SeeBeyond™ eBusiness Integration Suite

Secure Messaging Extension User's Guide

Release 5.0.1

Version 20031118211638.

Contents

Chapter 1

Introducing Secure Messaging Extension 5

Document Organization 6

Overview 6

Components 6

Supported Operating Systems 7

System Requirements 7

Introducing Secure Messaging Extension (SME) 8

Security Component 8

Compression Component 8

Introducing Multipurpose Internet Mail Extension (MIME) 9

Introducing Secure Multipurpose Internet Mail Extension (S/MIME) 10

Overview of SME Processes 11

SME Encryption/Decryption Process 11

SME Signature/Verification Process 13

SME Compression/Decompression Process 14

Chapter 2

Installation 15

Before Installing Secure Messaging Extension 15

Installing the Secure Messaging Extension 15

Installing eGate 15

Additional Files Required to Run SME 16

Chapter 3

Encrypted Message Formats, Digital Signature Formats, and Certificate Formats 17

Encrypted Message Formats 17

Digital Signature Formats 18

Signing and Attaching Signatures 22
Private Key Format 23
Certificate Formats 23

Chapter 4
Managing Keystores and Truststores 30
Overview 30
Steps Required to Create and Manage Private Keys 31
To Import a New Certificate: 31
To Manage a Public Certificate: 34
To Create a New Truststore: 37
To Import a Certificate into a Truststore 39

Chapter 5
S/MIME Collaboration Definitions 40
SME Collaborations 40
Available OTDs 41

Chapter 6
Using the Sample Project in eGate 42
Importing the SME_Sample_Project 42
Required Input Parameters 43
43
Configuring the File eWays 43
Configuring the JMS Clients 44
Creating an Environment 44
Creating and Activating the Deployment Profile 44
Running the Project 44

Index 45

Chapter 1

Introducing Secure Messaging Extension

This document describes how to install, configure, and use the SeeBeyond Technology Corporation's Secure Messaging Extension, referred to as SME throughout the rest of this document.

The topics in this chapter include:

- **"Document Organization" on page 6**
- **"Overview" on page 6**
- **"Supported Operating Systems" on page 7**
- **"System Requirements" on page 7**
- **"Introducing Secure Messaging Extension (SME)" on page 8**
- **"Introducing Multipurpose Internet Mail Extension (MIME)" on page 9**
- **"Introducing Secure Multipurpose Internet Mail Extension (S/MIME)" on page 10**
- **"Overview of SME Processes" on page 11**

1.1 Document Organization

This User's Guide is organized into two parts. The first part, consisting of Chapters 1-2, introduces SME and describes the procedures for installing and setting up the program. This part should be of particular interest to a System Administrator or other user charged with the task of getting the system up and running.

The second part, consisting of Chapters 3-6, describes the details of SME operation and configuration, including descriptions of encrypted message formats, instructions on Keystore management, and implementation of sample SME projects. This part should be of particular interest to a Developer involved in customizing SME for a specific purpose.

1.2 Overview

SME enables e*Gate to process Events using the S/MIME (Secure Multipurpose Internet Mail Extensions) message format. This format is the IETF RFC 2311 specification for encrypting and/or signing types of data.

SME supports encryption, decryption and authentication of messages and is interoperable with any other client applications that support the S/MIME standard.

SME adds the following features to transactions:

- privacy
- message (Event) authentication
- sender authentication
- nonrepudiation

1.2.1 Components

Components required to run SME include:

- eGate Integrator
- File eWay

1.3 Supported Operating Systems

Secure Messaging Extension is supported on the following operating systems:

- Windows 2000 SP3
- Windows XP SP1a
- Windows Server 2003
- Solaris 8 and 9
- AIX 5.1 and 5.2
- HP-UX 11.0 and HP-UX 11i (RISC)
- HP Tru64 5.1A
- Red Hat Linux 8 (Intel)
- Red Hat Linux Advanced Server (Intel)

1.4 System Requirements

To set up and run the IMS eWay with the eGate Enterprise Designer, you need the following:

- A TCP/IP network connection.
- Windows Server 2003, Windows 2000 SP3, or Windows XP. This is required for the User Interface.
- Microsoft Internet Explorer 6.0 SP1 or above.

Note: *Open and review the* ***Readme.txt*** *prior to installation for any additional requirements.*

1.5 Introducing Secure Messaging Extension (SME)

The SME product has the dual purpose of offering security features, which allow protected transmission of public domains such as the internet, and compression/decompression technology to effectively reduce/expand the size of files.

Security Component

As part of the security component, SME uses Public Key Infrastructure (PKI) technology to ensure the confidentiality of exchanges. This is done by digitally signing and encrypting messages as they are sent, and decrypting and authenticating messages when they are received.

SME performs the encryption and decryption of messages using the Secure/Multipurpose Internet Mail Extension (S/MIME). S/MIME is a specification for securing electronic mail, and is designed to add security to e-mail messages in MIME format.

S/MIME creates one-way hash algorithms that ensure data integrity by verifying no modifications are made to the message while in transit. In addition, the message sender's identity is verified through the use of digital signatures, proving that the message actually originated from the entity who claims to have sent it. For more information on the S/MIME format, see **"Introducing Secure Multipurpose Internet Mail Extension (S/MIME)" on page 10**

Security Services Offered Through SME Include:

- Encryption
- Decryption
- Sign Service
- Signature Verification Service

Compression Component

SME compression converts string and byte file formats, such as those found in text, graphics, audio, and video files, into smaller sized files. This is done using Java-based mathematical equations that scan and index repetitive patterns. If a file contains repetitive patterns—such as colors used in an image—then code is written to index the number of and exact placement of those patterns, effectively reducing the size of the file. When you decompress a file, the code that contains the index of repetitive patterns rebuilds the file to its original format.

Compression Services Offered Through SME Include:

- Compression
- Decompression

1.6 Introducing Multipurpose Internet Mail Extension (MIME)

MIME Message Format

As a specification for formatting non-ASCII messages, MIME enables the transfer and acceptance of files via the Internet mail system. MIME-compliant messages may contain any type of data, including the following:

- Text messages in US-ASCII
- Messages of unlimited length
- Binary files
- Character sets other than US-ASCII
- Multi-media: Image, Audio, and Video objects
- Multiple, nested objects in a single message

When later sent over a protocol such as HTTP or FTP, which provide a "binary clean" data path, MIME messages may be left in binary format. However, if the MIME message is sent via SMTP (E-mail) or other text-only protocols, binary objects must be encoded using the Base64 content transfer encoding format, which produces a textual representation of the original binary data.

Messages in MIME format consist of two parts: the header and the body. The header forms a collection of metadata in the form of keyword/value pairs structured to provide information necessary for the transmission and interpretation of the message. The body of the message contains the bulk data to be transferred. In turn, S/MIME defines the security services, adding digital signatures and encryption, thus preventing forgery and interception.

For more information regarding MIME, see the Internet Engineering Task Force Text Messages specification (RFC 822) and the MIME Message Body Format (RFC 2045), at http://www.ietf.org.

The S/MIME Version 3 specification (RFC 2623) is also found at http://www.ietf.org.

1.7 Introducing Secure Multipurpose Internet Mail Extension (S/MIME)

S/MIME is an encryption supported version of the MIME protocol. It is based on the Public Key Cryptography Standards (PKCS), which specify how the RSA public-key cryptographic algorithm should be used to implement enveloped encryption and digital signatures.

The RSA public-key system makes use of two related keys to perform the mathematical algorithms necessary to encrypt or decrypt data: a public key, which may be made available to any prospective correspondent, and a private key known only to the key's owner. A public key can be published openly, thereby assuring the ability of anyone to send secure messages that can only be decrypted by the owner of the respective private key.

Encryption can also be performed using one's private key, and decrypted with the corresponding public key. In this case, the encryption result is known as a digital signature, which guarantees to the intended recipient that the signed message is authentic and genuinely came from the stated originator of the message.

Digital signatures provide data integrity, authentication and non-repudiation of an electronic document. Successful verification of a digital signature ensures the recipient that the “document received” is identical to the “document sent” (data integrity) and confirms the identity of the sender (authentication). It also prevents any subsequent denial by the sender that the document originated with them (non-repudiation).

In practice, public keys are stored as certificates that comply with the X.509 standard. In addition to the public key, a certificate also contains information about the key owner's identity, the key's validity, and the issuer of the certificate, also known as a Certificate Authority.

1.8 Overview of SME Processes

The following diagrams outline the key activities involved in the SME processes, including:

- SME Encryption/Decryption Process
- SME Signature/Verification Process
- SME Compression/Decompression Process

1.8.1 SME Encryption/Decryption Process

This section describes the internal and external flow of the SME encryption, using the keypair encryption method. An illustration of the encryption method is also found in **Figure 1 on page 12**.

