SAS Publishing

Data Security Technologies in SAS® 9.1.2

The Power to Know®

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2004. *Data Security Technologies in SAS® 9.1.2*. Cary, NC: SAS Institute Inc.

Data Security Technologies in SAS® 9.1.2

ISBN 1–59047–494–5

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, April 2004

Contents

What’s New

Overview to Data Security Technologies

This document describes the technologies used by SAS to protect the confidentiality of data that is exchanged in client/server data transfers.

Details about Data Security Technologies

Data Security Technologies in SAS consolidates the information that was previously contained in multiple SAS documents. The data security technologies used by SAS is provided by:

- □ SASProprietary
- □ SAS/SECURE
- □ SSL (Secure Sockets Layer).

Note:

Transport Layer Security (TLS) is the successor to SSL V3.0. The Internet Engineering Task Force (IETF) adopted SSL V3.0 as the *de facto* standard and renamed it TLS. Throughout this document, any reference to SSL also applies to TLS.

PART 1

Data Security Technologies in SAS 9.1.2.

CHAPTER 1

Technologies for Data Security

Data Security Technologies: Overview

As e-business grows, there is a great need to ensure the confidentiality of business transactions over a network between an enterprise and its consumers, between enterprises, and within an enterprise. *Data security technologies* refers to the foundation SAS products and third-party strategies for protecting data and credentials (user IDs and passwords) that are exchanged in a networked environment. Fundamental to these technologies is the use of proven, industry-standard encryption algorithms for data protection.

Encryption is the transformation of intelligible data (plaintext) into an unintelligible form (ciphertext) by means of a mathematical process. The ciphertext is translated back to plaintext when the appropriate key that is necessary for decrypting (unlocking) the ciphertext is applied. Although encryption increases the protection of data, it does not prevent unauthorized access to data.

Authentication is the act of verifying the identity of an entity (such as a user). Authentication is used to confirm the authority of an entity to access protected resources. For details about authentication, see the documentation for your enterprise.

Data Security Technologies: Encryption Providers

SASProprietary Encryption Provider

SASProprietary is a fixed encoding algorithm that is included with Base SAS software. It requires no additional SAS product licenses. The SAS proprietary algorithm is strong enough to protect your data from casual viewing. SASProprietary provides a medium level of security. SAS/SECURE and SSL provide a high level of security.

SASProprietary System Requirements

SAS 9.1.2 supports SASProprietary under the following operating environments:

□ OpenVMS Alpha

□ UNIX

□ Windows

□ z/OS

SASProprietary Installation and Configuration

SASProprietary is part of Base SAS. Separate installation is not required.

For an example of configuring and using SASProprietary in your environment, see “SASProprietary for SAS/SHARE: Example” on page 22.

SAS/SECURE Encryption Provider

SAS/SECURE software is an add-on product that provides encryption algorithms in addition to the SASProprietary algorithm. SAS/SECURE requires a license, and it must be installed on each machine that runs a client and a server that will use the encryption algorithms. Although SAS/SECURE increases data security, it cannot completely prevent unauthorized access to your data.

SAS/SECURE System Requirements

SAS 9.1.2 supports SAS/SECURE under the following operating environments:

□ UNIX

  □ Compaq Tru64 UNIX

  □ HP UX on Itanium 64-bit platform

  □ HP UX on a 64-bit platform

  □ Linux for Intel Architecture on a 32-bit platform

  □ Solaris on a 64-bit platform

□ Windows

□ z/OS

Export Restrictions for SAS/SECURE

SAS/SECURE 9.1.2 is available to most commercial and government users inside and outside the U.S. However, some countries (for example, Russia, China, and France) have import restrictions on products that contain encryption, and the U.S. prohibits the export of encryption software to specific embargoed or restricted destinations.

SAS/SECURE for UNIX and z/OS includes the following encryption algorithms:

□ RC2 using 128-bit or 40-bit keys

□ RC4 using 128-bit or 40-bit keys

□ DES using 56-bit keys

□ TripleDES using 168-bit keys

SAS/SECURE for Windows uses the encryption algorithms that are available in Microsoft CryptoAPI. The level of the SAS/SECURE encryption algorithms under Windows depends on the level of the encryption support in Microsoft CryptoAPI under Windows. For this reason, SAS/SECURE for Windows has very few export restrictions.

SAS/SECURE Installation and Configuration

SAS/SECURE must be installed on the SAS server machine, the client machine, and possibly other machines, depending on the SAS software that requires encryption. For installation details, see the SAS documentation for the software that uses encryption.

For examples of configuring and using SAS/SECURE in your environment, see Chapter 3, "Data Security Technologies: Examples," on page 21.

Secure Sockets Layer (SSL) Encryption Provider

SSL is an abbreviation for Secure Sockets Layer, which is a protocol that provides network security and privacy. Developed by Netscape Communications, SSL uses encryption algorithms that include RC2, RC4, DES, TripleDES, IDEA, MD5, and others.

In addition to providing encryption services, SSL performs client and server authentication, and it uses message authentication codes to ensure data integrity. SSL is supported by both Netscape Navigator and Internet Explorer. Many Web sites use the protocol to protect confidential user information, such as credit card numbers. By convention, URLs that require an SSL connection begin with https: instead of http:. The SSL protocol is application independent and allows protocols such as HTTP, FTP, and Telnet to be transparently layered above it. SSL is optimized for HTTP. SSL includes software that was developed by the OpenSSL Project for use in the OpenSSL Toolkit. For more information see `www.OpenSSL.org`.

Note: Transport Layer Security (TLS) is the successor to SSL V3.0. The Internet Engineering Task Force (IETF) adopted SSL V3.0 as the *de facto* standard and renamed it TLS. Throughout this document, any reference to SSL also applies to TLS. △

SSL System Requirements

SAS 9.1.2 supports SSL under the following operating environments:

□ UNIX

□ Windows

SSL Concepts

Concepts that are fundamental to understanding SSL follow:

Certification Authorities (CAs)

Cryptography products provide security services by using digital certificates, public-key cryptography, private-key cryptography, and digital signatures. Certification authorities (CAs) create and maintain digital certificates, which also help preserve confidentiality.

Various commercial CAs, such as VeriSign and Thawte, provide competitive services for the e-commerce market. You can also develop your own CA by using products from companies such as RSA Security and Microsoft or from the Open Source Toolkit OpenSSL. From a trusted CA, members of an enterprise can obtain digital certificates to facilitate their e-business needs. The CA provides a variety of ongoing services to the business client that include handling digital certificate requests, issuing digital certificates, and revoking digital certificates.

