common_name=FinancePortalAccess
cert_corp_company=Acme Bank
jobTitle="Finance Manager"
keysize=2048
```

CHAPTER 4

4 PKI Web Service Sample Java Code

This chapter includes the following topics:

- About the PKI Web Service Sample Java Code
- DigiCert PKI Java Sample RA Application

4.1 About the PKI Web Service Sample Java Code

DigiCert PKI Web Service includes the sample Java code that is designed to illustrate how to integrate DigiCert PKI with your specific PKI solution. The sample Java code is an application (including sample files) that you can compile and run to perform lifecycle tasks against your DigiCert PKI account. Use this sample code to understand how the DigiCert PKI Web Service behaves so you can better integrate it into your specific PKI solution.

Communications between the sample Java code and DigiCert are secured using an SSL connection. An RA certificate authenticates it. You must already have obtained and installed an RA certificate to use this Java code. See "Obtaining Your Registration Authority Certificate" on page 21.

4.2 DigiCert PKI Java Sample RA Application

The sample Java code lets you perform the following operations:

Table 4-1 Sample code operations

Operation	Description
searchCertificate	Search for a certificate
enroll	Enroll for a certificate (including escrowing the key if configured)
renew	Renew a certificate
recover	Recover a key
revoke	Revoke certificates

Operation	Description
userpasscode	Create or update enrollment codes
	Get an enrollment code
	Create or update user
	Get user information
getStatus	Get health information for different operations

NOTE: The Java code is only a sample. It is designed to demonstrate the functionality of PKI Web Services; it should not take the place of your own, integrated RA application.

4.2.1 Using the DigiCert PKI Web Service Sample Java Code

Task 1. Obtain the sample Java code package

1. Extract the contents of the **digicert-pki-web-services.zip** file.
2. Once it is extracted, navigate to the pkiClient/sampleCode directory.

   This folder includes an /src folder that contains the .java files required to compile and run the Java code.

Task 2. Set up your environment

You need the following support applications to run the Java sample RA application:

- Java JDK. Download Java JDK 1.8 from http://www.oracle.com/technetwork/java/javase/downloads/index.html, and install it as described in the product documentation.

  NOTE: All DigiCert PKI components must run the same version of Java. For example, if your Web Service is running JRE 1.8, then the key escrow and recovery service of the PKI Enterprise Gateway must also run JRE 1.8.

- ant. Download ant from http://ant.apache.org, and install it as described in the product documentation.

Task 3. Configure secure communications with DigiCert

Obtain the RA certificate. See "Obtaining Your Registration Authority Certificate" on page 21.

Task 4. Configure the input parameters.

Using a standard text editor, configure the KEY_RECOVERY_ADMIN_ID input parameter in the com.verisign.pki.client.mpki8sample.ClientConfiguration file. Configuring this variable allows the sample RA application to escrow the private key during enrollment and recover the private key later.

Enter a base64-encoded value of an X.509 RA or DigiCert PKI administrator certificate for the account.

Task 5. Compile and run the sample Java code

Compile and run the sample Java code according to the steps:

1. From the folder where build.xml is located, run the following command to compile the Java code:

   ```
   ant
   ```

   The command builds the code and places the output jar file in the lib/ folder.

   NOTE: If you modify the sample code and want to build the modified code without regenerating and recompiling the stub code, run the following command:

   ```
   ant sample-client-jar
   ```

2. Run the following command to run the Java code.

   ```
   java -Djavax.net.ssl.keyStore=<path to RA certificate>
   -Djavax.net.ssl.keyStorePassword=<password to the RAkeystore>
   -Djavax.net.ssl.trustStore=<path to the SSL trust store in java>
   -Djavax.net.ssl.trustStorePassword=<password to the SSL trust store, typically
   "changeit"> -jar lib/samplepkiwsclient.jar
   <searchCertificate|enroll|renew|revoke|suspend|resume|recover|
   userpasscode|getStatus> <profile OID>
   ```

   For example, to enroll for a certificate, run the following command:

   ```
   java -Djavax.net.ssl.keyStore=./racert.ks -Djavax.net.ssl.
   keyStorePassword=password -Djavax.net.ssl.trustStore=
   ../certificates/cacerts -Djavax.net.ssl.trustStore Password=changeit -jar
   lib/samplepkiwsclient.jar enroll 2.16.840.1.113733.1.16.1.2.2.1.1.604911641
   ```

If the KEY_RECOVERY_ADMIN_ID variable is configured in the com.verisign.pki.client.mpki8sample.ClientConfiguration file, the enrollment operation also escrows the private key. Additionally, this variable must be set and a private key escrowed to run the command with the recover operation.

The results of the operation are displayed in your command line. The RA application outputs a policy file, certificate file, user information, or renewed certificate file to where the Java code was run.

NOTE: This sample code cannot be used against the profile that has been configured with a CA using DSA.

To simplify the process of running the Java code, you can use a script similar to the following to automate the command:

```
@echo off

REM Edit this variable to point to the sample code home directory set
PROJECT_HOME=E:\DigiCert\webservice\sampleCode
set CERTIFICATE_PROFILE_OID=2.16.840.1.113733.1.16.1.2.2.1.1.110565

java -Djavax.net.ssl.keyStore=%PROJECT_HOME%\certificates\ra.jks -
Djavax.net.ssl.keyStorePassword=1password -
Djavax.net.ssl.trustStore=%PROJECT_HOME%\certificates\cacerts -
Djavax.net.ssl.trustStorePassword=changeit -jar
lib/samplepkiwsclient.jar enroll %CERTIFICATE_PROFILE_OID%
```

NOTE: Refer to the readme.txt file for more detailed instructions.

CHAPTER 5

5 Integrating DigiCert PKI with Your RA Application

This chapter includes the following topics:

5.1 Introduction

You use the Certificate Enrollment Policy Protocol, Certificate Enrollment Protocol, and the Certificate Management Protocol to integrate operations into your RA application.

5.2 Certificate Enrollment Policy Protocol

Use the Certificate Enrollment Policy Protocol Web Service method to obtain the certificate policies that are configured for an account and CA. Certificate profiles are configured in PKI Manager. The certificate profile defines the data that is needed to create the certificate request. The certificate enrollment policy protocol lets you programmatically set the certificate type for which the end user enrolled. It also lets you set what data the enrollment request contains.

To obtain the certificate policies, your RA application must send a getPolicies request and obtain a response. The response includes all of the certificate profiles with identifying OIDs for that account and CA.

The first time your RA application enrolls for a certificate against a particular account, you must use this protocol before making the enrollment request. If you only make a change to the account settings, you can choose to have your RA application use the Certificate Enrollment Policy Protocol with each certificate enrollment request.

- Calling the protocol with each request ensures that the policy data is the most up-to-date, but requires additional steps in the enrollment flow.

- Calling the protocol only when a certificate profile is updated creates a less complicated enrollment flow, but it requires that you manually trigger the Certificate Enrollment Policy Protocol each time the profiles change.

5.2.1 Recommendations

You must call the Certificate Enrollment Policy Protocol before the initial enrollment request for an account and CA. DigiCert recommends that you cache the data that is returned. You can use it in future enrollments. You should only need to call the Certificate Enrollment Policy Protocol again if you add features to the account or make changes to the certificate policy.

5.2.2 Sample Certificate Enrollment Policy Protocol Request and Response

This section provides sample Certificate Enrollment Policy Protocol (getPolicies) requests and responses.

Sample getPolicies Request

The following is a sample getPolicies request.

See "About the Certificate Enrollment Policy Protocol" on page 91.

```
<ns1:getPolicies xmlns:ns1="http://schemas.verisign.com/
pkiservices/2009/07/policy"> <ns1:version>2.0</ns1:version>

<ns1:clientTransactionID>123456789 </ns1:clientTransactionID>

<ns1:client><ns1:lastUpdatetime>2009-11-06T10:11:12.000-08:00

</ns1:lastUpdatetime>

<ns1:preferredLanguage xmlns:xsi="http://www.w3.org/2001/

XMLSchema-instance" xsi:nil="1"/> </ns1:client>

<ns1:requestFilter><ns1:policyIDs><ns1:oid>
2.16.840.1.113733.1.16.1.2.6.1.1.601392593</ns1:oid>

</ns1:policyIDs></ns1:requestFilter>

</ns1:getPolicies>
```

Sample getPolicies Response

The following is a sample getPolicies response.

See "About the Certificate Enrollment Policy Protocol" on page 91.

```
<?xml version="1.0" encoding="UTF-8"?>

<ns1:getPoliciesResponse xmlns:ns1="http://schemas.verisign.com/
pkiservices/2009/07/policy">

      <ns1:clientTransactionID>123456789</ns1:clientTransactionID>

      <ns1:serverTransactionID>7b403e4718a3605b</ns1:server TransactionID>

      <ns1:response>

            <ns1:policyID>1</ns1:policyID>

            <ns1:policyFriendlyName>DigiCert Policy</ns1:policyFriendly Name>

            <ns1:nextUpdateHours xmlns:xsi="http://www.w3.org/2001/ XMLSchema-
            instance" xsi:nil="1"/>

            <ns1:policiesNotChanged>false</ns1:policiesNotChanged>

            <ns1:policies>

                  <ns1:policy>

                        <ns1:policyOIDReference>1</ns1:policyOIDReference>

                        <ns1:cAs>

                              <ns1:cAReference>1</ns1:cAReference>

                        </ns1:cAs>

                        <ns1:attributes>

                              <ns1:policySchema>1</ns1:policySchema>

                              <ns1:certificateValidity>

                                    <ns1:validityPeriodDays>4</ns1:validity

                                    PeriodDays>

                                    <ns1:renewalPeriodDays>4</ns1:renewalPeriod
                                    Days>

                              </ns1:certificateValidity>

                              <ns1:certificateOverrideValidity>

                                    <ns1:overrideFlag>true</ns1:overrideFlag>

      <ns1:overrideNameValuePair>$override

            ValidityDays</ns1:overrideNameValuePair>

      <ns1:overrideNameValuePair>$override

            ValidityStartDate</ns1:overrideNameValuePair>
```

```
<ns1:overrideNameValuePair>$override
      ValidityEndDate</ns1:overrideNameValuePair>
                  </ns1:certificateOverrideValidity>
                  <ns1:subjectNameInfo>
                        <ns1:subjectNameAttribute>
                        <ns1:subjectNameAttributecount>1</ns1:
                        subjectNameAttributecount>
                        <ns1:subjectNameAttributeNameValuePair>
                              <ns1:attributeName>pkcs-9-challenge
                              Password</ns1:attributeName>
                              <ns1:attributeNameValue mandatory= "false"
                              type="VT_UTF8_STRING">
$challenge</ns1:attributeName Value>
                              <ns1:attributeNameValueProperty>
                                    <ns1:value/>
                                    <ns1:source>USER_INPUT</ns1:source>
                                    <ns1:mandatory>false</ns1:
mandatory>
                                    <ns1:overridable>true</ns1:
overridable>
                              </ns1:attributeNameValueProperty>
                        </ns1:subjectNameAttributeNameValuePair>
                  </ns1:subjectNameAttribute>
                  <ns1:subjectNameAttribute>
                        <ns1:subjectNameAttributecount>1</ns1:
                        subjectNameAttributecount>
                        <ns1:subjectNameAttributeNameValuePair>
                              <ns1:attributeName>id-at-commonName
</ns1:attributeName>
                              <ns1:attributeNameValue mandatory=
                              "false" type="VT_UTF8_STRING">
$common_name</ns1:attributeName Value>
                              <ns1:attributeNameValueProperty>
                                    <ns1:value/>
                                    <ns1:source>USER_INPUT</ns1:source>
```

```
                                        <ns1:mandatory>false</ns1:

    mandatory>
                                        <ns1:overridable>true</ns1:
    overridable>
                                </ns1:attributeNameValueProperty>
                            </ns1:subjectNameAttributeNameValuePair>
                        </ns1:subjectNameAttribute>
                        <ns1:overrideSubjectNameFormat>false</ns1:
overrideSubjectNameFormat>
                            </ns1:subjectNameInfo>
                            <ns1:extensions>
                                <ns1:Extension>
                                    <ns1:extensionOIDReference>2</ns1:
                                    extensionOIDReference>
                                    <ns1:extensionCriticalFlag>false</ns1:
                                    extensionCriticalFlag>
                                    <ns1:extensionSyntax>
                                        <ns1:extensionAttributeNameValuePair>
                                            <ns1:attributeName>otherNameUPN
    </ns1:attributeName>
                                            <ns1:attributeNameValue
                                            mandatory=
    "true">$otherNameUPN</ns1: attributeNameValue>
                                            <ns1:attributeNameValue
                                            Property>
                                            <ns1:value/>
                                            <ns1:source>USER_INPUT</ns1:
        source>
                                            <ns1:mandatory>true</ns1:
        mandatory>
                                            <ns1:overridable>true</ns1:
        overridable>
                                            </ns1:attributeNameValue
```

```
                                                Property>
                                           </ns1:extensionAttributeNameValue
                                           Pair>
                                      </ns1:extensionSyntax>
                                 </ns1:Extension>
                            </ns1:extensions>
                            <ns1:privateKeyAttributes>
                                 <ns1:keysize>1536</ns1:keysize>
                                 <ns1:keyEscrowPolicy>
                                      <ns1:keyEscrowEnabled>true</ns1:keyEscrow
              Enabled>
         <ns1:keyRecoveryDualAdminApproval
         Required>false</ns1:keyRecoveryDual
         AdminApprovalRequired>
                            <ns1:keyEscrowDeploymentMode>CLOUD</ns1:
                            keyEscrowDeploymentMode>
                       </ns1:keyEscrowPolicy>
                       <ns1:keyexportable>true</ns1:keyexportable>
                       <ns1:algorithmOIDReference xmlns:xsi=
                       "http://www.w3.org/2001/XMLSchema-instance" xsi:nil="1"/>
                       <ns1:cryptoProviders xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance" xsi: nil="1"/>
                       </ns1:privateKeyAttributes>
                       <ns1:clientPolicy>
                            <ns1:certPublish>yes</ns1:certPublish>
                       </ns1:clientPolicy>
                       <ns1:systemInfo>
                            <ns1:cACertPublish>no</ns1:cACertPublish>
                            <ns1:certificateDeliveryFormat>
                            http://schemas.verisign.com/pkiservice
                            s/2009/07
                            /PKCS12</ns1:certificateDeliveryFormat>
                            <ns1:serviceEndpointList>
                                 <ns1:serviceEndpoint>
```

```
                    <ns1:type>DigiCertCertificate
                    EnrollmentPolicyEndpoint</ns1:

                    type>

                    <ns1:endpointURI>https://pki-ra.acme
                    bank.net/ra/policyService</ns1:

                    endpointURI>

                </ns1:serviceEndpoint>

                <ns1:serviceEndpoint>

                    <ns1:type>DigiCertCertificate
                    EnrollmentEndpoint</ns1:type>

                    <ns1:endpointURI>https://pki-ra.acme
                    bank.net/ra/enrollmentService

        </ns1:endpointURI>

                </ns1:serviceEndpoint>

                <ns1:serviceEndpoint>

                    <ns1:type>digitalIDCenterEndpoint

        </ns1:type>

                    <ns1:endpointURI>https://pki-ra.acme

                    bank.net/certificate-service/
                    didcservlet</ns1:endpointURI>

                </ns1:serviceEndpoint>

            </ns1:serviceEndpointList>

    <ns1:duplicateCertPolicy>DUPLICATE_CERT_
    ALLOWED</ns1:duplicateCertPolicy>

        </ns1:systemInfo>

        <ns1:rAPolicy xmlns:xsi=
        "http://www.w3.org/2001/XMLSchema-instance"
        xsi:nil="1"/>

        <ns1:seatIdInfo>

            <ns1:attributeNameValue mandatory="true"
            type="VT_UTF8_STRING">seat_id</ns1:

            attributeNameValue>

                <ns1:attributeNameValueProperty>

                    <ns1:value xmlns:xsi=
                                    Schema-instance"
                                    xsi:nil="1"/>

                                    <ns1:source>PASSCODE</ns1:s
                                    ource>

                                    <ns1:sourceAttributeName>se
                                    at_id</ns1:sourceAttributeN
                                    ame>

                                    <ns1:mandatory>true</ns1:ma
                                    ndatory>

                                    <ns1:overridable>false</ns1
                                    :overridable>

                              </ns1:attributeNameValueProperty>

                  </ns1:seatIdInfo>

                  <ns1:applicationInstructions xmlns:xsi=
                  "http://www.w3.org/2001/XMLSchema-instance" xsi: nil="1"/>

                  <ns1:deploymentMode>THIRDPARTY_INTEGRATION</ns1:
                  deploymentMode>

                  <ns1:status>ACTIVE</ns1:status>

            </ns1:attributes>

      </ns1:policy>

</ns1:policies>

</ns1:response>

      <ns1:cAs>

            <ns1:cA>

                  <ns1:uris>https://pki-ra.acmebank.net/ra/
                  enrollmentService</ns1:uris>

<ns1:certificate>MIIDsTCCApmgAwIBAgIQL4L8IVmOWK7g1mwrD/gTejANBgkqhkiG9w0BAQUFADBx
MQswCQYDVQQGEwJVUzEXMBUGA1UEChMOVmVyaVNpZ24sIEluYy4xHzAdBgNVBAsTFkZPUiBURVNUIFBVU
lBPU0VTIE9OTFkxKDAmBgNVBAMTH01hbmdtdW4gUEtJIC0gMSBMZXZlbCAtIFRlc3QgQ0EwHhcNMTEwMT
IwMDAwMDAwWhcNMjEwMTE5MjM1OTU5WjBxMQswCQYDVQQGEwJVUzEXMBUGA1UEChMOVmVyaVNpZ24sIEl
uYy4xHzAdBgNVBAsTFkZPUiBURVNUIFBVUlBPU0VTIE9OTFkxKDAmBgNVBAMTH01hbmdtdW4gUEtJIC0g
MSBMZXZlbCAtIFRlc3QgQ0EwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDPwANzYClK8chkW
/GmBztEBZZYSHWMcInbHaHKkNh7NUEjxwMrn4brDE3ReqCr8lGDrYdtaS+PEVNRF3Ijkr0pH040xxsFzL
b6huk9X7zUghkxLY2tHsltR+diWZVVwzTw21iuDMWOlkyBi27/rUQW8jtb6AYKsi4mZOjc5vgf1XKiH4j
f0U/Wi4GaLvKJTiuRZCAyaKKExImKeaOHsx2hd3aNqFU/sBKSmGZsko1Vz/bO7Q1WrQyj9fAunrMVEj5x
gWqDDwuVzsVbxpL85DdftmH+KXQ/gxJ48vmTI4KKmAHo05MED+JTlQ9V5W11kEQrTLbNGf1qqLjSe4Q/x
```

```
t1AgMBAAGjRTBDMBIGA1UdEwEB/wQIMAYBAf8CAQAwDgYDVR0PAQH/BAQDAgEGMB0GA1UdDgQWBBRUlZ7
imqBHnO8JYQr8k7wCaZ/rdDANBgkqhkiG9w0BAQUFAAOCAQEAetMwk5Eg/nTlZhofqKEZ+0WJaf+225DK
4lI2hLdiXFFPpUp+TJN8iPMVDU32Y1Y0v9/bQXBTmNtTicEn+xhuEo7l0n6CpxgNEPbrysj744IEI8dzE
zBw4lFHTVMallYhpDq1LBZMJyXZBOW3fOFZcad1CAs0nu/q4N2B45H+71BkN/PvIfbQG/q6dK4YFdBmaf
EWFhoPusGOQZNbOxOyLOOCsW1ZJfynA0182JccbrBc3mFNEg+aSt0fRiJxtFOABw4KPXBW7+0Vqt9qDFJ
YONfJMj2R2quuKqP1SKdNHHDpTfZcWgf8QbGtDUh0a4HZdY4tKT7UvkcxhuqTw2LdZA==</ns1:certif
icate>
                    <ns1:cAIssuerName>CN = DigiCert PKI – 1
                    Level - Test CA, OU = FOR TEST PURPOSES
                    ONLY, O = "VeriSign, Inc.", C =

                    US</ns1:cAIssuerName>

                    <ns1:cAReferenceID>1</ns1:cAReferenceID>

                    <ns1:cAType xmlns:xsi=
                    "http://www.w3.org/2001/XMLSchema-
                    instance" xsi:nil="1"/>

<ns1:rootCACertificate>MIIDsTCCApmgAwIBAgIQL4L8IVmOWK7
g1mwrD/gTejANBgkqhkiG9w0BAQUFADBxMQswCQYDVQQGEwJVUzEXMBUGA1UEChMOV
mVyaVNpZ24sIEluYy4xHzAdBgNVBAsTFkZPUiBURVNUIFBVUlBPU0VTIE9OTFkxKDAmBgNVBAMTH01hbm
dtdW4gUEtJIC0gMSBMZXZlbCAtIFRlc3QgQ0EwHhcNMTEwMTIwMDAwMDAwWhcNMjEwMTE5MjM1OTU5WjB
xMQswCQYDVQQGEwJVUzEXMBUGA1UEChMOVmVyaVNpZ24sIEluYy4xHzAdBgNVBAsTFkZPUiBURVNUIFBV
UlBPU0VTIE9OTFkxKDAmBgNVBAMTH01hbmdtdW4gUEtJIC0gMSBMZXZlbCAtIFRlc3QgQ0EwggEiMA0GC
SqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDPwANzYClK8chkW/GmBztEBZZYSHWMcInbHaHKkNh7NUEjxw
Mrn4brDE/3ReqCr8lGDrYdtaS+PEVNRF3Ijkr0pH040xxsFzLb6huk9X7zUghkxLY2tHsltR+diWZVVwz
Tw21iuDMWOlkyBi27/rUQW8jtb6AYKsi4mZOjc5vgf1XKiH4jf0U/Wi4GaLvKJTiuRZCAyaKKExImKeaO
Hsx2hd3aNqFU/sBKSmGZsko1Vz/bO7Q1WrQyj9fAunrMVEj5xgWqDDwuVzsVbxpL85DdftmH+KXQ/gxJ4
8vmTI4KKmAHo05MED+JTlQ9V5W11kEQrTLbNGf1qqLjSe4Q/xt1AgMBAAGjRTBDMBIGA1UdEwEB/wQIMA
YBAf8CAQAwDgYDVR0PAQH/BAQDAgEGMB0GA1UdDgQWBBRUlZ7imqBHnO8JYQr8k7wCaZ/rdDANBgkqhki
G9w0BAQUFAAOCAQEAetMwk5Eg/nTlZhofqKEZ+0WJaf+225DK4lI2hLdiXFFPpUp+TJN8iPMVDU32Y1Y0
v9/bQXBTmNtTicEn+xhuEo7l0n6CpxgNEPbrysj744IEI8dzEzBw4lFHTVMallYhpDq1LBZMJyXZBOW3f
OFZcad1CAs0nu/q4N2B45H+71BkN/PvIfbQG/q6dK4YFdBmafEWFhoPusGOQZNbOxOyLOOCsW1ZJfynA0
182JccbrBc3mFNEg+aSt0fRiJxtFOABw4KPXBW7+0Vqt9qDFJYONfJMj2R2quuKqP1SKdNHHDpTfZcWgf
8QbGtDUh0a4HZdY4tKT7UvkcxhuqTw2LdZA==</ns1:rootCACertificate>
             </ns1:cA>

      </ns1:cAs>

      <ns1:oIDs>

             <ns1:oID>

                    <ns1:value>id-ce-subjectAltName</ns1:value>

                    <ns1:oIDReferenceID>2</ns1:oIDReferenceID>

                    <ns1:group>3</ns1:group>
                    <ns1:defaultName xmlns:xsi=
                   "http://www.w3.org/2001/XMLSchema-instance"
                   xsi:nil="1"/>

            </ns1:oID>

            <ns1:oID>

                    <ns1:value>2.16.840.1.113733.1.16.1.2.6.1.1.601392593

</ns1:value>

                    <ns1:oIDReferenceID>1</ns1:oIDReferenceID>

                    <ns1:group>10</ns1:group>

                    <ns1:defaultName>For the Microsoft Windows Encrypting File
                    System. Enable folder and file encryption.

        </ns1:defaultName>

            </ns1:oID>

        </ns1:oIDs>

</ns1:getPoliciesResponse>
```

NOTE: For ECC and DSA, the PKCS#10 CSR format is supported, but the SPKAC format is not supported.

5.3 Certificate Enrollment Protocol

The Certificate Enrollment Protocol Web Service method is used to enroll for a certificate. Call this Web Service method after obtaining the policy with the Certificate Enrollment Policy Protocol Web Service method. To enroll for a certificate, your RA application must send a requestSecurityToken request and obtain a response. The response includes the certificate in one of the following formats, based on how your DigiCert PKI account is configured. And, it is also based on how it is defined in your certificate policy.

- PKCS#7
- PKCS#12

By default, DigiCert PKI automatically approves all correctly formatted requests that match the certificate policy. If the enrollment request is rejected, DigiCert PKI responds with a SOAP 1.1 fault.

You can send additional information using name-value pairs, and the public key for the certificate as a base64 string. Some examples of additional information include whether to have the certificate published in DigiCert Trust Network, or certificate search data such as $mail_firstname and $mail_lastname. See “Name/Value Pairs”.

5.3.1 Certificate Enrollment Protocol Enrollment Flow

The basic steps for certificate enrollment using the Certificate Enrollment Protocol are as follows:

1. Send a getPolicies call (required for the initial enrollment or if the policy has changed since the last enrollment) to DigiCert PKI. The response lists the available policies for this account. Each policy is identified with a certificate profile OID.

   All of the information that is needed to make an enrollment request is provided in the certificate policy. The certificate policy is returned by the getPolicies response. The information includes name-value pairs that are needed for enrollment, minimum key size, certificate format, and so on.

2. Your RA application parses the response from the getPolicies request to obtain the certificate policy for the type of certificate you want to enroll.
3. Using the information from the certificate policy, your subscriber's enrollment data, and the private key that is generated on your end user's computer, your RA application sends a requestSecurityToken call with a requestType of Issue to DigiCert PKI.

   If the key escrow store is set to DigiCert, DigiCert PKI generates the private key. The key escrow and recovery service generates the private key, if the key escrow store is set to a local user store.

4. DigiCert PKI returns a response including the certificate in the appropriate format (see Table 5-1).

5.3.2 Certificate Enrollment Protocol Renewal Flow

The renewal flow using the Certificate Enrollment Protocol is similar to the enrollment flow except that your RA application does not send a request for the certificate policy. The certificate is renewed using the same enrollment information that is used in the original enrollment request, along with a public key (DigiCert recommends that the key pair be newly generated; however, the RA application can use the original key pair).

For requests against profiles with key escrow enabled, a new key pair is always generated.

NOTE: If the policy information has changed since the original enrollment, you must enroll for a new certificate.

The steps for certificate enrollment using the Certificate Enrollment Protocol are as follows:

1. Send a requestSecurityToken call to the renew URI. Include the original public key or a newly generated one, the original certificate profile ID, and an x.509 file containing the

public key of the certificate being renewed. The certificate profile ID is a mandatory field; however, this value is ignored by the back-end during renewal.

2. Using the information from the certificate policy, your subscriber's enrollment data, and the private key that is generated on end-user computers, your RA application creates a renewal request for the appropriate certificate policy. Your RA application sends this request to DigiCert PKI.

   The private key is generated by DigiCert PKI, if the key escrow store is set to DigiCert. The private key is generated by the key escrow and recovery service, if the key escrow store is set to a local user store.

3. DigiCert PKI returns a response including the renewed certificate in the appropriate format.

5.3.3 Recommendations

DigiCert recommends that you call the Certificate Enrollment Policy Protocol before making the initial enrollment request, and then cache the resulting certificate information for use with future enrollment requests. If you already have this information, you can make the enrollment request without the Certificate Enrollment Policy Protocol call.

Renewal requests use the same policy information that is used for the original enrollment. If there has been a change to the policy or features for the account since the original enrollment request, the certificate should not be renewed as it does not contain the new information. Your RA application must enroll for a new certificate to pick up the policy or the feature changes.

5.3.4 Important Elements, Requests, and Responses

You can enroll in multiple token types, request types, and certificate formats. The URI and value you send (and the type of certificate you receive) depends on the type of enrollment, as defined in your certificate policy. Table 5-1 describes the most common Certificate Enrollment Protocol requests and responses.

NOTE: Refer to the appropriate element for the specific URI to send.

See "About the Certificate Enrollment Policy Protocol" on page 91.

Table 5-1 Common Certificate Enrollment Protocol request and responses

	Request			Response	
	Token Type URI	Token Type URI	BinarySecurity Token Value	RequestedSecurity Token Value	Sample Request and Response
Enroll without key escrow	PKCS#7	Issue	PKCS#10	PKCS#7	See "Sample Enrollment without Key Escrow Requests and Responses" on page 48.
Enroll with key escrow	PKCS#12	Issue	None	PKCS#12 and password	See "Sample Enrollment with Key Escrow Requests and Responses" on page 52.
Renew without key escrow	PKCS#7	Renew	X.509 v3 and PKCS#10	PKCS#7	See "Sample Renewal without Key Escrow Requests and Responses" on page 55.
Renew with key escrow	PKCS#12	Renew	X.509 v3 and PKCS#10	PKCS#12 and password	See "Sample Renewal with Key Escrow Requests and Responses" on page 57.

5.3.5 Sample Certificate Enrollment Protocol Requests and Responses

This section provides sample requests and responses for the following scenarios.

See "About the Certificate Enrollment Protocol" on page 134.

For requests and responses by key escrow, you must have implemented the key escrow option.

- See "Sample Enrollment without Key Escrow Requests and Responses" on page 48.
- See "Sample Enrollment with Key Escrow Requests and Responses" on page 52.
- See "Sample Renewal without Key Escrow Requests and Responses" on page 55.
- See "Sample Renewal with Key Escrow Requests and Responses" on page 57.

Sample Enrollment without Key Escrow Requests and Responses

Use RequestSecurityToken and RequestSecurityTokenResponse to request a certificate enrollment without escrowing the key.

Sample Enrollment without Key Escrow (RequestSecurityToken) Request

The following is a sample RequestSecurityToken request without key escrow.

See "About the Certificate Enrollment Protocol" on page 134.

```
<ns9:RequestSecurityToken xmlns:ns9="http://docs.oasis-open.org/ ws-sx/ws-
trust/200512/">

      <requestVSSecurityToken xmlns="http://schemas.verisign.com/
      pkiservices/2009/07/enrollment" preferredLanguage="en-US">

      <certificateProfileID>2.16.840.1.113733.1.16.1.2.3.1.1.60161

      3354</certificateProfileID>

      <clientTransactionID>20110703052428</clientTransactionID>

      <tokenType>http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss- X509-
      token-profile-1.0#PKCS7</tokenType>

      <requestType>http://docs.oasis-open.org/ws-sx/ws-trust/
      200512/Issue</requestType>

      <nameValuePair>

            <name>mail_firstname</name>

            <value>Juan</value>

      </nameValuePair>

      <nameValuePair>

            <name>mail_email</name>
```

```
        <value>jrodrigues@acmebank.com</value>
</nameValuePair>
<nameValuePair>
        <name>mail_lastname</name>
        <value>Rodrigues</value>
</nameValuePair>
<nameValuePair>
        <name>state</name>
        <value>California</value>
</nameValuePair>
<nameValuePair>
        <name>common_name</name>
        <value>Juan Rodrigues</value>
</nameValuePair>
<nameValuePair>
        <name>country</name>
        <value>US</value>
</nameValuePair>
<nameValuePair>
        <name>mailStop</name>
        <value>2-15</value>
</nameValuePair>
<nameValuePair>
        <name>publish_flag</name>
        <value>yes</value>
</nameValuePair>
<nameValuePair>
        <name>locality</name>
        <value>Mountain View</value>
</nameValuePair>
<nameValuePair>
        <name>overrideValidityDays</name>
        <value>2</value>
```

```
      </nameValuePair>

      <nameValuePair>

            <name>jobTitle</name>

            <value>Manager</value>

      </nameValuePair>

      <version>2.0</version>

      <binarySecurityToken xmlns:axis2ns1="http://docs.oasis-open.org/ ws-sx/ws-
trust/200512/" ValueType="http://schemas.verisign.com/
pkiservices/2009/07/PKCS10" axis2ns1:EncodingType= "http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity- secext-1.0.xsd#base64binary">

-----BEGIN CERTIFICATE REQUEST-----MIIDKTCCAhECAQAwgasxCzAJBgNVBA

YTAlVTMRMwEQYDVQQIEwpDYWxpZm9ybmlhMQswCQYDVQQHEwJNVjEXMBUGA1UEChM
OV1NBdXRvMjAxMTA1MTAxDjAMBgNVBAsTBUFETUlOMSYwJAYDVQQDEx1XU0F1dG8y

MDExMDUxMCBXU0F1dG8yMDExMDUxMDEpMCcGCSqGSIb3DQEJzrgfverstresreAxM
TA1MTBAeW9wbWFpbC5jb20wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQ

DH+/FbXH7yzcmupKAL5MzDReH2UQd10HXUoOUUeUXsMylY31lCg4ScZSVgCzQCQTQ
vP6XZGVjweTPpxZUKd8JJLC5JTXfFnxChkmiML1MtlTKhMvCx6KObU8dcvJ6Or0uG
o+6Y2m3tGgO0CjubMlVFqLBeYYH9rNuIxv2DczGnDeWI9vixjDGqi2wmZkSHE5LaK
w7MgBbGniwBgnaekT8eMMPbQ8XW/YSyiUvX/BC8tJ6NdHYXKz3/ReYwCP/0gKqnSs
eraklmklimkPOJpomp90ui9-kp0o9ui9j09-/5LxyL2EVzOzkX9tp8lBjHOA4UFqC
0PL8dU5HfUErtWJSbAgMBAAGgODAXBgkqhkiG9w0BCQcxChMIcGFzc3dvcmQwHQYJ
KoZIhvcNAQkCMRATDldTQXV0bzIwMTEwNTEwMA0GCSqGSIb3DQEBBQUAA4IBAQBzC

kh/hkpodpoOKOakS-0J-0Ik4tTZLFWH5TBV0HBi+HNGHgX86qX18aV5d8iqnqsduA
Qo2kSoyDyjlbIMLkEmvQhudYdVCIK/Zgc/GzWkI4gr5+4VDcPjF7goh22gljRmxoS

/9NI0N3I3GMkri6DqCniR45BUwNzwUb3Aqajtc5GYIfOLzVbVfu8NJCmNlsVONwqO
SoptNcmJeFSOIaw2cCNL647ubEKnk3L+HJ/qaO/+PVuN/D9LYTwmhyr2dhYGcMnNk
DiFShoE/4V5h7LQoltDfY4TQoJVrlYeYo3cWnbrd/K+pNDgT5jbEouZ11xARKQgH6 ACFFapy7zr8s4j

-----END CERTIFICATE REQUEST-----

</binarySecurityToken>

      </requestVSSecurityToken>

</ns9:RequestSecurityToken>
```

Sample Enrollment without Key Escrow (RequestSecurityTokenResponse) Response

The following is a sample RequestSecurityTokenResponse response without key escrow.

See "About the Certificate Enrollment Protocol" on page 134.

```
<RequestSecurityTokenResponse xmlns="http://docs.oasis-open.org/ ws-sx/ws-
rust/200512/">

      <RequestVSSecurityTokenResponse xmlns="http://schemas.verisign.
      com/pkiservices/2009/07/enrollment">

      <clientTransactionID>20110703052428</clientTransactionID>

      <serverTransactionID>a59965196bf0121a</serverTransactionID>

      <tokenType>http://docs.oasis-open.org/wss/2004/01/oasis-200401- wss-X509-
      token-profile-1.0#PKCS7</tokenType>

      <requestedVSSecurityToken>

      <binarySecurityToken EncodingType="http://docs.oasis-open.org/
wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd#base64binary "
ValueType="http://docs.oasis-open.org/wss/2004/01/

oasis-200401-wss-X509-token-profile-1.0#PKCS7">
MIAGCSqGSIb3DQEHAqCAMIACAQExADALBgkqhkiG9w0BBwGggDCCBGIwggNKoAMCAQI
CEGlcLAFZ8PtRk3KIgjErk+swDQYJKoZIhvcNAQELBQAwcTELMAkGA1UEBhMCVVMxFz
AVBgNVBAoTDlZlcmlTaWduLCBJbmMuMR8wHQYDVQQLExZGT1IgVEVTVCBQ&VVJQT1NF
UyBPTkxZMSgwJgYDVQQDEx9NYW5nbXVuIFBLSSAtIDEgTGV2ZWwgLSBUZXN0IENBMB4

XDTExMDgwMzAwMDAwMFoXDTExMDgwODIzNTk1OVowaTEiMCAGA1UEAwwZc3N3c3VzZX
JybndkIHNzd3N1c2Vycm53ZDEVMBMGA1UECwwMVlBOLVdJRkktV0VCMRYwFAYDVQQLD

A1NVzst/I30vvMYbiJ/EBkOT55dtL/kvHIvYRXM7ORf22nyUGMc4DhQWoLQ8vx1Tkd9
QSu1YlJsCAwEAAaOB/TCB+jAMBgNVHRMBAf8EAjAAMA4GA1UdDwEB/wQEAwID6DAWBg
NVHSUBAf8EDDAKBggrBgEFBQcDAjAOBgNVHREEBzAFiANRAQEwHwYDVR0jBBgwFoAUV

JWe4pqgR5zvCWEK/JO8Ammf63QwXwYDVR0fBFgwVjBUoFKgUIZOaHR0cDovL2Z0LWVu
dC1jcmwuYmJ0ZXN0Lm5ldC9jYV80ODk2ZWYwMjMwMzNlMGJhNjIyNWU5ZGVlNDUxYTI
0Ni9MYXRlc3RDUkwuY3JsMDAGCmCGSAGG+EUBEAMEIjAgBhNghkgBhvhF&ARABAgMBA

YKe78gqFgk2MDAwMTE5NjEwDQYJKoZIhvcNAQELBQADggEBAGA1hooesqBadELLa+WH
MEYfbzvh8+C3Q14Iv8JDPbwxKP3vGijTOF/x1Z8Hu9NImvy+CKiclYzUaRsS39T5D+v
3OYHNdiIsBKmUqihtBEoohA4N48dskT24FXnFGFqBD+NZZJsEcjo6eBAsKIE2DWA5wX
EYJCf1rOy+bTlaMbEK8lrpJwSEjpyYfMd0hRevCslWnguJHcejBDYU0d/BwUy+XUVJQ
ZS09tES2QE7P1y8eJj9GnlpHg5lwlXPd9E/rcyu3wEjfzdA3zOAHM/5ppNJxBMRmWZT
HOOoeHu+OiYZuTWmllUkajudXJ3IiGpmF4eG4TcszrcEXZ78QxwqU+gAADEAAAAAAAA

      </binarySecurityToken>

</requestedVSSecurityToken>

<version>2.0</version>

      </RequestVSSecurityTokenResponse>

</RequestSecurityTokenResponse>
```

Sample Enrollment with Key Escrow Requests and Responses

Use RequestSecurityToken and RequestSecurityTokenResponse to request a certificate enrollment that escrow the key.

Sample Enrollment with Key Escrow (RequestSecurityToken) Request

The following is a sample RequestSecurityToken request with key escrow.

See "About the Certificate Enrollment Protocol" on page 134.