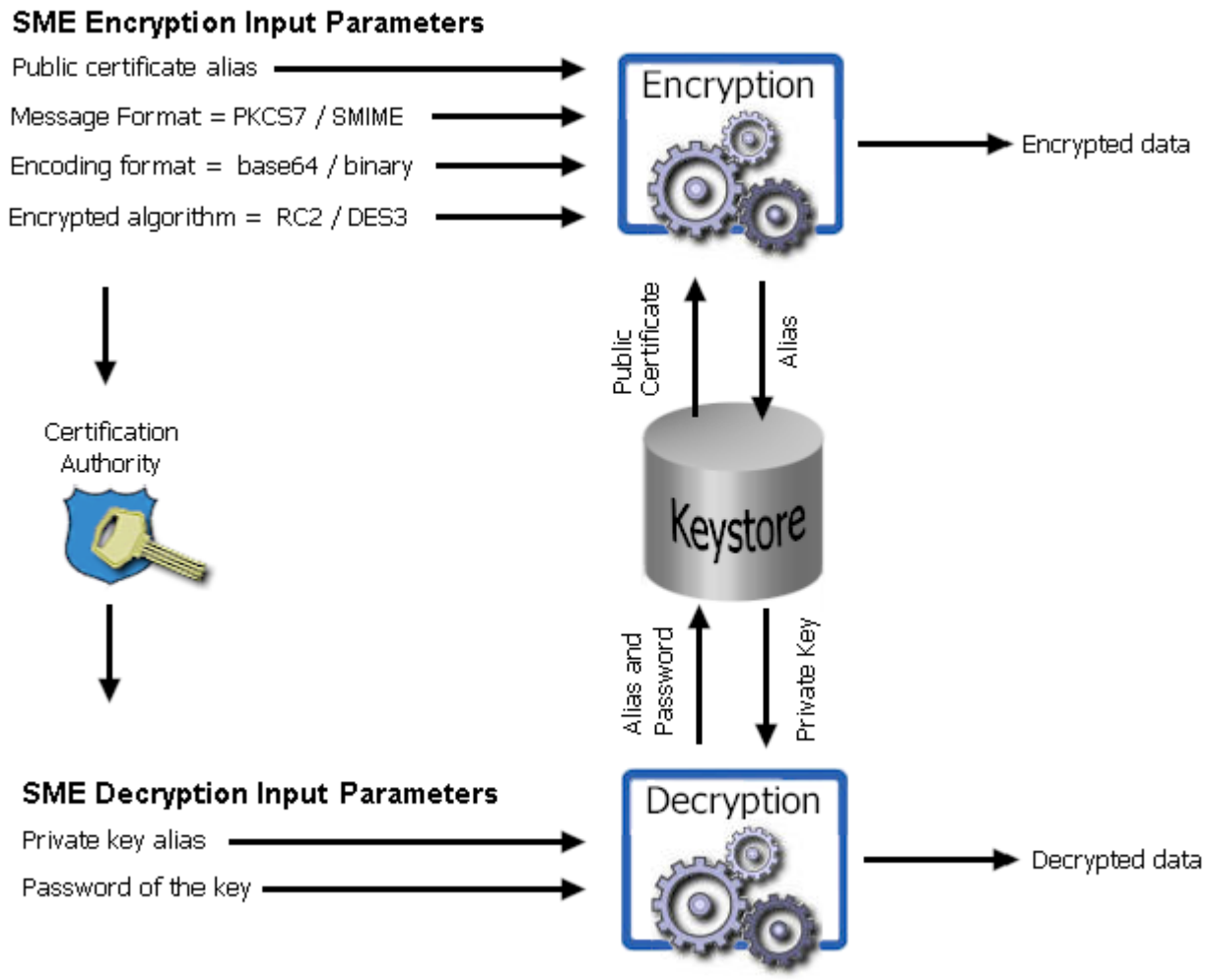
The encryption process begins when the sender's message is encrypted with the public key. The message is also signed by the sender, and the signature itself is encrypted with the sender's private key. When the reader receives the message, the encryption is decoded with the reader's private key. The sender's Public Certificate, located in the Keystore is used to verify the authenticity of the public key.

In addition to verifying the public key, public certificates also contain the sender's personal information, such as name, institution, and e-mail address, and are signed by a trusted Certificate Authority.

During encryption, a Public Certificate alias is used to identity the Public Certificate located in the Keystore. During decryption, the reader private key alias and password to access the Private Key from the Keystore and decrypt the message.

The encryption/decryption process illustrated in **Figure 1 on page 12**, details the SME Input Requirements for both encryption and decryption of data.

Figure 1 Secure Messaging Extension Encryption Process

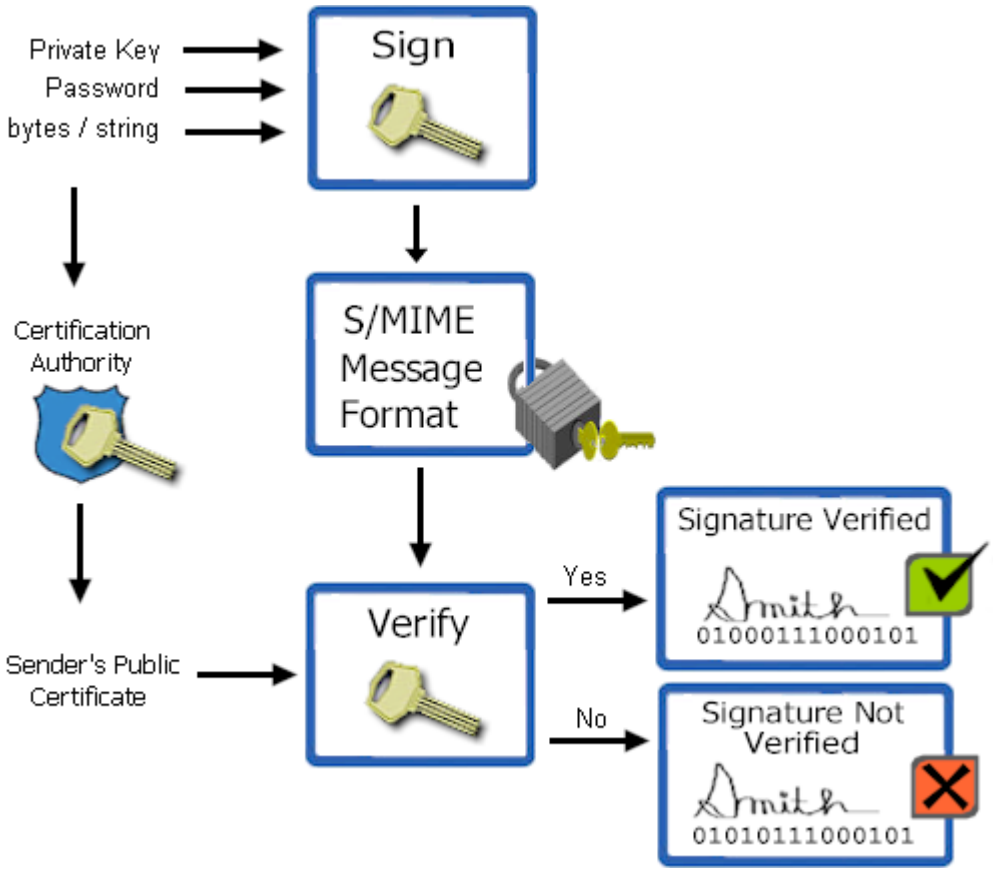


1.8.2 SME Signature/Verification Process

The SME signature/verification process begins when a subscriber publishes a certificate to a Certificate Authority. Published certificates contain the subscriber's identity and public key, and are digitally signed by the Certification Authority. The Certification Authority is also responsible for safeguarding access to the subscriber's private key, which is required during the verification process.

When a subscriber signs and sends a message, the SME Sign process converts the message from MIME to S/MIME format. The S/MIME message format also contains the digital footprint of the subscribers private key, so when the message is received by another user, the public key held by the Certification Authority "reads" and then verifies the digital signature created by the private key.

Figure 2 Secure Messaging Extension Signature Verification Process



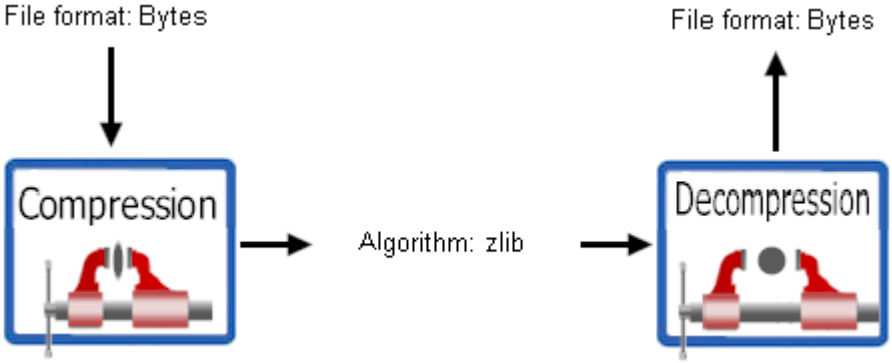
1.8.3 SME Compression/Decompression Process

The SME compression process converts byte type files into PKCS#7 format using the zlib compression library. For more information on the PKCS#7 see **"PKCS#7 encrypted message format" on page 17**.

For more information on the zlib compression library, visit the gzip home page at:

http://www.gzip.org

Figure 3 Secure Messaging Extension Compression Process



Chapter 2

Installation

This chapter describes the procedures for installing SME.

- **"Before Installing Secure Messaging Extension" on page 15**
- **"Installing the Secure Messaging Extension" on page 15**
- **"Additional Files Required to Run SME" on page 16**

2.1 Before Installing Secure Messaging Extension

Open and review either the Readme.txt for any additional information or requirements, prior to installation. The Readme.txt is located on the Repository CD-ROM.

2.2 Installing the Secure Messaging Extension

During the installation process, the Enterprise Manager, a web-based application, is used to select and upload the SME component (SMEWebServices.sar) from the eGate installation CD-ROM to the Repository.

Installing eGate

The eGate installation process includes the following components:

- Installing the eGate Repository
- Uploading products to the Repository
- Downloading components (including eGate Enterprise Designer and Logical Host)
- Viewing product information home pages

To install the SME component on an eGate supported system, follow the instructions for installing the eGate Integrator in the ICAN Installation Guide, and include the following steps:

1. During the procedures for uploading files to the eGate Repository using the Enterprise Manager, after uploading the **eGate.sar** file, select and upload the following file:
   - **SMEWebServices.sar** (to install the SME component)
   - **FileeWay.sar**
2. Continue installing the eGate Integrator as instructed in the *ICAN Installation Guide.*

2.3 Additional Files Required to Run SME

The SME supplies the following default JAR files that are required for binary encryption:

- local_policy.jar
- US_export_policy.jar

If you require encryption in DES3 format, then you must install updated JAR files that support this algorithm.

JAR files are located in the logicalhost at the following location:

logicalhost\jre\lib\security

Chapter 3

Encrypted Message Formats, Digital Signature Formats, and Certificate Formats

This chapter provides an overview of the encrypted message formats, digital signatures and certificates that are handled by SME. In addition, this chapter describes how to use Windows and Microsoft™ Internet Explorer tools to transfer certificate formats accepted by the SME.

3.1 Encrypted Message Formats

This section provides examples of encrypted message formats.