Public and Private Keys

Public-key cryptography uses a public and a private key pair. The public key can be known by anyone, therefore, anyone can send a confidential message. The private key is confidential and known only to the owner of the key pair, therefore, only the owner can read the encrypted message. The public key is used primarily for encryption, but it can also be used to verify digital signatures. The private key is used primarily for decryption, but it can also be used to generate a digital signature.

Digital Signatures

A digital signature affixed to an electronic document or to a network data packet is like a personal signature that concludes a hand-written letter or that validates a credit card transaction. Digital signatures are a safeguard against fraud. A unique digital signature results from using a private key to encrypt a message digest. Receipt of a document that contains a digital signature enables the receiver to verify the source of the document. Electronic documents can be verified if you know where the document came from, who sent it, and when it was sent. Another form of verification comes from MACs, which ensure that a document has not been changed since it was signed.

Digital Certificates

Digital certificates are electronic documents that ensure the binding of a public key to an individual or an organization. Digital certificates provide protection from fraud.

Usually, a digital certificate contains a public key, a user's name, and an expiration date. It also contains the name of the Certification Authority (CA) that issued the digital certificate and a digital signature that is generated by the CA. The CA's validation of an individual or an organization allows that individual or organization to be accepted at sites that trust the CA.

SSL Installation and Configuration

SSL includes software that you can download from **`www.OpenSSL.org`** at no charge and can build for your site. The instructions that you use to install and configure SSL at your site depend on whether you use UNIX or Windows. For complete details, see Appendix 1, "Installing and Configuring SSL under UNIX," on page 33 or Appendix 2, "Installing and Configuring SSL under Windows," on page 39.

For examples of configuring and using SSL in your environment, see Chapter 3, "Data Security Technologies: Examples," on page 21.

Data Encryption Algorithms

The following encryption algorithms are used by the data security technologies:

RC2
is a block cipher that encrypts data in blocks of 64 bits. A *block cipher* is an encryption algorithm that divides a message into blocks and encrypts each block. The RC2 key size ranges from 8 to 256 bits. SAS/SECURE uses a configurable key size of 40 or 128 bits. (The NETENCRYPTKEYLEN= system option is used to configure the key length.) The RC2 algorithm expands a single message to a maximum of 8 bytes. RC2 is a proprietary algorithm developed by RSA Data Security, Inc.

Note: RC2 is supported in SAS/SECURE and SSL. △

RC4
is a stream cipher. A *stream cipher* is an encryption algorithm that encrypts data 1 byte at a time. The RC4 key size ranges from 8 to 2048 bits. SAS/SECURE uses a configurable key size of 40 or 128 bits. (The NETENCRYPTKEYLEN= system option is used to configure the key length.) RC4 is a proprietary algorithm developed by RSA Data Security, Inc.

Note: RC4 is supported in SAS/SECURE and SSL. △

DES (Data Encryption Standard)
is a block cipher that encrypts data in blocks of 64 bits by using a 56-bit key. The algorithm expands a single message to a maximum of 8 bytes. DES was originally developed by IBM but is now published as a U.S. Government Federal Information Processing Standard (FIPS 46-3).

Note: DES is supported in SAS/SECURE and SSL. △

TripleDES
is a block cipher that encrypts data in blocks of 64 bits. TripleDES executes the DES algorithm on a data block three times in succession by using a single, 56-bit key. This has the effect of encrypting the data by using a 168-bit key. TripleDES expands a single message to a maximum of 8 bytes. TripleDES is defined in the American National Standards Institute (ANSI) X9.52 specification.

Note: TripleDES is supported in SAS/SECURE and SSL. △

SASProprietary
is a cipher that provides basic fixed encoding encryption services under all operating environments that are supported by SAS. Included in Base SAS, SASProprietary does not require additional SAS product licenses. The algorithm expands a single message to approximately one-third by using a 32-bit key.

Note: SASProprietary is supported only by the SASProprietary encryption provider. △

IDEA (International Data Encryption Algorithm)
is a 64-bit iterative block cipher that uses a 128-bit key.

Note: IDEA is supported only in SSL. △

MD5 (Message Digest)
is used for digital signature applications in which a large message must be securely compressed before being signed with a private key. The MD2, MD4, and MD5 family of algorithms share common structures, however, each design is unique. MD2 was optimized for 8-bit machines; MD4 and MD5 were designed for

32-bit machines. MD5 produces a 128-bit message digest from a message of arbitrary length.

Note: MD5 is supported only in SSL. △

Data Security Technologies: Comparison

A comparison of the features of the data security technologies follow:

Table 1.1 Summary of SASProprietary, SAS/SECURE, and SSL Features

Features	SASProprietary	SAS/SECURE	SSL
License Required	No	Yes	No
Encryption and Authentication	Encryption only	Encryption only	Encryption and Authentication
Encryption Level	Medium	High	High
Algorithms Supported	SASProprietary Fixed Encoding	RC2, RC4, DES, TripleDES	RC2, RC4, DES, TripleDES, IDEA, MD5, and others
Installation Required	No (part of Base SAS)	Yes	Yes
Operating Environments Supported	UNIX Windows z/OS OpenVMS Alpha	UNIX Windows z/OS	UNIX Windows
SAS Version Support	8 and later	8 and later	9 and later

Data Security Technologies: Implementation

The implementation of the installed data security technology depends on the environment that you work in. If you work in a SAS enterprise intelligence infrastructure, data security might be transparent to you because it has already been configured into your site's overall security plan. After the data security technology has been installed, the site system administrator configures the encryption method (level of encryption) to be used in all client/server data exchanges. All enterprise activity uses the chosen level of encryption, by default. For an example, see “SAS/SECURE for the IOM Bridge: Examples” on page 28.

If you work in a SAS session on a client machine that exchanges data with a SAS server, you will specify SAS system options that implement data security for the duration of the SAS session. If you connect a SAS/CONNECT client to a spawner, you will specify encryption options in the spawner start-up command. For details about SAS system options, see Chapter 2, “SAS System Options for Data Security,” on page 9. For examples, see Chapter 3, “Data Security Technologies: Examples,” on page 21.

CHAPTER 2

SAS System Options for Data Security

NETENCRYPT System Option

Specifies whether client/server data transfers are encrypted.

Client: Optional

Server: Optional

Default: NONETENCRYPT

Valid in: configuration file, OPTIONS statement, SAS System Options window, SAS invocation

See also: NETENCRYPTALGORITHM

Category: Communications: Networking and Encryption

PROC OPTIONS Group= Communications

Syntax

NETENCRYPT | NONETENCRYPT

Syntax Description

NETENCRYPT
specifies that encryption is required.