```
<ns9:RequestSecurityToken xmlns:ns9="http://docs.oasis-open.org/ ws-sx/ws-
trust/200512/">

      <requestVSSecurityToken xmlns="http://schemas.verisign.com/pkiservices/
      2009/07/enrollment" preferredLanguage="en-US">

      <certificateProfileID>2.16.840.1.113733.1.16.1.2.6.1.1.
      601392593</certificateProfileID>

      <clientTransactionID>20110629182601</clientTransactionID>

      <requestType>http://docs.oasis-open.org/ws-sx/ws-trust/200512/
      Issue</requestType>

      <nameValuePair>

            <name>mail_firstname</name>

            <value>Juan</value>

      </nameValuePair>

      <nameValuePair>

            <name>mail_email</name>

            <value>jrodrigues@acmebank.com</value>

      </nameValuePair>

      <nameValuePair>

            <name>mail_lastname</name>

            <value>Rodrigues</value>

      </nameValuePair>

      <nameValuePair>

            <name>state</name>

            <value>California</value>

      </nameValuePair>

      <nameValuePair>

            <name>common_name</name>

            <value>Juan Rodrigues</value>
      </nameValuePair>
      <nameValuePair>
            <name>country</name>
            <value>US</value>
      </nameValuePair>
      <nameValuePair>
            <name>mailStop</name>
            <value>2-15</value>
      </nameValuePair>
      <nameValuePair>
            <name>publish_flag</name>
            <value>yes</value>
      </nameValuePair>
      <nameValuePair>
            <name>locality</name>
            <value>Mountain View</value>
      </nameValuePair>
      <nameValuePair>
      <nameValuePair>
            <name>overrideValidityDays</name>
            <value>2</value>
      </nameValuePair>
      <nameValuePair>
      <nameValuePair>
            <name>jobTitle</name>
            <value>Manager</value>
      </nameValuePair>
            <version>2.0</version>
</requestVSSecurityToken></ns9:RequestSecurityToken>
```

Sample Enrollment with Key Escrow (RequestSecurityTokenResponse) Response

The following is a sample RequestSecurityTokenResponse response with key escrow.

See "About the Certificate Enrollment Protocol" on page 134.

```
<RequestSecurityTokenResponse xmlns="http://docs.oasis-open.org/ ws-sx/ws-
trust/200512/">

      <RequestVSSecurityTokenResponse xmlns="http://schemas.verisign.com/
      pkiservices/2009/07/enrollment">

            <clientTransactionID>20110629182601</clientTransactionID>

            <serverTransactionID>216d4cdd08063eb4</serverTransactionID>

            <tokenType>http://schemas.verisign.com/pkiservices/2009/07/
            PKCS12</tokenType>

      <requestedVSSecurityToken>

      <binarySecurityToken EncodingType="http://docs.oasis-open.org/

wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd#base64binary  "
ValueType="http://schemas.verisign.com/pkiservices/2009/07/PKCS12">
```

```
MIIO+AIBAzCCDrIGCSqGSIb3DQEHAaCCDqMEgg6fMIIOmzCCBMQGCSqGSIb3DQEHTx372

viCIEVQEiAvGuVdkoyyn8zBRdQSTOKPKCuYeWLWQgaFIUsSJzBIziA2sQNQtjwiUPul4Z
fskWN7FAOKxY2AN7RGUwzzRt2owYlY6BGBLJrEP0T4/X1PWzhunnmOGdFg3oKB8eV7hs1
G7p3RvxzNE81mlkP+ARkdWefkWS8wNHdbM4pf5yH2zLaUc9Y4wcPRQioAHRYlR5hDAYP6
zqqiEoOB4rswYnYtXSEc59w+AWQ4Z+quEzEsI4Va/PhO+vm1mAQ05QuT2ib8tgoL/IRKj
KwIfRLO6rtywukA4ILChSM/JGCeKVDSXiB8l/ChOQeRIB7kJ1K1soUU6yBhqhk9+iv8HZ
Lt5/YMYvTl/CtZx5F3OCqlv6rDJ2U2SF3gOF2wD/jYqr0YjPi22+dRuDQrdmiGyxJCwZl
S1Ymb2gxNEa6ntZC07i57hKGfBAXwLIhSGHo3wedQjKWrnGT4aFj6mcUM/VZ/6iTgadh7
wNVElxB/2YRD+jNAU5Nea8wAsZKArB+4qicnCS61wcBFEBgwJ8121emGwcQeDPjqu4ymY
YJ5qTotrBQ6JOcranQJLDRG7DuYcpo03Pjwe+sl3ne4Wx4wyCqcrv6I/S6LsOPrTF2xJg
rppdgATlgj/JVo6Br9jpL1pI/RR27JBSC9URj9Q2KKD3C9qHijdM1QAsrwpnWmXackESV
H3xDl+w/L2BT72spAD+5glYoq5I1F8OHrDYO4YZULfkun60wyU2D4Dh43UVkCbKO7J/eR
gYmWQC71W0lT/uTFeKyw/KG6E0qJ7IEuYSSNdC7bvbvEr5A30YltYeU9vO185zn9YqzVF
23kyFdaNUI2+AYaPVzzeHpd5i2H04IiPXi/dw6oFe6Nbf4+lfrxyk1/rOOCZbnH6JTJVO
qugfwaoEMMsBPdXC+cVnE0YbvfOpTnqQSjUq1ib7N2zIGgfbGFPF+E1B3E9QOz3vCPTs5
mrAoVJ9KieI8XDEoRo2OhNP3NV3GzyCQuWQI8BD5eosNx3S6QPyUrVyBWgVlzmBMyA0Oi
NEGOvAbExshuUQBYAJD2nf2mUlhbiAcfxhaJxdd/5P/7wf0Ff6ZywXCNtjgK6QSHwMx+O
dZ8ml9LUcc9/gkNF8E3eIvW0r+OymzuaowmR+yw8Z7PH2/o38OSAbkmtFcPpzCBVy8Z
ZZge0RoHn4P6iitRsKBNEdgkknvT+Lt2Hwpm1KKJXKve/ZTc0CUngJPicOkHTaXbqgA
Muu4aIiJU1HS27v5CctwbnMHn/QQzz2ig9qVYBbiXyjIzlRkVwMAna8t+mwf52qc1lc
u+8HMDCCEsTeon/XVR1z4rqO+auS8cq99iFAUMY0ZhrCsDz+upfvBQrCgXbAAbfJvr6
Y+WBHDfmpIoqfKfs4CKXiLeG4Y4VNcl/m/LJWSLSae8jePyQU8+HXzHnjyL5QizAMAr
loBt06o5vUTH/C6WgulPGGrNyyoZBcEK3u7Ly5wCgiJy2pKFIMyBnowld4L8wNBJXfk
vND8nV92Rh7uqNqQPwnEQ/Sq4V0TcVfTxlVyapAYICfPdvfW/fV7el8fn3IWllktwFb
M3rYrlYsND2naF5ANWXSVEC3qbr0DDDIfuP8jW/
```

```
      </binarySecurityToken>

      <pKCS12Password>DSheiSkJNuPO</pKCS12Password>
```

```
</requestedVSSecurityToken>

<version>2.0</version>

      </RequestVSSecurityTokenResponse>

</RequestSecurityTokenResponse>
```

Sample Renewal without Key Escrow Requests and Responses

Use RequestSecurityToken to request a certificate renewal without escrowing the key.

Sample Renewal without Key Escrow (RequestSecurityToken) Request

The following is a sample RequestSecurityToken request without key escrow.

See "About the Certificate Enrollment Protocol" on page 134.

```
<ns9:RequestSecurityToken xmlns:ns9="http://docs.oasis-open.org/ ws-sx/ws-
trust/200512/">

      <requestVSSecurityToken xmlns="http://schemas.verisign.com/
      pkiservices/2009/07/enrollment" preferredLanguage="en-US">

      <certificateProfileID>Magnum Renewal</certificateProfileID>

      <clientTransactionID>20110712190716</clientTransactionID>

            <requestType>http://docs.oasis-open.org/ws-sx/ws-trust/200512/
      Renew</requestType>

            <version>2.0</version>

      <binarySecurityToken xmlns:axis2ns1="http://docs.oasis-open.org/ ws-sx/ws-
trust/200512/" ValueType="http://docs.oasis-open.org/ wss/2004/01/oasis-200401-
wss-X509-token-profile-1.0#X509v3"

axis2ns1:EncodingType="http://docs.oasis-open.org/ wss/2004/01/oasis-200401-wss-
wssecurity-secext- 1.0.xsd#base64binary">
MIIEPDCCAySgAwIBAgIQfFJoZTvtW4asWV6JwHs1djANBgkqhkiG9w0BAQUFADBx
MQswCQYDVQQGEwJVUzEXMBUGA1UEChMOVmVyaVNpZ24sIEluYy4xHzAdBgNVBAsT
FkZPUiBURVNUIFBVUlBPU0VTIE9OTFkxKDAmBgNVBAMTH01hbmdtdW4gUEtJIC0g
MSBMZXZlbCAtIFRlc3QgQ0EwHhcNMTEwODEyMDAwMDAwWhcNMTEwODE5MjM1OTU5
WjBTMSIwIAYDVQQDDBllZ3dhdXRvdXNlcjEgZWd3YXV0b3VzZXIxMRUwEwYDVQQL
DAxWUE4tV0lGSS1XRUIxFjAUBgNVBAsMDU1VTFRJLUFMTE9XRUQwggEiMA0GCSqG

SIb3DQEBAQUAA4IBDwAwggEKAoIBAQDjRESfXffcVNvZXoz3BeDJW2hD9X9jsw6b
XvCLyHl2xKWfvM+/x1vgJEQU41TSksccxLXAyva67cIKlum/iQcSdidI42sWBuji

+JO1AMtmmOoEMFhK/gZIVkh4jdFusddPNk8DZVogTKh4xg873lDHWewF7RLfJ7/L
4pFCCpTsa+kjCrXzV6z+STHC9+lc4zz0Puh8eXGb4GWQM8dGBt3h1Tqd0b3L0KgP
L5ifVcWhaGB66c6UA4tKHCrcyPchyBAj4anNKWAR8lUYRaoE55hfevYFFhi7wmpg
yZqEkVijn5RJb/oxEoxCMYxHJK/bsDC/8ld5MLHCSspASAKeHMf3AgMBAAGjge0w
geowDAYDVR0TAQH/BAIwADAOBgNVHQ8BAf8EBAMCA+gwFgYDVR0lAQH/BAwwCgYI

KwYBBQUHAwIwHwYDVR0jBBgwFoAUVJWe4pqgR5zvCWEK/JO8Ammf63QwXwYDVR0f
BFgwVjBUoFKgUIZOaHR0cDovL2Z0LWVudC1jcmwuYmJ0ZXN0Lm5ldC9jYV80ODk2
```

```
ZWYwMjMwMzNlMGJhNjIyNWU5ZGVlNDUxYTI0Ni9MYXRlc3RDUkwuY3JsMDAGCmCG
SAGG+EUBEAMEIjAgBhNghkgBhvhFARABAgMBAYKe8804Fgk2MDAwMTMzOTYwDQYJ

KoZIhvcNAQEFBQADggEBAGJ/jinA219qKnfwxtaWss+vBXk49fDyJx0fUMR+ipXD
8K3KmXr24v3hODRPxGFNUiqsaYcMaefir5l17pieIUQxSDdJeIdI7xq88PkIF/
CeQANFX6r4NCwmWwY2Q8WJbfgsJYwkJNQfratBAjkG4MOGk6tZmxBnVrS5G
LlDXohvEUPgzNr6I3v8gJADzTRzy7ppMqFUHXWWkL+iriFNLv06rKP4TV0p9gtIk
89SVjMWMAZeu2NmGG9LQmBcrqq1tklVvhl2YoYps9nC7BR7EzHAPYwNx5LOS1uA/
MlCzEOUGBkpSdPZKeu4cFs36jsJ8gbvoThsIPbfi7YvYtpLF1k=

      </binarySecurityToken>

</requestVSSecurityToken>

</ns9:RequestSecurityToken>
```

Sample Renewal without Key Escrow (RequestSecurityTokenResponse) Response

The following is a sample RequestSecurityTokenResponse response without key escrow.

See "About the Certificate Enrollment Protocol" on page 134.

```
<RequestSecurityTokenResponse xmlns="http://docs.oasis-open.org/ ws-sx/ws-
trust/200512/">

<RequestVSSecurityTokenResponse xmlns="http://schemas.verisign.com/
pkiservices/2009/07/enrollment">

<clientTransactionID>20110712190716</clientTransactionID>

<serverTransactionID>22659fdc9726c47e</serverTransactionID>

<tokenType>"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss- X509-token-
profile-1.0#PKCS7</tokenType">

<requestedVSSecurityToken>

      <binarySecurityToken EncodingType="http://docs.oasis-open.org/
wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd#base64binary "
ValueType="http://docs.oasis-open.org/wss/2004/01/

oasis-200401-wss-X509-token-profile-1.0#PKCS7">
MIAGCSqGSIb3DQEHAqCAMIACAQExADALBgkqhkiG9w0BBwGggDCCBDwwggMkoAM
CAQICEDHkCfLBhPIT3b/gJDXoTb0wDQYJKoZIhvcNAQEFBQAwcTELMAkGA1UEBh
MCVVMxFzAVBgNVBAoTDlZlcmlTaWduLCBJbmMuMR8wHQYDVQQLExZGT1IgVEVTV
CBQVVJQT1NFUyBPTkxZMSgwJgYDVQQDEx9NYW5nbXVuIFBLSSAtIDEgTGV2ZWwg
LSBUZXN0IENBMB4XDTExMDgxMjAwMDAwMFoXDTExMDgyNjIzNTk1OVowUzEiMCA
GA1UEAwwZZWd3YXV0b3VzZXIxIGVnd2F1dG91c2VyMTEVMBMGA1UECwwMVlBOLV
dJRkktV0VCMRYwFAYDVQQLDA1NVUxUSS1BTExPV0VEMIIBIjANBgkqhkiG9w0BA
QEFAAOCAQ8AMIIBCgKCAQEA40REn1333FTb2V6M9wXgyVtoQ/V/Y7MOm17wi8h5

dsSln7zPv8db4CREFONU0pLHHMS1wMr2uu3CCpbpv4kHEnYnSONrFgbo4viTtQD
LZpjqBDBYy+KRQgqU7GvpIwq1&81es/kkxwvfpXOM89D7ofHlxm+BlkDPHRgbd4
dU6ndG9y9CoDy+Yn1XFoWhgeunOlAOLShwq3Mj3IcgQI+GpzSlgEfJVGEWqBOeY
X3r2BRYYu8JqYMmahJFYo5+USW/6MRKMQjGMRySv27Awv/JXeTCxwkrKQEgCnhz
H9wIDAQABo4HtMIHqMAwGA1UdEwEB/wQCMAAwDgYDVR0PAQH/BAQDAgPoMBYGA1
```

```
UdJQEB/wQMMAoGCCsGAQUFBwMCMB8GA1UdIwQYMBaAFFSVnuKaoEec7wlhCvyTv
AJpn+t0MF8GA1UdHwRYMFYwVKBSoFCGTmh0dHA6Ly9mdC1lbnQtY3JsLmJidGVz
dC5uZXQvY2FfNDg5NmVmMDIzMDMzZTBiYTYyMjVlOWRlZTQ1MWEyNDYvTGF0ZXN
0Q1JMLmNybDAwBgpghkgBhvhFARADBCIwIAYTYIZIAYb4RQEQAQIDAQGCnvPNOB
YJNjAwMDEzMzk2MA0GCSqGSIb3DQEBBQUAA4IBAQA5Ddn5zs91OyuagXIO5X4z7

STJm4cZQW70dEzqy6YGrtkZ3+Yxm0as06Z/IP8A8sn0ePVUDfwh2gjn5pB80Zkb
a1aUGT4B+1UWCMcCQHwDx0kBSZHUjSVJ7WuaCvnkHDnapaijTQ3d3vYAktNhQuC
2rbnUYMFzBXWjUWyarpgWd0n6xlq2Rb+O+VyUH9F+37R9+1zIresivy/DpIMlFA
GL5qTCpKt8qLg1TeCRJ+OFTuOlpIqDcNLwB+W+Z2J5m75i2zOKPlKRB6Bh+rsvH
3pQcaiOYDPbdfAD0jnwxpAVwt8t3PiU9Qnnn6B5==

      </binarySecurityToken>

</requestedVSSecurityToken>

<version>2.0</version>

</RequestVSSecurityTokenResponse></RequestSecurityTokenResponse>
```

Sample Renewal with Key Escrow Requests and Responses

Use RequestSecurityToken and RequestSecurityTokenResponse to request a certificate enrollment that escrows the key. Requests for certificate renewals where the key is escrowed are sent to the Key Management Server that is deployed at the enterprise site.

Sample Renewal with Key Escrow (RequestSecurityToken) Request

The following is a sample RequestSecurityToken request with key escrow.

See "About the Certificate Enrollment Protocol" on page 134.

```
<ns9:RequestSecurityToken xmlns:ns9="http://docs.oasis-open.org/ ws-sx/ws-
trust/200512/">

      <requestVSSecurityToken xmlns="http://schemas.verisign.com/
      pkiservices/2009/07/enrollment" preferredLanguage="en-US">

      <certificateProfileID>Certificate Renew</certificateProfileID>

      <clientTransactionID>20110712190102</clientTransactionID>

      <tokenType>http://schemas.verisign.com/pkiservices/2009/07/
      PKCS12</tokenType>

      <requestType>http://docs.oasis-open.org/ws-sx/ws-trust/200512/
      Renew</requestType>

<version>2.0</version>

      <binarySecurityToken xmlns:axis2ns1="http://docs.oasis-open.org/ws-sx/ ws-
trust/200512/" ValueType="http://docs.oasis-open.org/wss/2004/01/ oasis-200401-
wss-X509-token-profile-1.0#X509v3" axis2ns1:EncodingType= "http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity- secext-1.0.xsd#base64binary">
MIIF9TCCBN2gAwIBAgIQXZGPsgQBVVt4FEdD3XJy5jANBgkqhkiG9w0BAQUFADCB/DE
LMAkGA1UEBhMCVVMxHTAbBgNVBAoTFFMIIF9TCCBN2gAwIBAgIQXZGPsgQBVVt4FEdD
3XJy5jANBgkqhkiG9w0BAQUFADCB/DELMAkGA1UEBhMCVVMxHTAbBgNVBAoTFFN5bWF
```

```
udGVjIENvcnBvcmF0aW9uMR8wHQYDVQQLExZGT1IgVEVTVCBQVVJQT1NFUyBPTkxZMR
8wHQYDVQQLExZWZXJpU2lnbiBUcnVzdCBOZXR3b3JrMUIwQAYDVQQLEzlUZXJtcyBvZ

iB1c2UgYXQgaHR0cHM6Ly93d3cudmVyaXNpZ24uY29tL2Nwcy90ZXN0Y2EgKGMpMTEx
SDBGBgNVBAMTP1N5bWFudGVjIEMyIFNoYXJlZCBJbnRlcm1lZGlhdGUgVEVTVCBDZXJ
0aWZpY2F0ZSBBdXRob3JpdHkgLSBRQTAeFw0xMTA4MTIwMDAwMDBaFw0xMTA4MTkyMz
U5NTlaMHMxGDAWBgNVBAMMD2Vnd2F1dG8gZWd3YXV0bzEPMA0GA1UECwwGUy9NSU1FM
Q4wDAYDVQQKDAVXU05FVzE2MDQGA1UEAwwtZWd3YXV0byBlZ3dhdXRvV29YRHhmaXhn
eVB1UHdpZm9OQlotMjE5OTE1MjM1MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgK

CAQEAwcjpFuNSiQAAcj6LhZ8+CCryPy+lW60v2mwwSuZt6QI2kVtH/LdfTH1HkqUHoK

Br6lRjyZ9i8yELIw2lcaqaMwTKf7T2TmmtLJtJRJR4kiZbRBPwcqJjkzs1XNvLTqWt2
GhQrMTTfB3NQWM14Y03xTG1iEwdViidy6MhWX5NklHhhQwdY6ZcawVyKptm/TQF17K8
NT7H+hiZiUnySNHQvmjpsKXGt0tn0NM04ob39jRL7JFUId5mdxnKcVpsLlIbq+W/hr6
UgwdJZfHNNtOEQkiNvOap+6sR5KDdmQo3IY6mDReUlUTop05zgQGmBFfe+kan1fKmMR
DfjGGI7Ou9+wIDAQABo4IB+TCCAfUwDAYDVR0TAQH/BAIwADAOBgNVHQ8BAf8EBAMCB
aAwFgYDVR0lAQH/BAwwCgYIKwYBBQUHAwQwGgYDVR0RBBMwEYEPZWd3YXV0b0BlZ3cu
Y29tMB8GA1UdIwQYMBaAFKt41suOHxUtkFc9f9hwG0X3P4QkMDsGCCsGAQUFBwEBBC8

wLTArBggrBgEFBQcwAYYfImh0dHA6Ly9mdC1lbnQtb2NzcC5iYnRlc3QubmV0IjBfBg
NVHR8EWDBWMFSgUqBQhk5odHRwOi8vZnQtZW50LWNybC5iYnRlc3QubmV0L2NhXzMxM
TRlZTMzMDNjY2EwYWFlMzVkNmRiNTk0NWI0ZmNhL0xhdGVzdENSTC5jcmwwbAYDVR0g
BGUwYzBhBgtghkgBhvhFAQcXAjBSM

      </binarySecurityToken>

      </requestVSSecurityToken>

</ns9:RequestSecurityToken>
```

Sample Renewal with Key Escrow (RequestSecurityTokenResponse) Response

The following is a sample RequestSecurityTokenResponse response with key escrow.

See “About the Certificate Enrollment Protocol” on page 134.

```
<RequestSecurityTokenResponse xmlns="http://docs.oasis-open.org/ ws-sx/ws-
trust/200512/">

      <RequestVSSecurityTokenResponse xmlns="http://schemas.verisign.com/
      pkiservices/2009/07/enrollment">

      <clientTransactionID>20110712190102</clientTransactionID>

      <serverTransactionID>d3ce7d2e003e625f</serverTransactionID>

      <tokenType>http://schemas.verisign.com/pkiservices/2009/07/
      PKCS12</tokenType>

      <requestedVSSecurityToken>

      <binarySecurityToken EncodingType="http://docs.oasis-open.org/wss/
2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd#base64binary

" ValueType="http://schemas.verisign.com/pkiservices/2009/07/PKCS12">
MIIa8gIBAzCCGqwGCSqGSIb3DQEHAaCCGp0EghqZMIIalTCCBe4GCSqGSIb3DQEHAa
CCBd8EggXbMIIF1zCCBdMGCyqGSIb3DQEMCgECoIIE7jCCBOowHAYKKoZIhvcNAQwB
```

```
AzAOBAi1Qtg/OPSSFwICBAAEggTIhM6KIBBRViUh3pbB5tm//Oh0zQuSqw4Qs6fj2H
O71bKW+7zM53asVTIExobY1bqzNqhFvwedeqtdAlQv9gmQTsN6NH7lrBvkLCXBnE1x
5bsdKcUUoc3qehGuZfZ5AZaiE3H+hW8wkFOpKfvRNkTazY/9AwJJl1+cs8BtsalcYO
sr2IpnGtBH0C3UELxH3N03xSE0MRY+13fOddOcE+VMufswr2VEDKqdIcmU9cxHvElz
9NL1X+OV3MoGM1OvGdxZgIoLltXVimQ/PeT9B2edbeDTPvMTPwTVCWIj/WCbf4xU6+
Th4KxrOvanB5bz8s1hpiFTxUYjjX2G+ir3klJlnxtvVGg7G+MOxSCD+VdY9UjjIB9Y
IWMtCRmNqpRgprDdtGEWF6i+3LHDufVjVItEcp2HpLjvWEc80xB37SjHNZ8Dclta87
jgKmed8wpp+y0nHBnS8EpAELHmM2unLgEI8MJOliAZIfHq+6yYpxHts/wF0HEYRZ7G

tjr2D45MAlxY2Ac8BqDSTtk3fvoOkw/PqY8XgTOKWIQHzpxjJzqPDF4DJO83uPHmbP
33BNAOSVwy6AOSWs7TpsGFTZmWgpOHmucc8dOUrqMg7oxDdbgWwCIgwC26EnRpOZjw
F+QOIgcYEB5qWPRBVzIzl594mW3wEzDPvH83MpPEP9r5Bk+7SRB1ICfXF30yxe9EyR

vL5LdDMLipc6oqNpBxWV/K6im37T6DK1rlyEJmvV0vwbogF01op9gEEP+BlTQaoeLl
n2VHqdiGYs/PvlFLbEJaQm2nlKyEoFRqpM4KwKOm6q4pJgskXhBAdNDkDvfQVWORE2
fSSBpCHPwqdXYtH5SKSuwoX7NrACrzZFNbRnZDcPBLvCDatsFPy8ZStcSBDfDkqhy2
ZLxuSwi2BnXH/v1jkiuRtnJk/4HqtUkWzEGfAdCLb3yGJxqRPeVtg81G1ssJb9Wl7Z
QS3dTa7g78kTMvMzZdS6D6k/M3Q24OndfzYitaH7916tqOVBwfExg6v+NGpWa4w7fj
CmRU6+AEHHIFzBgXN+dDPA9sIq01EiRPAS7JOs5luDuCJayGYuHdCk/S30pfFQqwCx
hmIwPjnZCa21eQ3/u6VPiu4mTfJ4cMghKfossyB7KyEQsupuB0ONN6jlOACP9ymY/x
r8aX6tpDgerEpCAGg22c

      </binarySecurityToken>

      <pKCS12Password>EOSkinhaOs4q</pKCS12Password>

</requestedVSSecurityToken>

<version>2.0</version>

</RequestVSSecurityTokenResponse>

</RequestSecurityTokenResponse>
```

5.4 Certificate Management Protocol

The Certificate Management Protocol Web Service method is used to manage the certificates that have been issued to end users. This protocol includes the following functions:

- See “Searching for Certificates” on page 59.
- See “Changing the Status of a Certificate” on page 60.
- See “Recovering a Certificate” on page 60.

5.4.1 Searching for Certificates

This protocol allows the RA application to search for certificates based on specified search criteria.

Wildcards are supported for some search criteria. However, if you do a search which includes an asterisk as a literal character, the protocol treat it as a wildcard.

The response for a search request includes the number of matches that are found, as well as any certificates found (in base64-encoded x.509 format). It only returns the maximum configured number of results.

Creating Intelligent Searches

Depending upon the size of the database, search operations can take some time to complete. To avoid the chance of a search timing out, you should narrow your search criteria to return more accurate results. For example:

- Do not search across an entire account. Restrict searches to specific certificate profiles.
- Use seat IDs, serial numbers, and similar identifying information to further narrow searches.
- Wherever possible, search for exact matches. Do not use partial strings if you can use the entire search string. As an example, enter a user's full name rather than just the first name to search for a user.

5.4.2 Changing the Status of a Certificate

By default, an issued certificate is valid. The certificate can be used for normal operations (signing, encryption, or authentication, based on the certificate usage). This protocol allows the RA application to request a change to the status of a certificate, or set of the certificates that share a common seat ID, (updateCertificateStatus) to revoked. A revoked certificate can no longer be used. Set this status for the certificates that have been permanently lost, stolen, or compromised. Certificates with a revoked status cannot be returned to a valid status.

The response for a request to change status is a success message.

5.4.3 Recovering a Certificate

If you have escrowed your end-user private keys with the key escrow option, you can recover them in the event the keys become lost or unusable.

If you have imported your certificates in PKI Manager, you can recover your private keys.

The response for a request to recover a key is a password-protected PKCS#12 message, and password.

5.4.4 Certificate Management Protocol Flow

The steps for certificate management are:

- See “Search for Certificates” on page 61.
- See “Change Certificate Status” on page 61.
- See “Certificate Recovery (Escrow Set to DigiCert)” on page 62.
- See “Certificate Recovery (Escrow Set to Local User Store)” on page 62.
- See “CRL Checking and Changing the Certificate Status” on page 62.

5.4.4.1 Search for Certificates

1. Send a searchCertificateStatus call to DigiCert PKI identifying the search filter criteria.
2. If the request is authenticated and correct, DigiCert PKI performs the search. Otherwise, DigiCert PKI returns a SOAP Fault message.
3. The response provides the result of the request as a success message, which includes information about certificates found.

5.4.4.2 Change Certificate Status

1. Send an updateCertificateStatus call to DigiCert PKI identifying the certificates and specifying the new status. You can identify the certificates to be revoked, suspended or resumed by the certificate serial number or by the end user's unique identifier, or seat ID. This operation revokes, suspends, or resumes the certificate within the account and any sub-accounts.

   - Use certificate serial number to identify a single certificate. Any certificate that is not already revoked, suspended, or resumed can be revoked, suspended, or resumed by certificate serial number.
   - Use seat ID to identify one or more certificates. Only valid certificates or the certificates that have expired within the last 30 days can be revoked by seat ID.

2. If the request is authenticated and correct, DigiCert PKI updates the certificate status.

   - If the certificate serial number was used to identify the certificate, and the certificate was already revoked, suspended, or resumed, or no certificate was found, DigiCert PKI returns a SOAP Fault message.
   - If the seat ID was used to identify the certificates, DigiCert PKI always returns a success message and a count of the certificates revoked.

3. The response provides the result of the request as a success message.

NOTE: Revoke, Suspend, and Resume are the supported operations for updateCertificateStatus.

5.4.4.3 Certificate Recovery (Escrow Set to DigiCert)

1. The RA application sends a RequestKeyRecovery call, identifying the certificate to be recovered and including the administrator ID, to the PKI Web Service.
2. If the request is authenticated and correct, the PKI Web Service begins the certificate recovery process.
3. The DigiCert PKI generates the private key. The PKI Web Service securely processes the key recovery request and returns a PKCS #12 message and password, which is delivered back to the enterprise RA application.
4. The response provides the result (success or failure message) of the request. Provide this message and password to your end user or other authorized user in a secure manner.

5.4.4.4 Certificate Recovery (Escrow Set to Local User Store)

We support Single Administrator Approval or Dual Administrators Approval for Key recovery request over API.

This setting is based on Certificate profile. If profile indicate it is Dual Administrators Approval Required, you need two different Administrator approval API call to recover certificate.

Single Administrator Approval mode

1. The RA application sends a RequestKeyRecovery call, identifying the certificate to be recovered and including the administrator ID, to the key escrow and recovery service.
2. If the request is authenticated and correct, the key escrow and recovery service begins the certificate recovery process.
3. The key escrow and recovery service construct a PKCS#12 message and password, which is delivered back to the enterprise RA application.
4. The response provides the result (success or failure message) of the request. Provide this message and password to your end user or other authorized user in a secure manner.

Dual Administrators Approval mode

1. The RA application sends 1st RequestKeyRecovery call, identifying the certificate to be recovered and including the administrator ID, to the key escrow and recovery service.
2. If 1st request is authenticated and correct, return additional Administrator approval required message.

3. The RA application sends 2nd RequestKeyRecovery call, need identifying the certificate to be recovered and including another administrator ID, to the key escrow and recovery service.
4. If 2nd request is authenticated and correct, the key escrow and recovery service begins the certificate recovery process.
5. The key escrow and recovery service construct a PKCS#12 message and password, which is delivered back to the enterprise RA application.
6. The response provides the result (success, pending or failure message) of the request. Provide this message and password to your end user or other authorized user in a secure manner.

5.4.5 CRL Checking and Changing the Certificate Status

Once a certificate has been revoked, it cannot be returned to a valid status. Use of the certificate revocation list (CRL) determines if a certificate is valid or revoked.

For suspended certificates, CRL checking determines if a certificate is valid or suspended.

5.4.6 Important Elements, Requests, and Responses

The Certificate Management Protocol supports multiple functions. The values you send for each function (and the response you receive) depends on the type of request. Table 5-2 describes the common Certificate Management Protocol requests and responses.

Table 5-2 Common Certificate Management requests and responses

	Requests		Responses	
	Elements	Namespace	Response	Operation
Search	• Operation type • seatId • accountId • profileOID • commonName • status • emailAddress • serialNumber • issuingCA • validFrom	http://schemas. verisign.com/pkiservices/2009/07/management	• Success Message • Certificate count for the search request • Whether more than the maximum number of	searchCertificate operation See "Sample Search Certificates Requests and Responses" on page 64.

	Requests		Responses	
	• validTo • startIndex		results are found • Information for each certificate	
Revoke/ Suspend/ Resume	• Operation Type • certificateIssuer • certificateSerialNumber • seatID • revocationReason	http://schemas. verisign.com/ pkiservices/2009/07/ management	• Success code • Success Message • Number of certificates revoked	updateCertificate Status operation See “Sample Change Status Requests and Responses” on page 66.
Recover	• Operation Type • certificateIssuer • certificateSerialNumber • adminID (administrator certificate in x.509 format)	http://schemas. verisign.com/ pkiservices/2009/07/ management	PKCS#12 and password	keyRecovery operation See “Sample Recover Key Requests and Responses” on page 70.

5.4.7 Sample Certificate Management Protocol Requests and Responses

This section provides sample requests and responses for the following scenarios.

- See “Sample Search Certificates Requests and Responses” on page 65.
- See “Sample Change Status Requests and Responses” on page 67.
- See “Sample Recover Key Requests and Responses” on page 71.

Sample Search Certificates Requests and Responses

Use searchCertificate to search for certificates based on specified criteria.

Sample Search Certificates (searchCertificateRequest) Request

The following is a sample searchCertificateRequest request. This sample request returns one certificate.

See “About the Certificate Management Protocol” on page 159.

```
<?xml version="1.0" encoding="UTF-8"?>

<ns2:searchCertificateRequest xmlns:ns2="http://schemas.verisign.
com/pkiservices/2009/07/management">

      <ns2:clientTransactionID>123456787</ns2:clientTransaction ID>

            <ns2:seatId>jason_roberts@acme.com</ns2:seatId>

<ns2:version>1.0</ns2:version>

</ns2:searchCertificateRequest>
```

Sample Search Certificate (searchCertificateResponse) Response

The following is a sample searchCertificateResponse response.

See “About the Certificate Management Protocol” on page 159.