PKCS#7 encrypted message format

The PKCS#7 format, as specified by RFC 2315, is used for basic digitally signed and/or encrypted data. This format does not provide a MIME header, and produces mostly binary data, except for a few character strings in an embedded certificate, as shown in the following example:

```
0     *†H†÷  0  1,$0,  0^0,10UUS10\U

California1\0/UMonrovia1

0

U

STC10UDevelopment1'0%USTC Test Certificate Authority0*†H†÷
 V<±ïíà»,¯‡¾l-êÒTâž|g®<êÆ<õ¢\)Ç‰‡îQt£rµ»Ÿ½TûRP[Myß÷ ×ÚÚh-Íá-Ù¾—áô)Ã|bF©[_^HESM†2?k_
z¸~½ ï/ÈÕ+¶>æ³G¨šXK8yÃ!·Âyá—œB4U0 *†H†÷0*†H†÷b

4˜mDY  jE¯††,ë-]2žI¯e´G®†Ö¤ŸQÜ&ZÈX,¶Ê!4`RK”ÆE«9ýìÂPÝ Q- ní\=(-÷þÚïL
```

S/MIME2 encrypted message format (base64)

The S/MIME2 format is also used to represent digitally signed and/or encrypted data. This format provides a MIME header and encrypted results, with the binary data encoded as printable characters using the base64 method, as shown in the following example:

```
Content-Type: application/pkcs7-mime; name = "smime.p7m"
Content-Transfer-Encoding:base64
MIAGCSqGSIb3DQEHA6CAMIACAQAxggEkMIIBIAIBADCBiDCBgjELMAkGA1UEBhMCVVMxEzARBgNV
BAgTCkNhbGlmb3JuaWExETAPBgNVBAcTCE1vbnJvdmlhMQwwCgYDVQQKEwNTVEMxFDASBgNVBAsT
C0RldmVsb3BtZW50MScwJQYDVQQDEx5TVEMgVGVzdCBDZXJ0aWZpY2F0ZSBBdXRob3JpdHkCARMw
DQYJKoZIhvcNAQEBBQAEgYBR3Hwe+1JB2pZuR2XdNFS1DISYbgWHaXcmmpRZE+r35Ar5iaNlfRAj
ipc1RBW0HmidnWz3zBGYOml91btVjy2z6dmoDknnksgTI77YX727hESHgjCpxxcs+1kRzzI5ZUlU
WvvXeX/7wNkx3ZgJOrtIiXjfs6t8zW4edd1/13fQgjCABgkqhkiG9w0BBwEwFAYIKoZIhvcNAwcE
CBUeyy6UZb4koIAECOpD8MyUjNZ/BAjB0O2dStz8HgQIiPOI1H4tpfsECARjsNRDbMpqBAgtC3S1
7FnAWQQI8ymbLzoB4kUECF38LESRhXN2BAhcGnYwRqQDMgAAAAAAAAAAAA=
```

S/MIME2 encryption message format (binary)

This format represents a message as binary, non-printable data, with appropriate MIME headers, as shown in the following example:

```
Content-Type: application/pkcs7-mime; name = "smime.p7m"
Content-Transfer-Encoding:binary
0     *†H†÷  0  1,$0,  0^0,10UUS10\U
California1\0/UMonrovia10
U
STC10UDevelopment1'0%USTC Test Certificate Authority0*†H†÷
 V<±ïíà»,¯Qt£rµ»Ÿ½TûRP[Myß÷ ×ÚÚh-Íá-Ù¾—áô)Ã|bF©[_^HESM†2?k …Bmm_t1Gòz
~½ ï/ÈÕ+¶>æ³G¨šXK8yÃ!·Âyá—œB4U0 *†H†÷0*†H†÷b
4˜mDY  jE¯††,ë-]2žI¯e´G®†Ö¤ŸQÜ&ZÈX,¶Ê!4`RK”ÆE«9ýìÂPÝ Q- ní\=(-÷þÚïL
```

3.2 Digital Signature Formats

Although signatures normally are found attached to the message or file that they sign, detached signatures are also supported. A detached signature may be stored and transmitted separately from the message it signs.

Table 1 lists the features of each encrypted message format for attached signatures.

Table 1 Formats for attached signatures

PKCS#7 Format	S/MIME2 Format
▪ Includes original document in plain text, digital signature, and certificates involved, encapsulated, and encoded in Abstract Syntax Notation One (ASN.1) standard format. ***Note:*** *ASN.1 is an ISO/IEC standard for encoding rules used in ANSI X.509 certificates and PKCS documents.* **Example** 0 *†H†÷ 0 10+ 0 *†H†÷ $: This is only a test message! ,m0,i0,Ò0*†H†÷ 0,10UUS10\U California1\0/UMonrovia10 U STC10UDevelopment1'0%USTC Test Certificate Authority0020509184633Z030509184633Z0w10UUS10\U California1\0/UMonrovia10-U SeeBeyond1 0 URAD10USeeBeyond Test User 10Ÿ0*†H†÷ • 0% ®ŠGk•Éƒw¯¥S®¢_{!0Õ¢„&KÇéL›Ä,”1Än§lÏ»¶Õï¬©¥$lym´žÏ— ÍoÑLsuÉA#šk^# ü³ÅÅ§]ñsJAmf8ófsoU¢&mUþ„g,”>©kfÄXqÜ±Q½êÔú9P°KÍ~’ú“/ 0*†H†÷ _bšFïo7r ç¦«HêAßl“"zgÛæAÌœXú,‘Õ:Þ^=>P}°æå·ÌZ§R˜øüÅÌ(àØIãµ ÷Ñj #›òR1/”Œ80@ìûÍ,-/a†ÛZýý¥·s!ß¿ayS‘”#} …÷üç_"ëµÐÉµ4½¦1,-0,)0^0,10UUS10\U California1\0/UMonrovia10 U STC10UDevelopment1'0%USTC Test Certificate Authority0+ 0*†H†÷ ”Ö›»/éR8¶ZaÖ” ¡ÝXS*¿fuõURÑ^©pCËŸÂÍ,•Ÿ¶I/’{- ªÓIÊF62žSð ‡/ñI²e^ü#â„àðf·n(”aE±cÓ,Å¥>Ì°]2Åpæ2*Ì |êÏË{lÊ—0%#t<¥Œåœ ô> VÝ¹k	▪ Includes: ▪ MIME headers ▪ PKCS#7 attached signature object **Example** Content-Type: application/pkcs7-mime; name = "smime.p7m" Content-Transfer-Encoding:binary 0 *†H†÷ 0 10+ 0 *†H†÷ $: This is only a test message! ,m0,i0,Ò0*†H†÷ 0,10UUS10\U California1\0/UMonrovia10 U STC10UDevelopment1'0%USTC Test Certificate Authority0020509184633Z030509184633Z0w10UUS10\U California1\0/UMonrovia10-U SeeBeyond1 0 URAD10USeeBeyond Test User 10Ÿ0*†H†÷ • 0% ®ŠGk•Éƒw¯¥S®¢_{!0Õ¢„&KÇéL›Ä,”1Än§lÏ»¶Õï¬©¥$lym´žÏ— ÍoÑLsuÉA#šk^# ü³ÅÅ§]ñsJAmf8ófsoU¢&mUþ„g,”>©kfÄXqÜ±Q½êÔú9P°KÍ~’ú“/ 0*†H†÷ _bšFïo7r ç¦«HêAßl“"zgÛæAÌœXú,‘Õ:Þ^=>P}°æå·ÌZ§R˜øüÅÌ(àØIãµ ÷Ñj #›òR1/”Œ80@ìûÍ,-/a†ÛZýý¥·s!ß¿ayS‘”#} …÷üç_"ëµÐÉµ4½¦1,-0,)0^0,10UUS10\U California1\0/UMonrovia10 U STC10UDevelopment1'0%USTC Test Certificate Authority0+ 0*†H†÷ ”Ö›»/éR8¶ZaÖ” ¡ÝXS*¿fuõURÑ^©pCËŸÂÍ,•Ÿ¶I/’{- ªÓIÊF62žSð ‡/ñI²e^ü#â„àðf·n(”aE±cÓ,Å¥>Ì°]2Åpæ2*Ì |êÏË{lÊ—0%#t<¥Œåœ ô> VÝ¹k

Table 2 lists the features of each encrypted message format for detached signatures.

Table 2 Formats for detached signatures

PKCS#7 Format	S/MIME2 Format
▪ Includes signature and certificate without the signed data. ***Note:*** *RNIF1.1 uses PKCS#7 and detached format* **Example** 0 *†H†÷ 0 10+ 0 *†H†÷ ,m0,i0,Ò0*†H†÷ 0,10UUS10\U California1\0/UMonrovia10 U STC10UDevelopment1'0%USTC Test Certificate Authority0020509184633Z030509184633Z0w10UUS10\U California1\0/UMonrovia10-U SeeBeyond1 0 URAD10USeeBeyond Test User 10Ÿ0*†H†÷ • 0% ®ŠGk•Éƒw¯¥S®¢_{!0Õ¢„&KÇéL›Ä,”1Än§lÏ»¶Õï¬©¥$lym´žÏ– ÍoÑLsuÉA#šk^# ü³ÅÅ§]ñsJAmf8ófsoU¢&mUþ„g,”>©kfÄXqÜ±Q½êÔú9PªKÍ~’ú“/ 0*†H†÷ _bšFïo7r ç¦«HêAßl“"zgÛæAÌœXú,‘Õ:Þ^=>P}°æå·ÌZ§R˜øüÅÌ(àØIãµ ÷Ñj #›òR1/”Œ80@ìûÍ,-/a†ÛZýý¥·s!ß¿ayS‘”#} …÷üç_"ëµÐÉµ4½¦1,-0,)0^0,10UUS10\U California1\0/UMonrovia1 0 U STC10UDevelopment1'0%USTC Test Certificate Authority0+ 0*†H†÷ ”Ö>»/éR8¶ZaÖ” ¡ÝXS*¿fuõURÑ^©pCËŸÂÍ,•Ÿ¶I/’{- ªÓIÊF62žSð ‡/ñI²e^ü#â„àðf·n(”aE±cÓ,Å¥>Ì°]2åpÆ2*Ì |êÏË{lÊ–0%#t‹¥Œåœ ô› VÝ¹k .	▪ Includes a MIME multipart message consisting of the original data in one segment, and the binary format signature in a second segment. **Example** Content-Type: multipart/signed; protocol="application/pkcs7-signature"; micalg="SHA1"; boundary="Boundary_12e4421e_NOEWUDYA" --Boundary_12e4421e_NOEWUDYA Content-Type: text/plain This is only a test message! --Boundary_12e4421e_NOEWUDYA Content-Type: application/pkcs7-signature; name="smime.p7s" Content-Transfer-Encoding: binary Content-Disposition: attachment; filename=smime.p7s 0 *†H†÷ 0 10+ 0 *†H†÷ ,m0,i0,Ò0*†H†÷ 0,10UUS10\U California1\0/UMonrovia1 0 U STC10UDevelopment1'0%USTC Test Certificate Authority0020509184633Z030509184633Z0w10UUS10\U California1\0/UMonrovia10-U SeeBeyond1 0 URAD10USeeBeyond Test User 10Ÿ0*†H†÷ • 0% ®ŠGk•Éƒw¯¥S®¢_{!0Õ¢„&KÇéL›Ä,”1Än§lÏ»¶Õï¬©¥$lym´žÏ– ÍoÑLsuÉA#šk^# ü³ÅÅ§]ñsJAmf8ófsoU¢&mUþ„g,”>©kfÄXqÜ±Q½êÔú9PªKÍ~’ú“/ 0*†H†÷ _bšFïo7r ç¦«HêAßl“"zgÛæAÌœXú,‘Õ:Þ^=>P}°æå·ÌZ§R˜øüÅÌ(àØIãµ ÷Ñj #›òR1/”Œ80@ìûÍ,-/a†ÛZýý¥·s!ß¿ayS‘”#} …÷üç_"ëµÐÉµ4½¦1,-0,)0^0,10UUS10\U California1\0/UMonrovia1 0 U STC10UDevelopment1'0%USTC Test Certificate Authority0+ 0*†H†÷ ”Ö>»/éR8¶ZaÖ” ¡ÝXS*¿fuõURÑ^©pCËŸÂÍ,•Ÿ¶I/’{- ªÓIÊF62žSð ‡/ñI²e^ü#â„àðf·n(”aE±cÓ,Å¥>Ì°]2åpÆ2*Ì |êÏË{lÊ–0%#t‹¥Œåœ ô› VÝ¹k --Boundary_12e4421e_NOEWUDYA--