NONETENCRYPT
specifies that encryption is not required, but is optional.

Details

The default for this option specifies that encryption is used if the NETENCRYPTALGORITHM option is set and if both the client and the server are capable of encryption. If encryption algorithms are specified but either the client or the server is incapable of encryption, then encryption is not performed.

Encryption might not be supported at the client or at the server if

- □ You are using a release of SAS (prior to Version 8) that does not support encryption.
- □ Your site (the client or the server) does not have a security software product installed.
- □ You specified encryption algorithms that are incompatible in SAS sessions on the client and the server.

NETENCRYPTALGORITHM= System Option

Specifies the algorithm(s) to be used for encrypted client/server data transfers.

Client: Optional

Server: Required

Alias: NETENCRALG=

Valid in: configuration file, OPTIONS statement, SAS System Options window, SAS invocation

See also: NETENCRYPT

Category: Communications: Networking and Encryption

PROC OPTIONS Group= Communications

Syntax

NETENCRYPTALGORITHM=*algorithm* | *(“algorithm-1”... “algorithm-n”)*

Syntax Description

algorithm* | *(“algorithm–1”... “algorithm-n”)
specifies the algorithm(s) that can be used for encrypting data that is transferred between a client and a server across a network. When you specify two or more encryption algorithms, use a space or a comma to separate them, and enclose the algorithms in parentheses.

The following algorithms may be used:

- □ RC2
- □ RC4
- □ DES
- □ TripleDES
- □ SASProprietary
- □ SSL.

Details

The NETENCRYPTALGORITHM= option must be specified in the server session.

Use this option to specify one or more encryption algorithms that you want to use to protect the data that is transferred across the network. If more than one algorithm is specified, the client session negotiates the first specified algorithm with the server session. If the server session does not support that algorithm, the second algorithm is negotiated, and so on.

If either the client or the server session specifies the NETENCRYPT option (which makes encryption mandatory) but a common encryption algorithm cannot be negotiated, the client cannot connect to the server.

If the NETENCRYPTALGORITHM= option is specified in the server session only, then the server's values are used to negotiate the algorithm selection. If the client session supports only one of multiple algorithms that are specified in the server session, the client can connect to the server.

There is an interaction between either NETENCRYPT or NONETENCRYPT and NETENCRYPTALGORITHM.

Table 2.1 Client/Server Connection Outcomes

Server Settings	**Client Settings**	**Connection Outcome**
NONETENCRYPT NETENCRYPTALGORITHM=*algorithm*	No settings	If the client is capable of encryption, the client/server connection will be encrypted. Otherwise, the connection will not be encrypted.
NETENCRYPT NETENCRYPTALGORITHM=*algorithm*	No settings	If the client is capable of encryption, the client/server connection will be encrypted. Otherwise, the client/server connection will fail.
No settings	NONETENCRYPT NETENCRYPTALGORITHM=*algorithm*	A client/server connection will not be encrypted.
No settings	NETENCRYPT NETENCRYPTALGORITHM=*algorithm*	A client/server connection will fail.
NETENCRYPT or NONETENCRYPT NETENCRYPTALGORITHM=*algorithm-a*	NETENCRYPTALGORITHM=*algorithm-b*	Regardless of whether NETENCRYPT or NONETENCRYPT is specified, a client/server connection will fail.

Example

In the following example, the client and the server specify different values for the NETENCRYPTALGORITHM= option.

The client specifies two algorithms in the following OPTIONS statement:

```
options netencryptalgorithm=(rc2 tripledes);
```

The server specifies three algorithms and requires encryption in the following OPTIONS statement:

```
options netencrypt netencryptalgorithm=(ssl des tripledes);
```

The client and the server negotiate an algorithm that they share in common, TripleDES, for encrypting data transfers.

NETENCRYPTKEYLEN= System Option

Specifies the key length to use for encrypted client/server data transfers.

Client: Optional

Server: Optional

Alias: NETENCRKEY=

Default: 0

Valid in: configuration file, OPTIONS statement, SAS System Options window, SAS invocation

Category: Communications: Networking and Encryption

PROC OPTIONS Group= Communications

Syntax

NETENCRYPTKEYLEN= 0 | 40 | 128

Syntax Description

0
specifies that the maximum key length that is supported at both the client and the server is used.

40
specifies a key length of 40 bits for the RC2 and RC4 algorithms.

128
specifies a key length of 128 bits for the RC2 and RC4 algorithms. If either the client or the server does not support 128-bit encryption, the client cannot connect to the server.

Details

The NETENCRYPTKEYLEN= option supports only the RC2 and RC4 algorithms. The DES, TripleDES, or SSL algorithms are not supported.

Using longer keys consumes more CPU cycles. If you do not need a high level of encryption, set NETENCRYPTKEYLEN=40 to decrease CPU usage.

SSLCALISTLOC= System Option

Specifies the location of digital certificates for trusted certification authorities (CA).

Client: Required

Server: Optional

Valid in: configuration file, OPTIONS statement, SAS System Options window, SAS invocation

Operating Environment: UNIX

Category: Communications: Networking and Encryption

PROC OPTIONS Group= Communications

Syntax

SSLCALISTLOC="*file–path*"

Syntax Description

"file-path"
specifies the location of a file that contains the digital certificates for the trusted certification authority (CA).

Details

The SSLCALISTLOC= option identifies the certification authority that SSL should trust. This option is required at the client because at least one CA must be trusted in order to validate a server's digital certificate. This option is required at the server only if the SSLCLIENTAUTH option is also specified at the server.

SSLCERTISS= System Option

Specifies the name of the issuer of the digital certificate that SSL should use.

Client: Optional

Server: Optional

Valid in: configuration file, OPTIONS statement, SAS System Options window, SAS invocation

Operating Environment: Windows

Category: Communications: Networking and Encryption

PROC OPTIONS Group= Communications

Syntax

SSLCERTISS="*issuer-of-digital-certificate*"

Syntax Description

"issuer-of-digital-certificate"
specifies the name of the issuer of the digital certificate that should be used by SSL.

Details

The SSLCERTISS= option is used with the SSLCERTSERIAL= option to uniquely identify a digital certificate from the Microsoft Certificate Store.

SSLCERTLOC= System Option

Specifies the location of the digital certificate that is used for authentication.

Client: Optional

Server: Required

Valid in: configuration file, OPTIONS statement, SAS System Options window, SAS invocation

Operating Environment: UNIX

Category: Communications: Networking and Encryption

PROC OPTIONS Group= Communications

Syntax

SSLCERTLOC=*"file-path"*

Syntax Description

"file-path"
specifies the location of a file that contains a digital certificate.