```
<?xml version="1.0" encoding="UTF-8"?>

<ns2:searchCertificateRequest xmlns:ns2="http://schemas.
verisign.com/pkiservices/2009/07/management">

      <ns2:clientTransactionID>1337035116890</ns2:clientTransaction ID>

      <ns2:serverTransactionID>a70d6c67d8bbc38e</ns2:server TransactionID>

      <ns2:certificateCount>1</ns2:certificateCount>

      <ns2:certificateList>

            <ns2:certificateInformation>

<ns2:certificate>MIIGrjCCBZagAwIBAgIQFbKUv2WN9Aqut0W9gH8z2DA
NBgkqhkiG9w0BAQUFADCB/DELMAkGA1UEBhMCVVMxHTAbBgNVBAoTFFN5bWF
```

```
udGVjIENvcnBvcmF0aW9uMR8wHQYDVQQLExZGT1IgVEVTVCBQVVJQT1NFUyB
PTkxZMR8wHQYDVQQLExZWZXJpU2lnbiBUcnVzdCBOZXR3b3JrMUIwQAYDVQQ

LEzlUZXJtcyBvZiB1c2UgYXQgaHR0cHM6Ly93d3cudmVyaXNpZ2,o4kws9je
ry90ZXN0Y2EgKGMpMTExSDBGBgNVBAMTP1N5bWFudGVjIEMyIFNoYXJlZCBJ

bnRlcm1lZGlhdGUgVEVTVCBDZXJ0aWZpY2F0ZSBBdXRob3JpdHkgLSBRQTAe
Fw0xMjAyMDcwMDAwMDBaFw0xMjAyMTkyMzU5NTlaMDoxEjAQBgNVBAMMCXVz
cjMgd2ViMzEPMA0GA1UECwwGUy9NSU1FMRMwEQYDVQQKDApkaXBFR1dNKmn6

UF9tB2UuQyFWahHNVw1hN6RbKltCWP4qDbMsMIKeKKgwm7s7IpBhjCOOUEN0
InujqKGyW+qnD/YJdpmjWSkyZqw7+i8L7FaFCW9CLcWJorU5YKdeHFAJWVL+
uru8B6n9zxnyXZoo1JpVHLvm2BHd7EILdh3qHC04zFVSOEOiqjlhQ58a82Hh
ewiDdHHnSGPnqiPTboe61YnrnDD7wH3fmXqbKxMxm+3Atr3oT4evL49CNjjw
L+ePO5/cZERU+/VaihlfqfbscyCiqefnwOnTmMQaEt5KzhHv35TCO8VDtXVB

XzemPw85w83W25XUsoIxQIDAQABo4IC6zCCAucwDAYDVR0TAQH/BAIwADAOB
gNVHQ8BAf8EBAMCBaAwFgYDVR0lAQH/AwwCgYIKwYBBQUHAwQwHQYDVR0OBB

YEFFaFSmgtcRDwfRFiqt3Vq8LJCLn5MB8GA1UdEQQYMBaBFHdlYnVzcjNAcG
9va21haWwuY29tMB8GA1UdIwQYMBaAFKt41suOHxUtkFc9f9hwG0X3P4QkMH
QGCCsGAQUFBwEBBGgwZjApBggrBgEFBQcwAYYdaHR0cDovL2Z0LWVudC1vY3
NwLmJidGVzdC5uZXQwOQYIKwYBBQUHMAKGLWh0dHA6Ly9FVkludGwtYWlhLn
ZlcmlzaWduLmNvbS9FVkludGwyMDA2LmNlcjCB8wYDVR0fBIHrMIHoMIHloI
HioIHfhk5odHRwOi8vZnQtZW50LWNybC5iYnRlc3QubmV0L2NhXzMxMTRlZT
MzMDNjY2EwYWFlMzVkNmRiNTk0NWI0ZmNhL0xhdGVzdENSTC5jcmyGgYxsZG
FwOi8vZnQtZW50LWRpcmVjdG9yeS5iYnRlc3QubmV0L081MjAlM0QlMjBTeW
1hbnRlYyUyMENPcnAlMkMlMjBDJTIwJTNEJTIwVUslMkMlMjBDTiUyMCUzRC
UyMExEQVBDRFBUZXN0MT9jZXJ0aWZpY2F0ZXJldm9jYXRpb25saXN0O2Jpbm
FyeTBsBgNVHSAEZTBjMGEGC2CGSAGG+EUBBxcCMFIwJgYIKwYBBQUHAgEWGm

h0dHA6Ly93d3cuc3ltYXV0aC5jb20vY3BzMCgGCCsGAQUFBwICMBwaGmh0dH
A6Ly93d3cuc3ltYXV0aC5jb20vcnBhMEIGCSqGSIb3DQEJDwQ1MDMwCgYIKo
ZIhvcNAwcwCwYJYIZIAWUDBAEChe8n2IUneiUjnnka(I8kH87htyu8UY9BkW
CTYwMDAt56TmNFt5iA03foqe/UcL6ru6Xon/f4FB9W/rSygRR1Xdc7Ui48+U
icm3dnpzQJyXE/xP3SlWzW7aeNJWYx8pFKWa8i8573BEnTkHd8ZF1JrU0NZw
3X7giUXGbheT9pzaoEAHMcaCb7TLb5Xpv53iJKrpQ0gD7H8LbWR0IwzIBoFe
IzJeigyv2mclobzPjuZ1AbPJnEr/3FH4aQRKUd0ORXqg5JCrkoI2SzsluS6Y
Zd+t2PhLY9+GYVR==</ns2:certificate>

            <ns2:seatId>webusr3@pookmail.com</ns2:seatId>

            <ns2:commonName>USR3 WEB3</ns2:commonName>

            <ns2:accountId>600044916</ns2:accountId>

            <ns2:profileOID>2.16.840.1.113733.1.16.1.2.2.1.1.

6049116

      41</ns2:profileOID>

            <ns2:emailAddress>webusr3@pookmail.com</ns2:email Address>

            <ns2:status>EXPIRED</ns2:status>

            <ns2:validFrom>1328572800</ns2:validFrom>
            <ns2:validTo>1329695999</ns2:validTo>
            <ns2:serialNumber>15b294bf658df40aaeb745bd807f33d8<
/ns2:
      serialNumber>
            <ns2:isEscrowed>true</ns2:isEscrowed>
            </ns2:certificateInformation>
      </ns2:certificateList>
      <ns2:moreCertificateAvailable>false</ns2:moreCertificate Available>
      <ns2:version>1.0</ns2:version>
      </ns2:searchCertificateRequest>
```

Sample Change Status Requests and Responses

By default, issued certificates are in a valid status. Use updateCertificateStatus to change the status of (revoke, suspend, or resume) certificates.

Suspend or **Resume** operations are no longer supported for certificates issued under Public CA hierarchy.

Sample Revoke Certificate (updateCertificateStatusRequest) Request

The following is a sample updateCertificateStatus request with an operationType of Revoke. This sample request revokes one certificate.

For operationType suspend and resume, replace the operation type with **Suspend** or **Resume**. For Suspend, the revocation reason is 'CertificateHold' and Resume does not have a reason.

See "About the Certificate Management Protocol" on page 159.

```
<?xml version="1.0" encoding="UTF-8"?>
<ns2:updateCertificateStatusRequest xmlns:ns2="http://schemas.
verisign.com/pkiservices/2009/07/management">
      <ns2:clientTransactionID>123456787</ns2:clientTransactionID>
      <ns2:version>1.0</ns2:version>
      <ns2:certificateIssuer>"CN =DigiCert PKI - 1 Level,OU = FOR TEST PURPOSES
ONLY,O = "VeriSign, Inc.",C=US"
</ns2:certificateIssuer>
      <ns2:revocationReason>KeyCompromise</ns2:revocationReason>
      <ns2:comment>Key has been compromised</ns2:comment>
      <ns2:certificateSerialNumber>695c2c0159f0fb5193728882312b93eb
</ns2:certificateSerialNumber>
```

```
      <ns2:operationType>Revoke</ns2:operationType>
</ns2:updateCertificateStatusRequest>
```

Sample Revoke Certificate (updateCertificateStatusResponse) Response

The following is a sample updateCertificateStatusResponse response for the Revoke operationType.

See “About the Certificate Management Protocol” on page 159.

```
<?xml version="1.0" encoding="UTF-8"?>

<ns2:updateCertificateStatusResponse xmlns:ns2="http://schemas.
verisign.com/pkiservices/2009/07/management">

      <ns2:clientTransactionID>123456787</ns2:clientTransactionID>

      <ns2:serverTransactionID>ed7ce6383eaea7fb</ns2:server TransactionID>

      <ns2:version>1.0</ns2:version>

      <ns2:successCode>0</ns2:successCode>

      <ns2:successMsg>SUCCESS</ns2:successMsg>

</ns2:updateCertificateStatusResponse>
```

Sample Revoke Multiple Certificates (updateCertificateStatusRequest) Request

The following is a sample updateCertificateStatus request with an operationType of Revoke. This sample request revokes all certificates that are assigned to a specified seatID (a user's email address in this sample).

See “About the Certificate Management Protocol” on page 159.

```
<?xml version="1.0" encoding="UTF-8"?>

<ns2:updateCertificateStatusRequest xmlns:ns2="http://schemas.
verisign.com/pkiservices/2009/07/management">

      <ns2:clientTransactionID>123456787</ns2:clientTransactionID>

      <ns2:version>1.0</ns2:version>

      <ns2:certificateIssuer>"CN =DigiCert PKI - 1 Level,OU = FOR TEST PURPOSES
ONLY,O = "VeriSign, Inc.",C=US"

</ns2:certificateIssuer>

      <ns2:revocationReason>KeyCompromise</ns2:revocationReason>

      <ns2:comment>Key has been compromised</ns2:comment>

<ns2:seatID>jason_roberts@acme.com</ns2:seatID>

      <ns2:operationType>Revoke</ns2:operationType>

</ns2:updateCertificateStatusRequest>
```

Sample Revoke Multiple Certificates (updateCertificateStatusResponse) Response

The following is a sample updateCertificateStatusResponse response for the Revoke operationType.

See "About the Certificate Management Protocol" on page 159.

```
<?xml version="1.0" encoding="UTF-8"?>

<ns2:updateCertificateStatusResponse xmlns:ns2="http://schemas.
verisign.com/pkiservices/2009/07/management">

      <ns2:clientTransactionID>123456787</ns2:clientTransactionID>

      <ns2:serverTransactionID>ed7ce6383eaea7fb</ns2:server TransactionID>

      <ns2:version>1.0</ns2:version>

      <ns2:successCode>0</ns2:successCode>

      <ns2:successMsg>SUCCESS</ns2:successMsg>

<ns2:revocationCount>3</ns2:revocationCount>

</ns2:updateCertificateStatusResponse>
```

Sample Bulk Revoke by Certificate Serial Number Request

The following is a sample request for a bulk revoke by Certificate Serial Number. You can enter up to 100 certificate serial numbers to revoke multiple certificates in one request.

See "About the Certificate Management Protocol" on page 159.

```
<?xml version="1.0" encoding="UTF-8"?>

<ns2:bulkUpdateCertificateStatusRequest xmlns:ns2="http://schemas.
verisign.com/pkiservices/2009/07/management">

      <ns2:clientTransactionID>123456789</ns2:clientTransactionID>

      <ns2:version>1.0</ns2:version>

      <ns2:revocationReason>KeyCompromise</ns2:revocationReason>

      <ns2:comment>key is compromised</ns2:comment>

      <ns2: certificateSerialNumber>1816d47753810dff6ee949455615d7d7</
ns2:certificateSerialNumber>

      <ns2:certificateSerialNumber>7a3e0dc19faae63c5aebbe3d3eb227d7</
ns2:certificateSerialNumber>

      <ns2:certificateSerialNumber>sn-33</ns2:certificateSerialNumber>

      <ns2:operationType>Revoke</ns2:operationType>

</ns2:bulkUpdateCertificateStatusRequest>
```

Sample Bulk Revoke by Certificate Serial Number Response

The following is a sample response for a bulk revoke by Certificate Serial Number.

See "About the Certificate Management Protocol" on page 159.

```
<?xml version="1.0" encoding="UTF-8"?>

<ns2:bulkUpdateCertificateStatusResponse xmlns:ns2="http://schemas.
verisign.com/pkiservices/2009/07/management">

      <ns2:clientTransactionID>123456789</ns2:clientTransactionID>

      <ns2:serverTransactionID>1b987c48a52f956c</ns2:serverTransactionID>

      <ns2:version>1.0</ns2:version>

      <ns2:successCode>0</ns2:successCode>

      <ns2:successMsg>SUCCESS</ns2:successMsg>

      <ns2:revocationCount>2</ns2:revocationCount>

</ns2:bulkUpdateCertificateStatusResponse>
```

Sample Bulk Revoke by Certificate Seat ID Request

The following is a sample request for a bulk revoke by Certificate Seat ID. You can enter up to 100 Certificate Seat IDs to revoke multiple certificates in one request.

See "About the Certificate Management Protocol" on page 159.

```
<?xml version="1.0" encoding="UTF-8"?>

<ns2:xmlns:ns2="http://schemas.verisign.com/pkiservices/2009/ 07/management">

      <ns2:clientTransactionID>123456789</ns2:clientTransactionID>

      <ns2:version>1.0</ns2:version>

      <ns2:revocationReason>KeyCompromise</ns2:revocationReason>

      <ns2:comment>key is compromised</ns2:comment>

      <ns2:seatId>john_huang@digicert.com</ns2:seatId>

      <ns2:seatId>2013-04-17-scep1</ns2:seatId>

      <ns2:seatId>seatid-3333</ns2:seatId>

      <ns2:operationType>Revoke</ns2:operationType>

</ns2:bulkUpdateCertificateStatusRequest>
```

Sample Bulk Revoke by Certificate Seat ID Response

The following is a sample response for a bulk revoke by Certificate Seat ID.

See "About the Certificate Management Protocol" on page 159.

```
<?xml version="1.0" encoding="UTF-8"?>

<ns2:bulkUpdateCertificateStatusRequest xmlns:ns2="http://schemas.
verisign.com/pkiservices/2009/07/management">

      <ns2:clientTransactionID>123456789</ns2:clientTransactionID>

      <ns2:serverTransactionID>a927d2685358a5fb</ns2:serverTransactionID>

      <ns2:version>1.0</ns2:version>

      <ns2:successCode>0</ns2:successCode>

      <ns2:successMsg>SUCCESS</ns2:successMsg>

      <ns2:revocationCount>14</ns2:revocationCount>

</ns2:bulkUpdateCertificateStatusResponse>
```

Sample Recover Key Requests and Responses

Use requestKeyRecoveryMessage to recover a key. This section provides sample request and responses for key recovery requests.

Requests for key recovery are sent to the PKI Web Service.

Recover Key (requestKeyRecoveryMessage) Request

The following is a sample requestKeyRecoveryMessage request with single Administrator Approval mode for key recovery.

See "About the Certificate Management Protocol" on page 159.

```
<?xml version="1.0" encoding="UTF-8"?>

<ns2:requestKeyRecoveryMessage xmlns:ns2="http://schemas.
verisign.com/pkiservices/2009/07/management">

      <ns2:clientTransactionID>123456789</ns2:clientTransactionID>

      <ns2:certificateSerialNumber>24021485fced686cde8f1ba66868daf1

</ns2:certificateSerialNumber>

      <ns2:certificateIssuer>"CN =DigiCert PKI - 1 Level - Test CA,OU = FOR TEST
PURPOSES ONLY,O = "VeriSign, Inc. ",C=US"</ns2:

certificateIssuer>

      <ns2:adminID>MIIFkTCCBHmgAwIBAgIQfOZPxEdbt5Xnn/

+dXoy7bTANBgkqhkiG9w0BAQUFADCB+zELMAkGA1UEBhMCVVMxHTAbBgNVBAo
TFFN5bWFudGVjIENvcnBvcmF0aW9uMR8wHQYDVQQLExZGT1IgVEVTVCBQVVJQ
T1NFUyBPTkxZMR8wHQYDVQQLExZWZXJpU2lnbiBUcnVzdCBOZXR3b3JrMUIwQ
```

```
AYDVQQLEzlUZXJtcyBvZiB1c2UgYXQgaHR0cHM6Ly93d3cudmVyaXNpZ24uY2
9tL2Nwcy90ZXN0Y2EgKGMpMTExRzBFBgNVBAMTPlN5bWFudGVjIEMzIEFkbWl

uIEludGVybWVkaWF0ZSBURVNUIENlcnRpZmljYXRlIEF1dGhvcml0eSAtIFFB
MB4XDTExMDUwOTAwMDAwMFoXDTEyMDUwODIzNTk1OVowgZQxJjAkBgNVBAMMH

VdTQXV0bzIwMTEwNTEwIFdTQXV0bzIwMTEwNTEwMSkwJwYJKoZIhvcNAQkBFh
p3c2F1dG8yMDExMDUxMEB5b3BtYWlsLmNvbTEXMBUGA1UECgwOV1NBdXRvMjA
xMTA1MTAxDjAMBgNVBAsMBUFETUlOMRYwFAYDVQQLDA1NVUxUSS1BTExPV0VE
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAve7B/kMQ3W9516wPh

2x+n7OakWlU3nRAwka3Hh0H9JytoI5rhMpbCtyK654GcjTvyKmKT6dYaEJH/V
bIdmYVJLJXgKg1I7V4atb6SaQoBsA+rIzktpKn/eS0jM+xkZSpTRpc0oqJ/UK
gG+4WnMibofC2s2C2hAf1Co6m6f8Jycp9jzbPHBPL8815ejbwIqOKfYNIbP/w
5hHf3AENJ1HiDXDZaKVkPS5kTMWxXaK14G2sm5qpfe8ZX3EVameVFS1BaSDyX
kisPBY4LzHuklIAFiN7LEmQ9XLEfa3onFQV7v7QBwlFY1tImvSyNHc5KN9z/K
3Ofc4EnPP3z9WutXYewIDAQABo4IBdDCCAXAwCQYDVR0TBAIwADAOBgNVHQ8B
Af8EBAMCA+gwFgYDVR0lAQH/==

      </ns2:adminID>

      <ns2:version>1.0</ns2:version>

</ns2:requestKeyRecoveryMessage>
```

Recover Key (requestKeyRecoveryResponseMessage) Response

The following is a sample requestKeyRecoveryResponseMessage response with single Administrator Approval mode for key recovery.

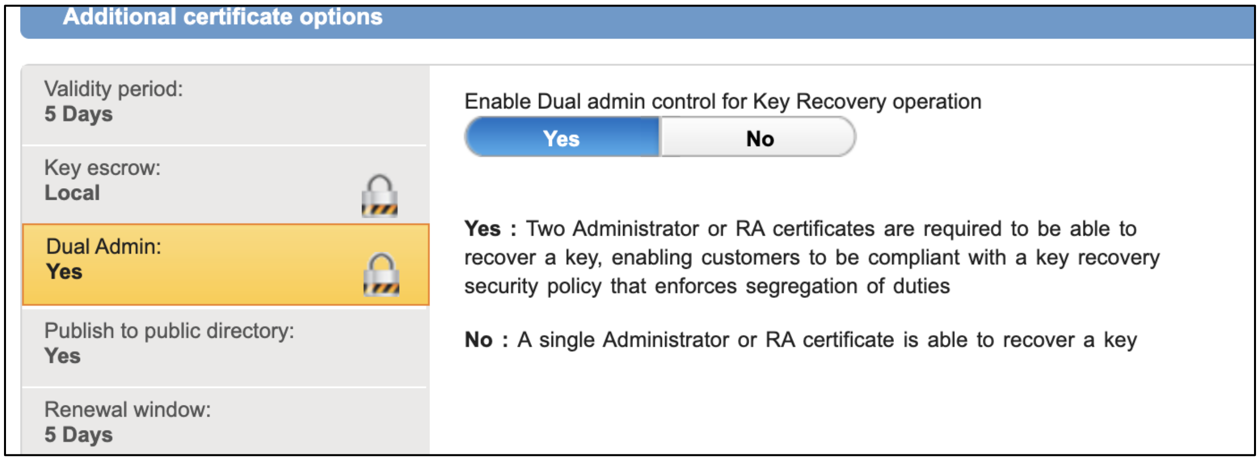
See “About the Certificate Management Protocol” on page 159.

```
<requestKeyRecoveryResponseMessage xmlns="http://schemas.
verisign.com/pkiservices/2009/07/management">

      <clientTransactionID>123456789</clientTransactionID>

      <serverTransactionID>cb2ed698b717f933</serverTransactionID>

      <adminApprovalPendingCount>0</adminApprovalPendingCount>

      <pKCS12Password>L8aNC63JGVSF</pKCS12Password>

      <pKCS12Message>
MIACAQMwgAYJKoZIhvcNAQcBoIAkgASCA+gwgDCABgkqhkiG9w0BBwGggCSABIID6DCC
BWUwggVhBgsqhkiG9w0BDAoBAqCCBQIwggT+MCgGCiqGSIb3DQEMAQMwGgQUUpC7sC25
7rWjVf50y5rghkAGqlMCAgQABIIE0ORbNBNTS8SZh29OPS1tTePfpxCODI0sgTl63eMT

AAlxTSTnYwpMlgU2/HO4nhjxTYYGBZQ6+pKQ2naERWWogBCBhluMzIb8bL1C/93EgQRL
IqP4e3utGVnYhPLte7FUH7wfFd0et2N83eNas/VEiAAhhml94MYVZeL8qmIOyDvctu5S
DDS2Mhjd6c+jeynkr9WJ8FIrynp+tCkYBIKllZV1sSznKSShEHRQ61GkyN6ctDw9QsM3
XZ2VWEJjuegjkfU5WLmuZY8UYIL1XqzgcMh/xwRDhMQ0i8UsVvGSzTKglrUgQJf1nRuN
6ZAIAL45gQnfGMrRUA/LoEeIIfbG9XvR0+3sZ/5BhU1vIhiG5S6VOmymMMC5EXe8Kjou
Lj4CMq9z3Vnnor1DBb4pIl5KO9Xjo7GHa4O5hfMjXz8fiA4BBSrFvptQgEI4ykl11eWI
grV8TY1QZBejyI5ed1z4vJVYcthUtwfG4/isNoKBI3MiOT50jMO6MUmzf4AChxsYCkrR
7e/UKPL89TxXQLdWe/a18EDwnNGxN2xvMtz8kunWS1UNkX6hEpOzOF9FM5qg/Afm3CuA
PEVD6WAQdwHd+ChwIBbzOuE+Baxr4zFKmyz8vS8IcVrS2YdmlrN1y+UGphUw+9ul9NI5
```

```
5QH+7FKh1xZtxDKYavAxafODkTRyEHmdXcnZ+u5ltAUijEaDEh79nGcZfQFj1du6i+wc
6okUfMV7zgIPDnZo3n+YABCLYSVMLV4lhH6B/wtwNb16/uI0b6NJOWHYIvQwVrNOViTa
1vX/1X4D5XNvkPR81AXgWk+AZmvCbZZ64oWxPlXeiBWpyab+C7u/Zp+u5h3bFqiyhubE
St3wlFtCoZoJxPzKdke3zs7mSiJWDCTTUozaz4+xa4s07um2ayHjojgaFHwvpwMWavY6
TknmzJKNDEjPt3oYhAUkPRixq9PXpyq4V8sfu6b2oaDTTyh7ycQpeEoYe/nyGC55Iegb
rbTxtQ1bw5tZOkYdf6lXAHrIgtLG4dJjSt7BIID6KfXaJaRUOxxgdRaPR

      </pKCS12Message>

            <version>1.0</version>

</requestKeyRecoveryResponseMessage>
```

To support Dual Admin control for Key Recovery:

- Enable Dual admin control for key recovery operation under Profile creation in PKI Manager and set it to "Yes".


Additional certificate options

Validity period: **5 Days**

Key escrow: **Local**

Dual Admin: **Yes**

Publish to public directory: **Yes**

Renewal window: **5 Days**

Enable Dual admin control for Key Recovery operation

[✓ Yes] No

Yes : Two Administrator or RA certificates are required to be able to recover a key, enabling customers to be compliant with a key recovery security policy that enforces segregation of duties

No : A single Administrator or RA certificate is able to recover a key

- The "Key Escrow and Recovery Service" should be upgraded to the latest version. Please refer to the DigiCert® PKI Enterprise Gateway Deployment Guide.

NOTE: Dual Admin key is only supported for local key escrow via API calls.

Recover Key (requestKeyRecoveryMessage) Request (1st Administrator)

The following is a sample 1st requestKeyRecoveryMessage request.

```
<?xml version="1.0" encoding="UTF-8"?>

<ns2:requestKeyRecoveryMessage
xmlns:ns2="http://schemas.verisign.com/pkiservices/2009/07/management">

<ns2:clientTransactionID>123456789</ns2:clientTransactionID>

<ns2:certificateSerialNumber>24021485fced686cde8f1ba66868daf1</ns2:certificateSer
ialNumber>

    <ns2:certificateIssuer>"CN =Managed PKI - 1 Level - Test CA,OU = FOR TEST
PURPOSES ONLY,O = "VeriSign, Inc.",C=US"</ns2:certificateIssuer>
```

```
 <ns2:adminID>MIIFkTCCBHmgAwIBAgIQfOZPxEdbt5Xnn/
+dXoy7bTANBgkqhkiG9w0BAQUFADCB+zELMAkGA1UEBhMCVVMxHTAbBgNVBAo
TFFN5bWFudGVjIENvcnBvcmF0aW9uMR8wHQYDVQQLExZGT1IgVEVTVCBQVVJQ
T1NFUyBPTkxZMR8wHQYDVQQLExZWZXJpU2lnbiBUcnVzdCBOZXR3b3JrMUIwQ
AYDVQQLEzlUZXJtcyBvZiB1c2UgYXQgaHR0cHM6Ly93d3cudmVyaXNpZ24uY2
9tL2Nwcy90ZXN0Y2EgKGMpMTExRzBFBgNVBAMTPlN5bWFudGVjIEMzIEFkbWl
uIEludGVybWVkaWF0ZSBURVNUIENlcnRpZmljYXRlIEF1dGhvcml0eSAtIFFB
MB4XDTExMDUwOTAwMDAwMFoXDTEyMDUwODIzNTk1OVowgZQxJjAkBgNVBAMMH
VdTQXV0bzIwMTEwNTEwIFdTQXV0bzIwMTEwNTEwMSkwJwYJKoZIhvcNAQkBFh
p3c2F1dG8yMDExMDUxMEB5b3BtYWlsLmNvbTEXMBUGA1UECgwOV1NBdXRvMjA
xMTA1MTAxDjAMBgNVBAsMBUFETUlOMRYwFAYDVQQLDA1NVUxUSS1BTExPV0VE
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAve7B/kMQ3W9516wPh
2x+n7OakWlU3nRAwka3Hh0H9JytoI5rhMpbCtyK654GcjTvyKmKT6dYaEJH/V
bIdmYVJLJXgKg1I7V4atb6SaQoBsA+rIzktpKn/eS0jM+xkZSpTRpc0oqJ/UK
gG+4WnMibofC2s2C2hAf1Co6m6f8Jycp9jzbPHBPL8815ejbwIqOKfYNIbP/w
5hHf3AENJ1HiDXDZaKVkPS5kTMWxXaK14G2sm5qpfe8ZX3EVameVFS1BaSDyX
kisPBY4LzHuklIAFiN7LEmQ9XLEfa3onFQV7v7QBwlFY1tImvSyNHc5KN9z/K
3Ofc4EnPP3z9WutXYewIDAQABo4IBdDCCAXAwCQYDVR0TBAIwADAOBgNVHQ8B
Af8EBAMCA+gwFgYDVR0lAQH/==
 </ns2:adminID>
  <ns2:version>1.0</ns2:version>
</ns2:requestKeyRecoveryMessage>
```

Recover Key (requestKeyRecoveryResponseMessage) Response (1st Administrator)

The following is a sample 1st requestKeyRecoveryResponseMessage response.

```
<requestKeyRecoveryResponseMessage
xmlns="http://schemas.verisign.com/pkiservices/2009/07/management">
<clientTransactionID>123456789</clientTransactionID>
<serverTransactionID>e10e379009a0380d</serverTransactionID>
<adminApprovalPendingCount>1</adminApprovalPendingCount>
<version>1.0</version>
</requestKeyRecoveryResponseMessage>
```

Recover Key (requestKeyRecoveryMessage) Request (2nd Administrator)

The following is a sample 2nd requestKeyRecoveryMessage request.

```
<?xml version="1.0" encoding="UTF-8"?>

<ns2:requestKeyRecoveryMessage
xmlns:ns2="http://schemas.verisign.com/pkiservices/2009/07/management">

<ns2:clientTransactionID>123456789</ns2:clientTransactionID>

<ns2:certificateSerialNumber>24021485fced686cde8f1ba66868daf1</ns2:certificateSer
ialNumber>

    <ns2:certificateIssuer>"CN =Managed PKI - 1 Level - Test CA,OU = FOR TEST
PURPOSES ONLY,O = "VeriSign, Inc.

",C=US"</ns2:certificateIssuer>

 <ns2:adminID>MIIFkTCCBHmgAwIBAgIQfOZPxEdbt5Xnn/

+aaay7bTANBgkqhkiG9w0BAQUFADCB+zELMAkGA1UEBhMCVVMxHTAbBgNVBAo

TFFN5bWFudGVjIENvcnBvcmF0aW9uMR8wHQYDVQQLExZGT1IgVEVTVCBQVVJQ

T1NFUyBPTkxZMR8wHQYDVQQLExZWZXJpU2lnbiBUcnVzdCBOZXR3b3JrMUIwQ

AYDVQQLEzlUZXJtcyBvZiB1c2UgYXQgaHR0cHM6Ly93d3cudmVyaXNpZ24uY2

9tL2Nwcy90ZXN0Y2EgKGMpMTExRzBFBgNVBAMTPlN5bWFudGVjIEMzIEFkbWl

uIEludGVybWVkaWF0ZSBURVNUIENlcnRpZmljYXRlIEF1dGhvcml0eSAtIFFB

MB4XDTExMDUwOTAwMDAwMFoXDTEyMDUwODIzNTk1OVowgZQxJjAkBgNVBAMMH

VdTQXV0bzIwMTEwNTEwIFdTQXV0bzIwMTEwNTEwMSkwJwYJKoZIhvcNAQkBFh

p3c2F1dG8yMDExMDUxMEB5b3BtYWlsLmNvbTEXMBUGA1UECgwOV1NBdXRvMjA

xMTA1MTAxDjAMBgNVBAsMBUFETUlOMRYwFAYDVQQLDA1NVUxUSS1BTExPV0VE

MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAve7B/kMQ3W9516wPh

2x+n7OakWlU3nRAwka3Hh0H9JytoI5rhMpbCtyK654GcjTvyKmKT6dYaEJH/V

bIdmYVJLJXgKg1I7V4atb6SaQoBsA+rIzktpKn/eS0jM+xkZSpTRpc0oqJ/UK

gG+4WnMibofC2s2C2hAf1Co6m6f8Jycp9jzbPHBPL8815ejbwIqOKfYNIbP/w

5hHf3AENJ1HiDXDZaKVkPS5kTMWxXaK14G2sm5qpfe8ZX3EVameVFS1BaSDyX

kisPBY4LzHuklIAFiN7LEmQ9XLEfa3onFQV7v7QBwlFY1tImvSyNHc5KN9z/K

3Ofc4EnPP3z9WutXYewIDAQABo4IBdDCCAXAwCQYDVR0TBAIwADAOBgNVHQ8B

Af8EBAMCA+gwFgYDVR0lAQH/==

 </ns2:adminID>

  <ns2:version>1.0</ns2:version>

</ns2:requestKeyRecoveryMessage>
```

Recover Key (requestKeyRecoveryResponseMessage) Response (2nd Administrator)

The following is a sample 2nd requestKeyRecoveryResponseMessage response.

```
<requestKeyRecoveryResponseMessage xmlns="http://schemas.
verisign.com/pkiservices/2009/07/management">
 <clientTransactionID>123456789</clientTransactionID>
 <serverTransactionID>cb2ed698b717f933</serverTransactionID>
 <adminApprovalPendingCount>0</adminApprovalPendingCount>
 <pKCS12Password>L8aNC63JGVSF</pKCS12Password>
 <pKCS12Message>
MIACAQMwgAYJKoZIhvcNAQcBoIAkgASCA+gwgDCABgkqhkiG9w0BBwGggCSABIID6DCC
BWUwggVhBgsqhkiG9w0BDAoBAqCCBQIwggT+MCgGCiqGSIb3DQEMAQMwGgQUUpC7sC25
7rWjVf50y5rghkAGqlMCAgQABIIE0ORbNBNTS8SZh29OPS1tTePfpxCODI0sgTl63eMT
AAlxTSTnYwpMlgU2/HO4nhjxTYYGBZQ6+pKQ2naERWWogBCBhluMzIb8bL1C/93EgQRL
IqP4e3utGVnYhPLte7FUH7wfFd0et2N83eNas/VEiAAhhml94MYVZeL8qmIOyDvctu5S
DDS2Mhjd6c+jeynkr9WJ8FIrynp+tCkYBIKllZV1sSznKSShEHRQ61GkyN6ctDw9QsM3
XZ2VWEJjuegjkfU5WLmuZY8UYIL1XqzgcMh/xwRDhMQ0i8UsVvGSzTKglrUgQJf1nRuN
6ZAIAL45gQnfGMrRUA/LoEeIIfbG9XvR0+3sZ/5BhU1vIhiG5S6VOmymMMC5EXe8Kjou
Lj4CMq9z3Vnnor1DBb4pIl5KO9Xjo7GHa4O5hfMjXz8fiA4BBSrFvptQgEI4ykl11eWI
grV8TY1QZBejyI5ed1z4vJVYcthUtwfG4/isNoKBI3MiOT50jMO6MUmzf4AChxsYCkrR
7e/UKPL89TxXQLdWe/a18EDwnNGxN2xvMtz8kunWS1UNkX6hEpOzOF9FM5qg/Afm3CuA
PEVD6WAQdwHd+ChwIBbzOuE+Baxr4zFKmyz8vS8IcVrS2YdmlrN1y+UGphUw+9ul9NI5
5QH+7FKh1xZtxDKYavAxafODkTRyEHmdXcnZ+u5ltAUijEaDEh79nGcZfQFj1du6i+wc
6okUfMV7zgIPDnZo3n+YABCLYSVMLV4lhH6B/wtwNb16/uI0b6NJOWHYIvQwVrNOViTa
1vX/1X4D5XNvkPR81AXgWk+AZmvCbZZ64oWxPlXeiBWpyab+C7u/Zp+u5h3bFqiyhubE
St3wlFtCoZoJxPzKdke3zs7mSiJWDCTTUozaz4+xa4s07um2ayHjojgaFHwvpwMWavY6
TknmzJKNDEjPt3oYhAUkPRixq9PXpyq4V8sfu6b2oaDTTyh7ycQpeEoYe/nyGC55Iegb
rbTxtQ1bw5tZOkYdf6lXAHrIgtLG4dJjSt7BIID6KfXaJaRUOxxgdRaPR
</pKCS12Message>
<version>1.0</version>
</requestKeyRecoveryResponseMessage>
```

5.5 User Management Protocol

The User Management Protocol Web Service method is used to manage the users to whom you issue (or have issued) certificates. This protocol includes the following functions:

- See "Creating or Modifying Users" on page 77.
- See "Getting User Information" on page 78.
- See "Generating or Replacing Enrollment Codes" on page 78.
- See "Getting Enrollment Code Information" on page 79.

5.5.1 Creating or Modifying Users

Before you use certificates to a user using PKI Web Services, the user must exist in your DigiCert PKI account. The createOrUpdateUser call lets you upload the data for a user or users into DigiCert PKI and create the user or users in the system. Although you can upload any or all of the following data for a user, at minimum a user must have a unique identifier (seat ID). This identifier must match the **User identifier** value in the **Customize user identification** section of the **Manage profiles** page in PKI Manager when you created the certificate template.

- Seat ID (Required)
- First name
- Last name
- Email address
- Work or mobile telephone numbers
- The other attributes that you define and should appear in the certificate

If the specified user already exists in the system, this call adds or overwrites the existing user data.

You can perform this operation for multiple users at one time by providing the user data for all of the users in one createOrUpdateUser call. The operation is performed on each user separately, so each user is added, or updated based upon whether that user exists in the system.

Error Handling for Creating or Modifying Users

User creation and modification requests only return a fault if the request is not processed (for example, a mismatched schema or version). All other request errors return a status code.

5.5.2 Getting User Information

Use the getUserInformation call to obtain information about the user and to obtain the valid certificates that are issued to that user. The call returns all of the data for the users that are stored at DigiCert. (It is either uploaded using the createOrUpdateUser or createOrUpdatePasscode calls, or added directly in PKI Manager.) This call can also be used to obtain all valid certificates for this user.

The getUserInformation call does not return data or certificates for the users that are stored locally.

5.5.3 Generating or Replacing Enrollment Codes

Your users require an enrollment code to pick up their certificates. This enrollment code verifies that the users are authorized to pick up their certificates. Use the createOrUpdatePasscode call to generate or replace enrollment codes for your users. Once an enrollment code is generated or replaced, you must provide this enrollment code to your users along with the enrollment URL created.

You can provide an enrollment code in the createOrUpdatePasscode call, or leave it blank to allow the system to generate the enrollment code for you.

- If you provide an enrollment code with this call, the enrollment code you provide is assigned to the user and replace any existing enrollment code. If the user does not exist, a user is created and assigned the enrollment code you provided.

  Enrollment codes must be between 6 and 32 characters, and should contain a mixture of upper- and lower-case letters, numbers, and special characters.

- If you do not provide an enrollment code, this call performs one of the following actions:

Table 5-3 createOrUpdatePasscode call behavior (if no enrollment code provided)

Situation	Result
User does not exist	The user is added and an enrollment code generated. The user data only includes the seat ID. You should use the createOrUpdateUser call to create a user with complete user data.
User exists, but does not have an enrollment code.	An enrollment code is generated.
User exists and has an enrollment code, but the enrollment code is locked due to too many failed pick-up attempts.	The existing enrollment code is unlocked and returned.
User exists and has an enrollment code, but has not picked up the certificate.	The existing enrollment code is returned.
User exists and has an enrollment code, but has already picked up the certificate.	If the profile allows multiple certificates for a user, a new enrollment code is generated. If the profile only allows one certificate for a user, an error is returned.

Error Handling for Generating or Replacing Enrollment Codes

Enrollment code generation and replacement requests only return a fault if the request is not processed (for example, a mismatched schema or version). All other request errors return a status code.

5.5.4 Getting Enrollment Code Information

Use the getPasscodeInformation call to obtain information about the enrollment code for the user, including:

- The enrollment code previously generated for the user. If the user has no enrollment code, an error is returned.
- When the enrollment code was created.
- When the enrollment code expires.

- The number of times the user has attempted to use the enrollment code to pick up a certificate and failed. This failure is typically due to entering the wrong enrollment code.
- Status of the enrollment code:
    - NEW - The enrollment code is ready to be used to pick up a certificate.
    - REDEEMED - The enrollment code has already been used to pick up a certificate. It can no longer be used again.
    - LOCKED - The enrollment code has been locked due to too many failed pick-up attempts.
    - EXPIRED - The enrollment code has expired.
- The unique identifier for the user (seatId)
- The OID for the certificate profile for which the enrollment code was generated.
- The certificate enrollment URL for the user to enroll for a certificate using the passcode.

5.5.5 Important Elements, Requests, and Responses

The User Management Protocol supports multiple functions. The values you send for each function (and the response you receive) depends on the type of request you make. Table 5-4 describes the common User Management Protocol requests and responses.

Table 5-4 Common Certificate Management requests and responses

	Requests		Responses	
	Elements	Namespace	Response	Operation
Create/Modify User	UserInformation	http://schemas.verisign.com/pkiservices/2011/08/usermanagement	User creation status	createOrUpdateUser See "Sample Add or Modify Users Requests and Responses" on page 81.
Get User Information	Seat ID	http://schemas.verisign.com/pkiservices/2011/08/usermanagement	User information	getUserInformation See "Sample Get User Information Requests and

	Requests		Responses	
				Responses” on page 84.
Generate/ Replace Passcode	Enrollment code information	http://schemas.verisign.com/pkiservices/2011/08/usermanagement	• Passcode information • Passcode creation status	createOrUpdatePasscode See “Sample Generate or Replace Enrollment Code Requests and Responses” on page 86.
Get Passcode Information	• Seat ID • Profile OID	http://schemas.verisign.com/pkiservices/2011/08/usermanagement	Passcode information	getPasscodeInformation See “Sample Get Enrollment Code Information Requests and Responses” on page 88.

5.5.6 Sample User Management Requests and Responses

This section provides sample requests and responses for the following scenarios.

- See “Sample Add or Modify Users Requests and Responses” on page 81.
- See “Sample Get User Information Requests and Responses” on page 84.
- See “Sample Generate or Replace Enrollment Code Requests and Responses” on page 86.
- See “Sample Get Enrollment Code Information Requests and Responses” on page 88.

Sample Add or Modify Users Requests and Responses

In order to issue certificates for and manage users, you must add them to your DigiCert PKI account. Use createOrUpdateUser to add a new user or users to your account. If the user or users already exist, this call automatically modifies the user.

Sample Add or Modify Users (createOrUpdateUser) Request

The following is a sample createOrUpdateUser request.

See “About the User Management Protocol” on page 178.