Table 2 Formats for detached signatures

PKCS#7 Format	S/MIME2 Format
	▪ Includes a MIME multipart message consisting of the original data in one segment, and the base64-encoded signature in a second segment **Example** Content-Type: multipart/signed; protocol="application/pkcs7-signature"; micalg="SHA1"; boundary="Boundary_12e4421e_FNGBRNRI" --Boundary_12e4421e_FNGBRNRI Content-Type: text/plain This is only a test message! --Boundary_12e4421e_FNGBRNRI Content-Type: application/pkcs7-signature; name="smime.p7s" Content-Transfer-Encoding: base64 Content-Disposition: attachment; filename=smime.p7s MIAGCSqGSIb3DQEHAqCAMIACAQExCzAJBgUrDgMCGgUAMIAGCSqG SIb3DQEHAQAAoIICbTCCAmkw ggHSAgETMA0GCSqGSIb3DQEBBAUAMIGCMQswCQYDVQQGEwJVUzET MBEGA1UECBMKQ2FsaWZvcm5p YTERMA8GA1UEBxMITW9ucm92aWExDDAKBgNVBAoTA1NUQzEUMBIG A1UECxMLRGV2ZWxvcG1lbnQx JzAlBgNVBAMTHlNUQyBUZXN0IENlcnRpZmljYXRlIEF1dGhvcml0 eTAeFw0wMjA1MDkxODQ2MzNa Fw0wMzA1MDkxODQ2MzNaMHcxCzAJBgNVBAYTAlVTMRMwEQYDVQQI EwpDYWxpZm9ybmlhMREwDwYD VQQHEwhNb25yb3ZpYTESMBAGA1UEChMJU2VlQmV5b25kMQwwCgYD VQQLEwNSQUQxHjAcBgNVBAMT FVNlZUJleW9uZCBUZXN0IFVzZXIgMTCBnzANBgkqhkiG9w0BAQEF AAOBjQAwgYkCgYEAropHa5XJ g3evpQFTrqJfeyEw1aKEJksfx+lMm8QsnTHEbqdsj8+7ttXvrKml JGx5bbSezzkIl81v0Uwfc3XJ QSOaA2teIxr8swvFDcWnXfFzSkFtkKM482Zzb1WiJhZtVf6EZywD nT6pAmujxFhx3LFRverU+jlQ ukvNfpL6ky8CAwEAATANBgkqhkiG9w0BAQQFAAOBgQBfE2IVmo9G 7xRvN3IZC+emq0jqE0HfbJMi eg1nf9vmQcycWBT6LJHVOt6IPZtQfbDmf+W3zFqnUpj4/ MUGzCjg2EnjtYD30Y9qI5vyF1IxL52M ODBA7PvNgq0vYYYY239a/f21t3Mh378dYXlTkZ0jfQmF9/ znXyLrtdDJtTS9pjGCAS0wggEpAgEB MIGIMIGCMQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5p YTERMA8GA1UEBxMITW9ucm92 aWExDDAKBgNVBAoTA1NUQzEUMBIGA1UECxMLRGV2ZWxvcG1lbnQx JzAlBgNVBAMTHlNUQyBUZXN0 IENlcnRpZmljYXRlIEF1dGhvcml0eQIBEzAHBgUrDgMCGjANBgkq hkiG9w0BAQEFAASBgJCd1gU+ uw/pUji2WmHWlCChE91YUyq/ o3X1VQVS0YipcEPLn8LNLI2ftkkvknuWqtNJykY2Mp5T8ICHFy/x SbJlXvwj4oTg8Ga3bgUdKJ1hRbFj0yzFpT7MsF0yxXDGMirMCnzq z8t7bMqBlzAlIw10i6WM5ZyA 9JsaH1bdE7lrAAAAAAAA --Boundary_12e4421e_FNGBRNRI--

3.3 Signing and Attaching Signatures

In an S/MIME message with a detached signature, the signature is calculated over on the entire payload data, in addition to its MIME header(s). The default Content-Type for such a MIME part is text/plain.

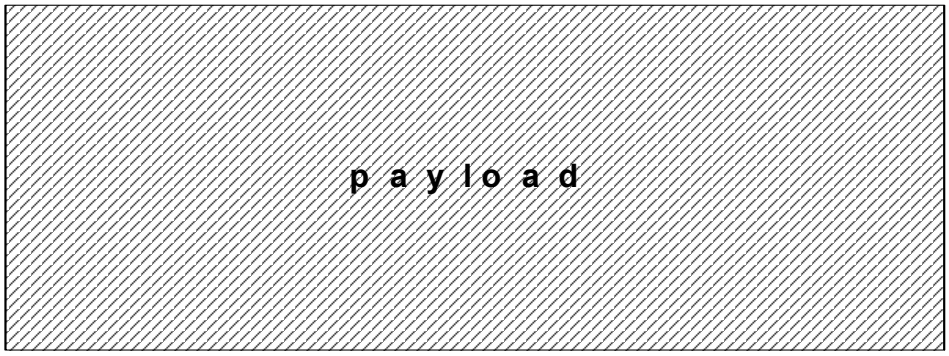
If signing a Content-Type other than text/plain, the user must generate a Content-Type header line for the payload. All other MIME headers and boundaries, including those of the detached signature part, are produced by SME.

An example XML message, digitally signed with a base64-encoded detached S/MIME signature is shown below.

```
MIME-Version: 1.0
Content-Type: multipart/signed;
protocol="application/x-pkcs7-signature"; micalg=sha1;
     boundary="----FA4D3A12E6192B82B05284F061C7CE55"

This is an S/MIME signed message

------FA4D3A12E6192B82B05284F061C7CE55
Content-Type: application/xml




------FA4D3A12E6192B82B05284F061C7CE55
Content-Type: application/x-pkcs7-signature; name="smime.p7s"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="smime.p7s"


------FA4D3A12E6192B82B05284F061C7CE55--
```

3.4 Private Key Format

Private keys, used by SME in the decryption and signing processes, are required to be in PKCS#12 format. If a key has been generated through a browser-based process and appears among your personal certificates in Microsoft Internet Explorer, it may be exported to a PKCS#12 file for use by SME. Procedures on converting and exporting certificate formats are included in section 3.5 of this chapter.

Note: *Remember the password you specify to encrypt the exported file; it is needed during the SME configuration process, in order to allow decryption and use the key.*

3.5 Certificate Formats

A Certificate, also called a Public Key Certificate, is an electronic message issued by a Certificate Authority that is used to match the value of the public key to the identity of the person, device, or service that holds the corresponding private key.

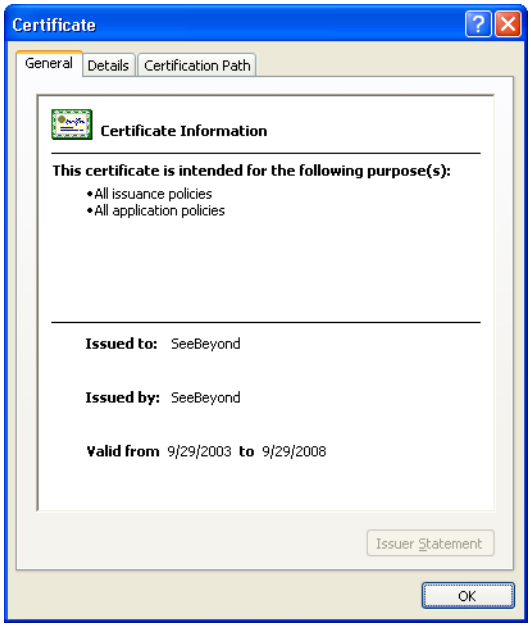
SME only accepts certificates in PKCS#7 format and DER encoded binary X.509.

Windows and Microsoft Internet Explorer provide a Certificate Wizard tool to convert between formats.

To convert from one certificate format to another

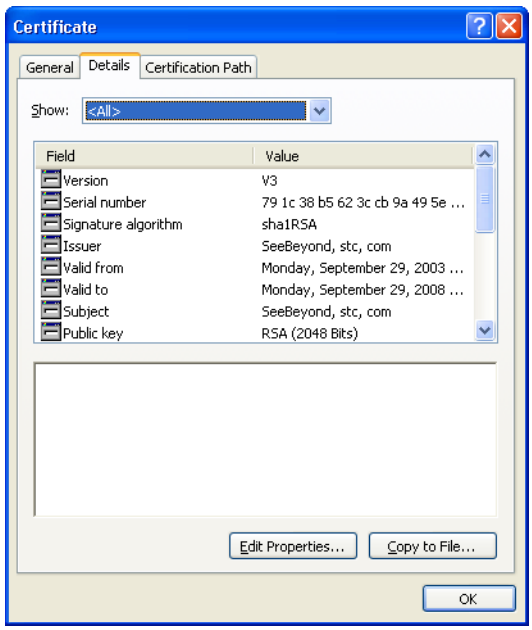
1 Double-click the certificate file to open the certificate properties, as shown in Figure 4.