Details

The SSLCERTLOC= option is required for a server. It is required at the client only if the SSLCLIENTAUTH option is specified at the server.

SSLCERTSERIAL= System Option

Specifies the serial number of the digital certificate that SSL should use.

Client: Optional

Server: Optional

Valid in: configuration file, OPTIONS statement, SAS System Options window, SAS invocation

Operating Environment: Windows

Category: Communications: Networking and Encryption

PROC OPTIONS Group= Communications

Syntax

SSLCERTSERIAL=“*serial-number*”

Syntax Description

“serial-number”
specifies the serial number of the digital certificate that should be used by SSL.

Details

The SSLCERTSERIAL= option is used with the SSLCERTISS= option to uniquely identify a digital certificate from the Microsoft Certificate Store.

SSLCERTSUBJ= System Option

Specifies the subject name of the digital certificate that SSL should use.

Client: Optional

Server: Optional

Valid in: configuration file, OPTIONS statement, SAS System Options window, SAS invocation

Operating Environment: Windows

Category: Communications: Networking and Encryption

PROC OPTIONS Group= Communications

Syntax

SSLCERTSUBJ=“*subject-name*”

Syntax Description

“subject-name”
specifies the subject name of the digital certificate that SSL should use.

Details

The SSLCERTSUBJ= option is used to search for a digital certificate from the Microsoft Certificate Store.

SSLCLIENTAUTH System Option

Specifies whether a server should perform client authentication.

Server: Optional

Valid in: configuration file, OPTIONS statement, SAS System Options window, SAS invocation

Operating Environments: UNIX, Windows

Category: Communications: Networking and Encryption

PROC OPTIONS Group= Communications

Syntax

SSLCLIENTAUTH | NOSSLCLIENTAUTH

Syntax Description

SSLCLIENTAUTH
specifies that the server should perform client authentication.

NOSSLCLIENTAUTH
specifies that the server should not perform client authentication.

Details

Server authentication is always performed, but the SSLCLIENTAUTH option enables a user to control client authentication. This option is valid only when used on a server.

SSLCRLCHECK System Option

Specifies whether a Certificate Revocation List (CRL) is checked when a digital certificate is validated.

Client: Optional

Server: Optional

Valid in: configuration file, OPTIONS statement, SAS System Options window, SAS invocation

Operating Environments: UNIX, Windows

Category: Communications: Networking and Encryption

PROC OPTIONS Group= Communications

Syntax

SSLCRLCHECK|NOSSLCRLCHECK

Syntax Description

SSLCRLCHECK
specifies that CRLs are checked when digital certificates are validated.

NOSSLCRLCHECK
specifies that CRLs are not checked when digital certificates are validated.

Details

A Certificate Revocation List (CRL) is published by a Certification Authority (CA) and contains a list of revoked digital certificates. The list contains only the revoked digital certificates that were issued by a specific CA.

The SSLCRLCHECK option is required at the server only if the SSLCLIENTAUTH option is also specified at the server. Because clients check server digital certificates, this option is relevant for the client.

SSLCRLLOC= System Option

Specifies the location of a Certificate Revocation List (CRL).

Client: Optional

Server: Optional

Operating Environment: UNIX

Category: Communications: Networking and Encryption

PROC OPTIONS Group= Communications

Syntax

SSLCRLLOC=“*file-path*”

Syntax Description

“file-path”
specifies the location of a file that contains a Certificate Revocation List (CRL).

Details

The SSLCRLLOC= option is required only when the SSLCRLCHECK option is specified.

SSLPVTKEYLOC= System Option

Specifies the location of the private key that corresponds to the digital certificate.

Client: Optional

Server: Optional

Valid in: configuration file, OPTIONS statement, SAS System Options window, SAS invocation

Operating Environment: UNIX

Category: Communications: Networking and Encryption

PROC OPTIONS Group= Communications

Syntax

SSLPVTKEYLOC="*file-path*"

Syntax Description

"file-path"
specifies the location of the file that contains the private key that corresponds to the digital certificate that was specified by using the SSLCERTLOC= option.

Details

The SSLPVTKEYLOC= option is required at the server only if the SSLCERTLOC= option is also specified at the server.

SSLPVTKEYPASS= System Option

Specifies the password that SSL requires for decrypting the private key.

Client: Optional

Server: Optional

Valid in: configuration file, OPTIONS statement, SAS System Options window, SAS invocation

Operating Environment: UNIX

Category: Communications: Networking and Encryption

PROC OPTIONS Group= Communications

Syntax

SSLPVTKEYPASS="*password*"

Syntax Description

"password"
specifies the password that SSL requires in order to decrypt the private key. The private key is stored in the file that is specified by using the SSLPVTKEYLOC= option.

Details

The SSLPVTKEYPASS= option is required only when the private key is encrypted. OpenSSL performs key encryption.

Note: No SAS system option is available to encrypt private keys. △

CHAPTER

3

Data Security Technologies: Examples

SAS/SECURE for SAS/CONNECT: Example

SAS/CONNECT Client under UNIX

The following statements configure the client. The NETENCRYPTALGORITHM= option specifies the use of the RC4 algorithm.

```
options netencryptalgorithm=rc4;
options remote=unxnode comamid=tcp;
signon;
```

SAS/CONNECT Server under UNIX

The following command starts a spawner on the machine that runs the server. The -NETENCRYPT option specifies that encryption is required for all clients that connect to the spawner. The -NETENCRYPTALGORITHM option specifies the use of the RC4 algorithm for encrypting all network data. The -SASCMD option specifies the SAS start-up command.

```
sastcpd -service spawner -netencrypt -netencryptalgorithm rc4 -sascmd mystartup
```

The spawner executes a UNIX shell script that executes the commands to start SAS.

```
#!/bin/ksh
#_________________
# mystartup
#_________________
. ~/.profile
sas dmr -noterminal -comamid tcp $*
```

SASProprietary for SAS/SHARE: Example

SAS/SHARE Client

In this example, the NETENCRYPTALGORITHM= option is set to SASProprietary to specify the use of the proprietary algorithm to encrypt the data between the client and the server. The NETENCRYPTALGORITHM= option must be set before the LIBNAME statement establishes the connection to the server.

```
options netencryptalgorithm=sasproprietary;
options comamid=tcp;
libname sasdata 'edc.prog2.sasdata' server=rmthost.share1;
```

SAS/SHARE Server

This example shows how to set the options for encryption services on a SAS/SHARE server. The NETENCRYPT option specifies that encryption is required by any client that accesses this server. The NETENCRYPTALGORITHM= option specifies that the SASProprietary algorithm be used for encryption of all data that is exchanged with connecting clients.

```
options netencrypt netencryptalgorithm=sasproprietary;
options comamid=tcp;
proc server id=share1;
run;
```

SSL for a SAS/CONNECT UNIX Spawner: Example

Startup of a UNIX Spawner on a SAS/CONNECT Server

After digital certificates are generated for the CA, the server, and the client, and a CA trust list for the client is created, you can start a UNIX spawner program that runs on a server that SAS/CONNECT clients connect to.