```
<?xml version="1.0" encoding="UTF-8"?>

<ns12:createOrUpdateUserRequest xmlns:ns12="http://schemas.
verisign.com/pkiservices/2011/08/usermanagement">

    <ns12:clientTransactionID>1316083481585</ns12:client

    TransactionID>

    <ns12:userInformation>

      <ns12:seatId>a2@test.com</ns12:seatId>

      <ns12:firstName>seat</ns12:firstName>

      <ns12:lastName>1</ns12:lastName>

      <ns12:emailAddress>seat1@test.com</ns12:emailAddress>

      <ns12:deskPhoneNumber>666-666-6666</ns12:deskPhoneNumber>

      <ns12:mobilePhoneNumber>555-555-5555</ns12:mobilePhone Number>

      <ns12:userAttribute>

            <ns12:name>organization</ns12:name>

            <ns12:value>test</ns12:value>

      </ns12:userAttribute>

      <ns12:userAttribute>

            <ns12:name>common_name</ns12:name>

            <ns12:value>seat1</ns12:value>

      </ns12:userAttribute>

      <ns12:userAttribute>

            <ns12:name>organizational_unit</ns12:name>

            <ns12:value>engineering#123/%</ns12:value>

      </ns12:userAttribute>

    </ns12:userInformation>

    <ns12:userInformation>

      <ns12:seatId>a3@test.com</ns12:seatId>

      <ns12:firstName>seat</ns12:firstName>

      <ns12:lastName>2</ns12:lastName>

      <ns12:emailAddress>seat1@test.com</ns12:emailAddress>

      <ns12:deskPhoneNumber>666-666-6666</ns12:deskPhoneNumber>
```

```
        <ns12:mobilePhoneNumber>555-555-5555</ns12:mobilePhone Number>
        <ns12:userAttribute>
              <ns12:name>organization</ns12:name>
              <ns12:value>test</ns12:value>
        </ns12:userAttribute>
        <ns12:userAttribute>
              <ns12:name>common_name</ns12:name>
              <ns12:value>seat1</ns12:value>
        </ns12:userAttribute>
        <ns12:userAttribute>
              <ns12:name>organizational_unit</ns12:name>
              <ns12:value>engineering#123/%</ns12:value>
        </ns12:userAttribute>
    </ns12:userInformation>
    <ns12:version>1.0</ns12:version>
</ns12:createOrUpdateUserRequest>
```

Sample Add or Modify Users (createOrUpdateUser) Response

The following is a sample createOrUpdateUser response.

See “About the User Management Protocol” on page 178.

```
<?xml version="1.0" encoding="UTF-8"?>
<createOrUpdateUserResponse xmlns="http://schemas.verisign.com/
pkiservices/2011/08/usermanagement">
      <clientTransactionID>1316083481585</clientTransactionID>
      <serverTransactionID>722b5c5a7606dc6e</serverTransactionID>
      <userCreationStatus>
            <seatId>a2@test.com</seatId>
            <statusCode>0</statusCode>
      </userCreationStatus>
      <userCreationStatus>
            <seatId>a3@test.com</seatId>
            <statusCode>0</statusCode>
      </userCreationStatus>
      <version>1.0</version>
</createOrUpdateUserResponse>
```

Sample Get User Information Requests and Responses

Use getUserInformation to obtain user information and all valid certificates that are assigned to a user that is stored at DigiCert.

Sample Get User Information (getUserInformation) Request

The following is a sample getUserInformation request.

See “About the User Management Protocol” on page 178.

```
<?xml version="1.0" encoding="UTF-8"?>

<ns12:getUserInformationRequest xmlns:ns12="http://schemas.
verisign.com/pkiservices/2011/08/usermanagement">

      <ns12:clientTransactionID>1316083628509</ns12:clientTransaction ID>

      <ns12:seatId>a1@test.com</ns12:seatId>

<ns12:getUserCertificate>true</ns12:getUserCertificate>

      <ns12:version>1.0</ns12:version>

</ns12:getUserInformationRequest>
```

Sample Get User Information (getUserInformation) Response

The following is a sample getUserInformation response.

See “About the User Management Protocol” on page 178.

```
<?xml version="1.0" encoding="UTF-8"?>

<getUserInformationResponse xmlns="http://schemas.verisign.com/
pkiservices/2011/08/usermanagement">

      <clientTransactionID>1316083628509</clientTransactionID>

      <serverTransactionID>9f9812678c60f092</serverTransactionID>

      <userInformation>

            <seatId>a1@test.com</seatId>

            <firstName>seat</firstName>

            <lastName>1</lastName>

            <emailAddress>seat1@test.com</emailAddress>

            <deskPhoneNumber>666-666-6666</deskPhoneNumber>

            <mobilePhoneNumber>555-555-5555</mobilePhoneNumber>

            <userAttribute>

                  <name>organization</name>

                  <value>test</value>

            </userAttribute>
            <userAttribute>
                  <name>organizational_unit</name>
                  <value>engineering#123/%</value>
            </userAttribute>
            <userAttribute>
                  <name>common_name</name>
                  <value>seat1</value>
            </userAttribute>
            <userAttribute>
                  <name>country</name>
                  <value>US</value>
            </userAttribute>
      </userInformation>
      <userValidCertificates>
<userCertificate>MIIFkTCCBHmgAwIBAgIQfOZPxEdbt5Xnn/
+dXoy7bTANBgkqhkiG9w0BAQUFADCB+zELMAkGA1UEBhMCVVMxHTAbBgNVBAoTFF
N5bWFudGVjIENvcnBvcmF0aW9uMR8wHQYDVQQLExZGT1IgVEVTVCBQVVJQT1NFUy
BPTkxZMR8wHQYDVQQLExZWZXJpU2lnbiBUcnVzdCBOZXR3b3JrMUIwQAYDVQQLEz
lUZXJtcyBvZiB1c2UgYXQgaHR0cHM6Ly93d3cudmVyaXNpZ24uY29tL2Nwcy90ZX
N0Y2EgKGMpMTExRzBFBgNVBAMTPlN5bWFudGVjIEMzIEFkbWluIEludGVybWVkaW
F0ZSBURVNUIENlcnRpZmljYXRlIEF1dGhvcml0eSAtIFFBMB4XDTExMDUwOTAwMD
AwMFoXDTEyMDUwODIzNTk1OVowgZQxJjAkBgNVBAMMHVdTQXV0bzIwMTEwNTEwIF
dTQXV0bzIwMTEwNTEwMSkwJwYJKoZIhvcNAQkBFhp3c2F1dG8yMDExMDUxMEB5b3
BtYWlsLmNvbTEXMBUGA1UECgwOV1NBdXRvMjAxMTA1MTAxDjAMBgNVBAsMBUFETU
lOMRYwFAYDVQQLDA1NVUxUSS1BTExPV0VEMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ
8AMIIBCgKCAQEAve7B/kMQ3W9516wPh2x+n7Oa/kWlU3nRAwka3Hh0H9JytoI5rh
MpbCtyK654GcjTvyKmKT6dYaEJH/VbIdmYVJLJXgKg1I7V4atb6SaQoBsA+rIzkt
pKn/eS0jM+xkZSpTRpc0oqJ/UKgG+4WnMibofC2s2C2hAf1Co6m6f8Jycp9jzbPH
BPL8815ejbwIqOKfYNIbP/w5hHf3AENJ1HiDXDZaKVkPS5kTMWxXaK14G2sm5qpf
e8ZX3EVameVFS1BaSDyXkisPBY4LzHuklIAFiN7LEmQ9XLEfa3onFQV7v7QBwlFY
1tImvSyNHc5KN9z/K3Ofc4EnPP3z9WutXYewIDAQABo4IBdDCCAXAwCQYDVR0TBA
IwADAOBgNVHQ8BAf8EBAMCA+gwFgYDVR0lAQH/==</userCertificate>
      </userValidCertificates>
      <version>1.0</version>
</getUserInformationResponse>
```

Sample Generate or Replace Enrollment Code Requests and Responses

Users require an enrollment code to pick up their certificate, to verify that they are authorized to do so. Use createOrUpdatePasscode to generate or replace enrollment codes for your users.

Sample Generate or Replace Enrollment Code (createOrUpdatePasscode) Request

The following is a sample createOrUpdatePasscode request.

See “About the User Management Protocol” on page 178.

```
<?xml version="1.0" encoding="UTF-8"?>

<ns12:createOrUpdatePasscodeRequest xmlns:ns12="http://schemas.
verisign.com/pkiservices/2011/08/usermanagement">

      <ns12:clientTransactionID>1316082833162</ns12:clientTransaction ID>

      <ns12:passcodeInformation>

            <ns12:passcode>abc123</ns12:passcode>

            <ns12:expiryDateTime>2011-09-18T00:00:00.000+00:00</ns12:

      expiryDateTime>

            <ns12:seatId>a2@test.com</ns12:seatId>

            <ns12:certificateProfileOid>2.16.840.1.113733.1.16.1.2.5.1.

      1.98395</ns12:certificateProfileOid>

      </ns12:passcodeInformation>

      <ns12:passcodeInformation>

            <ns12:expiryDateTime>2011-09-18T00:00:00.000+00:00</ns12:

      expiryDateTime>

            <ns12:seatId>a3@test.com</ns12:seatId>

            <ns12:certificateProfileOid>2.16.840.1.113733.1.16.1.2.5.1.

      1.983956</ns12:certificateProfileOid>

      </ns12:passcodeInformation>

      <ns12:version>1.0</ns12:version>

</ns12:createOrUpdatePasscodeRequest>
```

Sample Generate or Replace Enrollment Code (createOrUpdatePasscode) Response

The following is a sample createOrUpdatePasscode response.

See "About the User Management Protocol" on page 178.

```
<?xml version="1.0" encoding="UTF-8"?>

<createOrUpdatePasscodeResponse xmlns="http://schemas.verisign.com/
pkiservices/2011/08/usermanagement">

      <clientTransactionID>1316082833162</clientTransactionID>

      <serverTransactionID>7993f601bd3c8a9d</serverTransactionID>

      <passcodeCreationStatus>

            <passcodeInformation>

                  <passcode>abc123</passcode>

                  <numberOfBadAttempts>0</numberOfBadAttempts>

                  <passcodeStatus>NEW</passcodeStatus>

                  <expiryDateTime>2011-09-18T00:00:00.000+00:00

</expiryDateTime>

      <creationDateTime>2011-09-15T18:30:06.978+00:00</creationDateTime>

                  <seatId>a2@test.com</seatId>

                  <certificateProfileOid>2.16.840.1.113733.1.16.1.2.5.1.1.

98395</certificateProfileOid>

            </passcodeInformation>

            <statusCode>0</statusCode>

      </passcodeCreationStatus>

      <passcodeCreationStatus>

            <passcodeInformation>

                  <seatId>a3@test.com</seatId>

                  <certificateProfileOid>2.16.840.1.113733.1.16.1.2.5.1.1.

983956</certificateProfileOid>

      <enrollmentURL>https://pki.symauth.com/certificate-
service?x=xyzabcpki</enrollmentURL>

</passcodeInformation>

            <statusCode>A505</statusCode>

      </passcodeCreationStatus>

      <version>1.0</version>

</createOrUpdatePasscodeResponse>
```

Sample Get Enrollment Code Information Requests and Responses

Use getPasscodeInformation to obtain information about the user.

Sample Get Enrollment Code Information (getPasscodeInformation) Request

The following is a sample getPasscodeInformation request.

See “About the User Management Protocol” on page 178.

```
<?xml version="1.0" encoding="UTF-8"?>

<ns12:getPasscodeInformationRequest xmlns:ns12="http://schemas.
verisign.com/pkiservices/2011/08/usermanagement">

      <ns12:clientTransactionID>1316083681444</ns12:clientTransaction ID>

      <ns12:seatId>a1@test.com</ns12:seatId>

      <ns12:certificateProfileOid>2.16.840.1.113733.1.16.1.2.5.1.
      1.98395</ns12:certificateProfileOid>

<enrollmentURL>https://pki.symauth.com/certificate-
service?x=xyzabcpki</enrollmentURL>

<ns12:version>1.0</ns12:version>

</ns12:getPasscodeInformationRequest>
```

Sample Get PasscodeInformation (getPasscodeInformation) Response

The following is a sample getPasscodeInformation response.

See “About the User Management Protocol” on page 178.

```
<?xml version="1.0" encoding="UTF-8"?>

<getPasscodeInformationResponse xmlns="http://schemas.verisign.com/
pkiservices/2011/08/usermanagement">

      <clientTransactionID>1316083681444</clientTransactionID>

      <serverTransactionID>eb798624a78cc12f</serverTransactionID>

      <passcodeInformation>

            <passcode>abc123</passcode>

            <numberOfBadAttempts>0</numberOfBadAttempts>

            <passcodeStatus>NEW</passcodeStatus>

            <expiryDateTime>2011-09-18T00:00:00.000+00:00</expiryDate Time>

            <seatId>a1@test.com</seatId>

            <certificateProfileOid>2.16.840.1.113733.1.16.1.2.5.1.1.98395 </
      certificateProfileOid>

</passcodeInformation>

      <version>1.0</version>
```

```
</getPasscodeInformationResponse>
```

5.6 Health Check Protocol

Use the Health Check Protocol to retrieve status details for different operations. Currently supports checks for enroll operations.

5.6.1 Sample Health Check Protocol Request and Response

This section provides sample Health Check Protocol requests (getStatusRequestMessage) and responses (getStatusResponseMessage).

Sample getStatusRequestMessage Request

The following is a sample getStatusRequestMessage request.

```
<?xml version="1.0" encoding="UTF-8"?>

<ns16:getStatusRequestMessage xmlns:ns16=
"http://schemas.pki.digicert.com/pkiservices/healthcheck">

      <ns16:version>1.0</ns16:version>

      <ns16:clientTransactionID>693a1081b157850ef78b0df70def2b49

      </ns16:clientTransactionID>

      <ns16:operationType>ENROLL</ns16:operationType>

      <ns16:profileOid>2.16.840.1.113733.1.16.1.2.3.1.1.615642086

      </ns16:profileOid>

</ns16:getStatusRequestMessage>
```

Sample getStatusResponseMessage Response

The following is a sample getStatusResponseMessage request.

```
<?xml version="1.0" encoding="UTF-8"?>

<ns16:getStatusResponseMessage xmlns:ns16=

"http://schemas.pki.digicert.com/pkiservices/healthcheck">

      <ns16:version>1.0</ns16:version>

      <ns16:clientTransactionID>693a1081b157850ef78b0df70def2b49

      </ns16:clientTransactionID>

      <ns16:serverTransactionID>5a35249ee09f8c8d
```

```
        </ns16:serverTransactionID>

        <ns16:statusCode>UP</ns16:statusCode>

        <ns16:statusMessage>Operational</ns16:statusMessage>

</ns16:getStatusResponseMessage>
```

5.7 RA Application Recommendations

The DigiCert PKI protocols are designed to allow you flexibility to write your RA application to meet your needs. DigiCert has identified some recommendations you may want to follow:

- To make it easier to integrate with your existing security infrastructure, your enterprise RA application secures communication with DigiCert PKI using your RA certificate. However, since the RA application resides at your enterprise site, it depends upon your existing security infrastructure and practices for protection. DigiCert recommends that you implement a firewall and you use access rules to restrict access to the RA application to only the users that require it.
- Although you can set your RA application timeout to be any value, DigiCert recommends that the RA application timeout should not exceed 120 seconds.

APPENDIX A

6 Certificate Enrollment Policy Protocol Definition

This appendix includes the following topics:

6.1 About the Certificate Enrollment Policy Protocol

This section describes the protocol that downloads the policy file that is used to construct a certificate enrollment request. The request is in compliance with the customer's certificate profile.

6.2 Certificate Enrollment Policy Protocol Overview

The Certificate Enrollment Policy Protocol defines the interactions between a requesting RA application and DigiCert PKI for the exchange of a certificate policy. During account setup, the DigiCert PKI administrator creates certificate policy, defines rules, data and features governing the certificate enrollment process. The certificate policy has the information that is needed for your RA application to construct a valid Certificate Enrollment Protocol request for a DigiCert certificate.

The Certificate Enrollment Policy Protocol includes a single RA application request message (getPolicies) which requests the certificate policy from the DigiCert PKI. Additionally, it requests the matching server response message (getPoliciesResponse) which returns either the certificate policies or a SOAP fault message. Your RA application can parse the certificate policies in the response to dynamically construct the enrollment request. Parsing allows your RA application to download a new policy without requiring a customization change on the RA application.

A Certificate Enrollment Policy Protocol call is mandatory only for the first enrollment for a specific account. Or, if there have been changes to an account or a certificate profile. The first time an end user enrolls for a certificate, your RA application must call this protocol to construct a valid Certificate Enrollment Protocol request.

However, once the Certificate Enrollment Protocol request is constructed, subsequent enrollment requests can use the cached certificate policy information. Your RA application does not need to call the protocol again unless the account or the certificate profile has been changed.

- Calling the protocol with each request ensures that the policy data is the most up-to-date, but requires additional steps in the enrollment flow.
- Calling the protocol only when a certificate profile is changed creates a less complicated enrollment flow. But, it requires that you manually trigger the protocol each time the profiles change.

Table A-1 provides an overview of the Certificate Enrollment Policy Protocol WSDL and references to the sections that contain more information and code samples for its operations and messages.

Table A- 1 Certificate Enrollment Policy Protocol WSDL operations and messages

Name	Description	See...
Operations		
getPolicies	Protocol that fetches the certificate policies for the RA application.	See “ getPolicies” on page 102.
Messages		
getPoliciesMessage	getPolicies uses it to request a policy.	See “getPoliciesMessage” on page 102.
getPoliciesResponse	Response that is message received by the RA application.	See “getPoliciesResponse” on page 102.

Table A-2 provides an overview of the Certificate Enrollment Policy Protocol schema and references to the sections that contain more information and code samples for its elements.

Table A- 2 Certificate Enrollment Policy Protocol schema elements

Name	Description	See...
getPolicies	Contains the details of the policy request	See "getPolicies" on page 103.
VersionType	Defines the format for the version value	See "VersionType" on page 103.
TransactionIDType	Defines the format for the system-generated transaction ID	See "TransactionIDType" on page 103.
Client	Identifies the last time the RA application received the policy from DigiCert PKI. May include additional information	See "Client" on page 103.
RequestFilter	Filters the response to specific certificate policies. This feature is not supported in this release.	See "RequestFilter" on page 105.
FilterOIDCollection	Lists the certificate policies for which to filter.	See "FilterOIDCollection" on page 106.

Name	Description	See…
	This feature is not supported in this release.	
getPoliciesResponse	The response to the getPolicies request from DigiCert PKI	See "getPoliciesResponse" on page 106.
Response	The response object that contains all of the policies that are associated with an account	See "Response" on page 107.
CACollection	List of all of the CAs supported by DigiCert PKI for the specified account	See "CACollection" on page 108.
CA	Provides the CA information	See "CA" on page 108.
OIDCollection	List of OID objects referenced by the policies in the response.	See "OIDCollection" on page 109.
algorithmOIDReference	This element provides the OID object that corresponds to the asymmetric algorithm of the private key. This OID element is part of the OIDCollections in the response. It includes	See "Table A-5" on page 123.

Name	Description	See...
	reference to the Ecliptic Curve Cryptography (ECC) and Digital Signature Algorithm (DSA) OIDs. DigiCert PKI supports ECC 224, 384, and 521, as well as DSA 2048 (prime) - 256 (subprime) and 3072 (prime) - 256 (subprime). For RSA, this element is null as RSA is the default algorithm supported.	
OID	Identifies an object and provides generic attributes for that object	See "OID" on page 110.
PolicyCollection	List of DigiCert policies for the account	See "PolicyCollection" on page 111.
CertificateEnrollmentPolicy	Provides the policy information	See "CertificateEnrollmentPolicy" on page 112.
CAReferenceCollection	List of CA reference IDs associated with the CAs in the response	See "CAReferenceCollection" on page 112.
DuplicateCertPolicyEnum	Defines how duplicate	See "DuplicateCertPolicyEnum" on page 113.

Name	Description	See...
	certificates are handled	
Attributes	Lists the policy-specific elements	See "Attributes" on page 113.
PolicySchema	Indicates the schema version of the corresponding Certificate Enrollment Policy	See "policySchema" on page 114.
SystemInformation	Contains the system information that is not related to the certificate. For example, certificate delivery information or whether to publish the certificate in the DigiCert Trust Network (DTN).	See "SystemInformation" on page 114.
PersonalInfoType	Contains the name, email address, and telephone number of the contact	See "PersonalInfoType" on page 115.
ServiceEndPointListType	Provides the service endpoint information	See "ServiceEndPointListType" on page 116.
ServiceEndPointType	Contains a list of the service endpoints that the profile supports	See "ServiceEndPointType" on page 116.

Name	Description	See...
SeatInfoType	References the name-value pair that is used to send your seat ID. The seat ID is a unique identifier for the user. For each certificate that is issued to a user (identified by the seat ID), one seat is deducted from your seat count.	See “SeatInfoType” on page 117.
ApplicationInstructionsType	This element is not applicable for PKI Web Services.	See “ApplicationInstructionsType” on page 118.
DeliveryFormat	Contains the details about the format in which the issued certificate is delivered	See “DeliveryFormat” on page 118.
CACertPublishNameValuePair	Contains the details about the name-value pair that is provided in the enrollment which indicates whether to publish the issued certificate to the DTN.	See “CACertPublishNameValuePair ” on page 119.
Search CertificateData	Lists the name-value pairs the RA application send for certificate searches	See “SearchCertificateData” on page 119.

Name	Description	See...
CertificateValidity	Provides the details about the default certificate validity and renewal period	See "certificateValidity" on page 120.
OverrideValidity	Indicates if the certificate validity can be changed during enrollment	See "overrideValidity" on page 120.
validityNameValuePairNames	Allows the enrollment request to override the default certificate validity period	See "validityNameValuePairNames" on page 122.
PrivateKeyInfo	Provides the details about key material, such as the size of the key, key usage, and if the profile supports key escrow	See "PrivateKeyInfo" on page 123.
KeyEscrowPolicyType	Determines the key escrow policy for the certificate profile	See "KeyEscrowPolicyType" on page 124.
CryptoProviders	Lists the cryptographic providers available when the private key is generated	See "CryptoProviders" on page 125.
ClientPolicy	Contains the RA application-specific policies	See "ClientPolicy" on page 125.

Name	Description	See...
PublishCert	Identifies whether the certificate is published to the DigiCert Trust Network (DTN)	See “PublishCert” on page 127.
RAPolicy	Provides the general information for use between the RA and DigiCert PKI. This feature is not supported in this release.	See “RAPolicy” on page 127.
AuthorizationInfoType	This feature is not implemented for PKI Web Services.	See “AuthorizationInfoType” on page 128.
DirectoryInfoType	This feature is not implemented for PKI Web Services.	See “DirectoryInfoType” on page 128.
UserAuthorizationCollection	This feature is not implemented for PKI Web Services.	See “UserAuthorizationCollection” on page 129.
PollingPolicyType	This feature is not supported in this release.	See “PollingPolicyType” on page 129.
PollingTimeType	This feature is not supported in this release.	See “PollingTimeType” on page 129.
Extensions	Contains a list of extension elements	See “Extensions” on page 130.
Extension	Contains the extensions for which the RA	See “Extension” on page 130.

Name	Description	See...
	application must provide data	
extensionSyntax	Defines the attribute names and values for the extension	See “extensionSyntax” on page 131.
extensionValueType	Defines if the base64 encoded extension string value is mandatory or optional in the enrollment request	See “extensionValueType” on page 132.
AttributeNameValuePairType	Indicates the Subject DN extensions or attribute name and name-value pair that go into that attribute. This value is the name-value pair sent in the enrollment request.	See “AttributeNameValuePairType” on page 134.
AttributeValueType	Indicates whether the values are mandatory or optional, and what data type the value required can be	See “AttributeValueType” on page 135.
AttributeName ValuePropertyType	Provides the additional information that is required to build the RA	See “AttributeNameValuePropertyType” on page 135.

Name	Description	See...
	application which collects name value pairs from users enrolling for certificates against the policy	
subjectName	Provides the details about the subject DN for the certificate being requested	See "subjectName" on page 136.
subjectNameAttribute	Defines the attribute names in the subjectDN, and the values in the subjectDN attributes	See "subjectNameAttribute" on page 137.
CAType	Identifies whether the CA for the account is part of a public or a private hierarchy	See "CAType" on page 139.
MigrationOIDCollection	Contain a sequence of superceded OIDs.	See "migrationOIDCollection" on page 110.

6.3 Operations

The following WSDL operations are specific to the Certificate Enrollment Policy Protocol.

6.3.1 getPolicies

The Certificate Enrollment Policy protocol is a SOAP request consisting of one operation which fetches the certificate enrollment policy. Communication between the RA application and the server is secured through HTTPS, using the RA certificate for RA application authentication.

```
<wsdl:portType name="policy">

      <wsdl:operation name="getPolicies">

      <wsdl:input message="vscep:getPoliciesMessage" wsaw:Action=
      "http://schemas.verisign.com/pkiservices/2009/07/

      policy/getPolicies"/>

      <wsdl:output message="vscep:getPoliciesResponse" wsaw:

      Action="http://schemas.verisign.com/pkiservices/2009/07/
      policy/getPoliciesResponse" />

      </wsdl:operation>

</wsdl:portType>
```

6.4 Messages

The following WSDL message definitions are specific to this operation.

6.4.1 getPoliciesMessage

It is a request message used by the RA application to request a policy.

```
<wsdl:message name="getPoliciesMessage">

      <wsdl:part name="request" element="vscep:getPolicies" />

</wsdl:message>
```

- vscep:getPolicies has the actual details for the policy request.

6.4.2 getPoliciesResponse

The RA application receives a response message from the getPoliciesMessage request.

```
<wsdl:message name="getPoliciesResponse">

      <wsdl:part name="response" element="vscep:getPoliciesResponse" />
```

```
</wsdl:message>
```

- getPoliciesResponse contains the details for the policies.

6.5 Elements

The following elements are defined for this protocol.

6.5.1 getPolicies

This complex type element contains the details for the policy request, as well as an optional filtering element to filter any data from the policy.

NOTE: Filtering is not supported in this release.

```
<xs:element name="getPolicies">
      <xs:complexType>
            <xs:sequence>
                  <xs:element name="version" type="vscep:VersionType" />
                  <xs:element name="clientTransactionID" type="vscep:
                  TransactionIDType" minOccurs="0" />
                  <xs:element name="client" type="vscep:Client" minOccurs="0" />
                  <xs:element name="requestFilter" type="vscep:RequestFilter"
                  nillable="true" />
            </xs:sequence>
      </xs:complexType>
</xs:element>
```

- version - Use this element to identify the version of the getPolicy call. For this release, this value is always set to 2.0.
- clientTransactionID - Use this element to specify a transaction ID for the enterprise RA application (optional). The element may be useful if your RA application tracks the request and response.
- client - The <vscep:Client> element is an instance of the Client object. See “Client” on page 99. The <vscep:Client> element contains information about the caller: the caller's preferred language and the date and time of last policy retrieval. This is an optional field.

  NOTE: The preferred language feature is not supported in this release.

- requestFilter - Use this element to specify a filter on the policy data returned. See "RequestFilter".

NOTE: This feature is not supported in this release.

6.5.2 VersionType

Use the VersionType type element to restrict the version element to a numeric value (such as 1.0, 1.1, and so on).

```
<xs:simpleType name="VersionType" final="restriction">
      <xs:restriction base="xs:string">
            <xs:pattern value="\d{1,3}\.\d{0,3}" />
      </xs:restriction>
</xs:simpleType>
```

6.5.3 TransactionIDType

Use the TransactionIDType type element to restrict the system-generated transaction ID to no more than 40 characters.

```
<xs:simpleType name="TransactionIDType" final="restriction">
      <xs:restriction base="xs:string">
            <xs:maxLength value="40" />
      </xs:restriction>
</xs:simpleType>
```

6.5.4 Client

This complex type element provides details of the current policy update time, as well as any additional information that is required.

```
<xs:complexType name="Client">
      <xs:sequence>
            <xs:element name="lastUpdatetime" type="xs:dateTime" nillable
            ="true" />
            <xs:element name="preferredLanguage" type="xs:language" nillable
            ="true" />
            <xs:any namespace="##any" processContents="lax" minOccurs= "0"
            maxOccurs="unbounded" />
      </xs:sequence>
```

```
</xs:complexType>
```

- lastUpdatetime - Use this element to identify the last time the RA application received the policy from DigiCert PKI (in GMT format).

NOTE: In this release, the server always responds with the GetVSPoliciesResponse message.

- preferredLanguage: Use it as an xs:language element to indicate the preferred caller language. The getPoliciesResponse message is returned in the preferred RA application language. If the preferredLanguage value is absent from the LocalizedResources data element, the getPoliciesResponse message uses the language that is specified by the DefaultLanguage data element.

NOTE: The preferredLanguage element is not implemented in this release.

- ##any: The element is provided as a vendor-specific extension, if required.

6.5.5 RequestFilter

Use this complex type element to filter the response of the getPolicies call by specific policy IDs. This is an optional element.

This feature is not implemented in this release.

```
<xs:complexType name="RequestFilter">
      <xs:sequence>
            <xs:element name="policyIDs" type="vscep:FilterOIDCollection"
            nillable="true" />
            <xs:any namespace="##any" processContents="lax" minOccurs="0"
            maxOccurs="unbounded" />
      </xs:sequence>
</xs:complexType>
```

- policyIDs. This PolicyIDCollection element is a collection of policy IDs used by DigiCert PKI to filter the policies that are returned in the getPolicies response. Only the policy IDs included as a part of the collection is sent in the response. If the value is null, the filter element is not used.
  Use this element to construct the filter for specific PolicyIDs. The RA application must first obtain the policies for all the policy IDs associated with the account. As a result, the initial getPolicies call cannot use this filter. Once the RA application has policy OID details, the RA application can use the filter element to update a specific set of policies.

Implementations may allow the policy to be configured in a GUI (Policy Wizard equivalent) and the RA application can then obtain the policy OIDs.

- ##any - This element is reserved for future expansion.

6.5.6 FilterOIDCollection

Use the complex type element FilterOIDCollection to filter the getPolicies response by CertificateEnrollmentPolicy objects. DigiCert PKI returns any CertificateEnrollmentPolicy objects that match the list of OID values that are provided in the FilterOIDCollection element. The list is not ordered.

FilterOIDCollection is not implemented in this release.

```
<xs:complexType name="FilterOIDCollection">
      <xs:sequence>
            <xs:element name="oid" type="xs:string" maxOccurs="unbounded" />
      </xs:sequence>
</xs:complexType>
```

- oid: A string representing an OID. Each <oid> element must be unique in the FilterOIDCollection.

6.5.7 getPoliciesResponse

getPoliciesResponse is the response to the getPolicies request from DigiCert PKI.

```
<xs:element name="getPoliciesResponse">
      <xs:complexType>
            <xs:sequence>
                  <xs:element name="clientTransactionID" type="vscep:
                  TransactionIDType" minOccurs="0" />
                  <xs:element name="serverTransactionID" type="vscep:
                  TransactionIDType" />
                  <xs:element name="response" type="vscep:Response" />
                  <xs:element name="cAs" type="vscep:CACollection" nillable=
                  "true" />
                  <xs:element name="oIDs" type="vscep:OIDCollection" nillable=
                  "true" />
            </xs:sequence>
      </xs:complexType>
</xs:element>
```

- clientTransactionID - If an RA applicationTransactionID is sent in the getPolicies call, it is provided in the getPoliciesResponse.
- serverTransactionID - A transaction ID created by the DigiCert PKI server. The DigiCert PKI server can use this ID to find the transaction in the server log file for troubleshooting purpose.
- response - This is the list of all the policies that are returned for a particular account based on the request. See "Response" on page 107.
- cAs - This is a list of CAs that can issue certificates against the policies that are specified in the vscep: PolicyCollection. See "PolicyCollection" on page 111.
- OIDCollection - This is a list of OIDs that are referenced from the PolicyCollection in the response element.

6.5.8 Response

This is the response object that contains all of the policies that are associated with an account, based on the request.

```
<xs:complexType name="Response">

      <xs:sequence>

            <xs:element name="policyID" type="xs:string" />

            <xs:element name="policyFriendlyName" type="xs:string" minOccurs="0"
            maxOccurs="1" />

            <xs:element name="nextUpdateHours" type="xs:unsignedInt"
            nillable="true" />

            <xs:element name="policiesNotChanged" type="xs:boolean" />

            <xs:element name="policies" type="vscep:PolicyCollection"

            nillable="true" />

            <xs:any namespace="##any" processContents="lax" minOccurs="0"
            maxOccurs="unbounded" />

      </xs:sequence>

</xs:complexType>
```

- policyID: A unique identifier for the certificate enrollment policy. Multiple servers can respond with the same server ID in a getPoliciesResponse message, if they are configured to return the same Response object to the same requestor.

NOTE: For this release, this value is always set to **1**.

- policyFriendlyName: A human-readable friendly name for the certificate enrollment policy.

NOTE: For this release, this value is always set to **DigiCert Policy**.

- nextUpdateHours - An integer representing the number of hours that the DigiCert PKI recommends that the RA application wait before another getPolicies message is submitted. If the <nextUpdateHours> element is present and not nil, the <nextUpdateHours> element value must be a positive nonzero integer.

NOTE: This field not supported in this release.

- policiesNotChanged - This flag indicates whether the policies have changed since the lastUpdateTime sent by the RA application in the getPolicies call.

NOTE: This field not supported in this release. Value is always false.

- policies - This is a list of policies that are associated with the account. See "PolicyCollection" on page 111.
- ##any - This element is reserved for future expansion.

6.5.9 CACollection

This complex type element contains a list of all of the CAs supported by DigiCert PKI for the specified account.

```
<xs:complexType name="CACollection">
      <xs:sequence>
            <xs:element name="cA" type="vscep:CA" maxOccurs="unbounded" />
      </xs:sequence>
</xs:complexType>
```

- cA - This type element contains the vscep: CA element. See "CA".

6.5.10 CA

This complex type element provides a detailed structure for the CA information.

```
<xs:complexType name="CA">
      <xs:sequence>
            <xs:element name="uris" type="xs:anyURI" minOccurs="0"
            maxOccurs="unbounded" />
            <xs:element name="certificate" type="xs:base64Binary" />
            <xs:element name="cAIssuerName" type="xs:string" nillable= "true" />
            <xs:element name="cAReferenceID" type="xs:int" />
            <xs:element name="cAType" type="vscep:CAType" nillable="true" />
            <xs:element name="intermediateCACertificates" type="xs:
            base64Binary" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="rootCACertificate" type="xs:base64Binary"/>
            <xs:any namespace="##any" processContents="lax" minOccurs="0"
            maxOccurs="unbounded" />
      </xs:sequence>
</xs:complexType>
```

- uris - This is a list of URIs where the certificate can be enrolled. The URIs are specific to the CA issuing the certificate. It is intended to be the endpoint for the enrollment service.

NOTE: This field not supported in this release.

- certificate - base64encoded certificate for the CA. This field can be null.
- cAIssuerName - Issuer name for the CA. This field can be null.

NOTE: This field is informational only in this release.

- cAReferenceID - This element identifies the CA to the policies in the list.
- cAType - This element indicates whether the CA is for a public or a private account.
- intermediateCACertificates - A list of intermediate CAs for this profile. Each element is a base64 encoded certificate.
- rootCACertificate - The root CA for this profile. The element is a base64 encoded certificate and is a mandatory element.
- ##any - This element is reserved for future expansion.

6.5.11 OIDCollection

This complex type element is the list of OID objects that are referenced in the response policies. The structure is a sequence of OID elements which contain the OID data.

```
<xs:complexType name="OIDCollection">
      <xs:sequence>
            <xs:element name="oID" type="vscep:OID" minOccurs="1" maxOccurs=
            "unbounded" />
      </xs:sequence>
</xs:complexType>
```

- oID - An instance of vscep:OID which contains details about the object. See "OID" on page 110.

6.5.12 migrationOIDCollection

This complex type element contains a sequence of superceded OIDs. The migrationOIDCollection is of type:

```
<xs:complexType name="MigrationOIDCollection">
      <xs:sequence>
            <xs:element name="migratedFromOID" type="xs:string" minOccurs="0"
            maxOccurs="unbounded"/>
            <xs:element name="migratedToOID" type="xs:string" nillable="true"
            maxOccurs="unbounded"/>
            <xs:any namespace="##any" processContents="lax" minOccurs="0"
            maxOccurs="unbounded" />
      </xs:sequence>
</xs:complexType>
```

6.5.13 OID

This complex type element defines the details that identify an object and provides generic attributes for that object.

```
<xs:complexType name="OID">
      <xs:sequence>
            <xs:element name="value" type="xs:string" />
            <xs:element name="oIDReferenceID" type="xs:int" />
            <xs:element name="group" type="xs:unsignedInt" />
            <xs:element name="defaultName" type="xs:string" nillable= "true" />
            <xs:any namespace="##any" processContents="lax" minOccurs="0"
            maxOccurs="unbounded" />
      </xs:sequence>
</xs:complexType>
```

- value - This field the OID value, in the format 1.23.45.61
- oIDReferenceID - This field a reference ID that the policies use to identify the OID.
- group: This integer value identifies the type of object that the OID object represents. The <group> element must be one of the integer values in Table A-3.

Table A- 3 Integer values for the OID object

Integer Value	Meaning
1	Certificate policy identifier
2	Standard Certificate extension or attribute identifier
3	Private Certificate extension or attribute identifier
4	Hash algorithm identifier
5	"Signing" algorithm identifier
6	Public key identifier
7	Encryption algorithm identifier
8	key usage identifier
9	Extended key usage identifier
10	Enrollment object identifier

- defaultName - This field a human-readable friendly name of the OID.
- ##any - This element is reserved for future expansion.

6.5.14 PolicyCollection

This object contains the list of DigiCert policies for the account. The Policy element represents each policy and contains details about each policy.