Figure 4 Certificate File



2 Select the **Details** tab, as shown in **Figure 5 on page 24**.

Figure 5 Certificate Detail Tab

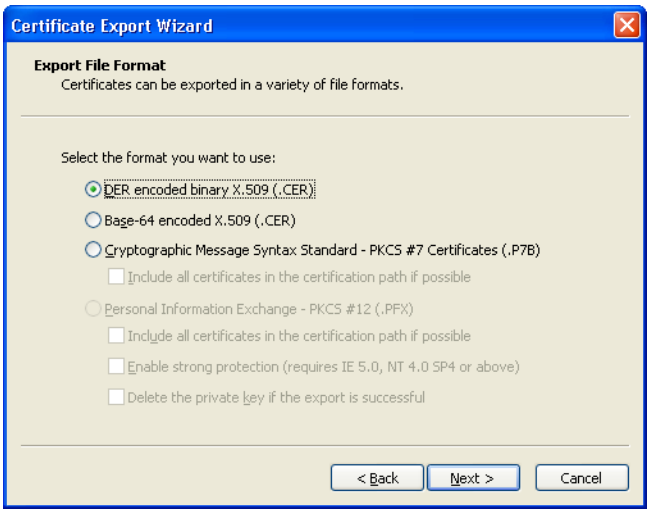


3 Click **Copy to File**. The Certificate Export Wizard appears, as shown in Figure 6.

Figure 6 Certificate Export Wizard

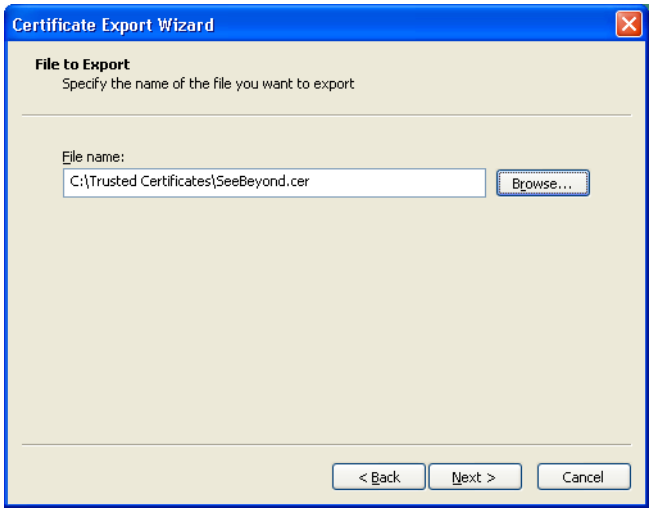
4 Click **Next** to open the certificate export file format, as shown in **Figure 7 on page 25**, and select the format.

Figure 7 File Format window



5 Click the **Next** button. The File to Export window appears, as shown in Figure 8.

Figure 8 File to Export window



6 Browse to and select the file name for the Certificate and choose the **Next** button. Details of the completed certificate appear, as shown in **Figure 9 on page 26**.

Figure 9 Completed Certificate Details window

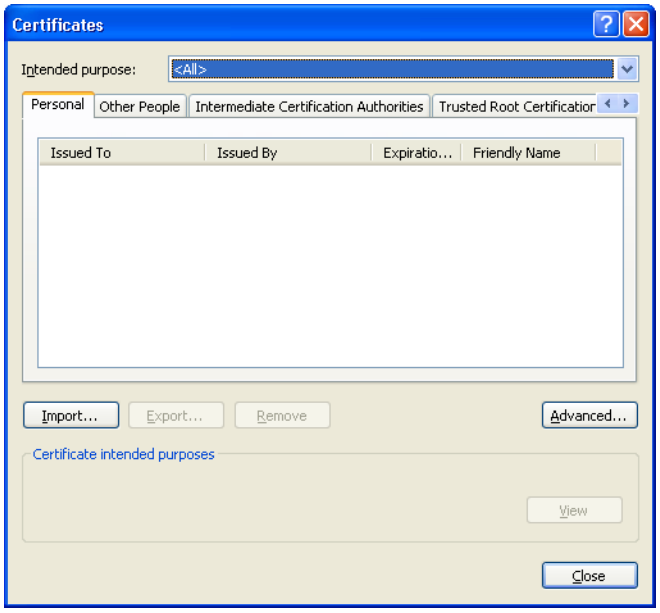
7 Click the **Finish** button to exit the Wizard.

To transfer the certificate formats using Microsoft Internet Explorer

1 From the Tools menu, click **Internet Options**.

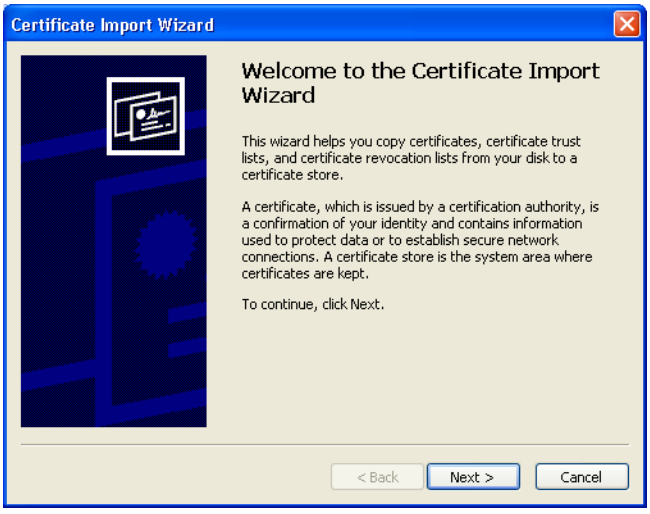
2 Click the **Content** tab and then click **Certificates**. The Certificates dialog appears, as shown in Figure 10.

Figure 10 Internet Explorer Certificates



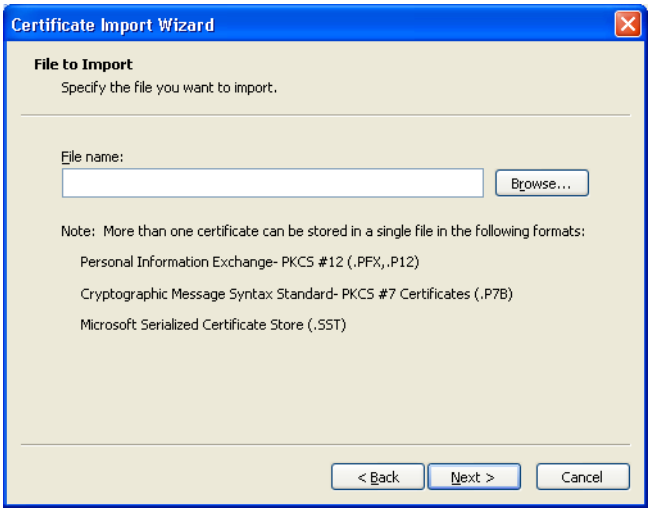
3 Click the **Import** button, the Certificate Import Wizard appears, as shown in **Figure 11 on page 27**.

Figure 11 Certificate Import Wizard



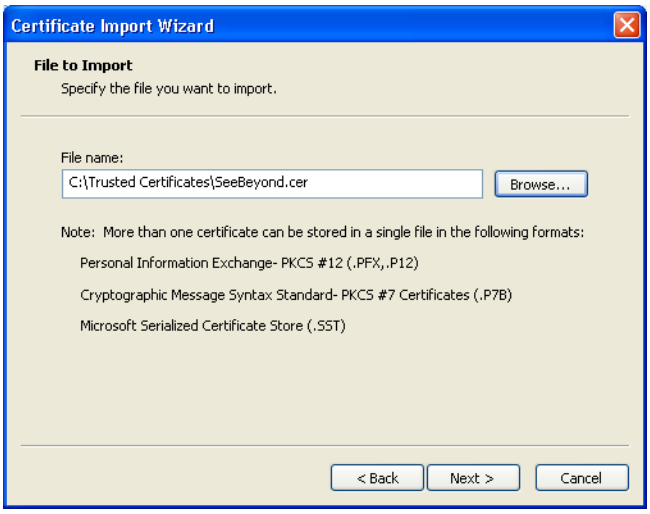
4 Click the **Next** button, the File to Import window appears, as shown in Figure 12.

Figure 12 File to Import window



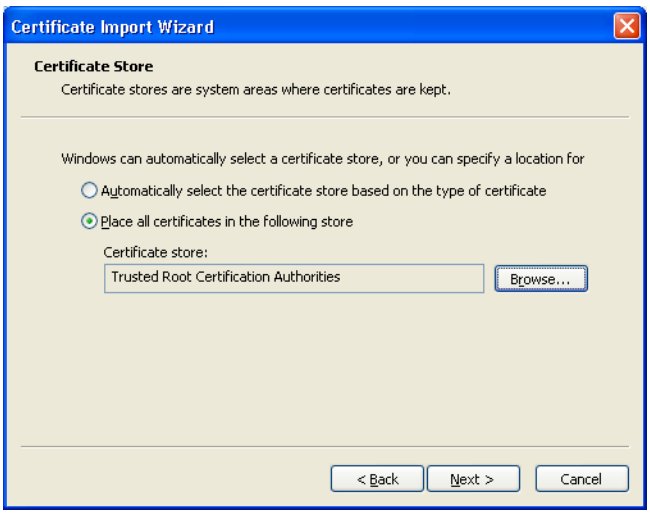
5 Click the **Browse** button and locate the certificate to open.

Figure 13 File to Import window



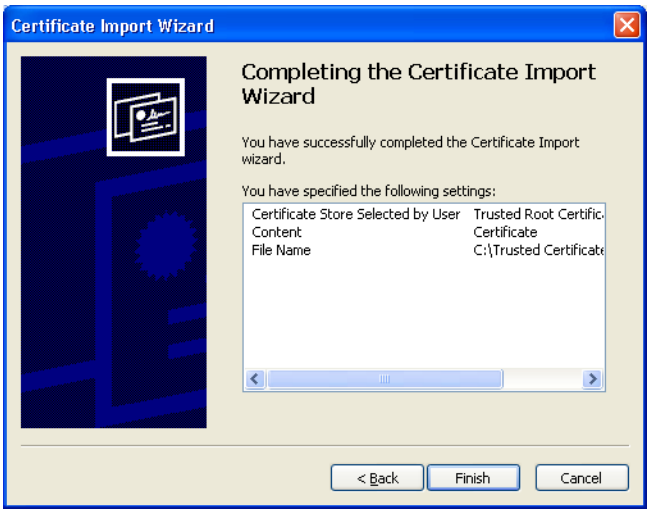
6 Click the **Next** button. The Certificate Store window appears, as shown in Figure 14.

Figure 14 Certificate Store window



7 Browse to the location where you want the certificate stored and click the **Next** button. Details of the completed certificate import appear, as shown in **Figure 15 on page 29**.

Figure 15 Completed Certificate Details window



8 Click the **Finish** button to exit the Wizard.

Chapter 4

Managing Keystores and Trueststores

This chapter describes the procedures for creating and managing Private Keys, Public Keys, and truststore certificates.