For example:

```
% sastcpd -service unxspawn -netencryptalgorithm ssl
-sslcertloc /users/server/certificates/server.pem
-sslpvtkeyloc /users/server/certificates/serverkey.pem
-sslpvtkeypass starbuck1
-sslcalistloc /users/server/certificates/sas.pem
-sascmd /users/server/command.ksh
```

The following table explains the SAS commands that are used to start a spawner on a SAS/CONNECT single-user server.

Table 3.1 SAS Commands for Spawner Start-Up Tasks

SAS Command	Function
sastcpd	starts the spawner
-service unxspawn	specifies the spawner service (configured in the services file)
-netencryptalgorithm ssl	specifies the SSL encryption algorithm
-sslcertloc /users/server/certificates/server.pem	specifies the file path for the location of the server's certificate
-sslpvtkeyloc /users/server/certificates/ serverkey.pem	specifies the file path for the location of the server's private key
-sslpvtkeypass password	specifies the password to access the server's private key
-sslcalistloc /users/server/certificates/sas.pem	specifies the CA trust list
-sascmd /users/server/command.ksh	specifies the name of an executable file that starts a SAS session when you sign on without a script file.

An example of an executable file follows:

```
#!/bin/ksh
#----------------------------------
# mystartup
#----------------------------------

. ~/.profile
sas -dmr -noterminal $*
#------------------------------
```

For complete information about starting a UNIX spawner, see *Communications Access Methods for SAS/CONNECT and SAS/SHARE*.

Connection of a SAS/CONNECT Client to a UNIX Spawner

After a UNIX spawner is started on a SAS/CONNECT server, a SAS/CONNECT client can connect to it.

The following example shows how to connect a client to a spawner that is running on a SAS/CONNECT server:

```
options netencryptalgorithm=ssl;
options sslcalistloc="/users/johndoe/certificates/cacerts.pem";
```

```
%let machine=apex.server.com;
signon machine.spawner user=_prompt_;
```

The following explains the SAS options that are used to connect to a SAS/CONNECT server.

Table 3.2 Client Access to a SAS/CONNECT Server

SAS Options	Client Access Tasks
NETENCRYPTALGORITHM=ssl	specifies the encryption algorithm
SSLCALISTLOC=cacerts.pem	specifies the CA trust list
SIGNON=*server-ID.service*	specifies the server and service to connect to
USER=_PROMPT_	prompts for the user ID and password to be used for authenticating the client to the server

The server-ID and the server's Common Name, which was specified in the server's digital certificate, must be identical.

For complete information about connecting to a UNIX spawner, see *Communications Access Methods for SAS/CONNECT and SAS/SHARE*.

SSL for a SAS/CONNECT Windows Spawner: Example

Startup of a Windows Spawner on a Single-User SAS/CONNECT Server

After digital certificates for the CA, the server, and the client have been generated and imported into the appropriate Certificate Store, you can start a spawner program that runs on a server that SAS/CONNECT clients connect to.

An example of how to start a Windows spawner on a SAS/CONNECT server follows:

```
spawner -security -netencryptalgorithm ssl -sslcertsubj "apex.pc.com"
 -sascmd mysas.bat
```

The following table shows the SAS commands that are used to start a spawner on a SAS/CONNECT single-user server.

Table 3.3 SAS Commands for Spawner Start-Up Tasks

SAS Command	Function
spawner	starts the spawner
-security	specifies the requirement that a client provide a user name and password to access the spawner
-netencryptalgorithm ssl	specifies the SSL encryption algorithm

SAS Command	Function
-sslcertsubj "apex.pc.com"	specifies the subject name that is used to search for a certificate from the Microsoft Certificate Store
-sascmd mysas.bat	specifies the name of an executable file that starts a SAS session when you sign on without a script file

In order for the Windows spawner to locate the appropriate server digital certificate in the Microsoft Certificate Store, you must specify the -SSLCERTSUBJ system option in the script that is specified by the -SASCMD option. -SSLCERTSUBJ specifies the subject name of the digital certificate that SSL should use. The subject that is assigned to the -SSLCERTSUBJ option and the machine that is specified in the client signon must be identical.

If the Windows spawner is started as a service, the -SERVPASS and -SERVUSER options must also be specified in the Windows spawner start-up command in order for SSL to locate the appropriate CA digital certificate.

For complete information about starting a Windows spawner, see *Communications Access Methods for SAS/CONNECT and SAS/SHARE*.

Connection of a SAS/CONNECT Client to a Windows Spawner on a SAS/CONNECT Server

After a spawner has been started on a SAS/CONNECT server, a SAS/CONNECT client can connect to it.

An example of how to make a client connection to a Windows spawner that is running on a SAS/CONNECT server follows:

```
options comamid=tcp netencryptalgorithm=ssl;
%let machine=apex.pc.com;
signon machine user=_prompt_;
```

The machine that is specified in the client signon and the subject (the -SSLCERTSUBJ option) that is specified at the server must be identical.

The following shows the SAS options that are used to connect to a Windows spawner that runs on a SAS/CONNECT server.

Table 3.4 Client Access to a SAS/CONNECT Server

SAS Options	Function
COMAMID=tcp	specifies the TCP/IP access method
NETENCRYPTALGORITHM=ssl	specifies the encryption algorithm
SIGNON=server-ID	specifies which server to connect to
USER=_PROMPT_	prompts for the user ID and password to be used for authenticating the client to the server

The server-ID and the server's Common Name, which was specified in the server's digital certificate, must be identical.

SSL for SAS/SHARE under UNIX: Example

Startup of a Multi-User SAS/SHARE Server

After certificates for the CA, the server, and the client have been generated, and a CA trust list for the client has been created, you can start a SAS/SHARE server.

An example of how to start a secured SAS/SHARE server follows:

```
%let tcpsec=_secure_;
options netencryptalgorithm=ssl;
options sslcertloc="/users/johndoe/certificates/server.pem";
options sslpvtkeyloc="/users/johndoe/certificates/serverkey.pem";
options sslpvtkeypass="password";
proc server id=shrserv;
run;
```

The following lists the SAS option or statement that is used for each task to start a server.