```
<xs:complexType name="PolicyCollection">
      <xs:sequence>
            <xs:element name="policy" type="vscep:CertificateEnrollment Policy"
            maxOccurs="unbounded" />
      </xs:sequence>
</xs:complexType>
```

- policy - instance of a vscep:CertificateEnrollmentPolicy element. It contains enrollment details for a particular policy.

6.5.15 CertificateEnrollmentPolicy

This complex type element is the policy structure for DigiCert policy. The response can have multiple policy entries.

```
<xs:complexType name="CertificateEnrollmentPolicy">
      <xs:sequence>
            <xs:element name="policyOIDReference" type="xs:int" />
            <xs:element name="cAs" type="vscep:CAReferenceCollection" />
            <xs:element name="attributes" type="vscep:Attributes" />
      </xs:sequence>
</xs:complexType>
```

- policyOIDReference - This element is an integer value that references the <oIDReferenceID> element of an existing OID object. A <policyOIDReference> element must be present in the set of <oIDReferenceID> element values in the corresponding getPoliciesResponse message.
- cAs - This element contains all of the policy-supported, CA reference IDs. A CA reference ID from this collection can be used to get the actual CAObject element from the CAReferenceCollection in the GetPoliciesResponse message.
- attributes -This element represents an instance of an Attributes object. See "Attributes" on page 113.

6.5.16 CAReferenceCollection

This complex type element contains a list of CA reference IDs associated with the CAObjects in the response. The policy instance uses this element to identify a specific CA.

```
<xs:complexType name="CAReferenceCollection">
      <xs:sequence>
            <xs:element name="cAReference" type="xs:int" maxOccurs=
            "unbounded" />
      </xs:sequence>
</xs:complexType>
```

- cAReference - An integer value that is associated with an existing <cAReferenceID> element in a CA object. The value of <cAReference> element must be unique within each CAReferenceCollection object and must reference a <cAReferenceID> element that is defined in the corresponding getPoliciesResponse.

6.5.17 DuplicateCertPolicyEnum

This element defines how duplicate certificates are handled.

```
<xs:simpleType name="DuplicateCertPolicyEnum">

      <xs:restriction base="xs:string">

            <xs:enumeration value="DUPLICATE_CERT_ALLOWED"/>

            <xs:enumeration value="DUPLICATE_CERT_NOT_ALLOWED"/>

      </xs:restriction>

</xs:simpleType>
```

- DUPLICATE_CERT_ALLOWED - This value indicates that a duplicate certificate (certificate with same subject DN) can be issued against this profile.
- DUPLICATE_CERT_NOT_ALLOWED - This value indicates that a duplicate certificate (certificate with same subject DN) cannot be issued against this profile.

6.5.18 Attributes

The Attribute object contains information about an EnrollmentPolicy sent in the response. It must be present for each Certificate EnrollmentPolicy instance that is sent in the PolicyResponse element.

```
<xs:complexType name="Attributes">

      <xs:sequence>

            <xs:element name="policySchema" type="xs:int" />

            <xs:element name="certificateValidity" type="vscep:Certificate
            Validity" />

            <xs:element name="certificateOverrideValidity" type="vscep:
            OverrideValidity" minOccurs="0" maxOccurs="1" />

            <xs:element name="subjectNameInfo" type="vscep:subjectName"
            nillable="true" />

            <xs:element name="extensions" type="vscep:Extensions"
            nillable="true"/>

            <xs:element name="privateKeyAttributes" type="vscep:

            PrivateKeyInfo" />

            <xs:element name="clientPolicy" type="vscep:ClientPolicy"
            nillable="true" />

            <xs:element name="systemInfo" type="vscep:SystemInformation" min
            Occurs="0" maxOccurs="1" />

            <xs:element name="rAPolicy" type="vscep:RAPolicy" nillable="true" />
            <xs:element name="seatIdInfo" type="vscep:SeatInfoType"
            nillable="true"/>

            <xs:element name="applicationIntegrationInfo" type="vscep:
            ApplicationIntegrationInfoType" nillable="true"/>

            <xs:element name="migrationOIDs" type="vscep:
            MigrationOIDCollection" nillable="true"/>

            <xs:element name="status" type="xs:string"/>

            <xs:any namespace="##any" processContents="lax" minOccurs="0"
            maxOccurs="unbounded" />

      </xs:sequence>

</xs:complexType>
```

policySchema

This element is an integer value representing the version of the corresponding Certificate Enrollment policy data. The <policySchema> element should be an integer value of 1, 2, or 3.

NOTE: For this release, this value is 1.

```
<xs:element name="policySchema" type="xs:int" />
```

6.5.19 SystemInformation

This complex type element contains information and name-value pair details that the RA application must send. This information is not included in the certificate, but is needed to define other system behaviors.

```
<xs:complexType name="SystemInformation">

      <xs:sequence>

            <xs:element name="searchCertificateData" type="vscep:
            SearchCertificateData" minOccurs="0" maxOccurs="1" />

            <xs:element name="cACertPublish" type="vscep:PublishCert"
            minOccurs="0" maxOccurs="1" />

            <xs:element name="cACertPublishNameValuePair" type="vscep:
            CACertPublishNameValuePair" minOccurs="0" maxOccurs="1" />

            <xs:element name="certificateDeliveryFormat" type="vscep:

            DeliveryFormat" />

            <xs:element name="adminInfo" type="vscep:PersonalInfoType"

            minOccurs="0" />
            <xs:element name="serviceEndpointList"type="vscep:Service
            EndpointListType"/>

            <xs:element name="duplicateCertPolicy" type="vscep:Duplicate
            CertPolicyEnum"/>

            <xs:any namespace="##any" processContents="lax" minOccurs="0"
            maxOccurs="unbounded" />

      </xs:sequence>

</xs:complexType>
```

- searchCertificateData - This element contains the name-value pair names for certificate searches. If you send this information, it is stored. The RA application can use this data to search for a specific certificate using LDAP.

NOTE: This feature is not supported in this release.

- cACertPublish - This element indicates whether or not to publish the certificate to the VTN (**yes**, **no**, or **Client Provided**).
- cACertPublishNameValuePair - This element is only present when the cAcertPublish is set to clientProvided. The element contains the name-value pair that the RA application sends to indicate if the certificate should be published to the VTN (**yes** or **no**).
- certificateDeliveryFormat - This element describes the delivery format (set during account configuration) for the certificate in the enrollment request. The RA application can use this information to set the correct Token type in the enrollment request, and to correctly process the response.
- serviceEndpointList - A list of client-required, service endpoints.
- duplicateCertPolicy - This element indicates whether a duplicate certificate with same Subject DN is allowed for this profile. See "DuplicateCertPolicyEnum" on page 113.
- ##any - For future expansion.

6.5.20 PersonalInfoType

This complex type element contains the name, email, and phone number of a contact.

```
<xs:complexType name="PersonalInfoType">

      <xs:sequence>

            <xs:element name="name" type="vscep:string"/>

            <xs:element name="email" type="vscep:string"/>

            <xs:element name="phone" type="vscep:string" nillable="true"/>

            <xs:any namespace="##any" processContents="lax" minOccurs="0"
            maxOccurs="unbounded" />
```

```
    </xs:sequence>
</xs:complexType>
```

- name - This element contains name of the contact.
- email - This element contains the email address of the contact.
- phone - This element contains the phone number of the contact.
- ##any - For future expansion.

6.5.21 ServiceEndPointListType

This complex type element contains a list of profile-supported service endpoints.

```
<xs:complexType name="ServiceEndpointListType">
    <xs:sequence>
        <xs:element name="serviceEndpoint" type="vscep: ServiceEndpointType"
        maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

6.5.22 ServiceEndPointType

This complex type element contains the list of endpoints that the profile supports. Each endpoint contains a type indicating the type of endpoint supported.

```
<xs:complexType name="ServiceEndpointType">
    <xs:sequence>
        <xs:element name="endpointURI" type="xs:anyURI"/>
        <xs:element name="type" type="xs:string"/>
        <xs:any namespace="##any" processContents="lax" minOccurs="0"
        maxOccurs="unbounded" />
    </xs:sequence>
</xs:complexType>
```

Table A-4 list the possible service endpoints that serviceEndpointList returns.

Table A- 4 Service endpoints returned

URL Type	Endpoint Description
digicertCertificateEnrollment PolicyEndpoint	Web Service endpoint of the Certificate Enrollment Policy service
digicertCertificateEnrollmentEndpoint	Web Service endpoint of the Certificate Enrollment service
digicertCertificateManagement Endpoint	Web Service endpoint of the Certificate Management service
enterpriseCertificateEnrollment KeyEscrowEndpoint	Web Service endpoint of the key escrow and recovery service

6.5.23 SeatInfoType

This complex type element contains the seatID name-value pair element.

```
<xs:complexType name="SeatInfoType">

      <xs:sequence>

            <xs:element name="attributeNameValue" type="vscep:
            AttributeValueType"/>

            <xs:element name="attributeNameValueProperty" type="
            vscep:AttributeNameValuePropertyType"/>

            <xs:any namespace="##any" processContents="lax" minOccurs ="0"

            maxOccurs="unbounded" />

      </xs:sequence>

</xs:complexType>
```

6.5.24 ApplicationInstructionsType

This complex element captures the download URL described in the certificate profile.

NOTE: This element is not applicable for PKI Web Services.

```
<xs:complexType name="ApplicationIntegrationInfoType">
      <xs:sequence>
            <xs:element name="applicationInstructionUrl" type="xs: anyURI"/>
      </xs:sequence>
</xs:complexType>
```

6.5.25 DeliveryFormat

This type element contains details about the format in which the issued certificate is delivered. The enrollment service delivers the certificate. The default value is base64 PKCS7; however base64 X.509 and PKCS#7, MIME, and base64 PKCS INDEF are also supported.

```
<xs:simpleType name="DeliveryFormat">
      <xs:restriction base="xs:string">
            <xs:enumeration value="http://docs.oasis-open.org/wss/2004/01/
            oasis-200401-wss-X509-token-profile-1.0#X509v3" />
            <xs:enumeration value="http://docs.oasis-open.org/wss/2004/01/
            oasis-200401-wss-X509-token-profile-1.0#PKCS7" />
            <xs:enumeration value="http://schemas.verisign.com/pkiservices/
            2009/07/PKCS12" />
      </xs:restriction>
</xs:simpleType>
```

6.5.26 CACertPublishNameValuePair

This is an optional element that contains details about the name-value pair. The RA application sends the name-value pair during enrollment, indicating whether to publish the issued certificate to the DTN. This element should only be sent if the cACertPublish flag is set to clientProvided.

The value that is sent in this element is $publish_flag.

```
<xs:simpleType name="CACertPublishNameValuePair">

      <xs:restriction base="xs:string">

            <xs:enumeration value="$publish_flag" />

      </xs:restriction>

</xs:simpleType>
```

- $publish_flag - This element determines whether or not to publish the certificate to the DTN. The values can be **yes** (publish the certificate) or **no** (do not publish the certificate).

6.5.27 SearchCertificateData

This complex type element provides the name-value pairs the RA application sends for certificate searches. This element consists of a sequence of SearchNameValuePairs elements. For this release, this element refers to the email address name-value pair that you can send in the enrollment request.

DigiCert recommends that you send the name-value pair, even if it is not a part of the certificate data, to allow searches in PKI Manager.

```
<xs:complexType name="SearchCertificateData">

      <xs:sequence>

            <xs:element name="searchAttributeNameValuePair"
            type="vscep:AttributeNameValuePairType" minOccurs="0"
            maxOccurs="unbounded" />

      </xs:sequence>

</xs:complexType>
```

6.5.28 certificateValidity

This complex type element contains details about the certificate validity and renewal period.

```
<xs:complexType name="CertificateValidity">
      <xs:sequence>
            <xs:element name="validityPeriodDays" type="xs:unsignedLong" />
            <xs:element name="renewalPeriodDays" type="xs:unsignedLong" />
      </xs:sequence>
</xs:complexType>
```

Use separate complex attributes with a list of name-value pairs to override validity and renewal options. You can override certificate validity for enrollment or renewal using either number of days or start and end dates.

- validityPeriodDays is the recommended validity period for a certificate. If overrideValidity is set to true, the RA application can send in validityNameValuePairNames to change the default validity period.
- renewalPeriodDays is the default renewal overlap period for a certificate. The renewal overlap period (or renewal grace period) is the number of days before a certificate expires when it can be renewed.

6.5.29 overrideValidity

This complex type element allows the enrollment request to override the default values for certificate validity.

```
<xs:complexType name="OverrideValidity">
      <xs:sequence>
            <xs:element name="overrideFlag" type="xs:boolean" minOccurs= "0" />
            <xs:element name="overrideNameValuePair" type="vscep:
            validityNameValuePairNames" minOccurs="0" maxOccurs= "unbounded" />
      </xs:sequence>
</xs:complexType>
```

- overrideFlag - Boolean flag that specifies the enrollment request to override the default set values for certificate validity.
- overrideNameValuePair - Lists the name-value pairs that are used to send the override validity.

6.5.30 validityNameValuePairNames

The enumeration-type element indicates the name-value pairs that can be passed with the validityNameValuePair to override the certificate validity that is set for the account.

Regardless of the values that are provided here, DigiCert PKI does not issue a certificate with a validity beyond the validity of the issuing CA.

```
<xs:simpleType name="validityNameValuePairNames">
      <xs:restriction base="xs:string">
            <xs:enumeration value="$overrideValidityDays" />
            <xs:enumeration value="$overrideValidityStartDate" />
            <xs:enumeration value="$overrideValidityEndDate" />
      </xs:restriction>
</xs:simpleType>
```

- overrideValidityDays - Use this element to override the default number of days the certificate is valid starting from the date of issue.
  If this value is greater than default-validity-period plus validity-grace-period plus one day, then default-validity plus validity-grace-period is used.
- overrideValidityStartDate - Use this element to override the default validity period start date. (mm/dd/yyyy) The certificate is issued with a start time of "00:00:00" GMT on the validity period start date. If overrideValidityDays is also sent, the end date is calculated based on the number of days in overrideValidityDays. Due to the behavior of DigiCert PKI, this value cannot be earlier than 730 days (2 years) from the current date.
- overrideValidityEndDate - Use this element to override the default validity period end date. (Use mm/dd/yyyy format; certificates are issued with an end time of 23:59:59 GMT on the validity period end date.) If overrideValidityDays is sent, it overrides the overrideValidityEndDate value.
  If value is greater than the overrideValidityStartDate attribute plus default-validity-period plus validity-grace-period plus three days, then the value overrides ValidityStartDate plus default-validity-period plus validity-grace-period.

6.5.31 GeneralizedTime

The GeneralizedTime type is a standard ASN.1 type for variable precision representation of time. GeneralizedTime supports the standard X.509 certificate format that is defined in RFC 5280 for a certificate expiry time format

Optionally, the GeneralizedTime field can include a representation of the time differential between local and Greenwich Mean Time (GMT). GeneralizedTime expresses values for the year, month, day, hour, time, minute, and second in any of three forms:

1. Local time only: "YYYYMMDDHHMMSS".
2. Greenwich Mean Time only: "YYMMDDHHMMSSZ".
3. Difference between local and GMT times. "YYYYMMDDHHMMSS+-HHMM".

GeneralizedTime values must be expressed in Greenwich Mean Time and must include seconds, even where the number of seconds is zero. GeneralizedTime values must not include fractional seconds.

6.5.32 PrivateKeyInfo

This complex type attribute contains details that are relevant to key material. Details may include the size of the key, the key usage, and whether the profile supports key escrow.

```
<xs:complexType name="PrivateKeyInfo">

      <xs:sequence>

            <xs:element name="keysize" type="xs:int" />

            <xs:element name="keyescrowpolicy" type="vscep:KeyEscrowPolicyType"
            minOccurs="0" />

            <xs:element name="keyexportable" type="xs:boolean" />

            <xs:element name="keyprotect" type="xs:boolean" minOccurs="0" />

            <xs:element name="algorithmOIDReference" type="xs:int" nillable=
            "true" />

            <xs:element name="cryptoProviders" type="vscep:CryptoProviders"
            nillable="true" />

      </xs:sequence>

</xs:complexType>
```

- Key size and curve size - This element is determined by the type, either RSA, ECC or DSA, determines this element. The element determines the key size and curve size.
- keyescrowpolicy - This element indicates the key escrow policy for the certificate policy.
- keyexportable - This element is a Boolean flag indicating if the private key can be exported.

- keyprotect - This element is a Boolean flag indicating if key is to be protected.
- algorithmOIDReference - This element provides the OID object that corresponds to the asymmetric algorithm of the private key. This OID element is part of the OIDCollections in the response. The OID element includes reference to the Ecliptic Curve Cryptography (ECC) and Digital Signature Algorithm (DSA) OIDs. Currently, for DSA enrollments, only P=2048 (prime) Q=256 (subprime) and P=3072 (prime) Q=256 (subprime) are supported. These details are provided in the defaultName element of the corresponding OID element in OIDCollection. For RSA, this element is null as RSA is the default algorithm supported.

Table A- 5 Values for DSA and ECC in Web Services OID Representation

OID	Description
1.3.132.0.35	ECC Algorithm OID for secp521r1
1.3.132.0.34	ECC Algorithm OID for secp384r1
1.2.840.10045.3.1.7	ECC Algorithm OID for secp256r1
1.3.14.3.2.12	DSA Algorithm OID

- cryptoProviders - This element lists the cryptographic service providers that are configured for the account.

6.5.33 KeyEscrowPolicyType

This complex type element determines the key escrow policy for the certificate profile. The structure is as follows.

```
                <xs:complexType name="KeyEscrowPolicyType">
                       <xs:sequence>
                     <xs:element name="keyEscrowEnabled" type="xs:boolean" />
                     <xs:element name="keyEscrowDeploymentMode" type="xs:string "
                        minOccurs="0" />
                     <xs:element name="keyEscrowDualAdminApprovalRequired" type=
                        "xs:boolean" minOccurs="0"/>
                     <xs:element name="doKeyRecoveryForAdditionalEnrollRequest"
                        type="xs:boolean" nillable="true"/>
                     <xs:any namespace="##any" processContents="lax"
minOccurs="0"
```

```
                    maxOccurs="unbounded"
                    />
                        </xs:sequence>
                    </xs:complexType>
```

- keyEscrowEnabled - Indicates whether key escrow is enabled for this profile.
- keyEscrowDeploymentMode - Indicates the deployment mode for the key manager if key escrow is enabled. Values are:
  - CLOUD if the key escrow store is set to Symantec.
  - ENTERPRISE if the key escrow store is set to a local user store
- keyEscrowDualAdminApprovalRequired – support this option with Local Key Management (ENTERPRISE) only, when return true, we do need dual Administrators Approval request, if false, single Administrator request is enough. Not support on Cloud Key Management. This setting is on certificate profile, need administrator set dual administrator required on each profile.

  If Dual Administrators required for Key Recovery, we need
  - between 1st and 2nd Request should be within 1 hour. If exceeded, request is handled as 1st request and no error
  - If API use same Administrator for 2nd request, get A707 error on API return
- doKeyRecoveryForAdditionalEnrollRequest - Indicates whether multiple device enrollments are supported for this certificate profile. This feature is available for a remote key that has an escrowed profile like an S/MIME profile. When certificates are enrolled for the same user from different devices, Web Service returns the same certificate that is issued during the first enrollment. The exception is when the certificate is in the renewal period. When the latest user's certificate is in the renewal period, a new certificate is returned.

6.5.34 CryptoProviders

This complex type attribute lists the cryptographic providers available for use in generating the private key. The list is not ordered.

```
<xs:complexType name="CryptoProviders">
      <xs:sequence>
            <xs:element name="provider" type="xs:string" maxOccurs=
            "unbounded" />
      </xs:sequence>
</xs:complexType>
```

- provider - This element is the string name of a cryptographic provider. For example, Microsoft Enhanced Cryptographic Provider v1.0.

6.5.35 ClientPolicy

This complex type element contains the RA application-specific policies. The RA application parses these policies and integrates the results into the behavior of the RA application.

NOTE: This element is not supported in this release.

```
<xs:complexType name="ClientPolicy">
      <xs:sequence>
            <xs:element name="clientName" type="xs:string" minOccurs="0" />
            <xs:element name="maxPinLength" type="xs:int" minOccurs="0" />
            <xs:element name="minPinLength" type="xs:int" minOccurs="0" />
            <xs:element name="noOfBadAttempts" type="xs:int" minOccurs= "0" />
            <xs:element name="certRenewalOverlap" type="xs:int"
            minOccurs= "0" />
            <xs:element name="renewExpiredCerts" type="xs:boolean"
            minOccurs="0" />
            <xs:element name="certRenewalMsg" type="xs:string" minOccurs= "0" />
            <xs:element name="certCleanUp" type="xs:boolean" minOccurs= "0" />
            <xs:element name="certPublish" type="vscep:PublishCert" />
            <xs:any namespace="##any" processContents="lax" minOccurs= "0"
            maxOccurs="unbounded" />
      </xs:sequence>
</xs:complexType>
```

- clientName - This element is the name of the RA application to be integrated.
- maxPinLength - This element provides the maximum PIN length for which the RA application prompts the user if the RA application implements PIN-based authentication.
- minPinLength - This element provide the minimum PIN length for which the RA application prompts the user if the RA application implements PIN-based authentication.
- noOfBadAttempts - This element defines the number of bad attempts that are allowed before the account is locked, if the RA application implements PIN-based authentication.

- certRenewalOverlap - This element defines the number of days before the certificate expires that the RA application can make a renewal request.
- renewExpiredCerts - This element is a Boolean value indicating if the RA application can renew an expired certificate.
- certRenewalMsg - This element defines the message string that the RA application displays for renewal messages.
- certCleanUp - This element identifies whether the RA application must remove revoked or expired certificates.
- certPublish - This element defines whether the RA application should publish the certificate to the local datastore.
- ##any - For future expansion.

6.5.36 PublishCert

This enumeration type element sets the certificate publish flag. The flag can be set to one of the following three:

1. yes (publish the certificate to the DTN)
2. no (do not publish the certificate to the DTN)
3. clientProvided (allow the end user to decide to publish the certificate to the DTN).

This value should match the value that is set for cACertPublish.

```
<xs:simpleType name="PublishCert">

      <xs:restriction base="xs:string">

            <xs:enumeration value="yes" />

            <xs:enumeration value="no" />

            <xs:enumeration value="clientProvided" />

      </xs:restriction>

</xs:simpleType>
```

6.5.37 RAPolicy

This complex type element contains general Registration Authority Service information for the account.

NOTE: This element is not supported in this release.

```
<xs:complexType name="RAPolicy">

      <xs:sequence>

            <xs:element name="registerUser" type="xs:boolean" />

            <xs:element name="verifyUser" type="xs:boolean" />

            <xs:element name="authorizationInfo" type="vscep:
            AuthorizationInfoType" minOccurs="0" maxOccurs="unbounded"/>

            <xs:element name="certPublish" type="vscep:PublishCert"
            minOccurs="0"/>

            <xs:element name="pollingPolicy" type="vscep:PollingPolicy Type"
            nillable="true"/>

      </xs:sequence>

</xs:complexType>
```

- registerUser - This element is a Boolean value that indicates if the Registration Authority Service should publish the issued certificate to the local store.

- verifyUser - The element is a Boolean value that indicates if the enrollment request needs to be approved. The Registration Authority Service approves the enrollment request. The approval method is either manual or automated. **Yes** indicates automated approval, and **No** in the enrollment request dictates manual approval.
- ##any - This element is reserved for future expansion

6.5.38 AuthorizationInfoType

This complex type contains the information that the RA authorization module needs to authorize requests.

NOTE: This element is not applicable to PKI Web Services.

```
<xs:complexType name="AuthorizationInfoType">
      <xs:sequence>
            <xs:element name="userAuthorizationInfo" type="vscep:
            UserAuthorizationCollection"/>
            <xs:element name="directoryInfo" type="vscep:Directory InfoType"/>
      </xs:sequence>
</xs:complexType>
```

6.5.39 DirectoryInfoType

This complex type element contains the domain, IP address, and host name that describes the directory.

NOTE: This element is not applicable to PKI Web Services.

```
<xs:complexType name="DirectoryInfoType">
      <xs:sequence>
            <xs:element name="directoryType" type="xs:string"/>
            <xs:element name="domainName" type="xs:string"/>
            <xs:choice>
                  <xs:element name="ipAddress" type="xs:string"/>
                  <xs:element name="hostName" type="xs:string"/>
            </xs:choice>
      </xs:sequence>
</xs:complexType>
```

6.5.40 UserAuthorizationCollection

This complex type element contains a list of directory groups that are used for authorization by the RA.

NOTE: This element is not applicable to PKI Web Services.

```
<xs:complexType name="UserAuthorizationCollection">
      <xs:sequence>
            <xs:element name="authorizedGroup" type="xs:string"
            maxOccurs="unbounded"/>
      </xs:sequence>
</xs:complexType>
```

6.5.41 PollingPolicyType

This complex type element defines the time interval. The enterprise gateway uses the time interval to download a policy from the policy service.

NOTE: This element is not applicable to PKI Web Services.

```
<xs:complexType name="PollingPolicyType">
      <xs:sequence>
            <xs:element name="gatewayPollingTime" type="vscep:
            PollingTimeType"/>
            <xs:any namespace="##any" processContents="lax"
      </xs:sequence>
</xs:complexType>
```

6.5.42 PollingTimeType

This complex type element indicates the number of hours after which the gateway should request a policy.

NOTE: This element is not applicable to PKI Web Services.

```
<xs:complexType name="PollingTimeType">
      <xs:sequence>
            <xs:element name="nextUpdateHours" type="xs:int"/>
      </xs:sequence>
```

```
</xs:complexType>
```

6.5.43 Extensions

The Extension type is used to provide an X.509v3 Certificate Extension.

```
<xs:complexType name="Extensions">
      <xs:sequence>
            <xs:element name="Extension" type="vscep:Extension"
            minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
</xs:complexType>
```

Extension

The extension element contains the details about extensions for which your RA application needs to provide enrollment data. The RA application can provide the data in name-value pairs or in PKCS10 format.

NOTE: Only the name-value pair method is supported for this release.

```
<xs:complexType name="Extension">
      <xs:sequence>
            <xs:element name="extensionOIDReference" type="xs:int" />
            <xs:element name="extensionCriticalFlag" type="xs:boolean" />
            <xs:element name="extensionSyntax" type="vscep:extensionSyntax"
            nillable="true" />
      </xs:sequence>
</xs:complexType>
```

- extensionOIDReference - This element is a reference ID for the OID element (oIDReferenceID) in the OIDCollections element of the response message. The group element of the OID referenced by this field classifies the extension as a standard extension or as a private extension.
- extensionCriticalFlag - This element is a Boolean flag to indicate if the extension is critical or not.

- extensionSyntax - This value element provides a list of AttributeNames and AttributesValues. The RA application can send the AttributeNames and AttributesValues as name-value pairs in the enrollment request. Or, it can construct the extension with the specified AttributeNames and corresponding values in PKCS10 format.

  NOTE: Only the name-value pair method is supported for this release.

extensionSyntax

This element defines the attribute names and values for the extension. The RA application needs to collect this information and sends it to DigiCert PKI either as name value pairs or in PKCS10 format.

NOTE: Only the name-value pair method is supported for this release.

```
<xs:complexType name="extensionSyntax">
      <xs:sequence>
            <xs:choice>
                  <xs:element name="extensionAttributeNameValuePair" type=
                  "vscep:AttributeNameValuePairType" minOccurs="0" maxOccurs
                  ="unbounded" />
                  <xs:element name="extensionValue" type="vscep:extensionValue
                  Type" minOccurs="0" maxOccurs="unbounded" />
            </xs:choice>
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
            maxOccurs="unbounded" />
      </xs:sequence>
</xs:complexType>
```

- extensionAttributeNameValuePair -The name in this element defines the extension attribute name that is added to the certificate. The value contains the enrollment data name-value that DigiCert PKI uses to replace the corresponding value in the extension or the extension attribute. This name-value pair is prefixed with $.
- extensionValue - This optional element is the string representation of the extension value. The value of this element is the name of the enrollment data name-value that DigiCert PKI uses to replace the corresponding value in the extension. This name-value pair is prefixed with $.
- ##other - This element is reserved for future expansion

extensionValueType

The ExtensionValueType element defines if the extension string value is mandatory or optional in the enrollment request. If the mandatory attribute is set to true, the stringValue for the extension is mandatory; otherwise, the stringValue for the extension is optional.

The actual value of this element is the name-value pair sent in the enrollment request, and is added to the certificate extension.

```
<xs:complexType name="extensionValueType">
      <xs:simpleContent>
            <xs:extension base="xs:string">
            <xs:attribute name="mandatory" type="xs:boolean" />
            <xs:attribute name="type" type="xs:string" default="string" />
            </xs:extension>
      </xs:simpleContent>
</xs:complexType>
```

The following are example implementations of an Extension element:

Example 1

```
<ns1:extensions>
      <ns1:Extension>
            <ns1:extensionOIDReference>1</ns1:extensionOIDReference>
                  <ns1:extensionCriticalFlag>false</ns1:extensionCriticalFlag>
                        <ns1:extensionSyntax>
                              <ns1:extensionValue mandatory="true"
                              type="string">$extensions
                        </ns1:extensionValue>
                        </ns1:extensionSyntax>
      </ns1:Extension>
<ns1:Extension>
      <ns1:extensionOIDReference>3</ns1:extensionOIDReference>
            <ns1:extensionCriticalFlag>false</ns1:extensionCriticalFlag>
                  <ns1:extensionSyntax>
                        <ns1:extensionValue mandatory="true" type="VT_BASE64_
                        STRING">$fche
                        ck</ns1:extensionValue>
                  </ns1:extensionSyntax>
            </ns1:Extension>
</ns1:extensions>
```

Example 2

```
<ns1:extensions>
      <ns1:Extension>
            <ns1:extensionOIDReference>2</ns1:extensionOIDReference>
            <ns1:extensionCriticalFlag>false</ns1:extensionCriticalFlag>
            <ns1:extensionSyntax>
                  <ns1:extensionAttributeNameValuePair>
                        <ns1:attributeName>rfc822name</ns1:attributeName>
                              <ns1:attributeNameValue
                              mandatory="true">$mail_email
                        </ns1:attributeNameValue>
                  </ns1:extensionAttributeNameValuePair>
            </ns1:extensionSyntax>
      </ns1:Extension>
</ns1:extensions>
```

The results of the sample implementations are:

Example 1 shows an extension which does not have attributes. The values for the custom extension can be provided using the custom_extension name-value pair. Append _n to the name-value pair to custom_extension to provide multiple custom extensions. The number of extensions may be from _1 to _n. n is the total number of custom extensions defined. These name-value pairs should be sent in the enrollment request.

This example uses two custom extensions (referenced by extensionOIDReference 1 and 3) to identify the extensions and the fcheck name-value pairs. The user sends the values as part of the enrollment request.

In Example 2, the subjectAltName (represented by extensionOIDRefernce=2) has one attribute named rfc822name. The RA application needs to send the value as part of the mail_email name-value pair in the enrollment data to populate this extension.

6.6 Subject Distinguished Name (DN)

Table A-6 defines the certificate request name-value pairs available for use in the RA application, and maps them to the corresponding attribute in the DigiCert PKI. Except as noted, all values support the VT_PRINTABLE_STRING, VT_IA5_STRING, VT_T61_STRING, and VT_UTF8_STRING data types.

Table A- 6 Data Type mappings

VT_ Data Types	Standard Data Types
VT_IA5_STRING	IA5 string
VT_PRINTABLE_STRING	PrintableString
VT_T61_STRING	T.61 (or Teletext 61) string
VT_UTF8_STRING	UTF-8 string

The following values do not appear in the issued certificate, but do determine how the issued certificate behaves.

- overrideValidityDays
- overrideValidityStartDate
- overrideValidityEndDate
- publish_flag
- seat_id

6.6.1 AttributeNameValuePairType

The complex type element AttributeNameValuePairType identifies the name-value pair for the Subject DN or extension attribute. This name-value pair is sent in the enrollment request.

```
<xs:complexType name="AttributeNameValuePairType">
      <xs:sequence>
            <xs:element name="attributeName" type="xs:string" />
            <xs:element name="attributeNameValue" type="vscep:Attribute
            ValueType"/>
            <xs:element name="attributeNameValueProperty" type=" vscep:
            AttributeNameValuePropertyType" />
      </xs:sequence>
</xs:complexType>
```

- attributeName - This element is the name of the attribute being defined as part of the attributeNameValuePair. It is used to identify the attribute name for Subject DN or an extension.
- attributeNameValue - This element is the value of the attribute being defined as part of the attributeNameValuePair. It is used to define the attribute value for Subject DN or an extension.

6.6.2 AttributeValueType

AttributeValueType indicates whether the name-value pair value is mandatory or not, and what data type the value is required to be. The value is an enrollment name-value pair and should be sent in an enrollment request.

```
<xs:complexType name="AttributeValueType">
      <xs:simpleContent>
            <xs:extension base="xs:string">
                  <xs:attribute name="mandatory" type="xs:boolean" />
                  <xs:attribute name="type" type="xs:string" />
            </xs:extension>
      </xs:simpleContent>
</xs:complexType>
```

- mandatory - This attribute is a Boolean value that indicates if the field is a mandatory field in the enrollment request. This field is optional. If not present it means that the field is optional.
- type - This field indicates the data type for the Subject DN attribute value. If not present it means that the field is treated as a Unicode (UTF-16) string.

6.6.3 AttributeNameValuePropertyType

The element provides the additional information that is required for you to build the RA application. The RA application collects name-value pairs from users who enroll for certificates against the policy.

```
<xs:complexType name="AttributeNameValuePropertyType">
      <xs:sequence>
            <xs:element name="value" type="xs:string" nillable="true"/>
            <xs:element name="source" type="xs:string" minOccurs="0"/>
            <xs:element name="sourceAttributeName" type="xs:string"
            minOccurs="0"/>
            <xs:element name="mandatory" type="xs:boolean" nillable="true"/>
            <xs:element name="overridable" type="xs:boolean" nillable="true"/>
      </xs:sequence>
</xs:complexType>
```

- source - The client uses this element to determine from where the attribute value should be obtained. Possible values include:
  - CONSTANT - Value is obtained from the defaultDisplayLabel element.
  - USER_INPUT - Value is obtained from the user input. The value is not applicable for PKI Web Services.
  - PASSCODE - The enrollment service obtains this value from passcode information. This value is not applicable for PKI Web Services.
  - ACTIVE_DIRECTORY - Value should be obtained from the Active Directory. This value is not applicable for PKI Web Services.
  - WEBSERVICE_CLIENT - RA-application-sent value.
- internalName - The client uses this value to fetch information from an LDAP directory or AD.
- defaultDisplayLabel - If the value source is CONSTANT, the client uses this value as the constant value for the element that is set in the profile.
- defaultDisplayID - The client uses this value as a key that can be looked up in an internationalization resource bundle to display a label to collect information for this attribute
- mandatory - Indicates whether this attribute value is mandatory or not.
- overridable - Indicates whether this attribute value can be administratively overridden.

6.6.4 subjectName

This element provides the details about the Subject DN for the certificate.

```
<xs:complexType name="subjectName">
      <xs:sequence>
            <xs:element name="subjectNameAttribute" type="vscep:subjectName
            Attribute" minOccurs="0" maxOccurs="unbounded" />
            <xs:element name="overrideSubjectNameFormat" type="xs: boolean" />
            <xs:any namespace="##any" processContents="lax" minOccurs="0"
            maxOccurs="unbounded" />
      </xs:sequence>
</xs:complexType>
```

- subjectNameAttribute - This element defines the syntax of the SubjectName. If overrideSubjectNameFormat is set to true, then this is an optional field.

- overrideSubjectNameFormat - This Boolean element indicates whether the SubjectName format is enforced. If this value is true the SubjectNameAttribute is null, and the enterprise RA application sends the SubjectName format in PKCS10 format.

  This flag is always false in this release.
- ##any - This element is reserved for future expansion.

6.6.5 subjectNameAttribute

This element defines the attribute names in the SubjectDN, and the values in the subjectDN attributes. The RA application needs to collect these values and send them to DigiCert PKI. The RA application can choose to send the values as name value pairs or in PKCS10 format.

NOTE: Only the name-value pair method is supported in this release.

All of the SubjectDN attributes are optional; however, one attribute should be sent with the enrollment data. That attribute guarantees uniqueness of the certificate.

```
<xs:complexType name="subjectNameAttribute">
      <xs:sequence>
            <xs:element name="subjectNameAttributecount" type="xs:int"
            nillable="true" />
            <xs:element name="subjectNameAttributeNameValuePair" type=
            "vscep:AttributeNameValuePairType" minOccurs="1" maxOccurs=
            "unbounded" />
      </xs:sequence>
</xs:complexType>
```

- subjectNameAttributecount - This element is an integer value that specifies the number of times the attribute appears in subjectDN. The element is optional.
- subjectNameAttributeNameValuePair - This element defines the attributes to be added to the Subject DN. This element can appear in the policy response multiple times (equal to the value of the subjectNameAttributeCount element).
  - For the name-value pair method that defines the Subject DN attributes:
    - The name of this name-value pair indicates the attribute name in the Subject DN.
    - The value of this name-value pair indicates the name of the pair that the RA application sends with the enrollment data for a given SubjectName AttributeName. The name must be prefixed with $ to indicate that the name-value pair value that is sent in the enrollment request is used in the certificate.

- If using the PKCS10 method to define the component attributes, construct the Subject DN in the PKCS10 format using the subjectNameAttributecount and subjectNameAttributeNameValuePair.
    - The subjectNameAttributecount determines the number of times the element can appear in the response.
    - The subjectNameAttributeNameValuePair defines the subject Name attributes in the Subject DN, including the type and mandatory attributes of the value element.