This Chapter Includes

- **"Overview" on page 30**
- **"Steps Required to Create and Manage Private Keys" on page 31**

4.1 Overview

Keystores are repositories for the sensitive cryptographic key information required for self-authentication. Key entries are typically private keys and are accompanied by the certificate chain for the corresponding public key.

Truststores hold all public key certificates belonging to the other party, or in the case with SME, the message sender. Certificates held in the Trust Store are considered Trusted Certificates since the Key Store owner trusts that the public key in the certificate belongs to the identity provided by the subject or owner of the certificate.

During runtime, one Keystore is created for each environment, but several truststores may exist to accommodate the different relationships between trading partners. ICAN commonly groups both Keystores and truststores under the common name "Keystore", however, both are considered separate entities.

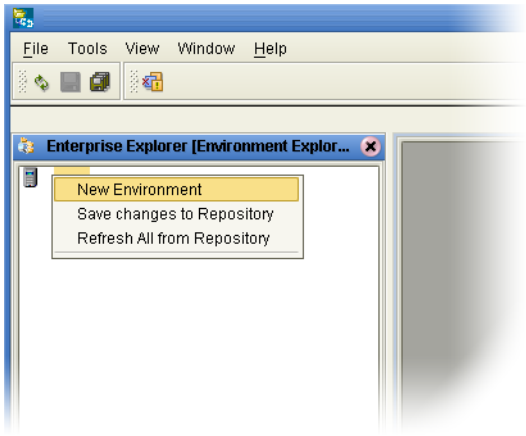
4.2 Steps Required to Create and Manage Private Keys

eGate Integrator includes Keystore and truststore management functionality. Using Environment Explorer, you first create a new Keystore environment, and then import or export private keys, create new truststores, and manage public certificates.

To Import a New Certificate:

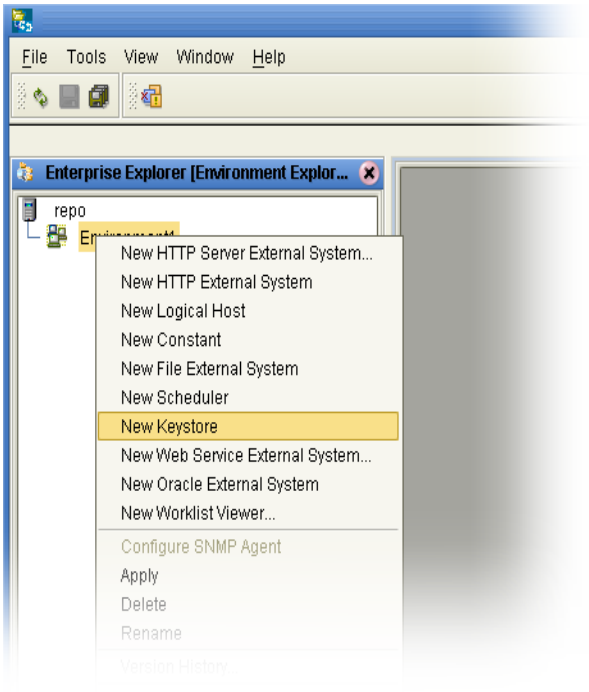
1 From the Environment Explorer, right-click the ican50 icon and choose **New Environment**.

Figure 16 Creating a New Environment



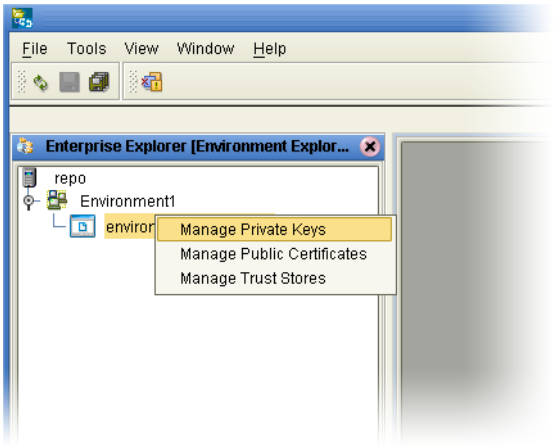
2 Right-click the new Environment and choose **New Keystore** from the selection menu, as shown in **Figure 17 on page 32**. This creates a new Keystore called Environment-ks-store.

Figure 17 New Keystore selection



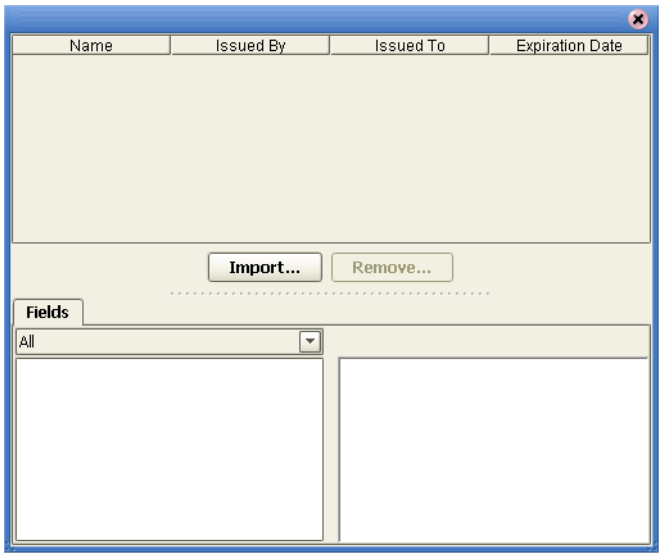
3 Right-click Environment-ks-store and choose **Manage Private Keys** from the selection menu.

Figure 18 Manage Private Keys



4 The **Manage Private Keys** window appears.

Figure 19 Manage Private Keys window

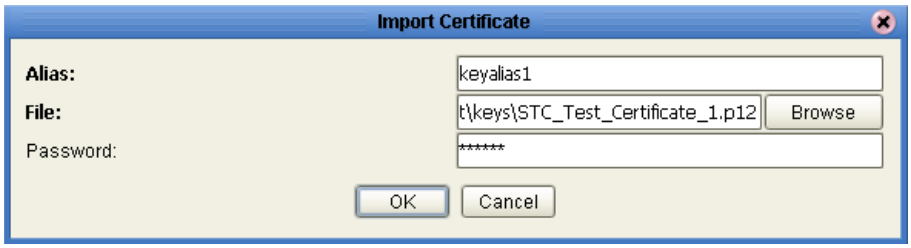


5 Click the **Import** button, the Import Certificate window appears.

Enter the following information:

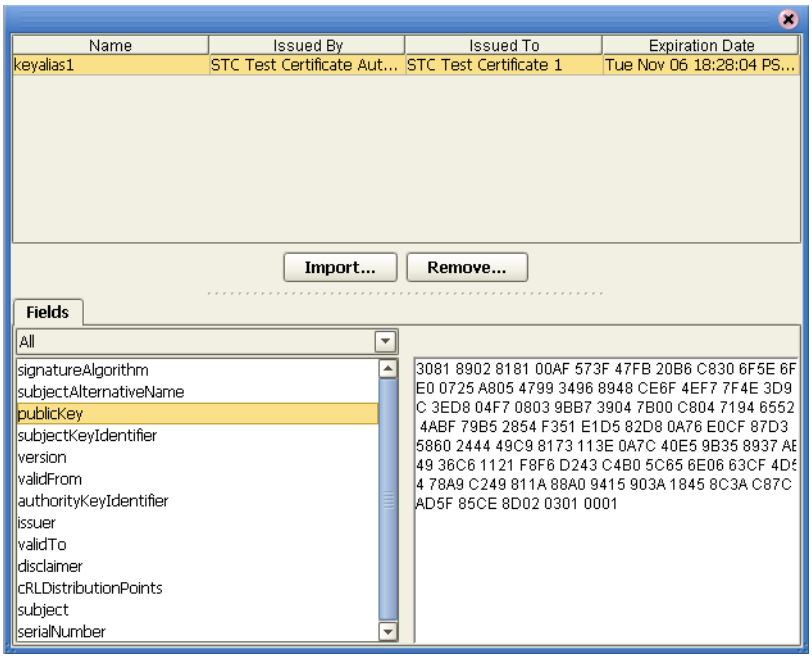
- **Alias**—the name you want associated with the certificate
- **File**—the location of the certificate
- **Password**—the password required to access the Private Key

Figure 20 Completed Import Certificate



6 Click the **OK** button, The **Key Pair Description** window appears, displaying the name and details of the imported certificate, as shown in **Figure 21 on page 34**.

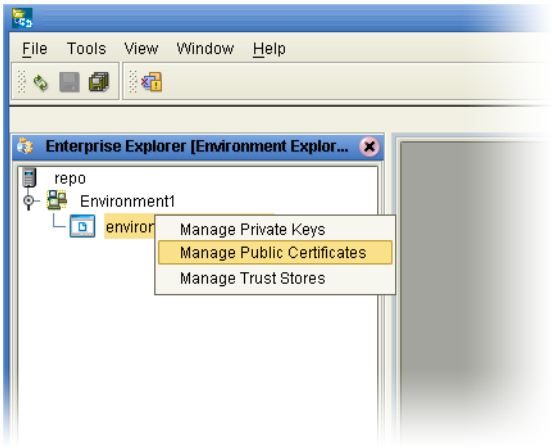
Figure 21 Key Pair Description window



To Manage a Public Certificate:

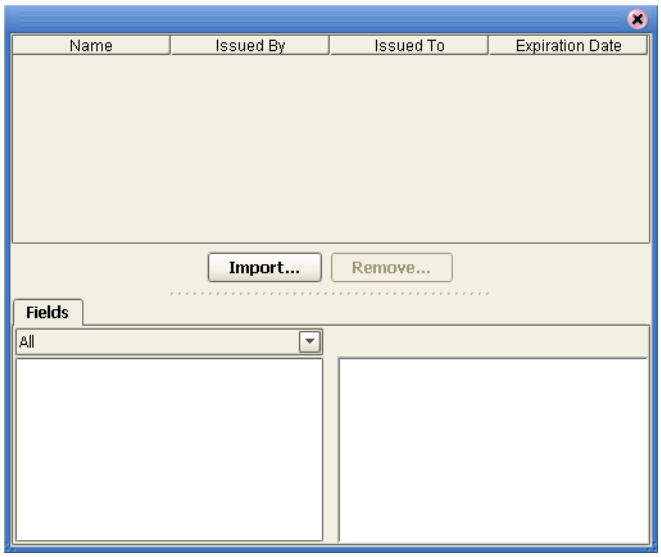
1 Right-click Environment-ks-store and choose **Manage Public Certificates** from the selection menu.