Table 3.5 Server Start-Up Tasks

Server Start-Up Tasks	SAS Options and Statements
Secure the server	TCPSEC= _SECURE_
Specify SSL as the encryption algorithm	NETENCRALG=SSL
Specify the filepath for the location of the server's certificate	SSLCERTLOC=server.pem
Specify the filepath for the location of the server's private key	SSLPVTKEYLOC=serverkey.pem
Specify the password to access server's private key	SSLPVTKEYPASS="password"
Start the server	PROC SERVER ID=shrserv;

Note: As an alternative to using the SSLPVTKEYPASS= option to protect the private key, you might prefer that the private key remain unencrypted, and use the file system permissions to prevent read and write access to the file that contains the private key. To store the private key without encrypting it, use the –NODES option when requesting the certificate. △

SAS/SHARE Client Access of a SAS/SHARE Server

After a SAS/SHARE server has been started, the client can access it.

An example of how to make a client connection to a secured SAS/SHARE server follows:

```
options sslcalistloc="/users/johndoe/certificates/cacerts.pem";
%let machine=apex.server.com;
libname a '.' server=machine.shrserv user=_prompt_;
```

The following lists the SAS options that are used to access a SAS/SHARE server from a client.

Table 3.6 Tasks for Accessing a SAS/SHARE Server from a Client

Client Access Tasks	SAS Options
Specify the CA trust list	SSLCALISTLOC=cacerts.pem
Specify the machine and server to connect to	SERVER=machine.shrserv
Prompt for the user ID and password to be used for authenticating the client to the server	USER=_PROMPT_

The server-ID and the server's Common Name, which was specified in the server's certificate, must be identical.

SSL for SAS/SHARE under Windows: Examples

Startup of a Multi-User SAS/SHARE Server

After certificates for the CA, the server, and the client have been generated, and imported into the appropriate certificate store, you can start a SAS/SHARE server.

An example of how to start a secured SAS/SHARE server follows:

```
%let tcpsec=_secure_;
options comamid=tcp netencryptalgorithm=ssl;
options sslcertiss="Glenn's CA";
options sslcertserial="0a1dcfa3000000000015";
proc server id=shrserv;
run;
```

The following contains a list of tasks for starting a server and the SAS option or statement that is used for each task.

Table 3.7 Server Start-Up Tasks

Server Start-Up Tasks	SAS Options and Statements
Secure the server	TCPSEC= _SECURE_
Specify the TCP/IP access method	COMAMID=tcp
Specify SSL as the encryption algorithm	NETENCRALG=SSL
Specify the name of the issuer of the digital certificate that SSL should use.	SSLCERTISS="Glenn's CA"

Server Start-Up Tasks	SAS Options and Statements
Specify the serial number of the digital certificate that SSL should use.	SSLCERTSERIAL="0a1dcfa3000000000015"
Start the server	PROC SERVER ID=shrserv;

SAS/SHARE Client Access of a SAS/SHARE Server

After a SAS/SHARE server has been started, the client can access it.

An example of how to make a client connection to a secured SAS/SHARE server follows:

```
options comamid=tcp;
%let machine=apex.server.com;
libname a '.' server=machine.shrserv user=_prompt_;
```

The following contains a list of tasks for accessing a server from a client and the SAS option that is used for each task.

Table 3.8 Tasks for Accessing a SAS/SHARE Server from a Client

Client Access Tasks	SAS Options
Specify the TCP/IP access method	COMAMID=tcp
Specify the machine and server to connect to	SERVER=machine.shrserv
Prompt for the user ID and password to be used for authenticating the client to the server	USER=_PROMPT_

The server-ID and the server's Common Name, which was specified in the server's certificate, must be identical.

SAS/SECURE for the IOM Bridge: Examples

IOM Bridge Encryption Configuration

The IOM Bridge for SAS clients can use SAS/SECURE to encrypt network data between SAS and its clients.

SAS/SECURE must be installed at the SAS server and at the SAS client. SAS clients include COM clients and Java clients.

You can configure encryption properties in either metadata or in code.

□ "IOM Bridge for SAS Clients: Metadata Configuration" on page 28

□ "IOM Bridge for COM: Configuration in Code" on page 29

□ "IOM Bridge for Java: Configuration in Code" on page 30

IOM Bridge for SAS Clients: Metadata Configuration

In order to connect a SAS client to a SAS server, the CreateObjectByLogicalName function must obtain encryption information from metadata that is stored in the metadata repository. SAS Management Console can be used to configure encryption properties into the metadata repository, as follows:

Required encryption level
In SAS Management Console, follow this path:

```
<Connection> →  Options  → Encryption →  Required Encryption Level
```

Valid values for required encryption levels are:

None
No encryption

Credentials
Only user credentials (ID and password) are encrypted. This is the default.

Everything
All client/server transfers, including credentials, are encrypted.

Server encryption algorithm
In SAS Management Console, follow this path:

```
<Connection> →  Options  → Advanced Options → Encryption →
Server → Encryption Algorithms
```

Valid values for server encryption algorithms are: RC2, RC4, DES, TRIPLEDES, and SASPROPRIETARY (the default).

For complete details about using SAS Management Console to configure the IOM Bridge, visit **supportexp.unx.sas.com/rnd/itech/doc9/admin_oma/sasserver/iombridge**.

IOM Bridge for COM: Configuration in Code

When using the CreateObjectByServer function to connect a Windows client to a SAS server, specify the following properties in your client code in the ServerDef object:

□ BridgeEncryptionLevel

□ BridgeEncryptionAlgorithm

An example follows:

```
obServerDef.BridgeEncryptionLevel=EncryptAll;
obServerDef.BridgeEncryptionAlgorithm="TripleDes";
```

EncryptAll
causes all data, including credentials (user IDs and passwords), to be encrypted in client/server transfers.

TripleDes
is the specific encryption algorithm to be applied to data transfers.

For a complete list of encryption values, see the SAS Object Manager class reference (sasoman.chm).

IOM Bridge for Java: Configuration in Code

When using the BridgeServer object to connect a Java client to a SAS server, use the following functions to specify your encryption settings:

- □ setEncryptionContent
- □ setEncryptionAlgorithms
- □ setEncryptionPolicy

An example follows:

```
obBridgeServer.setEncryptionContent(BridgeServer.ENCRYPTION_CONTENT_ALL);
obBridgeServer.setEncryptionAlgorithms(BridgeServer.ENCRYPTION_ALGORITHM_TRIPLEDES);
obBridgeServer.setEncryptionPolicy(BridgeServer.ENCRYPTION_POLICY_REQUIRED);
```

ENCRYPTION_CONTENT_ALL
causes all data, including credentials (user ID and password), to be encrypted in client/server transfers.