NOTE: This method is not supported in this release.

The following is an example implementation of Subject Name attributes:

```
<subjectNameInfo>
      <subjectDNAttribute>
            <SubjectDNAttributecount>2</SubjectDNAttributecount>
            <subjectNameAttributeNameValuePair>
                  <attributeName>AT_ORG_UNIT</attributeName>
                  <attributevalue type="VT_T61_STRING" mandatory="false">
                  $cert_org_unit</attributevalue>
            </subjectNameAttributeNameValuePair>
            <subjectNameAttributeNameValuePair>
                  <attributeName>AT_ORG_UNIT/attributeName>
                  <attributevalue type="VT_T61_STRING" mandatory="false">
                  $cert_org_unit_1</attributevalue>
            </subjectNameAttributeNameValuePair>
      </subjectDNAttribute>
      <subjectDNAttribute>
            <SubjectDNAttributecount>1</SubjectDNAttributecount>
            <subjectNameAttributeNameValuePair>
                  <attributeName>AT_COMMON_NAME</attributeName>
                  <attributevalue type="VT_T61_STRING" mandatory="false">
                  $common_name</attributevalue>
            </subjectNameAttributeNameValuePair>
      </subjectDNAttribute>
</subjectNameInfo>
```

The results of this sample implementation are:

This subjectNameInfo object specifies two attributes in the subject DN (AT_ORG_UNIT & AT_COMMON_NAME).

- AT_ORG_UNIT appears in the certificate twice. Its value is equal to the values of the cert_org_unit and cert_org_unit_1 name-value pairs from the enrollment request.
- AT_COMMON_NAME appears in the certificate once and its value is equal to the value of the common_name name-value pair from the enrollment request.

6.6.6 CAType

This enumeration type element specifies whether the CA for the account is part of a public or a private hierarchy.

```
<xs:simpleType name="CAType">
      <xs:restriction base="xs:string">
            <xs:enumeration value="public" />
            <xs:enumeration value="private" />
      </xs:restriction>
</xs:simpleType>
```

6.6.7 SearchNameValuePairs

This simple type element contains the name-value pair names that the RA application can send in the enrollment request for certificate searches.

```
<xs:simpleType name="SearchNameValuePairs">
      <xs:restriction base="xs:string">
            <xs:enumeration value="$mail_firstname" />
            <xs:enumeration value="$mail_lastname" />
      </xs:restriction>
</xs:simpleType>
```

- $mail_firstname - This element sets the subscriber first name as a certificate search field.
- $mail_lastname - This element sets the subscriber last name as a certificate search field.

APPENDIX B

7 Certificate Enrollment Protocol Definition

This appendix includes the following topics:

7.1 About the Certificate Enrollment Protocol

This section describes the protocol that allows an RA application to enroll for a new certificate.

7.2 Certificate Enrollment Protocol (Request Security Token) Overview

Enrolling for a certificate is basically a two-step process:

1. Send a requestSecurityToken call to the DigiCert PKI. This call includes all of the enrollment data that is required to successfully enroll for a certificate.
2. Receive a requestSecurityTokenResponse response from DigiCert PKI.

    - If the certificate request is approved, this response includes the approved certificate. The certificate can be in PKCS#7, or, for the profiles that support key escrow, PKCS#12 format.
    - If the certificate cannot be issued immediately, DigiCert PKI responds with a pending message and a request ID. The RA application can use the request ID to query DigiCert PKI for the certificate status.

    NOTE: This option is not supported in this release.

    - If the certificate cannot be issued, DigiCert PKI responds with a SOAP Fault Message.

The RA application may require additional steps (for example, obtaining the subscriber name, email address), but these are handled by your RA application and are not described here.

DigiCert PKI HTTPS client-server certificate authentication secures communication with DigiCert PKI. A client certificate or RA certificate is used to identify the RA application request. The RA certificate is used to verify the DigiCert PKI account and configured policies.

See “About PKI Web Services” on page 16.

Additionally, WS-Trust and WS-Security are defined in the enrollment schema file. If you develop your own certificate enrollment solution for your RA application, you need to incorporate WS-Trust and WS-Security into that certificate enrollment solution.

7.2.1 Certificate Enrollment Protocol Details

The first time an enrollment is made for an account, the RA application must first use the Certificate Enrollment Policy Protocol. For each subsequent enrollment request, use of the Certificate Enrollment Policy Protocol is optional. The RA application uses the protocol to download the certificate policies that were configured during account set-up. The Certificate Enrollment Policy Protocol response is a policy file containing the OID values identifying the policies available for the account. The enrollment request must include the specific name-value pairs that are associated with the policy for which the enrollment request is made.

In its response, the Certificate Enrollment Policy Protocol defines the enrollment data that the RA application must send with the enrollment request. The response also contains non-subscriber data that defines how the certificate behaves (whether to publish the certificate in the VTN). The RA application gathers the data and sends it with the RequestSecurityToken request as name-value pairs. The RA application also sends the public key in PKCS #10 format.

7.2.2 Name/Value Pairs

Standard name-value pairs are used to send values for the subject DN attribute and standard extension values (as defined by X.509 standard).

The name-value pairs in your enrollment request must match the pairs in your certificate policies that were returned in the Certificate Enrollment Policy Protocol response. The name-values pairs in your certificate policies were configured for the account during account set-up.

Do not use the percentage character (%) or URL encoding for values in name-value pairs. For example, use Acme Bank rather than Acme%20Bank.

7.2.3 Custom Extensions

The Certificate Enrollment Policy Protocol defines the details for standard extensions. However, the Certificate Enrollment Policy Protocol response may contain customer-specific private extensions called custom extensions. If the value to be sent is an ASN1 or binary blob then it should be sent as a base64-encoded string representation of the blob.

If your certificate policy requires multiple custom_extension name value pairs, send them using custom_extension_1...custom_extension_n (where "n" is the number of policy custom extensions).

7.2.4 Certificate Enrollment Protocol Details for Renewal

Renewal is a special case of enrollment request. The same RequestSecurityToken operation is used for renewing the certificate, with a different request type element. Renewal must be performed in the renewal overlap period, or DigiCert PKI returns a duplicate certificate error.

- Enrolling for a new certificate using the original name-value pair information requires the RA application to collect the end user's original enrollment information (name-value pairs). The RA application then sends a RequestSecurityToken operation with an Issue request type element. The operation must be done in the renewal overlap period.
- Renewal with the user's x.509 certificate – This renewal method uses the end user's x.509 certificate to send renewal information. The RA application sends in two binary security tokens. One token is a PKCS#10/SPKAC value type token containing the public key to be used to issue the renewed certificate. The public key can be newly generated, or the same public key that is sent in the original enrollment (DigiCert recommends generating a new key.)

The second binary security token must be either a base64 encoded x.509 certificate or a PKCS#7 message. DigiCert PKI uses it as proof of possession to renew the certificate. The base64 encoded x.509 certificate is the original certificate to be renewed. Your RA application needs to obtain the certificate in a secure manner. For example, your RA application must obtain it from your certificate datastore after the end user is authenticated. Or, you must authenticate the end user to your RA application using the end user's certificate, and provide that certificate to DigiCert PKI.

For SPKAC and PKCS#10 CSR formats, you must send the appropriate request type element:

- http://schemas.verisign.com/pkiservices/2009/07/PKCS10
- http://schemas.verisign.com/pkiservices/2009/07/SPKAC

Note: During the RA renewal process only, the CSR passes through the request process, but the new key is ignored in the back end. The back end uses the existing private key and issues certificates against it.

NOTE: For ECC and DSA, the PKCS#10 CSR format is supported, but the SPKAC format is not supported.

Table B-1 provides an overview of RequestSecurityToken WSDL and its prerequisites, and cross-references to the sections in this guide that contain more information and code samples for its operations and messages.

Table B- 1 Certificate Enrollment Protocol WSDL operations and messages

Name	Description	See...
Operations		
RequestSecurityToken	Protocol that requests certificate enrollment for the RA application	See "RequestSecurityToken" on page 145.
Messages		
RequestSecurityToken Msg	Message used by RequestSecurityToken Msg to request a certificate	See "RequestSecurityTokenMsg" on page 145.
RequestSecurity TokenResponseMsg	Response message received by the RA application	See "RequestSecurityTokenResponse Msg" on page 145.

Table B-2 provides an overview of RequestSecurityToken schema and its prerequisites, and cross-references to the sections in this guide that contain more information and code samples for its elements.

Table B- 2 Certificate Enrollment Protocol schema elements

Name	Description	See...
Request Elements	Provides the information about the request	See "Request Elements" on page 146.
Request Elements (Complex Types)		
RequestVSSecurityToken EnrollmentType	Contains the elements for the certificate request in the RequestSecurityToken message	See "RequestVSSecurityTokenEnrollmentType" on page 147.
RequestVSSecurityToken ResponseEnrollmentType	Contains the elements that are part of a server response to the RequestSecurityToken message	See "RequestVSSecurityTokenResponseEnrollmentType" on page 150.
RequestedVSSecurityToken EnrollmentType	Defines the contents of the certificate	See "RequestedVSSecurityTokenEnrollmentType" on page 152.
NameValueType	Communicates the name-value pairs in the request	See "NameValueType" on page 153.

7.3 Operations

The following WSDL operations are specific to the Certificate Enrollment Protocol.

7.3.1 RequestSecurityToken

The RequestSecurityToken operation performs certificate enrollment requests, renewal requests, as well as retrieval of a pending certificate.

```
<wsdl:operation name="RequestSecurityToken">
      <wsdl:input message="wst:RequestSecurityTokenMsg" />
      <wsdl:output message="wst:RequestSecurityTokenResponseMsg" />
</wsdl:operation>
```

7.4 Messages

The following WSDL message definitions are specific to this operation.

7.4.1 RequestSecurityTokenMsg

The message is the request message that is used for enrollment and renewal operations.

```
<wsdl:message name="RequestSecurityTokenMsg">
      <wsdl:part name="request" element="wst:RequestSecurityToken" />
</wsdl:message>
```

- RequestSecurityToken contains the actual details for the enrollment or the renewal request. See “RequestSecurityToken” on page 131.

7.4.2 RequestSecurityTokenResponseMsg

The message is the response message that is used for enrollment and renewal operations.

```
<wsdl:message name="RequestSecurityTokenResponseMsg">
      <wsdl:part name="response" element="wst:RequestSecurityTokenResponse" />
</wsdl:message>
```

- RequestSecurityTokenResponse element contains the details for the certificate enrollment. See “RequestSecurityTokenResponse” on page 140.

7.5 Elements

The following elements are defined for this protocol.

7.5.1 Request Elements

VersionType

```
<xs:simpleType name="VersionType" final="restriction">
      <xs:restriction base="xs:string">
            <xs:pattern value="\d{1,3}\.\d{0,3}" />
      </xs:restriction>
</xs:simpleType>
```

The type is a restricted string type. It is used to communicate the version of the RA application to the server, so the server can respond properly to the RA application.

TransactionIDType

```
<xs:simpleType name="TransactionIDType" final="restriction">
      <xs:restriction base="xs:string">
            <xs:maxLength value="40" />
      </xs:restriction>
</xs:simpleType>
```

The type is a restricted string type, which identifies the transaction ID elements, such as clientTransactionID and serverTransactionID. The clientTransactionID is used to match RA application and server requests. The serverTransactionID is used to identify the transaction for troubleshooting purposes.

RequestVSSecurityToken

Similar to RequestSecurityToken, this element is of type RequestVSSecurityTokenEnrollmentType. See "RequestVSSecurityTokenEnrollmentType".

RequestSecurityTokenResponse

Similar to RequestSecurityTokenResponse, this element is of type RequestVSSecurityTokenResponseEnrollmentType. See "RequestVSSecurityTokenResponseEnrollmentType".

RequestedVSSecurityToken

Similar to RequestedSecurityToken, this element is of type RequestedVSSecurityTokenEnrollmentType. See "RequestedVSSecurityTokenEnrollmentType".

RequestType

RequestType is a DigiCert -defined element similar to the RequestType element that is defined in WS - Trust 1.3. Use this element to indicate the request type: issue a certificate, renew a certificate, or query the certificate status.

RequestSecurityToken

RequestSecurityToken is defined in WS - Trust 1.3. This element contains the details about the certificate being requested.

RequestSecurityTokenResponse

RequestSecurityTokenResponse is defined in WS - Trust 1.3. This element contains the certificate requested.

TokenType

TokenType element is defined in WS - Trust 1.3. This element defines the type of certificate being requested.

7.5.2 Request Elements (Complex Types)

RequestVSSecurityTokenEnrollmentType

Similar to RequestSecurityTokenType, this complex type contains the elements for the certificate request in the RequestSecurityToken message. It is an RA application-provided object for a certificate enrollment request. It is an extension to wst:RequestSecurityTokenType.

```
<xs:complexType name="RequestVSSecurityTokenEnrollmentType">

      <xs:sequence>

            <xs:element name="certificateProfileID" type="xs:string" />

            <xs:element name="clientTransactionID" type="vswstep:
            TransactionIDType" minOccurs="0" />

            <xs:element name="tokenType" type="vswstep:TokenType"

            minOccurs="0" />

            <xs:element name="requestType" type="vswstep:RequestTypeEnum" />

            <!--
            For enrollment or renewal operation

            valueType for certificate enrollment/renewal/pick operations
            http://schemas.verisign.com/pkiservices/2009/07/PKCS10
            http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-

            X509-token-profile-1.0#PKCS7

            http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss- X509-token-
            profile-1.0#X509v3

            -->

            <xs:element name="binarySecurityToken" type="wsse:
            BinarySecurityTokenType" minOccurs="0" maxOccurs="unbounded" />

            <xs:element name="additionalContext" type="auth:
            AdditionalContextType" minOccurs="0" />

            <xs:element name="pendingTokenReferenceID" type="xs:string"
            minOccurs="0" />

            <!-- For future extensions -->

            <xs:element name="nameValuePair" type="vswstep:NameValueType "
            minOccurs="0" maxOccurs="unbounded" />

            <xs:element name="version" type="vswstep:VersionType" />

            <xs:any namespace="##any" processContents="lax" minOccurs="0"
            maxOccurs="unbounded" />

      </xs:sequence>

      <xs:attribute name="preferredLanguage" type="xs:language"

      use="optional" />

      <xs:anyAttribute namespace="##other" processContents="lax" />

</xs:complexType>
```

- certificateProfileID is the ID for certificate profile. It is the Policy OID value as defined for the VSOID OID defined in the Certificate Enrollment Policy protocol. If the Policy OID value element is not included in the request, the DigiCert Enrollment Service returns an error.
- clientTransactionID is the transaction ID sent by the RA application to track responses from DigiCert PKI.
- tokenType can be one of the following (based on how the policy was configured):

  - x509 (http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-X509-token-profile-1.0#X509v3) for end-user certificate enrollment returning the end-user certificate only.
  - PKCS#7 (http://docs.oasis-open.org/wss/2004/01/oasis-200401-%20wss-X509-token-profile-1.0#PKCS7) for end-user certificate enrollment returning the end-user certificate and the CA chain.

  - PKCS#12 (http://schemas.verisign.com/pkiservices/2009/07/PKCS12) for returning an escrowed key, the associated certificate, and its password.

- requestType is a DigiCert-defined element based on WS-Trust 1.3. The possible values for requestType are:

  - Issue (http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue) for new certificate enrollment.
  - Renew (http://docs.oasis-open.org/ws-sx/ws-trust/200512/Renew) for certificate renewal.
  - QueryTokenStatus (http://schemas.verisign.com/pkiservices/2009/07/QueryTokenStatus) for token status query.

  NOTE: QueryTokenStatus is not supported in this release.

- binarySecurityToken is a wsse:BinarySecurityToken element. It provides the PKCS10 public key, base64-encoded format in the certificate enrollment or renewal request.

  - For new enrollments, the value type for binarySecurityToken is PKCS#10 (http://schemas.verisign.com/pkiservices/2009/07/PKCS10).
  - For renewal request, ValueType for BinarySecurityToken can be:

    - One wsse:BinarySecurityTokens: ValueType of PKCS#10 (http://schemas.verisign.com/pkiservices/2009/07/PKCS10) for the certificate request.
    - Two binarySecurityTokens: ValueType of PKCS#10 (http://schemas.verisign.com/pkiservices/2009/07/PKCS10) for the certificate request. The other is for the end-user certificate, ValueType: (http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-X509-token-profile-1.0#X509v3).

- additionalContext provides extra authentication information in a wst:RequestSecurityToken message. This is an optional element, and should be omitted if there is no extra information to be passed.

NOTE: additionalContext is not supported in this release.

- pendingTokenReferenceID provides a certificate pickup ID. If the enrollment request is in a pending state, the RA application can use this element to check the status of the request. When the administrator approves the certificate request, the certificate is downloaded automatically.

NOTE: pendingTokenReferenceID is not supported in this release.

- nameValuePair is used to notify the server of the name value pairs that are included in a request operation. (A PKCS#10 message in a certificate request contains only the public key. All other information is provided in name value pairs.) This element can appear multiple times.
- version is the schema version that the RA application supports. This element allows the RA application to specify which version it expects in the response.

NOTE: For this release, this value should be 2.0.

- ##any - This element is reserved for future expansion.
- preferredLanguage defines the preferred language to be used in a server response.

NOTE: Only English (en-us) is supported in this release.

- ##other - This element is reserved for future expansion.

RequestVSSecurityTokenResponseEnrollmentType

The RequestVSSecurityTokenResponseEnrollmentType complex type contains the elements that are part of a server response to the RequestSecurityToken message.

```
<xs:complexType name="RequestVSSecurityTokenResponseEnrollmentType">

      <xs:sequence>

            <xs:element name="clientTransactionID" type="vswstep:
            TransactionIDType" minOccurs="0" />

            <xs:element name="serverTransactionID" type="vswstep:
            TransactionIDType" />

            <xs:element name="tokenType" type="vswstep:TokenType"

            minOccurs="0" />

            <xs:element name="dispositionMessage" type="xs:string"

            minOccurs="0" /><!--

            valueType for certificate enrollment/renewal/pick operations
            (pkcs12) http://schemas.verisign.com/pkiservices/2009/07/

            PKCS12(certificate chain or CRL)

            http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss- X509-token-
            profile-1.0#PKCS7 (certificate)

            http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss- X509-token-
            profile-1.0#X509v3-->

            <xs:element name="binarySecurityToken" type="wsse:
            BinarySecurityTokenType" minOccurs="0" />
            <xs:element name="requestedVSSecurityToken"
            type="vswstep:RequestedVSSecurityTokenEnrollmentType"

            minOccurs="0" />

            <xs:element name="version" type="vswstep:VersionType" />

            <xs:any namespace="##targetNamespace" processContents= "lax"
            minOccurs="0" maxOccurs="unbounded" />

      </xs:sequence>

    <xs:attribute name="preferredLanguage" type="xs:language"

    use="optional" />

    <xs:anyAttribute namespace="##other" processContents="lax" />

</xs:complexType>
```

- clientTransactionID is the transaction ID provided by the RA application to track responses from DigiCert PKI. The value is sent back only when clientTransactionID is supplied in the request.
- serverTransactionID is the transaction ID created by the DigiCert PKI server. The DigiCert PKI server can use this ID to find the transaction in the server log file for troubleshooting purpose.
- tokenType can be one of the following (based on how the policy is configured):
  - (http://docs.oasis-open.org/wss/2004/01/oasis-200401-%20wss-X509-token-profile-1.0#X509v3) for end-user certificate enrollment returning the end-user certificate only.
  - PKCS#7 (http://docs.oasis-open.org/wss/2004/01/oasis-200401-%20wss-X509-token-profile-1.0#PKCS7) for end-user certificate enrollment returning the end-user certificate and the CA chain.
  - PKCS#12 (http://schemas.verisign.com/pkiservices/2009/07/PKCS12) for returning an escrowed key, the associated certificate, and its password.
- dispositionMessage can be empty or it can be used to communicate more information to the RA application. For example, certificate issued, certificate request pending, and so on.
- binarySecurityToken is used to deliver the base64 encoded string for the certificate in the response. ValueType: PKCS#7 (http://docs.oasis-open.org/wss/2004/01/oasis-200401-%20wss-X509-token-profile-1.0#PKCS7).
- version is the schema version. The DigiCert PKI server communicates the schema version in the response to the RA application so that it knows whether it can consume the schema.
- preferredLanguage defines the preferred language that is used in the server response.

NOTE: Only English (en-us) is supported in this release.

RequestedVSSecurityTokenEnrollmentType

The RequestedVSSecurityTokenEnrollmentType complex type appears as follows:

```
<xs:complexType name="RequestedVSSecurityTokenEnrollmentType">
      <xs:choice>
            <xs:sequence>
                  <xs:element name="binarySecurityToken" type="wsse:
                  BinarySecurityTokenType" />
                  <xs:element name="pKCS12Password" type="xs:string"
                  minOccurs="0" />
            </xs:sequence>
            <xs:element name="pendingTokenReferenceID" type="xs:string" />
      </xs:choice>
</xs:complexType>
```

DigiCert-issued certificates are in ASN1 format and they are base-64 encoded. RequestedVSSecurityTokenEnrollmentType has four possible contents:

- The value type for wsse:BinarySecurityToken is (http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-X509-token-profile-1.0#X509v3). The certificate is delivered as an X509 certificate. There is no PKCS12Password, nor PendingTokenReferenceID element.
- The value type for wsse:BinarySecurityToken is PKCS#7 (http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-X509-token-profile-1.0#PKCS7). The certificate is delivered as a PKCS7 message, which contains the end-user certificate and its intermediate CAs. There is no PKCS12Password, nor pendingTokenReferenceID element.
- The value type for wsse:BinarySecurityToken is PKCS#12 (http://schemas.verisign.com/pkiservices/2009/07/PKCS12). The certificate is delivered in PKCS12 format with a PKCS12Password. There is no PendingTokenReferenceID element.
- A PendingTokenReferenceID is present. No BinarySecurityToken or PKCS12Password elements.

NOTE: The type is not supported in this release.

NameValueType

NameValueType is a DigiCert-defined data type. It is used to communicate name-value pairs.

- If the value is a string, stringValue element is used.
- If the value is a binary blob, base64 encoded string value is used.
- For key escrow profiles, an additional NameValueType element (with name as “keysize” and value as defined by the certificate profile) should be sent in the enrollment request.

```
<xs:complexType name="NameValueType">
      <xs:sequence>
            <xs:element name="name" type="xs:string" />
            <xs:element name="value" type="xs:string" />
      </xs:sequence>
</xs:complexType>
```

7.5.3 RequestSecurityTokenType

The RequestSecurityTokenType is a complex type (defined in WS - Trust 1.3) that contains the elements for the certificate request in the RequestSecurityToken message. It is the RA application-provided object for a certificate enrollment request.

```
<xs:complexType name="RequestSecurityTokenType">
      <xs:annotation>
            <xs:documentation>
Actual content model is non-deterministic, hence wildcard.
The following shows intended content model:
                  <xs:element ref=‘wst:TokenType’ minOccurs=‘0’ />
                  <xs:element ref=‘wst:RequestType’ />
                  <xs:element ref=‘wsp:AppliesTo’ minOccurs=‘0’ />
                  <xs:element ref=‘wst:Claims’ minOccurs=‘0’ />
                  <xs:element ref=‘wst:Entropy’ minOccurs=‘0’ />
                  <xs:element ref=‘wst:Lifetime’ minOccurs=‘0’ />
                  <xs:element ref=‘wst:AllowPostdating’ minOccurs=‘0’ />
                  <xs:element ref=‘wst:Renewing’ minOccurs=‘0’ />
                  <xs:element ref=‘wst:OnBehalfOf’ minOccurs=‘0’ />
                  <xs:element ref=‘wst:Issuer’ minOccurs=‘0’ />
```

```
                <xs:element ref='wst:AuthenticationType' minOccurs='0' />

                <xs:element ref='wst:KeyType' minOccurs='0' />

                <xs:element ref='wst:KeySize' minOccurs='0' />

                <xs:element ref='wst:SignatureAlgorithm' minOccurs='0' />

                <xs:element ref='wst:Encryption' minOccurs='0' />

                <xs:element ref='wst:EncryptionAlgorithm' minOccurs='0' />

                <xs:element ref='wst:CanonicalizationAlgorithm'

                minOccurs= '0' />

                <xs:element ref='wst:ProofEncryption' minOccurs='0' />

                <xs:element ref='wst:UseKey' minOccurs='0' />

                <xs:element ref='wst:SignWith' minOccurs='0' />

                <xs:element ref='wst:EncryptWith' minOccurs='0' />

                <xs:element ref='wst:DelegateTo' minOccurs='0' />

                <xs:element ref='wst:Forwardable' minOccurs='0' />

                <xs:element ref='wst:Delegatable' minOccurs='0' />

                <xs:element ref='wsp:Policy' minOccurs='0' />

                <xs:element ref='wsp:PolicyReference' minOccurs='0' />

                <xs:any namespace='##other' processContents='lax'
            minOccurs='0' maxOccurs='unbounded' />

            </xs:documentation>

        </xs:annotation>

        <xs:sequence>

            <xs:any namespace="##any" processContents="lax"
            minOccurs="0" maxOccurs="unbounded" />

        </xs:sequence>

        <xs:attribute name="Context" type="xs:anyURI" use="optional" />

        <xs:anyAttribute namespace="##other" processContents="lax" />

    </xs:complexType>
```

7.5.4 RequestSecurityTokenResponseType

The wst:RequestSecurityTokenResponseType is a complex type (defined in WS - Trust 1.3) that contains the elements that are part of a server response to a wst:RequestSecurityToken message.

```
<xs:complexType name="RequestSecurityTokenResponseType">
      - <xs:annotation>
            <xs:documentation>
Actual content model is non-deterministic, hence wildcard. The following shows
intended content model:
                  <xs:element ref=‘:TokenType’ minOccurs=‘0’ />
                  <xs:element ref=‘:RequestType’ />
                  <xs:element ref=‘:RequestedSecurityToken’ minOccurs=‘0’ />
                  <xs:element ref=‘wsp:AppliesTo’ minOccurs=‘0’ />
                  <xs:element ref=‘:RequestedAttachedReference’ minOccurs
                  =‘0’ />
                  <xs:element ref=‘:RequestedUnattachedReference’ minOccurs
                  =‘0’ />
                  <xs:element ref=‘:RequestedProofToken’ minOccurs=‘0’ />
                  <xs:element ref=‘:Entropy’ minOccurs=‘0’ />
                  <xs:element ref=‘:Lifetime’ minOccurs=‘0’ />
                  <xs:element ref=‘:Status’ minOccurs=‘0’ />
                  <xs:element ref=‘:AllowPostdating’ minOccurs=‘0’ />
                  <xs:element ref=‘:Renewing’ minOccurs=‘0’ />
                  <xs:element ref=‘:OnBehalfOf’ minOccurs=‘0’ />
                  <xs:element ref=‘:Issuer’ minOccurs=‘0’ />
                  <xs:element ref=‘:AuthenticationType’ minOccurs=‘0’ />
                  <xs:element ref=‘:Authenticator’ minOccurs=‘0’ />
                  <xs:element ref=‘:KeyType’ minOccurs=‘0’ />
                  <xs:element ref=‘:KeySize’ minOccurs=‘0’ />
                  <xs:element ref=‘:SignatureAlgorithm’ minOccurs=‘0’ />
                  <xs:element ref=‘:Encryption’ minOccurs=‘0’ />
                  <xs:element ref=‘:EncryptionAlgorithm’ minOccurs=‘0’ />
                  <xs:element ref=‘:CanonicalizationAlgorithm’ minOccurs =‘0’ />
```

```
                <xs:element ref=‘:ProofEncryption’ minOccurs=‘0’ />
                <xs:element ref=‘:UseKey’ minOccurs=‘0’ />
                <xs:element ref=‘:SignWith’ minOccurs=‘0’ />
                <xs:element ref=‘:EncryptWith’ minOccurs=‘0’ />
                <xs:element ref=‘:DelegateTo’ minOccurs=‘0’ />
                <xs:element ref=‘:Forwardable’ minOccurs=‘0’ />
                <xs:element ref=‘:Delegatable’ minOccurs=‘0’ />
                <xs:element ref=‘wsp:Policy’ minOccurs=‘0’ />
                <xs:element ref=‘wsp:PolicyReference’ minOccurs=‘0’ />
                <xs:any namespace=‘##other’ processContents=‘lax’ minOccurs
                =‘0’ maxOccurs=‘unbounded’ />
            </xs:documentation>
        </xs:annotation>
        - <xs:sequence>
            <xs:any namespace="##any" processContents="lax" minOccurs="0"
            maxOccurs="unbounded" />
        </xs:sequence>
        <xs:attribute name="Context" type="xs:anyURI" use="optional" />
        <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:complexType>
```

Limitations of Certificate Attributes

The following are the Certificate Attribute field names and Length Limitations of the Certificate Attribute fields:

Table B- 3 Names and Length Limitations of Certificate Attributes

Certificate Attribute Name	Maximum Length Supported
ub-name	32768
ub-common-name	64
ub-locality-name	128
ub-state-name	128
ub-organization-name	64
ub-organization-unit-name	64
ub-title	64
ub-serial-number	64
ub-match	128
ub-emailaddress-length	255
ub-common-name-length	64
ub-country-name-alpha-length	2
ub-country-name-numeric-length	3
ub-domain-defined-attributes	4
ub-domain-defined-attribute-type-length	8
ub-domain-defined-attribute-value-length	128
ub-domain-name-length	16
ub-extension-attributes	256
ub-e163-4-number-length	15

Certificate Attribute Name	Maximum Length Supported
ub-e163-4-sub-address-length	40
ub-generation-qualifier-length	3
ub-given-name-length	16
ub-initials-length	5
ub-integer-options	256
ub-numeric-user-id-length	32
ub-organization-name-length	64
ub-organization-unit-name-length	32
ub-organizational-units	4
ub-pds-name-length	16
ub-pds-parameter-length	30
ub-pds-physical-address-lines	6
ub-postal-code-length	16
ub-pseudonym	128
ub-surname-length	40
ub-terminal-id-length	24
ub-unformatted-address-length	180
ub-x121-address-length	16

NOTE: By default, the ub-common-name field is constructed as mail_firstName<SPACE>mail_lastName. The total length of mail_firstName plus mail_lastName should be 63 or less. The maximum length for ub-common-name is 64.

APPENDIX C

8 Certificate Management Protocol Definition

This appendix includes the following topics:

8.1 About the Certificate Management Protocol

This section describes the protocol that allows an administrator to perform the administrative tasks for end-user certificates using the RA application.

8.2 Certificate Management Protocol Overview

The Certificate Management Protocol supports the following administrative operations that allow the RA application to perform the following:

- Search. This operation allows the RA application to search for certificates based on specified criteria. The response includes the resulting certificates in base64-encoded x.509 format. If the search finds more than 50 certificates, only the first 50 are returned.

  See "Creating Intelligent Searches" on page 60.

- Recover. (Available only if the profile supports key escrow). The operation allows an administrator to restore to the user a certificate that has been reported lost, stolen, or damaged.
- Suspend/Resume. This operation allows the administrator to suspend a valid certificate. You can also resume a suspended certificate.
- Revoke. This operation allows an administrator to permanently invalidate an end user's certificates. Revoke is necessary if a certificate has been lost, stolen, or compromised. Revoked certificates appear on the next updated version of the CRL.

  A revoked certificate cannot be returned to a valid state again.

Table C-1 provides an overview of the Certificate Management Protocol WSDL and its prerequisites, and cross-references to the sections in this guide that contain more information and code samples for its operations and messages.

Table C- 1 Certificate Management Protocol WSDL operations and messages

Name	Description	See...
Operation		
updateCertificateStatus	Protocol that requests certificate status changes for the RA application	See “updateCertificateStatus” on page 163.
keyRecovery	Protocol that recovers the private key for an end user's certificate	See “keyRecovery” on page 163.
searchCertificate	Protocol that searches for and returns certificates based on specified criteria	See “searchCertificate” on page 163.
bulkUpdateCertificateStatus	Protocol that requests bulk certificate status changes for the RA application.	See “bulkUpdateCertificateStatus” on page 164.
Messages		
RequestKeyRecoveryMessage	RequestKeyRecovery uses it to request a certificate recovery	See “RequestKeyRecoveryMessage” on page 164.
RequestKeyRecoveryResponse Message	The RA application receives a response to the RequestKeyRecoveryMessage call	See “RequestKeyRecoveryResponse Message” on page 165.
updateCertificateStatusRequest	updateCertificateStatus uses it to request a certificate status change	See “updateCertificateStatus Request” on page 167.

Name	Description	See...
updateCertificateStatusResponse	The RA application receives a response to the updateCertificateStatus call	See "updateCertificateStatusResponse" on page 168.
searchCertificateRequest	searchCertificate uses it to request certificates based on specific search criteria	See "searchCertificateRequest" on page 172.
searchCertificateResponse	The RA application receives a response to the searchCertificateRequest call	See "searchCertificateResponse" on page 173.
bulkUpdateCertificateStatus Request	UpdateCertificateStatus uses it to request a bulk certificate status change.	See "bulkUpdateCertificateStatusRequest" on page 169.
bulkUpdateCertificateStatus Response	The RA application receives the response message to the bulkUpdateCertificateStatus call.	See "bulkUpdateCertificateStatusResponse" on page 171.

Table C-2 provides an overview of the Certificate Management Protocol schema and its prerequisites, and cross-references to the sections in this guide that contain more information and code samples for its elements.

Table C- 2 Certificate Management Protocol schema elements

Name	Description	See...
OperationTypeEnum	Indicates all of the supported certificate status change operations	See "OperationTypeEnum" on page 174.
RevokeReasonCodeEnum	The RA application specifies the reason code when revoking a certificate	See "RevokeReasonCodeEnum" on page 174.
CertificateSearchResultType	Indicates all of the supported search criteria	See "CertificateSearchResultType" on page 175.
CertificateListType	Provides the returned certificates in base64-encoded x.509 format	See "CertificateListType" on page 176.
CertificateStatusEnum	Indicates the status of the certificates returned in the response	See "CertificateStatusEnum" on page 177.

8.3 Operations

The following administration operations are defined for the DigiCert PKI interface:

8.3.1 updateCertificateStatus

This operation is invoked to update the certificate status. The operation can request a change of status to revoked or suspended.

```
<wsdl:portType name="certificateManagementOperations">
      <wsdl:operation name="updateCertificateStatus">
            <wsdl:input message="tns:updateCertificateStatusRequest" />
            <wsdl:output message="tns:updateCertificateStatusResponse" />
      </wsdl:operation>
</wsdl:portType>
```

8.3.2 keyRecovery

This operation is invoked to recover the private key for an end user's certificate. The operation returns the private key as a password-protected PKCS#12 message, along with the corresponding password.

```
<wsdl:portType name="certificateManagementOperations">
      <wsdl:operation name="keyRecovery">
            <wsdl:input message="tns:requestKeyRecoveryMessage" />
            <wsdl:output message="tns:requestKeyRecoveryResponseMessage" />
      </wsdl:operation>
</wsdl:portType>
```

8.3.3 searchCertificate

This operation is invoked to search for and return a specified certificate. The operation returns the certificate as a PKCS#12 message.

```
<wsdl:portType name="certificateManagementOperations">
      <wsdl:operation name="searchCertificate">
            <wsdl:input message="tns:searchCertificateRequest" />
            <wsdl:output message="tns:searchCertificateResponse" />
      </wsdl:operation>
</wsdl:portType>
```

8.3.4 bulkUpdateCertificateStatus

This operation is invoked to update a bulk certificate status. The operation can request a change of status to revoked.

```
<wsdl:portType name="certificateManagementOperations">
      <wsdl:operation name="bulkUpdateCertificateStatus">
            <wsdl:input message="tns:bulkUpdateCertificateStatusRequest" />
            <wsdl:output message="tns:bulkUpdateCertificateStatusResponse" />
      </wsdl:operation>
</wsdl:portType>
```

8.4 Messages

The following WSDL message definitions are specific to this operation.

8.4.1 RequestKeyRecoveryMessage

This method is used by the RA application to request key recovery. The request must include the certificate serial number. The response contains the PKCS#12 file and the PKCS#12 password.

```
<xs:element name="requestKeyRecoveryMessage" type="vsmgmt:
RequestKeyRecoveryMessageType"/>
      <xs:complexType name="RequestKeyRecoveryMessageType">
            <xs:sequence>
                  <xs:element name="clientTransactionID"
                  type="vsmgmt:TransactionID Type" minOccurs="0" />
                  <xs:element name="pKCS12Password" type="xs:string"minOccurs
                  ="0" />
                  <xs:element name="certificateSerialNumber" type="xs:string" />
                  <xs:element name="certificateIssuer" type="xs:string" />
                  <xs:element name="adminID" type="xs:string" />
                  <xs:element name="version" type="vsmgmt:VersionType" />
                  <xs:any namespace="##any" processContents="lax" minOccurs="0"
                  maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>
```

- version is the schema version. The DigiCert PKI server communicates the schema version in its response to the RA application. Consequently, the RA application knows whether it can consume it

NOTE: For this release, this should be 1.0.

- clientTransactionID - Use this element to specify a transaction ID for the enterprise RA application (optional). The element may be useful, if your RA application tracks the request and response.
- pKCS12Password is the password definition for the PKCS12 message as defined in Certificate Enrollment Protocol.

NOTE: This field is not supported in this release.

- certificateSerialNumber is the certificate serial number for which the key recovery operation is to be performed. The same certificate and associated private key is recovered and sent back in PKCS#12 format.
- certificateIssuer is the name of the issuing certification authority. Obtain this value from the certificate's Issuer Distinguished Name.

NOTE: Unless the CA has been rekeyed, this value matches the Issuer Distinguished Name in the certificate policy that is returned in the getPolicyResponse. If the CA is rekeyed, the Issuer Distinguished Name format must match the format in the certificate policy that is returned in the getPolicyResponse.

- adminID is the identifier for the RA certificate or account administrator who can approve the key recovery request.
- ##any - This element is reserved for future expansion.