Figure 22 Manage Public Certificates



2 The **Manage Public Certificates** window appears, as shown in **Figure 23 on page 35**.

Figure 23 Manage Public Certificates window

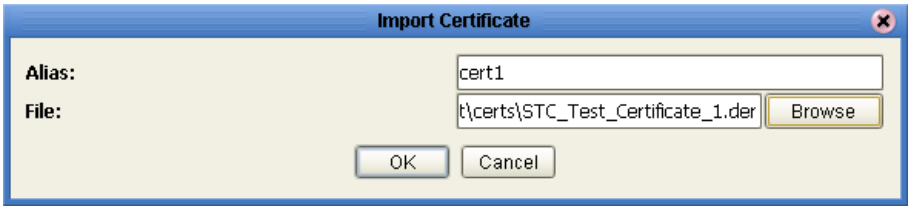


3 Click the **Import** button. The **Import Certificate** window appears.

Enter the following information:

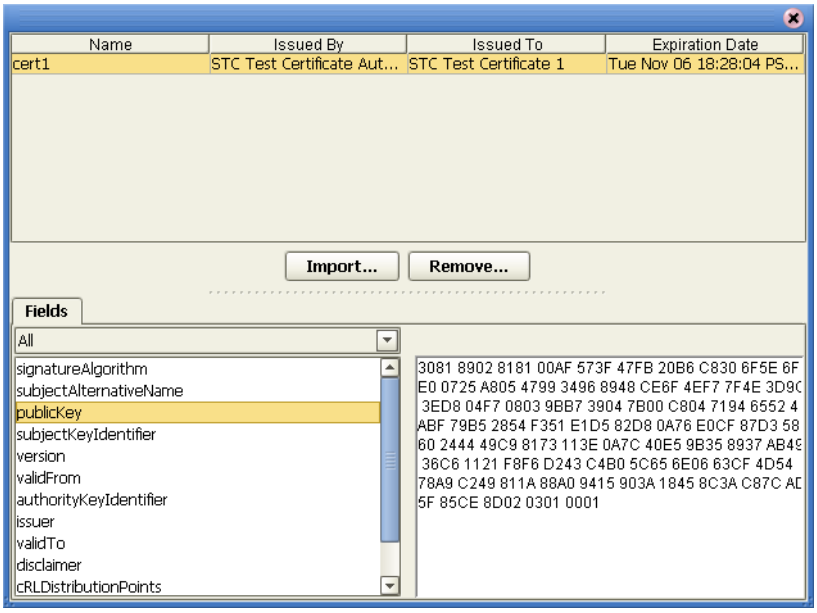
- **Alias**—the name you want associated with the certificate
- **File**—the location of the certificate

Figure 24 Import Certificate window



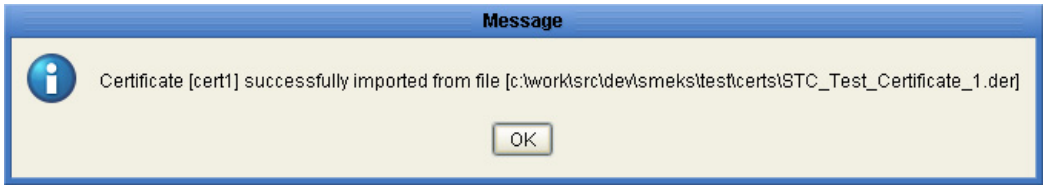
4 Click the OK button. The **Import Public Certificate Description** window appears, as shown in **Figure 25 on page 36**.

Figure 25 Import Public Certificate Description window



5 Click the Import button. A message appears confirming the import.

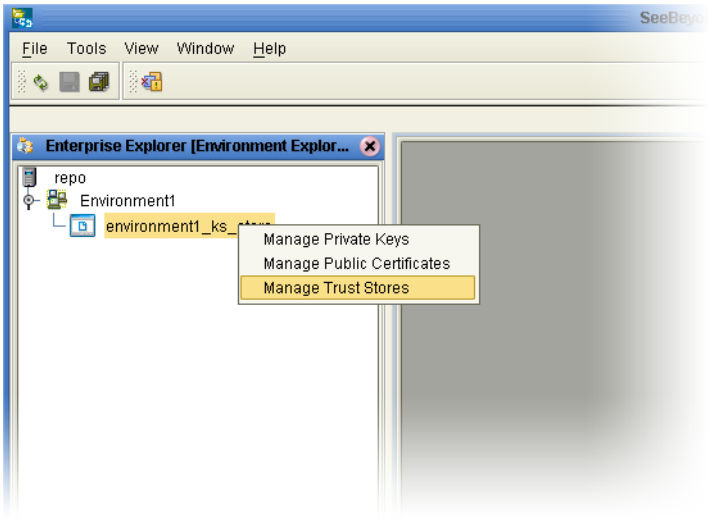
Figure 26 Import Confirmation Message



To Create a New Truststore:

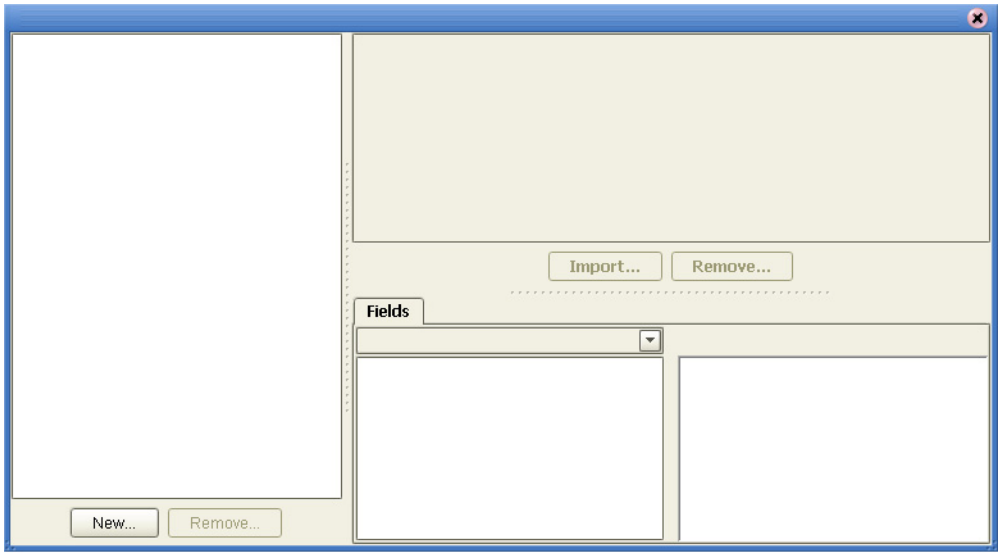
1 Right-click Environment-ks-store and choose **Manage Truststores** from the selection menu.

Figure 27 Manage Truststores



2 The Manage Truststore window appears.

Figure 28 Manage Truststore window



3 Click the New button. The New TrustStore window appears, as shown in **Figure 29 on page 38**

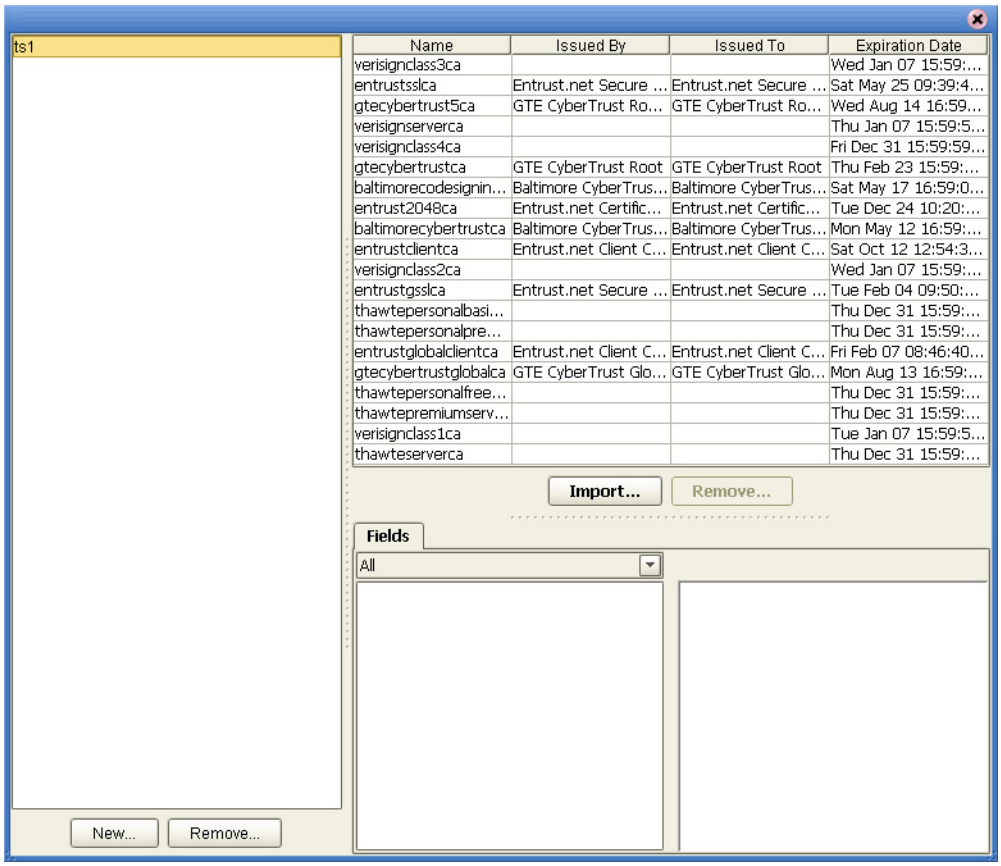
Figure 29 New Truststore

4 Enter an Alias to identify the truststore and click the **OK** button.

5 The Truststore Description window appears.

A number of trust certificates also appear in the right pane. These are industry known Trust Certificates loaded by default by the JDK.