ENCRYPTION_ALGORITHM_TripleDes
is the specific encryption algorithm to be applied to data transfers.

ENCRYPTION_POLICY_REQUIRED
specifies that encryption is required. If the server does not support encryption, the connection fails.

For a complete list of encryption values, see the Java reference for com.sas.services.connection at **www.support.sas.com**.

PART 2

Installing and Configuring SSL

APPENDIX 1

Installing and Configuring SSL under UNIX

SSL under UNIX: System and Software Requirements

The system and software requirements for using SSL under UNIX operating environments are:

- □ A computer that runs UNIX.
- □ Internet access and a Web browser such as Netscape Navigator or Internet Explorer.
- □ The TCP/IP communications access method.
- □ Access to the OpenSSL utility at **`www.openssl.org/source`** if you plan to use the OpenSSL CA.
- □ Knowledge of your site's security policy, practices, and technology. The properties of the digital certificates that you request are based on the security policies that have been adopted at your site.

Setting Up Digital Certificates for SSL under UNIX

Perform the following tasks to set up and use SSL:

1. Download and build SSL
2. Create a digital certificate request
3. Generate a digital certificate from the request
4. View digital certificates
5. End OpenSSL
6. Create a CA trust list for the SSL client application.

Step 1. Download and Build SSL

If you want to use OpenSSL as your trusted Certification Authority (CA), follow the instructions for downloading and building OpenSSL that are given at **www.openssl.org/source**. For complete documentation about the OpenSSL utility, visit **www.openssl.org/docs/apps/openssl.html**.

Information about alternative CAs and their Web sites follows:

□ For VeriSign, see **www.verisign.com**

□ For Thawte, see **www.thawte.com**

Step 2. Create a Digital Certificate Request

The tasks that you perform to request a digital certificate for the CA, the server, and the client are similar; however, the values that you specify will be different.

In this example, Proton, Inc. is the organization that is applying to become a CA by using OpenSSL. After Proton, Inc. becomes a CA, it can serve as a CA for issuing digital certificates to clients (users) and servers on its network.

Perform the following tasks:

1. Select the **apps** subdirectory of the directory where OpenSSL was built.
2. Initialize OpenSSL.

```
$ openssl
```

3. Issue the appropriate command to request a digital certificate.

Table A1.1 Open SSL Commands for Requesting a Digital Certificate

Request Certificate for	OpenSSL Command
CA	req -config ./openssl.cnf -new -out sas.req -keyout saskey.pem -nodes
Server	req -config ./openssl.cnf -new -out server.req -keyout serverkey.pem
Client	req -config ./openssl.cnf -new -out client.req -keyout clientkey.pem

Table A1.2 Arguments and Values Used in OpenSSL Commands

OpenSSL Arguments and Values	Functions
req	requests a certificate
-config ./openssl.cnf	specifies the storage location for the configuration details for the OpenSSL program

OpenSSL Arguments and Values	Functions
-new	identifies the request as new
-out sas.req	specifies the storage location for the certificate request
-keyout saskey.pem	specifies the storage location for the private key
-nodes	prevents the private key from being encrypted

4 Informational messages are displayed and prompts for additional information appear according to the specific request.

To accept a default value, press the Return key. To change a default value, type the appropriate information and press the Return key.

Note: Unless the -NODES option is used in the OpenSSL command when creating a digital certificate request, OpenSSL will prompt you for a password before allowing access to the private key. △

The following is an example of a request for a digital certificate:

```
OpenSSL> req -config ./openssl.cnf -new -out sas.req -keyout saskey.pem -nodes
Using configuration from ./openssl.cnf
Generating a 1024 bit RSA private key
............................++++++
..........................................++++++
writing new private key to 'saskey.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:
State or Province Name (full name) [North Carolina]:
Locality Name (city) [Cary]:
Organization Name (company) [Proton Inc.]:
Organizational Unit Name (department) [IDB]:
Common Name (YOUR name) []: proton.com
Email Address []:Joe.Bass@proton.com
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
OpenSSL>
```

The request for a digital certificate is complete.

Note: For the server, the Common Name must be the name of the machine that the server runs on; for example, apex.serv.com. △

Step 3. Generate a Digital Certificate from the Request

Perform the following tasks to generate a digital certificate for a CA, a server, and a client.

1 Issue the appropriate command to generate a digital certificate from the digital certificate request.

Table A1.3 OpenSSL Commands for Generating Digital Certificates under UNIX

Generate Certificate for	OpenSSL Command
CA	x509 req -in sas.req -signkey saskey.pem -out sas.pem
Server	ca -config ./openssl.cnf -in server.req -out server.pem
Client	ca -config ./openssl.cnf -in client.req -out client.pem

The functions performed by the OpenSSL arguments and values follow.

Table A1.4 Arguments and Values Used in OpenSSL Commands under UNIX

OpenSSL Arguments and Values	Functions
x509	identifies the certificate display and signing utility
req	specifies that a certificate be generated from the request
ca	identifies the Certification Authority utility
-config ./openssl.cnf	specifies the storage location for the configuration details for the OpenSSL utility
-in filename.req	specifies the storage location for the input for the certificate request
-out filename.pem	specifies the storage location for the certificate
-signkey saskey.pem	specifies the private key that will be used to sign the certificate that is generated by the certificate request

2 Informational messages are displayed and prompts for additional information appear according to the specific request.

To accept a default value, press the Return key. To change a default value, type the appropriate information, and press the Return key.

Sample dialog for creating a server digital certificate follows:

Note: The password is for the CA’s private key. △

```
Using configuration from ./openssl.cnf
Enter PEM pass phrase: password
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
countryName           :PRINTABLE:’US’
stateOrProvinceName   :PRINTABLE:’NC’
localityName          :PRINTABLE:’Cary’
organizationName      :PRINTABLE:’Proton, Inc.’
organizationalUnitName:PRINTABLE:’IDB’
commonName            :PRINTABLE:’proton.com’
Certificate is to be certified until Oct 16 17:48:27 2003 GMT (365 days)
Sign the certificate? [y/n]:y
1 out of 1 certificate requests certified, commit? [y/n]y
```

```
Write out database with 1 new entries Data Base Updated
```

The subject's Distinguished Name is obtained from the digital certificate request.