8.4.2 RequestKeyRecoveryResponseMessage

The RA application receives this response from the RequestKeyRecoveryMessage request. The response contains the PKCS#12 file and the PKCS#12 password.

```
<xs:element name="requestKeyRecoveryResponseMessage">

      <xs:annotation>

            <xs:documentation>Comment describing your root element</xs:
            documentation>

      </xs:annotation>

      <xs:complexType>

            <xs:sequence>

                  <xs:element name="clientTransactionID" type="vsmgmt:
```

```
                TransactionID

                Type" minOccurs="0" />

                <xs:element name="serverTransactionID" type="vsmgmt:
                TransactionID

                Type" />

                <xs:element name="adminApprovalPendingCount" type="xs:int" min
                Occurs="0" />

                <xs:element name="pKCS12Password" type="xs:string" minOccurs

                ="0" />

                <xs:element name="pKCS12Message" type="xs:string" minOccurs

                ="0" maxOccurs="unbounded" />

                <xs:element name="version" type="vsmgmt:VersionType" />

                <xs:element name="any" type="xs:anyType" minOccurs="0"
                maxOccurs="unbounded" />

            </xs:sequence>

        </xs:complexType>

</xs:element>
```

- clientTransactionID - Contains the transaction ID sent in the original request (optional). The ID may be useful if your RA application tracks the request and response.
- serverTransactionID is the transaction ID created by the DigiCert PKI server The DigiCert PKI server can use this ID to find the transaction in the server log file for troubleshooting purpose.
- adminApprovalPendingCount is sent in case of dual control or multi-control accounts. The count indicates to the RA application the number of administrators that must approve this request before the key can be recovered. This field is not sent in the response, if the PKCS#12 message is sent.
- pKCS12Password is the password for the PKCS#12 file as defined in the enrollment protocol
- pKCS12Message - The message is the base64-encoded string for the certificate, along with the private key in PKCS#12 format.
- version is the schema version - The DigiCert PKI server communicates the schema version in its response to the RA application. The communication lets the RA application know whether it can consume the version.
- ##any - This element is reserved for future expansion.

8.4.3 updateCertificateStatusRequest

The request is the method used by the RA application to update the certificate status. The method can request a change of status to revoked to update the certificate's validity.

NOTE: certificateIssuer and certificateSerialNumber are the only valid parameters supported for Suspend or Resume operations.

```
<xs:element name="updateCertificateStatusRequest" type="vsmgmt:Update
CertificateStatusRequestType" />

<xs:complexType name="updateCertificateStatusRequestType">

      <xs:sequence>

            <xs:element name="clientTransactionID" type="vsmgmt:
            TransactionIDType" minOccurs="0" />

            <xs:element name="version" type="vsmgmt:VersionType" />

            <xs:element name="certificateIssuer" type="xs:string" />

            <xs:element name="revocationReason" type="vsmgmt:RevokeReason
            CodeEnum" minOccurs="0" />

            <xs:element name="challenge" type="xs:string" minOccurs="0" />

            <xs:element name="comment" type="vsmgmt:CommentType" minOccurs

            ="0" />

            <xs:element name="certificateSerialNumber" type="xs:string" />

            <xs:element name="seatID" type="xs:string" />

            <xs:element name="operationType" type="vsmgmt:Operation TypeEnum" />

            <xs:any namespace="any" processContents="lax" minOccurs="0" max
            Occurs="unbounded" />

      </xs:sequence>

</xs:complexType>
```

- clientTransactionID - Use this element to specify a transaction ID for the enterprise RA application (optional). If your RA application tracks requests and responses, use the element.
- version is the schema version. The DigiCert PKI server communicates the schema version in its response to the RA application. The version lets the RA application know whether it can consume it.

  NOTE: For this release, this should be 1.0.

- certificateIssuer is the name of the issuing Certification Authority. This element has been deprecated.

NOTE: Unless the CA has been rekeyed, this value matches the Issuer Distinguished Name in the certificate policy that is returned in the getPolicyResponse. If the CA has been rekeyed, the format for the Issuer Distinguished Name must match the format in the certificate policy that is returned in the getPolicyResponse.

- revocationReason is the optional revocation reason code. It appears in the DigiCert database and CRL status. If not set, a reason code of Superseded is used. See "RevokeReasonCodeEnum".
- challenge is used to contain a challenge phrase if the enterprise chooses to send the user a challenge phrase for the revocation operation. The field is optional.

NOTE: This field is not supported in this release.

- comment allows the RA application to associate a comment with the revocation of a certificate. The field is useful for audit purposes. The field can contain up to 512 ASCII characters.
- certificateSerialNumber is the certificate serial number for which the certificate status is to be updated. If this element is used, do not use seatID.
- seatID is the unique identifier for the end user (seat ID) whose certificates are to be updated. If this element is used, do not use certificateSerialNumber.
- ##any - This element is reserved for future expansion.

8.4.4 updateCertificateStatusResponse

The RA application receives this response message from the updateCertificateStatusRequest request.

```
<xs:element name="updateCertificateStatusResponse" type="vsmgmt:
UpdateCertificateStatusResponseType" />

<xs:complexType name="updateCertificateStatusResponseType">

      <xs:sequence>

            <xs:element name="clientTransactionID" type="vsmgmt:
            TransactionIDType" minOccurs="0" />

            <xs:element name="serverTransactionID" type="vsmgmt:
            TransactionIDType" />

            <xs:element name="version" type="vsmgmt:VersionType" />

            <xs:element name="successCode" type="xs:int" />

            <xs:element name="successMsg" type="xs:string" />

            <xs:element name="revocationCount" type="xs:int" />
```

```
        <xs:any namespace="##any" processContents="lax" minOccurs="0"
        maxOccurs="unbounded" />

    </xs:sequence>

</xs:complexType>
```

- clientTransactionID - Use this element to specify a transaction ID for the enterprise RA application (optional). The element is useful if your RA application tracks the request and response.
- serverTransactionID is the transaction ID created by the DigiCert PKI server. The DigiCert PKI server can use this ID to find the transaction in the server log file for troubleshooting purpose.
- version is the schema version.
- successCode is an integer value indicating the success status. The value "0" indicates that the operation was successful.
- successMsg is a message indicating the success status. The value "Success" indicates that the operation was successful.
- revocationCount is the number of certificates whose status was updated to revoked.
- The count is applicable for suspended or resumed certificates.
- ##any - This element is reserved for future expansion.

8.4.5 bulkUpdateCertificateStatusRequest

The following is a sample request for a bulkUpdateCertificateStatusRequest used by the RA application to update the bulk certificate status. You can request a change of status to revoke to update certificate validity.

```
<xs:element name="bulkUpdateCertificateStatusRequest" type="vsmgmt:
BulkUpdateCertificateStatusRequestType" />

<xs:complexType name="BulkUpdateCertificateStatusRequestType">

    <xs:sequence>

        <xs:element name="clientTransactionID"
        type="vsmgmt:TransactionIDType" minOccurs="0" />

        <xs:element name="version" type="vsmgmt:VersionType" />

        <xs:element name="revocationReason"
        type="vsmgmt:RevokeReasonCodeEnum" minOccurs="0" />

        <xs:element name="comment" type="vsmgmt:CommentType"

        minOccurs="0" />

    <xs:choice>

        <xs:element name="certificateSerialNumber" type="xs:string"

        maxOccurs="100" />
```

```
            <xs:element name="seatId" type="xs:string" maxOccurs="100" />

            <xs:element name="profileOID" type="xs:string" maxOccurs="100" />

        </xs:choice>

            <xs:element name="operationType" type="vsmgmt:OperationTypeEnum" />

            <xs:any namespace="any" processContents="lax" minOccurs="0"
            maxOccurs="unbounded" />

        </xs:sequence>

</xs:complexType>
```

- clientTransactionID - Use this element to specify a transaction ID for the enterprise RA application (optional). This may be useful if your RA application tracks the request and response.
- version is the schema version. DigiCert PKI server communicates the schema version in its response to the RA application so that the RA application knows whether it can consume it.

NOTE: For this release, this should be 1.0.

- revocationReason is the optional revocation reason code. It appears in the DigiCert database, CRL, and OCSP status and CRL status. If not set, a reason code of Superseded is used. See "RevokeReasonCodeEnum" .
- comment allows the RA application to associate a comment with the revocation of a certificate. Comments are useful for audit purposes.
- certificateSerialNumber is the certificate serial number for which the certificate status is to be updated. If this element is used, do not use seatID. You can enter up to 100 certificateSerialNumber elements to revoke multiple certificates in one request.
- seatID is the unique identifier for the end user (seat ID) whose certificates are to be updated. If this element is used, do not use certificateSerialNumber. You can enter up to 100 seatID elements to revoke multiple certificates in one request.

NOTE: ProfileOID is not supported for bulk revocation.

- ##any - This element is reserved for future expansion.

8.4.6 bulkUpdateCertificateStatusResponse

The RA application receives the response message from a bulkUpdateCertificateStatusRequest.

```
<xs:element name="bulkUpdateCertificateStatusResponse"
type="vsmgmt:BulkUpdateCertificateStatusResponseType" />

<xs:complexType name="BulkUpdateCertificateStatusResponseType">

      <xs:sequence>

            <xs:element name="clientTransactionID"
            type="vsmgmt:TransactionIDType" minOccurs="0" />

            <xs:element name="serverTransactionID"
            type="vsmgmt:TransactionIDType" minOccurs="0" />

            <xs:element name="version" type="vsmgmt:VersionType" />

            <xs:element name="successCode" type="xs:int" />

            <xs:element name="successMsg" type="xs:string" />

            <xs:element name="revocationCount" type="xs:int" />

            <xs:any namespace="##any" processContents="lax" minOccurs="0"
            maxOccurs="unbounded" />

      </xs:sequence>

</xs:complexType>
```

- clientTransactionID - Use this element to specify a transaction ID for the enterprise RA application (optional). If your RA application tracks requests and responses, IDs are useful.
- serverTransactionID is the transaction ID created by the DigiCert PKI server. The DigiCert PKI server can use this ID to find the transaction in the server log file for troubleshooting purpose.
- version is the schema version.
- successCode is an integer value indicating the success status. The value "0" indicates that the operation was successful. All other values are reserved for future expansion.
- successMsg is a message indicating the success status. The value "Success" indicates that the operation was successful. All other values are reserved for future expansion.
- revocationCount is the number of certificates whose status was updated to revoked.
- ##any - This element is reserved for future expansion.

8.4.7 searchCertificateRequest

The RA application uses the request method to search for certificates by specified criteria. The method returns up to 50 matching certificates.

```
<xs:element name="searchCertificateRequest" type="vsmgmt:Search
CertificateRequestType" />

<xs:complexType name="SearchCertificateRequestType">

      <xs:sequence>

            <xs:element name="clientTransactionID" type="vsmgmt:Transaction
            IDType" minOccurs="0" />

            <xs:element name="seatId" type="xs:string" minOccurs="0" />

            <xs:element name="accountId" type="xs:string" minOccurs="0" />

            <xs:element name="profileOID" type="xs:string"  minOccurs="0" />

            <xs:element name="commonName" type="xs:string" minOccurs="0" />

            <xs:element name="status" type="vsmgmt:CertificateStatusEnum"
            minOccurs="0" />

            <xs:element name="emailAddress" type="xs:string" minOccurs= "0" />

            <xs:element name="serialNumber" type="xs:string" minOccurs= "0" />

            <xs:element name="issuingCA" type="xs:base64Binary" minOccurs

            ="0" />

            <xs:element name="validFrom" type="xs:long" minOccurs="0" />

            <xs:element name="validTo" type="xs:long" minOccurs="0" />
            xs:element name="version" type="vsmgmt:VersionType" /> xs:element
            name="startIndex" type="xs-int" />

            <xs:any namespace="##any" processContents="lax" minOccurs="0"
            maxOccurs="unbounded" />

      </xs:sequence>

</xs:complexType>
```

- clientTransactionID - transaction ID for the enterprise RA application (optional). IDs are useful if your RA application tracks requests and responses.
- seatId - unique identifier for the user that owns the certificate. The same identifier is configured as the User identifier in PKI Manager.
- accountId - the account or sub-account to search for the certificates.
- profileOID - the OID of the certificate profile under which the certificate was issued.
- commonName - the common name of the certificate.
- status - the status (valid, pending, expired, revoked, or suspended) of the certificate.
- emailAddress - the email address of the user that owns the certificate.

- serialNumber- the certificate serial number.
- issuingCA - the CA that issued the certificate.
- validFrom - certificate issue date.
- validTo - certificate expiration date.
- version - the schema version
- startIndex - the index from where you want the certificate results returned. By default, the index starts from 0.
- ##any - This element is reserved for future expansion.

NOTE: searchCertificateRequest does not return a certificate that is in the Deleted status.

8.4.8 searchCertificateResponse

The RA application receives the response message from the searchCertificateStatusRequest request.

```
<xs:element name="searchCertificateResponse" type="vsmgmt:Search
CertificateResponseType" />

xs:complexType name="SearchCertificateResponseType">

      <xs:sequence>

            <xs:element name="clientTransactionID" type="vsmgmt:
            TransactionIDType" minOccurs="0" />

            <xs:element name="serverTransactionID" type="vsmgmt:
            TransactionIDType" />

            <xs:element name="certificateCount" type="xs:int" />

            <xs:element name="certificateList" type="vsmgmt:Certificate
            ListType" minOccurs="0" />

            <xs:element name="moreCertificateAvailable" type="xs:boolean"
            minOccurs="0" />

            <xs:element name="version" type="vsmgmt:VersionType" />

            <xs:any namespace="##any" processContents="lax" minOccurs="0"
            maxOccurs="unbounded" />

      </xs:sequence>

/xs:complexType>
```

- clientTransactionID - a transaction ID for the enterprise RA application (optional). IDs may be useful if your RA application tracks the request and response.
- serverTransactionID - the transaction ID created by the DigiCert PKI server. The DigiCert PKI server can use this ID to find the transaction in the server log file for troubleshooting purpose.

- certificateCount - the number of certificates that are returned in the response.
- certificateList - contains the list of certificates returned.
- moreCertificateAvailable - identifies if more certificates are available than the configured startIndex and the configured amount.
- version - the schema version.
- ##any - This element is reserved for future expansion.

8.5 Elements

The following describe the WSDL elements for the Certificate Management Protocol.

8.5.1 Elements for requestKeyRecovery and updateCertificateStatus Operations

The following WSDL element definitions are specific to the requestKeyRecovery and updateCertificateStatus operations.

OperationTypeEnum

This element is of type enumeration. This element indicates all of the operations that are supported when updating the certificate status. The element is part of updateCertificateStatusRequest.

```
<xs:simpleType name="OperationTypeEnum">
      <xs:restriction base="xs:string">
            <xs:enumeration value="Revoke" />
            <xs:enumeration value="Suspend" />
            <xs:enumeration value="Resume" />
      </xs:restriction>
</xs:simpleType>
```

RevokeReasonCodeEnum

These are the various reason codes that the RA application can specify when revoking a certificate. The field is optional. If the code is not specified, DigiCert PKI uses Superseded. This element is part of updateCertificateStatusRequest.

Important Note:

- “Unspecified”, “PrivilegeWithdrawn “and “AACompromise” reason codes are no longer supported and if used, will be automatically replaced with “Superseded”
- “CACompromise” reason code is no longer supported and if used, will be automatically replaced with “CessationOfOperation”.

```
<xs:simpleType name="RevokeReasonCodeEnum">
      <xs:restriction base="xs:string">
            <xs:enumeration value="KeyCompromise" />
            <xs:enumeration value="AffiliationChanged" />
            <xs:enumeration value="CessationOfOperation" />
            <xs:enumeration value="Superseded" />
      </xs:restriction>
</xs:simpleType>
```

8.5.2 Elements for searchCertificate Operations

The following WSDL element definitions are specific to the searchCertificate operation.

CertificateSearchResultType

This element indicates all of the search criteria that are supported when in search of a certificate. This element is part of searchCertificateResponse.

```
<xs:complexType name="CertificateSearchResultType">
      <xs:sequence>
            <xs:element name="certificate" type="xs:base64Binary" />
            <xs:element name="seatId" type="xs:string" />
            <xs:element name="commonName" type="xs:string" />
            <xs:element name="accountId" type="xs:string" />
            <xs:element name="profileOID" type="xs:string" />
            <xs:element name="emailAddress" type="xs:string" nillable= "true"/>
            <xs:element name="status" type="vsmgmt:CertificateStatusEnum" />
            <xs:element name="revokeAt" type="xs:long" minOccurs="0" />
            <xs:element name="revokeReason" type="vsmgmt:RevokeReasonCode Enum"
            minOccurs="0" />
            <xs:element name="validFrom" type="xs:long" />
            <xs:element name="validTo" type="xs:long" />
            <xs:element name="serialNumber" type="xs:string" />
            <xs:element name="isEscrowed" type="xs:boolean" />
            <xs:any namespace="##any" processContents="lax" minOccurs="0"
            maxOccurs="unbounded" />
      </xs:sequence>
</xs:complexType>
```

- certificate - This element contains the base4-encoded x.509 certificates returned. Each certificate is contained in a separate certificateInformation element.
- seatId - a unique identifier for the user that owns the certificate. The ID must be the same attribute that is configured as the seat ID in PKI Manager.
- commonName - the common name of the certificate.
- accountId - the unique identifier for the account or sub-account.
- profileOID - the OID of the certificate profile.
- emailAddress - the email address of the user that owns the certificate.
- status - the status (valid, pending, expired, revoked, or suspended) of the certificate.
- revokedAt - if the certificate is revoked, This is the date that the certificate was revoked.
- revokeReason - the optional revocation reason code.
- validFrom - certificate issue date.
- validTo - certificate expiration date.
- serialNumber - the certificate serial number.
- enrollmentNotes - any notes added to the certificate during enrollment. Notes can be up to 512 ASCII characters in length. Include these notes by adding a notes=<value> name/value pair in the input.txt file when enrolling for certificates.
- revokeComments - any comments added to a certificate during revocation. Comments can be up to 512 ASCII characters in length. Include these comments by adding a certmgmt.comment=<value> name/value pair in the certmgmt.txt.revoke file during revocation.
- isEscrowed - indicates whether the private key is stored (for key recovery).
- true - indicates that the private key is escrowed.
- false - indicates that the private key is not escrowed.
- ##any - this element is reserved for future expansion

CertificateListType

This element indicates all of the operations that are supported when a certificate status is updated. This element is part of searchCertificateResponse.

```
<xs:complexType name="CertificateListType">
      <xs:sequence>
            <xs:element name="certificateInformation" type="vsmgmt:
            CertificateSearchResultType" maxOccurs="unbounded" />
      </xs:sequence>
</xs:complexType>
```

- certificateInformation - This element contains the base4-encoded x.509 certificates returned. Each certificate is contained in a separate certificateInformation element.

CertificateStatusEnum

This element indicates the status of each certificate returned. This element is part of searchCertificateResponse.

```
<xs:simpleType name="CertificateStatusEnum">
      <xs:restriction base="xs:string">
            <xs:enumeration value="VALID" />
            <xs:enumeration value="EXPIRED" />
            <xs:enumeration value="REVOKED" />
            <xs:enumeration value="SUSPENDED" />
      </xs:restriction>
</xs:simpleType>
```

- enumeration value - The enumeration value specifies the status of the certificates that are returned in the searchCertificateResponse. The value can be:
  - VALID
  - EXPIRED
  - REVOKED
  - SUSPENDED

  Suspend is supported for CAs who have enabled this feature.

APPENDIX D

9 User Management Protocol Definition

This appendix includes the following topics:

9.1 About the User Management Protocol

This section describes the protocol that allows an administrator to perform tasks for end-user management (add or modify users, and generate or replace enrollment codes).

9.2 User Management Protocol Overview

The User Management Protocol supports four administrative operations:

- Add or modify users. This operation lets you create new users and upload data for the users. If the users already exist, this operation modifies the user data.
- Obtain user information. This operation returns the user information previously uploaded to the DigiCert PKI account.
- Generate or replace enrollment codes. This operation lets you generate and replace enrollment codes for users, and update the enrollment code status (bad attempts counter, expiration date).
- Obtain enrollment code information. This operation returns information about an individual enrollment code that is assigned to a user.

Table D-1 provides an overview of the User Management Protocol WSDL and its prerequisites, and cross-references to the sections in this guide that contain more information and code samples for its operations and messages.

Table D- 1 User Management Protocol WSDL operations and messages

Name	Description	See...
Operation		
createOrUpdateUser	Protocol that requests user creation or modification and uploads user data	See "Operations" on page 181.
getUserInformation	Protocol that requests and returns user data and valid user certificates	
createOrUpdatePasscode	Protocol that requests enrollment code generation or replacement	
getPasscodeInformation	Protocol that requests and returns enrollment code information	
Messages		
createOrUpdateUserRequest	createOrUpdateUser message to request user creation or modification	See "createOrUpdateUserRequest" on page 181.
createOrUpdateUserResponse	RA application receives response to the createOrUpdateUserRequest call	See "createOrUpdateUserResponse" on page 183.
getUserInformationRequest	getUserInformation uses it to request user information and valid user certificates	See "getUserInformationRequest" on page 184.
getUserInformationResponse	RA application receives response to the	See "getUserInformationResponse" on page 185.

Name	Description	See...
	getUserInformationRequest call	
createOrUpdatePasscodeRequest	createOrUpdatePasscode uses it to request enrollment code generation or replacement	See "createOrUpdatePasscodeRequest" on page 187.
createOrUpdate PasscodeResponse	RA application receives response to the createOrUpdatePasscode Request call	See "createOrUpdatePasscodeResponse" on page 190.
getPasscodeInformationRequest	getPasscodeInformation uses it to request enrollment code information	See "getPasscodeInformationRequest" on page 192.
getPasscode InformationResponse	RA application receives response to the getPasscodeInformationRequest call	See "getPasscodeInformationResponse" on page 193.
deleteUserRequest	deleteUser uses it to delete a user and revoke any certificates issued to the user.	See "Single user delete request" on page 194.
deleteUserResponse	RA application receives response to the deleteUserRequest call	See "Single user delete response" on page 195.
bulkDeleteUserRequest	bulkDeleteUser uses it to delete multiple users and revoke any certificates issued to them.	See "Bulk user delete request" on page 196.
bulkDeleteUserResponse	RA application receives response to the bulkUserDeleteRequest call	See "Bulk user delete response" on page 196.

9.3 Operations

The following administration operations are defined for the DigiCert PKI interface:

- createOrUpdateUser - This operation is invoked to create or update users and upload user data to the system.
- CreateOrUpdatePasscode - This operation is invoked to generate or replace enrollment codes for users.
- getUserInformation - This operation is invoked to obtain data about a user previously uploaded to DigiCert. This operation can also obtain the valid certificates that are issued to a user.
- getPasscodeInformation - This operation is invoked to obtain information about an individual enrollment code that is assigned to a user.

9.4 Messages

The following WSDL message definitions are specific to this operation.

9.4.1 createOrUpdateUserRequest

The request is used to create or modify users and upload new or modified user data. The request must include a unique seat ID. It must match the **User identifier** in the **Customize user identification** section of the **Manage this profile** page in PKI Manager.

```
<?xml version="1.0" encoding="UTF-8"?>

<ns12:createOrUpdateUserRequest xmlns:ns12="http://schemas.
verisign.com/pkiservices/2011/08/usermanagement">

      <ns12:clientTransactionID>1316083481585</ns12:client TransactionID>

      <ns12:userInformation>

            <ns12:seatId>a2@test.com</ns12:seatId>

            <ns12:firstName>seat</ns12:firstName>

            <ns12:lastName>1</ns12:lastName>

            <ns12:emailAddress>seat1@test.com</ns12:emailAddress>

            <ns12:deskPhoneNumber>666-666-6666</ns12:deskPhoneNumber>

            <ns12:mobilePhoneNumber>555-555-5555</ns12:mobilePhone Number>

            <ns12:userAttribute>

                  <ns12:name>organization</ns12:name>

                  <ns12:value>test</ns12:value>

            </ns12:userAttribute>
            <ns12:userAttribute>
                  <ns12:name>common_name</ns12:name>
                  <ns12:value>seat1</ns12:value>
            </ns12:userAttribute>
            <ns12:userAttribute>
                  <ns12:name>organizational_unit</ns12:name>
                  <ns12:value>engineering#123/%</ns12:value>
            </ns12:userAttribute>
      </ns12:userInformation>
      <ns12:userInformation>
            <ns12:seatId>a3@test.com</ns12:seatId>
            <ns12:firstName>seat</ns12:firstName>
            <ns12:lastName>2</ns12:lastName>
            <ns12:emailAddress>seat1@test.com</ns12:emailAddress>
            <ns12:deskPhoneNumber>666-666-6666</ns12:deskPhoneNumber>
            <ns12:mobilePhoneNumber>555-555-5555</ns12:mobilePhone Number>
            <ns12:userAttribute>
                  <ns12:name>organization</ns12:name>
                  <ns12:value>test</ns12:value>
            </ns12:userAttribute>
            <ns12:userAttribute>
                  <ns12:name>common_name</ns12:name>
                  <ns12:value>seat1</ns12:value>
            </ns12:userAttribute>
            <ns12:userAttribute>
                  <ns12:name>organizational_unit</ns12:name>
                  <ns12:value>engineering#123/%</ns12:value>
            </ns12:userAttribute>
      </ns12:userInformation>
      <ns12:version>1.0</ns12:version>
</ns12:createOrUpdateUserRequest>
```

- clientTransactionID - Use this element to specify a transaction ID for the enterprise RA application (optional). The IDs are useful if your RA application tracks the request and response.

- UserInformation - This element contains the information about the user. This element appears in the request once for every user being created or modified. The following elements are included in UserInformationType. Only seatId is required.

  To create or modify multiple users in one operation, include a UserInformation element for each user to be created or modified.

  - seatId - a unique identifier for the user. The element can be a custom element or another UserInformation element, if the same attribute is configured as the User identifier in PKI Manager.
  - firstName - First name of the user.
  - lastName - Last name (or surname) of the user.
  - emailAddress - Email address for the user.
  - deskPhoneNumber - Landline-based telephone number for the user.
  - mobilePhoneNumber - Mobile telephone number for the user.
  - userAttribute - Use the element to add the custom user attributes in a certificate. Use the format List of Set<Name<String>,Value<String>>.

- version is the schema version. The DigiCert PKI server communicates the schema version in its responses to the RA application. Consequently, the RA application knows whether it can consume it.

NOTE: For this release, this field should be 1.0.

9.4.2 createOrUpdateUserResponse

The message is the response that the RA application receives from the createOrUpdateUserRequest request.

```
<?xml version="1.0" encoding="UTF-8"?>

<createOrUpdateUserResponse xmlns="http://schemas.verisign.com/
pkiservices/2011/08/usermanagement">

      <clientTransactionID>1316083481585</clientTransactionID>

      <serverTransactionID>722b5c5a7606dc6e</serverTransactionID>

      <userCreationStatus>

            <seatId>a2@test.com</seatId>

            <statusCode>0</statusCode>

      </userCreationStatus>

      <userCreationStatus>

            <seatId>a3@test.com</seatId>

            <statusCode>0</statusCode>
      </userCreationStatus>

      <version>1.0</version>

</createOrUpdateUserResponse>
```

- clientTransactionID - Contains the transaction ID sent in the original request (optional). The IDs are useful if your RA application tracks its requests and responses.
- serverTransactionID is the transaction ID created by the DigiCert PKI server. The DigiCert PKI server can use this ID to find the transaction in the log file for troubleshooting purposes.
- UserCreationStatus - Identifies whether the user was created or modified. This element appears in the response once for each user that is sent in the request. This element returns the following for each user:
  - seatId is the unique identifier for the user that is sent in the request.
  - successCode is an integer value indicating the success status. The value "0" indicates that the operation was successful.
- version is the schema version. The DigiCert PKI server communicates the schema version in response to the RA application. Consequently, the RA application knows whether it can consume it.

NOTE: For this release, this field should be 1.0.

9.4.3 getUserInformationRequest

The RA application uses the message to return information about the user.

```
<?xml version="1.0" encoding="UTF-8"?>

<ns12:getUserInformationRequest xmlns:ns12="http://schemas.
verisign.com/pkiservices/2011/08/usermanagement">

      <ns12:clientTransactionID>1316083628509</ns12:clientTransaction ID>

      <ns12:seatId>a1@test.com</ns12:seatId>

<ns12:getUserCertificate>true</ns12:getUserCertificate>

      <ns12:version>1.0</ns12:version>

</ns12:getUserInformationRequest>
```

- clientTransactionID - Use this element to specify a transaction ID for the enterprise RA application (optional). If your RA application tracks its requests and responses, the IDs are useful.
- seatId - a unique identifier for the user. The ID can be a custom element or another UserInformation element, if the same attribute is configured as the User identifier in PKI Manager.

- getUserCertificate - Use this element to obtain a valid certificate for the user, if one has been issued.
- version is the schema version. The DigiCert PKI server communicates the schema version in response to the RA application. Consequently, the RA application knows whether it can consume it.

NOTE: For this release, this field should be 1.0.

9.4.4 getUserInformationResponse

The RA application receives this message from the getUserInformationRequest request.

```
<?xml version="1.0" encoding="UTF-8"?>

<getUserInformationResponse xmlns="http://schemas.verisign.com/
pkiservices/2011/08/usermanagement">

      <clientTransactionID>1316083628509</clientTransactionID>

      <serverTransactionID>9f9812678c60f092</serverTransactionID>

      <userInformation>

            <seatId>a1@test.com</seatId>

            <firstName>seat</firstName>

            <lastName>1</lastName>

            <emailAddress>seat1@test.com</emailAddress>

            <deskPhoneNumber>666-666-6666</deskPhoneNumber>

            <mobilePhoneNumber>555-555-5555</mobilePhoneNumber>

            <userAttribute>

            <name>organization</name>

            <value>test</value>

            </userAttribute>

            <userAttribute>

                  <name>organizational_unit</name>

                  <value>engineering#123/%</value>

            </userAttribute>

            <userAttribute>

                  <name>common_name</name>

                  <value>seat1</value>

            </userAttribute>

            <userAttribute>
```

```
                <name>country</name>
                <value>US</value>
            </userAttribute>
        </userInformation>
        <userValidCertificates>
<userCertificate>MIIFkTCCBHmgAwIBAgIQfOZPxEdbt5Xnn/
+dXoy7bTANBgkqhkiG9w0BAQUFADCB+zELMAkGA1UEBhMCVVMxHTAbBgNVBAoTFF
N5bWFudGVjIENvcnBvcmF0aW9uMR8wHQYDVQQLExZGT1IgVEVTVCBQVVJQT1NFUy
BPTkxZMR8wHQYDVQQLExZWZXJpU2lnbiBUcnVzdCBOZXR3b3JrMUIwQAYDVQQLEz
lUZXJtcyBvZiB1c2UgYXQgaHR0cHM6Ly93d3cudmVyaXNpZ24uY29tL2Nwcy90ZX
N0Y2EgKGMpMTExRzBFBgNVBAMTPlN5bWFudGVjIEMzIEFkbWluIEludGVybWVkaW
F0ZSBURVNUIENlcnRpZmljYXRlIEF1dGhvcml0eSAtIFFBMB4XDTExMDUwOTAwMD
AwMFoXDTEyMDUwODIzNTk1OVowgZQxJjAkBgNVBAMMHVdTQXV0bzIwMTEwNTEwIF
dTQXV0bzIwMTEwNTEwMSkwJwYJKoZIhvcNAQkBFhp3c2F1dG8yMDExMDUxMEB5b3
BtYWlsLmNvbTEXMBUGA1UECgwOV1NBdXRvMjAxMTA1MTAxDjAMBgNVBAsMBUFETU
lOMRYwFAYDVQQLDA1NVUxUSS1BTExPV0VEMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ
8AMIIBCgKCAQEAve7B/kMQ3W9516wPh2x+n7Oa/kWlU3nRAwka3Hh0H9JytoI5rh
MpbCtyK654GcjTvyKmKT6dYaEJH/VbIdmYVJLJXgKg1I7V4atb6SaQoBsA+rIzkt
pKn/eS0jM+xkZSpTRpc0oqJ/UKgG+4WnMibofC2s2C2hAf1Co6m6f8Jycp9jzbPH
BPL8815ejbwIqOKfYNIbP/w5hHf3AENJ1HiDXDZaKVkPS5kTMWxXaK14G2sm5qpf
e8ZX3EVameVFS1BaSDyXkisPBY4LzHuklIAFiN7LEmQ9XLEfa3onFQV7v7QBwlFY
1tImvSyNHc5KN9z/K3Ofc4EnPP3z9WutXYewIDAQABo4IBdDCCAXAwCQYDVR0TBA
IwADAOBgNVHQ8BAf8EBAMCA+gwFgYDVR0lAQH/==</userCertificate>
        </userValidCertificates>
        <version>1.0</version>
</getUserInformationResponse>
```

- clientTransactionID - Transaction IDs are sent in the original requests (optional). The IDs may be useful if your RA application tracks its requests and responses.
- serverTransactionID is the transaction ID created by the DigiCert PKI server The DigiCert PKI server can use this ID to find the transaction in the log file for troubleshooting purposes.
- UserInformation - This element contains the information about the user. This element appears in the request once for every user being created or modified. The following elements are included in UserInformationType.
    - seatId - a unique identifier for the user.
    - firstName - First name of the user.
    - lastName - Last name (or surname) of the user.
    - emailAddress - Email address for the user.
    - deskPhoneNumber - Landline-based telephone number for the user.

  - mobilePhoneNumber - Mobile telephone number for the user.
  - userAttribute - The custom user attributes that appear in the certificate, in the format List of Set<Name<String>, Value<String>>.

- userValidCertificates - This element contains the certificate that is issued to the user.
- version is the schema version. The DigiCert PKI server communicates the schema version in its response to the RA application. Consequently, the RA application knows whether it can consume it.

NOTE: For this release, this field should be 1.0.

9.4.5 createOrUpdatePasscodeRequest

The request is a method used to create users and upload a new passcode for the user. The request must include a unique seat ID that matches the attribute that is set as the User identifier. The User identifier is in the Customize user identification section of the Manage this profile page in PKI Manager.

```
<?xml version="1.0" encoding="UTF-8"?>

<ns14:createOrUpdatePasscodeRequest xmlns:ns14="http://schemas.
verisign.com/pkiservices/2011/08/usermanagement">

            <ns14:clientTransactionID>1433918052018</ns14:clientTransactionID>

            <ns14:passcodeInformation>

            <ns14:passcode>passcode!</ns14:passcode>

            <ns14:expiryDateTime>2015-06-15T00:00:00.000+05:30</ns14:

            expiryDateTime>

            <ns14:seatId>test_user_passcode_001@pookmail.com</ns14: seatId>

            <ns14:firstName>Test User 001</ns14:firstName>

            <ns14:lastName>Passcode 001</ns14:lastName>

            <ns14:email>test_user_passcode_001@pookmail.com</ns14:email>

            <ns14:certificateProfileOid>2.16.840.1.113733.1.16.1.5.3.2.1.
            41747424</ns14:certificateProfileOid>

            <ns14:userAttribute>

                  <ns14:name>uniformResourceIdentifier</ns14:name>

                  <ns14:value>test</ns14:value>

            </ns14:userAttribute>

            <ns14:userAttribute>

                  <ns14:name>custom_encode_dnsName</ns14:name>

                  <ns14:value>seat1</ns14:value>
            </ns14:userAttribute>
        </ns14:passcodeInformation>
        <ns14:passcodeInformation>
<ns14:passcode>passcode!</ns14:passcode>
            <ns14:expiryDateTime>2015-06-15T00:00:00.000+05:30</ns14:
            expiryDateTime>
            <ns14:seatId>test_user_passcode_002@pookmail.com</ns14: seatId>
            <ns14:firstName>Test User 002</ns14:firstName>
            <ns14:lastName>Passcode 002</ns14:lastName>
            <ns14:email>test_user_passcode_002@pookmail.com</ns14: email>
            <ns14:certificateProfileOid>2.16.840.1.113733.1.16.1.5.3.2.1.
            41747424</ns14:certificateProfileOid>
            <ns14:userAttribute>
                <ns14:name>custom_encode_dnsName</ns14:name>
                <ns14:value>Test User 002</ns14:value>
            </ns14:userAttribute>
            <ns14:userAttribute>
                <ns14:name>uniformResourceIdentifier</ns14:name>
                <ns14:value>value!</ns14:value>
            </ns14:userAttribute>
        </ns14:passcodeInformation>
        <ns14:version>1.0</ns14:version>
</ns14:createOrUpdatePasscodeRequest>
```

- clientTransactionID - Use this element to specify a transaction ID for the enterprise RA application (optional). The IDs may be useful if your RA application tracks requests and responses.
- version is the schema version. The DigiCert PKI server communicates the schema version in response to the RA application. Consequently, the RA application knows whether it can consume it.

  NOTE: For this release, this field should be 1.0.

- passcodeInformation is PasscodeInformationType. This element contains the information about the enrollment code to be generated or replaced for the user. This element appears in the request once for each user for whom an enrollment code is generated or replaced.

To generate or replace multiple enrollment codes in one operation, include a PasscodeInformation element for each enrollment code to be generated or replaced.

The following elements are included in PasscodeInformationType:

- passcode - The enrollment code to be assigned to the user. If no enrollment code is provided, an enrollment code is generated and returned. Enrollment codes must be between 6 and 32 characters, and should contain a mixture of upper- and lower-case letters, numbers, and special characters.
  For security purposes, DigiCert recommends that you do not provide an enrollment code, and allow it to be generated randomly.
- numberOfBadAttempts - Providing a custom value for this element is not supported in this release.
- passcodeStatus - Providing a custom value for this element is not supported in this release.
- expiryDateTime. Use this element to set when the enrollment code expires. (The format is: YYYY-MM-DDT:HH:MM:SS.mmm+HH:MM. For example, 2015-09-18T00:00:00.000+00:00.) The value can be from the time of the request to 14 days from the time and date of the request.
  If not provided, the enrollment codes expire 10 days from the date and time of the request.
- creationDateTime - Providing a custom value for this element is not supported in this release.
- firstName - First name of the user.
- lastName - Last name (or surname) of the user.
- email - Email address for the user.
- seatId - A unique identifier for the user. The ID can be a custom element or it can be another UserInformation element. The ID requires that the same attribute is configured as the User identifier in PKI Manager.
- certificateProfileOid is the OID of the certificate profile. The value is mandatory, as every enrollment code is associated with a certificate profile.
- userAttribute - The custom user attributes that appear in the certificate, in the format List of Set<Name<String>, Value<String>>.