Figure 30 Truststore Description window



To Import a Certificate into a Truststore

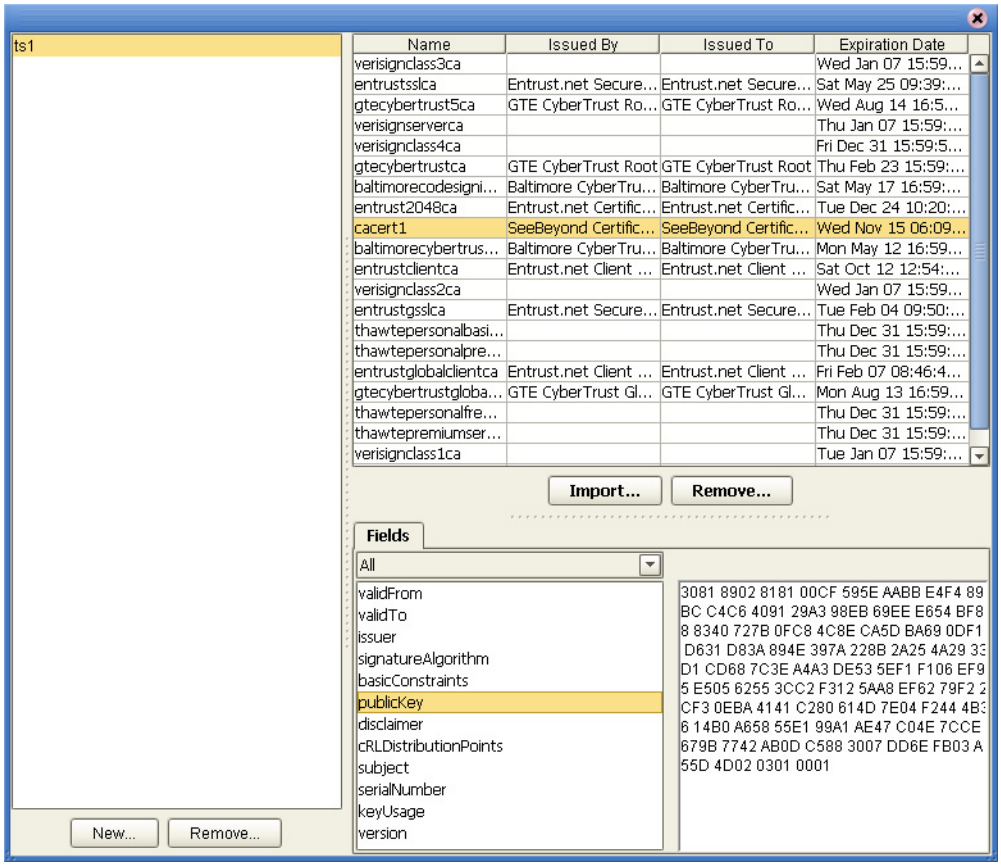
1 Click the **Import** button. The Import Certificate window appears.

2 Enter the Alias and File location of certificate and click the **OK** button.

3 A message appears confirming the import. Click the **OK** button.

4 The Manage Truststore Certificate Description window appears, containing the imported certificate.

Figure 31 Manage Truststore Certificate Description window



Chapter 5

S/MIME Collaboration Definitions

This chapter lists the various Collaboration Definitions and OTDs used in SME.

SME includes several completed Collaboration Definitions containing the encoded business rules used to compress, decrypt, and create digital signatures.

Every Collaboration Definition is also associated with both an input and an output OTD. The structure and rules defined in each OTD define the necessary data transformations required to complete each function. You select OTDs from the OTD Library, located on the root of the SME node in Enterprise Explorer.

5.1 SME Collaborations

Collaboration Definitions used in SME include:

- **CompressService Collaboration Definition:** used to compress data
- **DecompressService Collaboration Definition:** used to decompress data
- **EncryptService Collaboration Definition:** used to encrypt data
- **DecryptService Collaboration Definition:** used to decrypt data
- **SignService Collaboration Definition:** used to electonically sign data
- **VerifySignatureService Collaboration Definition:** used to verify electonically signed data.

5.2 Available OTDs

Several OTDs are available for use, including:

- SMIMECompressInput_SMIMECompressInput
- SMIMECompressOutput_SMIMECompressOutput
- SMIMEDecompressInput_SMIMEDecompressInput
- SMIMEDecompressOutput_SMIMEDecompressOutput
- SMIMEDecryptInput_SMIMEDecryptInput
- SMIMEDecryptOutput_SMIMEDecryptOutput
- SMIMEEncryptInput_SMIMEEncryptInput
- SMIMEEncryptOutput_SMIMEEncryptOutput
- SMIMESignInput_SMIMESignInput
- SMIMESignOutput_SMIMESignOutput
- SMIMEVerifyInput_SMIMEVerifyInput
- SMIMEVerifyOutput_SMIMEVerifyOutput

Chapter 6

Using the Sample Project in eGate

This chapter describes how to set the properties of the SME.

This Chapter Includes:

- **Importing the SME_Sample_Project** on page 42
- **Required Input Parameters** on page 43
- **Configuring the File eWays** on page 43
- **Configuring the JMS Clients** on page 44
- **Creating an Environment** on page 44
- **Creating and Activating the Deployment Profile** on page 44
- **Running the Project** on page 44

See the *SeeBeyond ICAN Suite Installation Guide* for additional information.

6.0.1 Importing the SME_Sample_Project

SME includes a Sample Project that you can use as a guide when building your own projects. Before you can use the Sample Project, you must import it into Enterprise Designer using the Enterprise Designer Project Import utility. The Sample Project (**SME_Messenging_Ext_Sample.zip**) file is located in the SME sample folder on the installation CD-ROM.

To import a sample eWay Project to the Enterprise Designer:

1. Extract the samples from the Enterprise Manager to a local file. Sample files are uploaded with the eWay's documentation .sar file and downloaded from the Enterprise Manager's Documentation tab.
2. From the Enterprise Designer's Project Explorer pane, right-click the Repository and select **Import Project** from the shortcut menu. The **Select File to Import** dialog box appears.
3. Browse to the directory that contains the sample project zip file. Select the **SME_Sample_Project.zip** sample file and click **Open**.
4. From the File Destination dialog box, select **Import to a new Project**, enter the name **SME_Sample_Project**, and click **OK**.
5. When the import has successfully completed, right-click the Repository and select **Refresh All from Repository** from the shortcut menu.

6 To run an imported sample Project, you must do the following:

- Create an **Environment**
- Configure the eWays for your specific system
- Create a **Deployment Profile**.

6.0.2 Required Input Parameters

The following tables detail input requirements for the encryption and decryption process. You enter these format requirements when you are creating your business rules.

For more information on how these requirements are used in the encryption and decryption process, see **Figure 1 on page 12**.

Table 3 SME Encryption Input Parameters

Requirement	Valid Values
Public certificate alias	any alphanumeric
Message Format	PKCS7 or SMIME
Encoding Format	base64 or binary
Encrypted Algorithm	RC2 or DES3

Table 4 SME Decryption Input Parameters

Requirement	Valid Values
Private Key Alias	any alphanumeric
Password of the key	any alphanumeric

6.0.3 Configuring the File eWays

The sample uses an inbound and an outbound File eWay. To configure the sample projects eWays, use the following information.

1 Double-click the **Inbound File eWay**, select **Inbound File eWay** in the Templates dialog box and click **OK**.

2 The **Parameters** dialog box opens to the Inbound File eWay configuration. Modify the configuration for your system, including the settings for the **Inbound** File eWay in Table 5, and click **OK**. The configuration settings are saved for the eWay.

Table 5 Inbound File eWay Settings

Inbound eWay Connection Parameters	
Directory	C:/temp
Input file name	Input*.txt

3 In the same way, modify the **Outbound** File eWay configuration for your system, including the settings in Table 6, and click **OK**.

Table 6 Outbound File eWay Settings

Outbound eWay Connection Parameters	
Directory	C:/temp
Output file name	output%.txt

6.0.4 Configuring the JMS Clients

When a Service is linked with a Queue (or Topic), the Enterprise Designer adds a JMS properties handle that facilitates the transfer and, if necessary, translation of data within the eGate system. JMS configuration properties must be configured in both the Connectivity Map and the Environment Explorer.

For more information on JMS configuration parameters see the *eGate Integrator User's Guide*.

6.0.5 Creating an Environment

Environments include the external systems, Logical Hosts, integration servers and message servers used by a Project and contain the configuration information for these components. Environments are created using the Enterprise Designer's Environment Explorer and Environment Editor.

To create the external environment for the Sample Project:

On the Environment Explorer, highlight and right-click the eWay profile. Select Properties. Enter the configuration information required for your Outbound eWay.

6.0.6 Creating and Activating the Deployment Profile

A Deployment Profile is used to assign Collaborations and message destinations to the integration server and message server. Deployment Profiles are created using the Deployment Editor.

To deploy a project, please see the *"eGate Integrator's User's Guide"*.

6.0.7 Running the Project

For instruction on how to run the Sample project, see the *"eGate Tutorial"*.

On the Environment Explorer, highlight and right-click the eWay profile. Select **Properties**. Enter the configuration information required for your Outbound eWay.

Index

A

Abstract Syntax Notation One (ASN.1) **19**
Alias **33**, **35**, **38**

B

base64 method **18**

C

Certificate Authority **11**
Certificate Wizard **23**
Certification Authority **13**
Collaboration Definitions **40**
components **6**
compression **8**

D

data integrity **10**
decompression **8**
decryption **8**
Deployment Profile **44**
DER **23**

E

eGate.sar **16**
encryption **8**

F

File **33**, **35**
FileeWay.sar **16**
Format
    IETF RFC 2311 **6**

G

gzip **14**

I

implementation **42**
installation
    Windows **15**
Internet Engineering Task Force **9**

J

JDK **38**
JMS Client
    properties **44**

K

Key Pair Description **33**
keypair **11**
Keystore **11**
Keystore - new **31**
Keystores **30**

M

Manage Public Certificates **34**
Manage Truststores **37**
MEWebServices.sar **16**
MIME **22**
MIME Message Body Format **9**

N

New Keystore **31**
non-ASCII **9**
non-repudiation **10**

O

OTD **40**
overview **6**

P

Password **33**
PKCS#12 **23**
PKCS#7 **14**, **17**, **23**
Private Key **33**
private key **11**
properties
    JMS Client **44**
Public Certificate **11**
Public Certificate alias **11**
Public Key Cryptography Standards (PKCS) **10**
Public Key Infrastructure (PKI) **8**

R

RFC 2315 **17**
RSA **10**
running a project **44**

S

S/MIME **9**, **10**, **22**
    introduction **9**
S/MIME2 **18**
Secure Messaging Extension
    introduction **8**
Sign Service **8**
Signature Verification Service **8**
SME
    introduction **8**
SMTP (E-mail) **9**
Supported Operating Systems **7**
system requirements **7**

T

TCP/IP **7**
Trust Certificates - default **38**
Truststore **38**
Truststores **30**

U

US-ASCII **9**

X

X.509 **23**
X.509 standard **10**
XML message **22**