9.4.6 createOrUpdatePasscodeResponse

The RA application receives the message from the createOrUpdatePasscodeRequest request.

```
<?xml version="1.0" encoding="UTF-8"?>

<createOrUpdatePasscodeResponse xmlns="http://schemas.verisign.com/
pkiservices/2011/08/usermanagement">

      <clientTransactionID>1433918052018</clientTransactionID>

      <serverTransactionID>61297e4249548d80</serverTransactionID>

      <passcodeCreationStatus>

            <passcodeInformation>

<passcode>passcode!</passcode>

                  <numberOfBadAttempts>0</numberOfBadAttempts>

                  <passcodeStatus>NEW</passcodeStatus>

                  <expiryDateTime>2015-06-14T18:30:00.000+00:00</expiry
                  DateTime>

                  <creationDateTime>2015-06-10T06:33:51.851+00:00</creation
                  DateTime>

                  <seatId>test_user_passcode_001@pookmail.com</seatId>

                  <certificateProfileOid>2.16.840.1.113733.1.16.1.5.3.2.1.
                  41747424</certificateProfileOid>

                  <enrollmentURL>https://pki.symauth.com/certificateservice

                  ?x=xyzabcpki</enrollmentURL>

            </passcodeInformation>

            <statusCode>0</statusCode>

      </passcodeCreationStatus>

      <passcodeCreationStatus>

            <passcodeInformation>

      <passcodeInformation>

                  <passcode>passcode!</passcode>

                  <numberOfBadAttempts>0</numberOfBadAttempts>

                  <passcodeStatus>NEW</passcodeStatus>

                  <expiryDateTime>2015-06-14T18:30:00.000+00:00</expiryDate
                  Time>

                  <creationDateTime>2015-06-10T06:33:51.974+00:00</creation
                  DateTime>
```

```
                    <seatId>Test_User_002@pookmail.com</seatId>

                    <certificateProfileOid>2.16.840.1.113733.1.16.1.5.3.2.1.
                    41747424</certificateProfileOid>

                    <enrollmentURL>https://pki.symauth.com/certificateservice

                    ?x=xyzabcpki</enrollmentURL>

            </passcodeInformation>

            <statusCode>0</statusCode>

        </passcodeCreationStatus>

        <version>1.0</version>

</createOrUpdatePasscodeResponse>
```

- clientTransactionID - Contains the transaction ID sent in the original request (optional). The ID may be useful if your RA application tracks requests and responses.
- serverTransactionID is the transaction ID created by the DigiCert PKI server. The DigiCert PKI server can use this ID to find the transaction in the log file for troubleshooting purposes.
- passcodeInformation is PasscodeInformationType. This element contains the information about the enrollment code to be generated or replaced for the user. This element appears in the request once for each user for whom an enrollment code is generated or replaced. The following elements are included in PasscodeInformationType:
  - passcode - The enrollment code to be assigned to the user.
  - numberOfBadAttempts - The number of times the user has attempted to use the enrollment code to pick up a certificate and failed.
  - passcodeStatus - Status of the enrollment code:
    - NEW - The enrollment code is ready to be used to pick up a certificate.
    - REDEEMED - The enrollment code has already been used to pick up a certificate. It can no longer be used again.
    - LOCKED - The enrollment code has been locked due to too many failed pick-up attempts.
    - EXPIRED - The enrollment code has expired
  - expiryDateTime - Expiration of the enrollment code. (The format is: YYYY-MM-DDT:HH:MM:SS.mmm+HH:MM. For example,
  - 2015-09-18T00:00:00.000+00:00)
  - creationDateTime - Creation of the enrollment code. (The format is: YYYY-MM-DDT:HH:MM:SS.mmm+HH:MM. For example,
  - 2015-09-18T00:00:00.000+00:00)
  - seatId is the unique identifier for the user who is sent in the request.

  - certificateProfileOid - the OID of the certificate profile.
  - enrollmentURL - the enrollment URL for the user to enroll for a certificate with a passcode

- passcodeCreationStatus is PasscodeCreationStatusType. This element identifies whether the enrollment code was successfully generated (or assigned). This element appears in the request once for each user for whom an enrollment code is generated or replaced. The following elements are included in PasscodeCreationStatusType:

  - passcodeInformationType. This element is the same as passcodeInformationType.
  - statusCode is an integer value indicating the success status. The value "0" indicates that the operation was successful.

- version is the schema version. The DigiCert PKI server communicates the schema version in its response to the RA application. Consequently, the RA application knows whether it can consume it.

NOTE: For this release, this field should be 1.0.

NOTE: The 'enrollmentURL' response parameter will not contain an URL for requests submitted against Public SMIME profiles (for example, Secure Email, S/MIME (Digital Signature only), S/MIME (Encryption only)). Instead, the 'enrollmentURL' response will contain this value: <enrollmentURL>URL not provided for Public SMIME certificate enrollments</enrollmentURL>

9.4.7 getPasscodeInformationRequest

The RA application uses this method to return information about an individual enrollment code for a user.

```
<?xml version="1.0" encoding="UTF-8"?>

<ns12:getPasscodeInformationRequest xmlns:ns12="http://schemas.
verisign.com/pkiservices/2011/08/usermanagement">

      <ns12:clientTransactionID>1316083681444</ns12:clientTransaction ID>

      <ns12:seatId>a1@test.com</ns12:seatId>

      <ns12:certificateProfileOid>2.16.840.1.113733.1.16.1.2.5.1.
      1.98395</ns12:certificateProfileOid>

      <ns12:version>1.0</ns12:version>

</ns12:getPasscodeInformationRequest>
```

- clientTransactionID - Use this element to specify a transaction ID for the enterprise RA application (optional). The IDs may be useful if your RA application tracks requests and responses.

- seatId - A unique identifier for the user. The ID can be a custom element or it can be another UserInformation element. The ID requires that the same attribute is configured as the User identifier in PKI Manager
- certificateProfileOid - the OID of the certificate profile. The value is mandatory, as every enrollment code is associated with a certificate profile.
- version is the schema version. The DigiCert PKI server communicates the schema version in its response to the RA applications. Consequently, the RA application knows whether it can consume it.

NOTE: For this release, this field should be 1.0.

9.4.8 getPasscodeInformationResponse

The RA application receives this message from the getPasscodeInformationRequest request.

```
<?xml version="1.0" encoding="UTF-8"?>

<getPasscodeInformationResponse xmlns="http://schemas.verisign.com/
pkiservices/2011/08/usermanagement">

      <clientTransactionID>1316083681444</clientTransactionID>

      <serverTransactionID>eb798624a78cc12f</serverTransactionID>

      <passcodeInformation>

            <passcode>abc123</passcode>

            <numberOfBadAttempts>0</numberOfBadAttempts>

            <passcodeStatus>NEW</passcodeStatus>

            <expiryDateTime>2011-09-18T00:00:00.000+00:00</expiryDate Time>

            <seatId>a1@test.com</seatId>

            <certificateProfileOid>2.16.840.1.113733.1.16.1.2.5.1.

            1.98395</certificateProfileOid>

            <enrollmentURL>https://pki.symauth.com/certificate-
            service?x=xyzabcpki</enrollmentURL>

      </passcodeInformation>

      <version>1.0</version>

</getPasscodeInformationResponse>
```

- clientTransactionID - Contains the transaction ID sent in the original request (optional). The IDs may be useful if your RA application tracks the requests and responses.

- serverTransactionID is the transaction ID created by the DigiCert PKI server. The DigiCert PKI server can use this ID to find the transaction in the log file for troubleshooting purposes.
- PasscodeInformation - This element contains the information about the enrollment code. This element appears in the request once for every enrollment code being updated or replaced. The following elements are included in PasscodeInformationType.
  - passcode - The enrollment code to be assigned to the user.
  - numberOfBadAttempts - The number of times the user has attempted to use the enrollment code to pick up a certificate and failed.
  - passcodeStatus - Status of the enrollment code:
    - NEW - The enrollment code is ready to be used to pick up a certificate.
    - REDEEMED - The enrollment code has already been used to pick up a certificate. It can no longer be used again.
    - LOCKED - The enrollment code has been locked due to too many failed pick-up attempts.
    - EXPIRED - The enrollment code has expired.
  - expiryDateTime - When the enrollment code expires. (The format is: YYYY-MM-DDT:HH:MM:SS.mmm+HH:MM. For example, 2011-09-18T00:00:00.000+00:00)
  - creationDateTime - When the enrollment code was created. (The format is: YYYY-MM-DDT:HH:MM:SS.mmm+HH:MM. For example, 2011-09-18T00:00:00.000+00:00)
    - seatId is the unique identifier for the user who is sent in the request.
    - certificateProfileOid - the OID of the certificate profile.
    - enrollmentURL - the enrollment URL for the user to enroll for a certificate with a passcode
- version is the schema version. The DigiCert PKI server communicates the schema version in its response to the RA application. Consequently, the RA application knows whether it can consume it.

NOTE: For this release, this field should be 1.0.

NOTE: The 'enrollmentURL' response parameter will not contain an URL for requests submitted against Public SMIME profiles (for example, Secure Email, S/MIME (Digital Signature only), S/MIME (Encryption only)). Instead, the 'enrollmentURL' response will contain this value: <enrollmentURL>URL not provided for Public SMIME certificate enrollments</enrollmentURL>

9.4.9 Single user delete request

The RA application uses the following sample request for a deleteUserRequest request to delete a user and revoke their certificate.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/ soap/envelope/"
xmlns:user="http://schemas.verisign.com/pkiservices/2011/08/ usermanagement">

<soapenv:Header/>

<soapenv:Body>

<user:deleteUserRequest>

<!--Optional:-->

<user:clientTransactionID>1234569</user:clientTransactionID>

<user:seatId>abcdefghijklm</user:seatId>

<!--Optional:-->

<user:revocationReason>KeyCompromise</user:revocationReason>

<user:version>1.0</user:version>

<!--You may enter ANY elements at this point-->

</user:deleteUserRequest>

</soapenv:Body>

</soapenv:Envelope>
```

9.4.10 Single user delete response

The following is the sample response for single user delete.

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">

<S:Body>

<deleteUserResponse xmlns="http://schemas.verisign.com/
pkiservices/2011/08/usermanagement">

<clientTransactionID>1234569</clientTransactionID>

<serverTransactionID>12345678</serverTransactionID>
```

```
<deleteUserStatus>
<status>Success</status>
<errorCode>0</ errorCode >
<seatId>abcdefghijklm</seatId>
<revocationCount>10</revocationCount>
</deleteUserStatus>
<version>1.0</version>
</deleteUserResponse>
</S:Body>
</S:Envelope>
```

9.4.11 Bulk user delete request

The RA application uses the following sample request for a bulkDeleteUserRequest request to delete multiple users and revoke their certificates. You can enter up to 25 Certificate Seat IDs to revoke multiple users in one request.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/ soap/envelope/"
xmlns:user="http://schemas.verisign.com/pkiservices/2011/08/ usermanagement">
<soapenv:Header/>
<soapenv:Body>
<user:bulkDeleteUserRequest>
<!--Optional:-->
<user:clientTransactionID>123456789</user:clientTransactionID>
<!--1 to 25 repetitions:-->
<user:seatId>seatid-1111</user:seatId>
<user:seatId>seatid-2222</user:seatId>
<!--Optional:-->
<user:revocationReason>KeyCompromise</user:revocationReason>
<user:version>1.0</user:version>
<!--You may enter ANY elements at this point-->
</user:bulkDeleteUserRequest>
</soapenv:Body>
</soapenv:Envelope>
```

9.4.12 Bulk user delete response

The following is the sample response for bulk user delete.

```
<bulkDeleteUserResponse xmlns="http://schemas.verisign.com/
pkiservices/2011/08/usermanagement">

<clientTransactionID>123456789</clientTransactionID>

<serverTransactionID>c0445cd199aab660</serverTransactionID>

<deleteUserStatus>

<status>Failed with error</status>

<errorCode>A409</errorCode>

<seatId>1111</seatId>

<revocationCount>2</revocationCount>

</deleteUserStatus>

<deleteUserStatus>

<status>Success</status>

<errorCode>0</errorCode>

<seatId>2222</seatId>

<revocationCount>1</revocationCount>

</deleteUserStatus>

<version>1.0</version>

</bulkDeleteUserResponse>
```

APPENDIX E

10 Health Check Protocol Definition

This appendix includes the following topics:

- About the Health Check Protocol
- Health Check Protocol Overview
- Operations
- Messages

10.1 About the Health Check Protocol

This section describes the Health Check protocol. Use it to retrieve status details for different operations.

Currently supports checks for enroll operation.

10.2 Health Check Protocol Overview

To better manage you certificate issuance schedule, use the getStatus operation. You can use the status information to troubleshoot service outages and resumptions.

10.3 Operations

The following WSDL operations are for Health Check Protocol.

10.3.1 getstatus

Invoke this operation to get the health status for different operations. For the current release, only ENROLL operation is supported.

```
<wsdl:portType name="HealthcheckServiceOperations">
      <wsdl:operation name="getStatus">
            <wsdl:input message="tns:statusRequest" />
            <wsdl:output message="tns:statusResponse" />
      </wsdl:operation>
</wsdl:portType>
```

10.4 Messages

The following WSDL message definitions are specific to this operation.

getStatusRequestMessage

```
<xs:element name="getStatusRequestMessage"
type="symhc:GetStatusRequestMessageType" />

<xs:complexType name="GetStatusRequestMessageType">

      <xs:sequence>

            <xs:element name="version" type="symhc:VersionType" />

            <xs:element name="clientTransactionID"
            type="symhc:TransactionIDType" minOccinOccurs="0" />

            <xs:element name="operationType" type="symhc:OperationType"
            minOccurs="1" />

            <xs:element name="profileOid" type="xs:string" minOccurs="0"
            maxOccurs="1" />

      </xs:sequence>

</xs:complexType>
```

- version - This is the schema version. The DigiCert PKI server communicates the schema version in its response to the RA application. Consequently, the RA application knows whether it can consume it.

NOTE: For this release, this should be 1.0.

- clientTransactionID - Use this element to specify a transaction ID for the enterprise RA application (optional). The element may be useful, if your RA application tracks the request and response.
- operationType - Use this element to specify the operation for which the health check needs to be performed.

NOTE: For this release, use ENROLL.

```
<xs:simpleType name="OperationType">

   <xs:restriction base="xs:string">

         <xs:enumeration value="ENROLL" />

   </xs:restriction>

</xs:simpleType>
```

- profileOid - If the operationType is ENROLL, then use this element to specify the ProfileOId for which you want to perform the healthcheck operation. This is a mandatory element for ENROLL operation type.

getStatusResponseMessage

```
<xs:element name="getStatusResponseMessage"
type="symhc:GetStatusResponseMessageType" />

      <xs:complexType name="GetStatusResponseMessageType">

            <xs:sequence>

                  <xs:element name="version" type="symhc:VersionType" />

                  <xs:element name="clientTransactionID"
                  type="symhc:TransactionIDType" minOccurs="0" />

                  <xs:element name="serverTransactionID"
                  type="symhc:TransactionIDType" />

                  <xs:element name="statusCode" type="symhc:StatusCode" />

                  <xs:element name="statusMessage" type="xs:string" />

            </xs:sequence>

</xs:complexType>
```

- version - This is the schema version. The DigiCert PKI server communicates the schema version in its response to the RA application. The communication lets the RA application know whether it can consume the version.
- clientTransactionID - This is the transaction ID sent in the original request (optional). The ID may be useful if your RA application tracks the request and response.
- serverTransactionID - This is the transaction ID created by the DigiCert PKI server The DigiCert PKI server can use this ID to find the transaction in the server log file for troubleshooting purposes.
- statusCode - This is the health status of the operation specified in the Request. The value would be either "UP" or "DOWN". "UP" – represents state of system where in the specified operation in request is expected to complete successfully. "DOWN" – represents state of system where in the specified operation in the request might fail.

```
<xs:simpleType name="StatusCode">

      <xs:restriction base="xs:string">

            <xs:enumeration value="UP" />

            <xs:enumeration value="DOWN" />

      </xs:restriction>

</xs:simpleType>
```

- statusMessage - This provides some additional information about the current state of the service.

APPENDIX F

11 Error Codes and Troubleshooting

This appendix includes the following topics:

11.1 Error Handling

DigiCert PKI communicates errors between the RA application and the service using SOAP Fault messages. The SOAP 1.1 Fault element defines four sub-elements:

- faultcode: The faultcode element is intended for use by software to provide an algorithmic mechanism for identifying the fault. The faultcode must be present in a SOAP Fault element and the faultcode value must be a qualified name. SOAP defines a small set of SOAP fault codes covering SOAP faults. DigiCert defines additional errors specific to DigiCert PKI. Refer to Table F-1 for these errors and possible solutions.
- faultstring: The faultstring element is intended to provide a human readable explanation of the fault and is not intended for algorithmic processing. The faultstring must be present in a SOAP Fault element and should provide at least some information explaining the nature of the fault.
- faultactor: The faultactor element is intended to provide information about what component caused the fault within the message path. The value of the faultactor attribute is a URI identifying the source of the fault. Applications that do not act as the ultimate destination of the SOAP message must include the faultactor element in a SOAP Fault element.
- detail: The detail element is intended to carry the specific error information that is related to the Body element. It must be present if the contents of the Body element could not be successfully processed. It must not be used to carry information about error information belonging to header entries (this information must be carried within header entries).

  If the detail element is not present in the Fault element, the fault is not related to processing of the Body element. The information can be used to distinguish whether or not the Body element was processed in case of a fault situation.

The manner in which DigiCert PKI handles error situations depends on the error category:

- High-level error situations (such as missing or unsupported parameters or authorization errors) are communicated using the format:

  ```
  <faultcode> soap:client </faultcode>
  ```

  Where <faultstring/> is used to present human readable information.

- If the service is not available or is not able to process the request, the error is communicated as using the format:

  ```
  <faultcode>soap:server</faultcode>
  ```

  Where <faultcode/> is used to present human readable information.

- DigiCert PKI-specific error codes and details message are part of the SOAP Fault <details> element. See “DigiCert PKI Error Codes” on page 205. This section lists the error codes specific to DigiCert PKI, as well as suggested solutions.

11.1.1 Schema Representation:

```
<xs:Envelope xmlns:wst="http://docs.oasis-open.org/ ws-sx/ws-trust/200512/
      xmlns:xs=http://schemas.xmlsoap.org/soap/envelope/">
            <xs:Body>
                  <xs:Fault>
                        <faultcode> wst:InvalidRequest</faultcode>
                        <faultstring>The request was invalid or malformed
</faultstring>
                        <detail>
                              <vswstep:faultdetails>
                                    <message>
                                          My application didn't work
                                    </message>
                                    <errorcode> 1001
                                    </errorcode>
                              </e:myfaultdetails>
                        </detail>
                  </SOAP-ENV:Fault>
            </SOAP-ENV:Body>
```

```
</SOAP-ENV:Envelope>
```

11.1.2 Sample Client Code to Handle Fault Response as Exception:

The example java axis client catches faults and parses the response. Optionally, refer to the processPKIServiceFault method of the PKIWSClientHelper class in the Java sample for details on handling fault exceptions.

See "About the PKI Web Service Sample Java Code" on page 31.

```
try{

// Call the service with input data & get the output EnrollmentServiceResponse
resp = RequestSecurityToken.request (requestVariable);

// Parse the response here

}

catch (Exception e)  { if(e.getCause() instanceof AxisFault)

{

AxisFault af = (AxisFault)e.getCause(); String errorDetails = "";

if(af.getFaultDetailElement() != null)

{

Iterator<OMElement> iter = af.getFaultDetailElement().getChildren();


while(iter!= null && iter.hasNext())

{

// Get all elements in details element of the fault response OMElement element =
iter.next();

errorDetails += element.getText() + " ";

}

System.out.println(af.getFaultMessageContext().getEnvelope());

}

}

}
```

11.2 DigiCert PKI Error Codes

Table F-1 lists the DigiCert PKI error codes, as well as suggested solutions. This table also identifies which protocol can throw the error.

If you contact DigiCert Technical Support for assistance with any of these errors, provide the following information. The information helps Technical Support with troubleshooting:

- DigiCert PKI operation that received the error
- Client Transaction ID
- Server Transaction ID
- Fault error code and message

These error codes are available in your Fault response.

Table F- 1 DigiCert PKI error codes

Error Message Code	Protocol That can Throw the Error	Error Message	Solution
A200	All protocols	Your request is invalid. Schema verification failed.	The request did not validate against the schema. Rebuild the request correctly and try again. You can encounter this error during the renewal of a Public S/MIME certificate if the renewal request is trying to update the rfc822Name value within the SAN extension. Ensure the value is the same as the to-be renewed certificate.
A201	Certificate Enrollment Policy Protocol	The status of the requested certificate profile is inactive.	The certificate profile in the enrollment request is inactive. Make the certificate profile active or use another profile in the request.

Error Message Code	Protocol That can Throw the Error	Error Message	Solution
A202	Certificate Enrollment Protocol or User Management Protocol	Enrollment information did not contain a mandatory attribute.	As stated. Review the policy to determine the missing attribute, and then re-send the request.
A203	Certificate Enrollment Protocol or User Management Protocol or Certificate Enrollment Policy Protocol	This operation is not supported for a migrated certificate profile.	As stated. Review the policy to determine the superseededOID.
A204	Certificate Enrollment Protocol or User Management Protocol or Certificate Enrollment Policy Protocol	This operation is not supported for a deleted certificate profile.	As stated. Perform the operation with an active profile.
A30B	Get Single Policy, Certificate Enrollment Protocol, Key Recovery, Certificate Renewal Protocol	The certificate profile used in this request is mapped to a different RA certificate.	The authentication request failed. Review if your certificate profile is mapped to the RA certificate sent in the request and try again. Authentication can fail for the Get Single Policy, Certificate Enrollment Protocol, Key Recovery, or Certificate Renewal Protocol if the certificate profile is not mapped to the RA certificate. Ensure that you use the correct RA certificate with this request.

Error Message Code	Protocol That can Throw the Error	Error Message	Solution
			If this error persists, contact DigiCert Technical Support.
A30C	Certificate Enrollment Protocol	Organization not authenticated. Please make sure the organization name matches the account organization and is approved.	Ensure that, if the request contains cert_corp_company value, then it should match the account organization name and the organization should be approved. If the request does not contain cert_corp_company, then the account organization should be approved.
A30D	Certificate Enrollment Protocol	Domain not authenticated. Please make sure the domain exists in your account and it is approved.	Ensure that the domain name from mail_email value should exist in that account and it should be approved.
A300	All protocols	Failed to authenticate request.	The authentication request failed. Review your authentication method and try again. Authentication can occur for the User Management Protocol if the RA certificate is not valid for the specified certificate profile. Ensure that you use the correct RA certificate with this request. If this error persists, contact DigiCert Technical Support.

Error Message Code	Protocol That can Throw the Error	Error Message	Solution
A301	All protocols	Authentication certificate has expired.	Obtain a valid certificate and try again.
A302	All protocols	Authentication certificate has been revoked.	Obtain a valid certificate and try again.
A303	All protocols	VeriSign Security Token cannot be verified.	Obtain a valid certificate and try again.
A304	All protocols	VeriSign Security Token expired.	Obtain a valid certificate and try again.
A305	All protocols	Security Token Service internal error.	Obtain a valid certificate and try again. If this error persists, contact DigiCert Technical Support for assistance.
A306	All protocols	Passcode has been redeemed.	As stated.
A307	All protocols	Passcode has expired.	Contact DigiCert Technical Support for assistance.
A308	Certificate Enrollment Protocol	Invalid PKCS#7 in renewal request. Rebuild the request and retry the operation.	As is stated.
A309	All protocols	Passcode is deleted.	Contact DigiCert Technical Support for assistance.
A310	All protocols	Passcode is locked.	Contact DigiCert Technical Support for assistance.
A400	All protocols	The specified request failed.	Retry the operation later. If you continue to receive this error, contact DigiCert Technical Support for assistance.

Error Message Code	Protocol That can Throw the Error	Error Message	Solution
A401	All protocols	Feature not supported.	The requested feature is not supported for your account. The Key Management Server also returns the error, if it receives a request for a non-recovery operation.
A402	All protocols	Database is not accessible. Try again later.	Retry the operation later. If you continue to receive this error, contact DigiCert Technical Support.
A403	All protocols	Operation not supported.	The requested operation is not supported for your account.
A404	All protocols	DigiCert PKI configuration error.	This account is not properly configured. Contact DigiCert Technical Support for assistance.
A405	• Certificate Enrollment Policy Protocol • Certificate Enrollment Protocol	The account's policy file cannot be read. Run the Policy Wizard again. If this error persists, contact your administrator for assistance.	This account is not properly configured. Contact DigiCert Technical Support for assistance.
A406	Certificate Enrollment Protocol	Cannot determine the certificate profile based on the input.	Cannot determine the certificate profile based on the input. Provide the profile ID or run getPolicies to obtain the correct policy profileOID.

Error Message Code	Protocol That can Throw the Error	Error Message	Solution
A407	Certificate Enrollment Protocol	Key Management Server configuration error.	The Key Management Server is not properly configured. Contact DigiCert Technical Support for assistance.
A408	User Management Protocol	Could not find a user for this Seat ID and Profile ID. Rebuild the request and retry the operation.	As stated.
A409	User Management Protocol	Could not find the Seat ID in the system.	Provide a valid value for the Seat ID.
A500	All protocols	Your request did not include any data.	Your request did not include any data. Submit the request again with proper content.
A501	All protocols	Malformed request	The request does not meet the DigiCert WSDL or schema requirements. Rebuild the request correctly and try again.
A502	All protocols	Invalid or empty soap action	The SOAP action header element does not meet the DigiCert WSDL or schema requirements. Construct the SOAP action correctly and try again.
A503	All protocols	Request version is not supported	The version of your SOAP request is not supported.
A504	Certificate Enrollment Protocol	The PKCS#10 object that is provided is not valid. Rebuild the object correctly and retry the operation.	As stated.

Error Message Code	Protocol That can Throw the Error	Error Message	Solution
A505	Certificate Enrollment Protocol	The certificate profile ID provided in the request is invalid.	Policy does not support the certificate profile ID provided in the enrollment request, or no profile ID was provided. Provide the profile ID or run getPolicies to obtain the correct policy profileOID. An invalid profile ID can occur if you use DigiCert PKI Web Service with an account that stores user data in an enterprise store. DigiCert PKI Web Services is supported only for accounts with user data stored at DigiCert.
A506	Certificate Enrollment Protocol	An expected attribute is missing from the enrollment request. Correct the request and retry the operation.	Verify the name-value pairs that are specified in the request against those required by the policy.
A507	Certificate Enrollment Protocol	Data type validation that has failed for an attribute value in the enrollment request.	In the request, the data type for an attribute value does not match the data type specified by the policy. Correct the attribute value and retry the operation.
A508	Certificate Enrollment Protocol	The key length of the binary security token that is provided does not match the policy.	Check the policy file and ensure that PKCS10 is created with a key length greater than or equal to the key that is specified in the policy.

Error Message Code	Protocol That can Throw the Error	Error Message	Solution
A509	Certificate Enrollment Protocol	The length of an attribute value is invalid. Correct the request and retry the operation.	Attribute value is too long (refer to RFC5280). Country attribute only supports printable string with length of 2.
A40A	User Management Protocol	Request failed due to failure in delete user operation.	Rebuild the request and retry the operation.
A50A	Certificate Enrollment Protocol	certPublishFlag is not set correctly according to the policy file.	certPublishFlag name value pair is not sent properly. If clientProvided, publish_flag needs to be sent as **yes** or **no**.
A50B	Certificate Enrollment Protocol	The token type that is requested is not supported. Correct the request and retry the operation.	The token type that is specified in the request does not match the token type that is configured in the policy. Correct the request and retry the operation.
A50C	Certificate Enrollment Protocol	The PKCS#10 object that is provided contains the values that are not supported. Rebuild the object correctly and retry the operation.	The PKCS10 contains values other than the public key. Create the PKCS10 with only the public key. No additional attributes are included.
A50D	Certificate Management Protocol	The PKCS#12 password that is provided is not valid. The password can contain only alphanumeric characters and should conform to password policy.	The enterprise RA application password is not correctly formatted. Correct the password and try the operation again.

Error Message Code	Protocol That can Throw the Error	Error Message	Solution
A50F	Certificate Enrollment Protocol	A mandatory name value pair is absent from the enrollment request. Correct the request and retry the operation.	As stated. Review the policy to determine the missing name-value pair, and then re-send the request.
A51A	User Management Protocol	An expected attribute Seat ID is missing in the delete user request. Correct the request and retry the operation.	Provide a valid value for the Seat ID.
A51B	Certificate Enrollment Protocol	The RSA public key in PKCS#10 object is encoded incorrectly. Rebuild the object correctly and retry the operation.	As stated.
A51C	Certificate Enrollment Protocol	Multiple email addresses not allowed in SMIME certificates. Correct the request and retry the operation.	Ensure that, if the certificate request contains emailAddress and otherNameUPN fields, then they should match the mail_email value (RFC822 Name).
A51D	Certificate Enrollment Protocol	CA ISSUER URI is missing in AIA extension. Please contact DigiCert Support with your Account and Issuing CA Names to update your CA policy and include the required URI.	Contact DigiCert Support with your Account details and Public Issuing CA name for the certificate that caused the error, who will request that your CA Policy is updated to include the required CA Issuer URL inside your AIA extension.
A510	Certificate Enrollment Protocol	The PKCS#7 object that is provided is not valid or cannot be constructed	As stated.

Error Message Code	Protocol That can Throw the Error	Error Message	Solution
		from the information provided. Rebuild the object correctly and retry the operation.	
A511	Certificate Enrollment Protocol	The request type in the request is invalid. Please correct the request and retry the operation.	As stated.
A512	Certificate Enrollment Protocol	The name in the name value pair that is provided is empty or invalid. Correct the request and retry the operation.	As stated.
A513	Certificate Enrollment Protocol	To-be-renewal certificate is outside of the renewal grace period window.	In the request, the certificate that is sent for renewal is past its available renewal date. You must enroll for a replacement certificate.
A514	User Management Protocol	The enrollment code that is provided cannot be less than 6 or more than 32 characters long. Correct the request and retry the operation.	Provide an enrollment code between 6 and 32 characters inclusive.
A515	User Management Protocol	The seat ID in the request was invalid or cannot be found. Correct the request and retry the operation.	Provide a valid value for seatId.
A516	User Management Protocol	The passcode information provided was invalid or the passcode information is	For createOrUpdatePasscode, one or more attributes for the enrollment code that is

Error Message Code	Protocol That can Throw the Error	Error Message	Solution
		not available for the response. Correct the request and retry the operation or create the passcode and try the operation again.	provided are invalid. For example, the expiry time may be longer than 14 days or in the past. For getPasscodeInformation, an enrollment code is not assigned to the user for the specified certificate profile. Create an enrollment code and try the operation again.
A517	User Management Protocol	User information in the request is invalid. Correct the request and retry the operation.	The request contains invalid information. Correct the request and then re-send the request.
A518	Certificate Enrollment Protocol	The ECDSA curve size of the binary security token that is provided does not match the policy.	Check the policy file and make sure PKCS10 is created with a key pair as per the curve size.
A519	User Management Protocol	An expected attribute Seat ID is missing in the delete user request. Correct the request and retry the operation.	Provide a valid value for the Seat ID.
A600	User Management Protocol	An internal service error occurred. Retry the operation later.	Values that are provided do not meet the length requirements. Ensure that the values that are provided meet the following character limits (inclusive): UserInformationType attributes • seatId - 1-255 • firstName - 0-255 • lastName - 0-255 • emailAddress - 0-255

Error Message Code	Protocol That can Throw the Error	Error Message	Solution
			• deskPhoneNumber – 0-64 • mobilePhoneNumber - 0-64 NameValueType attributes • name - 1-64 • value - 1-255
A600	All protocols	An internal service error occurred. Retry the operation later.	For all other situations, contact DigiCert Technical Support for assistance.
A601	Certificate Enrollment Protocol	Enrollment request is incomplete or incorrect. Correct the enrollment request and retry the operation.	Verify the name-value pairs that are specified in the request against those required by the policy. Make sure that all required name-value pairs are provided.
A602	All protocols	The administrator or the user certificate for this account is invalid or revoked. Retry the operation with a valid certificate.	As stated.
A603	All protocols	An internal service error occurred. Retry the operation later.	Contact DigiCert Technical Support for assistance. Could also occur if the value length is invalid.
A604	Certificate Enrollment Protocol	A certificate has already been issued with this enrollment information.	As stated. Retry the request with unique enrollment information.
A605	Certificate Enrollment Protocol	The enrollment request cannot be processed as it exceeds the number of	No seats are available for this account. Contact your

Error Message Code	Protocol That can Throw the Error	Error Message	Solution
		certificates available for this account.	DigiCert representative to purchase more seats.
A606	Certificate Management Protocol	Key escrow is not supported for this certificate ID. Correct the enrollment request and retry the operation.	The certificate profile ID provided does not support key escrow.
A607	Certificate Enrollment Protocol	You are attempting to issue a certificate from a legacy CA chain.	Please contact your administrator to configure a profile using an Issuing CA that chains up to a DigiCert trusted Root CA.
A608	Certificate Enrollment Protocol	You are attempting to issue a certificate from a legacy CA chain.	Please contact your administrator to configure a profile using an Issuing CA that chains up to a DigiCert trusted Root CA
A701	Certificate Management Protocol	Certificate Management Service is unable to perform the operation. The certificate is already revoked.	You cannot revoke a certificate that is already revoked.
A702	Certificate Management Protocol	Certificate Management Services revoke operation failed due to missing parameters. Correct the request and try again.	Revocation request is incomplete or incorrect. Correct the revocation request and retry the operation.
A703	Certificate Management Protocol	The Certificate Management Service failed, because the certificate for the request was not found. Correct the request and try again.	The certificate that is identified in the request was not found. Correct the request and retry the operation.

Error Message Code	Protocol That can Throw the Error	Error Message	Solution
A704	Certificate Management Protocol	Certificate Management Service failed due to invalid certificate serial number. Correct the request and try again.	The serial number in the request was incorrect or not for a valid certificate. Correct the request and retry the operation.
A705	Certificate Management Protocol	Certificate Management Services revoke operation failed due to invalid reason code. Correct the request and try again.	The revocation reason code in the request is not valid. Correct the reason code and retry the operation.
A706	Certificate Management Protocol	The value type that is provided for this admin ID is invalid. Correct the request and try again.	Correct the value type that is provided for this administrator ID and retry the operation.
A707	Certificate Management Protocol	The Key Management Server cannot find the key escrow record for this certificate serial number. Correct the request and try again. If Key Recovery required dual Administrators and send 2nd request with same administrator of 1st request, encounter this error code.	Update the request with a valid certificate serial number and retry the operation. The key escrow record for the given certificate serial number must also be available. If dual Administrators approval required, please use different Administrator on 2nd request.
A708	Certificate Management Protocol	The administrator may not have the correct role or the administrator certificate for this account is invalid or revoked.	The administrator certificate that is used to perform this operation must be valid. The administrator must also have the correct

Error Message Code	Protocol That can Throw the Error	Error Message	Solution
		Correct the request and try again.	role. Correct these issues and retry the operation.
A709	Certificate Management Protocol	Unable to perform Certificate Management Operation as the data that is provided in the request is invalid. Correct the request and try again.	Information in the request is in an invalid format. Correct the request and retry the operation.
A70B	Certificate Management Protocol	Certificate status is not valid. It might have been suspended already.	Verify the certificate status and try again.
A70C	Certificate Management Protocol	Certificate status is not in suspend state. It might have been resumed already.	You cannot resume a certificate that is not in a suspended state.
A70D	Certificate Management Protocol	Unable to perform certificate management operation for the input parameters. Correct the request and try again.	Operation type is not supported. Contact DigiCert Technical Support for assistance.
AD01	Certificate Enrollment Protocol	You have exceeded the number of certificates available for this certificate type. Contact your administrator for assistance.	The seat pool has reached the maximum number of allowed certificates. Add some seats to the seat pool.
AF01	Health Check Protocol	profileOid is missing. profileOid is mandatory for ENROLL operationType.	profileOid parameter is either not provided or is empty. For ENROLL operationType, profileOid parameter is mandatory. Add the profileOid parameter to the request and retry.

11.3 DigiCert PKI Errors without Error Codes

This section lists some errors that may not have a specific error code but may have an error string.

11.3.1 Non-unique body parts

If your WS-trust WSDL defines two operations with the same signatures, you may encounter an error similar to the following:

```
[Error] Non-unique body parts! In a port, as per BP 1.1 R2710 operations must
have unique operation signature on the wire for successful dispatch. In port
VeriSignCertServiceSOAP, Operations "RequestSecurityToken2" and
"RequestSecurityToken" have the same request body block {http://docs.oasis-
open.org/ ws-sx/ws-trust/200512/} RequestSecurityToken.
```

The JAX-WS schema compiler needs to use the -extension switch to compile the wsdl. Axis2 schema compilers need to make similar changes.

To resolve this issue, your RA applications must use the defined SOAP action to invoke the required operation.

11.3.2 Bouncy Castle Error

PKI Web Services sample client displays bouncy castle error messages while it tries to parse the certificate enrollment response for a non-key escrow profile. The enrollment works fine. But, the sample client struggles to parse the certificate in the response as bouncy castle jar files are not available in Java path.

The resolution is to place the bouncy castle jar files in Java class path under

```
JRE_HOME/lib/ext. The jar files bcmail-jdk16-1.46.jar and bcprov-jdk16-1.46.jar
are available in the package under sampleClient\tools\clientApp\lib\thirdparty.
```

Index