A root CA digital certificate is self-signed, which means that the digital certificate is signed with the private key that corresponds to the public key that is in the digital certificate. Except for root CAs, digital certificates are usually signed with a private key that corresponds to a public key that belongs to someone else, usually the CA.

The generation of a digital certificate is complete.

Step 4. View Digital Certificates

To view a digital certificate, issue the following command:

```
openssl> x509  -text -in filename.pem
```

A digital certificate contains data that was collected to generate the digital certificate timestamps, a digital signature, and other information. However, because the generated digital certificate is encoded (usually in PEM format), it is unreadable.

Step 5. End OpenSSL

To end OpenSSL, type **quit** at the prompt.

Step 6. Create a CA Trust List for the SSL Client Application

After generating a digital certificate for the CA, the server, and the client (optional), you must identify for the OpenSSL client application one or more CAs that are to be trusted. This list is called a *trust list*.

If there is only one CA to trust, in the client application, specify the name of the file that contains the OpenSSL CA digital certificate.

If multiple CAs are to be trusted, create a new file and copy-and-paste into it the contents of all the digital certificates for CAs to be trusted by the client application.

Use the following template to create a CA trust list:

```
Certificate for OpenSSL CA

-----BEGIN CERTIFICATE-----

<PEM encoded certificate>

-----END CERTIFICATE-----


Certificate for Keon CA

-----BEGIN CERTIFICATE-----

<PEM encoded certificate>

-----END CERTIFICATE-----
```

```
Certificate for Microsoft CA

-----BEGIN CERTIFICATE-----


-----END CERTIFICATE-----
```

Because the digital certificate is encoded, it is unreadable. Therefore, the content of the digital certificate in this example is represented as **<PEM encoded certificate>** . The content of each digital certificate is delimited with a **-----BEGIN CERTIFICATE-----** and **-----END CERTIFICATE-----** pair. All text outside the delimiters is ignored. Therefore, you might want to use undelimited lines for descriptive comments. In the preceding template, the file that is used contains the content of digital certificates for the CAs: OpenSSL, Keon, and Microsoft.

Note: If you are including a digital certificate that is stored in DER format, you must first convert it to PEM format. For more information, see “Converting between PEM and DER File Formats for SSL” on page 43. △

Converting between PEM and DER File Formats for SSL

By default, OpenSSL files are created in PEM (Privacy Enhanced Mail) format. SSL files that are created in Windows operating environments are created in DER (Distinguished Encoding Rules) format.

Under Windows, you can import a file that is created in either PEM or DER format. However, a digital certificate that is created in DER format must be converted to PEM format before it can be included in a trust list under UNIX.

An example of converting a server digital certificate from PEM input format to DER output format follows:

```
OpenSSL> x509 -inform PEM -outform DER -in server.pem -out server.der
```

An example of converting a server digital certificate from DER input format to PEM output format follows:

```
OpenSSL> x509 -inform DER -outform PEM -in server.der -out server.pem
```

APPENDIX

2

Installing and Configuring SSL under Windows

SSL under Windows: System and Software Requirements

The system and software requirements for using SSL under the Windows operating environment are:

□ A computer that runs Windows 2000 (or later).

□ Depending on your configuration, it might be useful to have access to the Internet and a Web browser such as Netscape Navigator or Internet Explorer.

□ The TCP/IP communications access method.

□ Microsoft Certificate Services add-on software.

□ If you will run your own CA, the Microsoft Certification Authority application (which is accessible from your Web browser).

□ For SAS/CONNECT, in order for a SAS/CONNECT client session to connect to a SAS/CONNECT server session via a Windows spawner using SSL encryption, ensure that the client session runs on a machine that has a Trusted CA Certificate.

The Windows spawner must run on a machine that has a Trusted CA Certificate and a Personal Certificate.

□ Knowledge of your site's security policy, practices, and technology. The properties of the digital certificates that you request will depend on the security policies that have been adopted at your site.

Setting Up Digital Certificates for SSL under Windows

Perform the following tasks to set up digital certificates for SSL:

□ Configure SSL

□ Request a digital certificate.

Step 1. Configure SSL

Complete information about configuring your Windows operating environment for SSL is contained in the Windows installation documentation and at **`www.microsoft.com`**.

The following keywords might be helpful when searching the Microsoft Web site:

- □ digital certificate services
- □ digital certificate authority
- □ digital certificate request
- □ site security planning.

Step 2. Request a Digital Certificate

The method of requesting a digital certificate depends on the CA that you use:

- □ the Microsoft Certification Authority
- □ a certification authority that is not Microsoft.

Request a Digital Certificate from the Microsoft Certification Authority

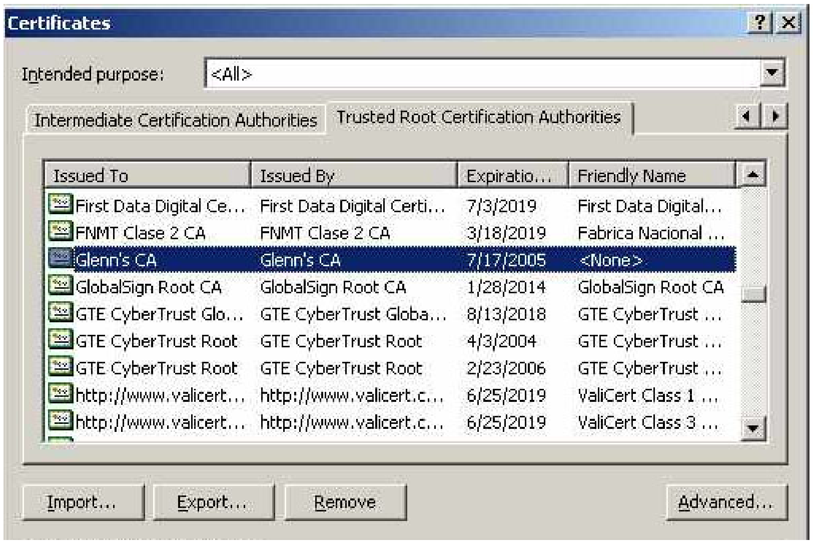
Perform the following tasks to request digital certificates that are issued by the Microsoft Certification Authority:

1. *System administrator*: If you are running your own CA, use Microsoft Certificate Services to create an active Certification Authority (CA).
2. *User*:

   a. Use the Certificate Request wizard to request a digital certificate from an active enterprise CA. The Certificate Request wizard lists all digital certificate types that the user can install.
   b. Select a digital certificate type.
   c. Select security options.
   d. Submit the request to an active CA that is configured to issue the digital certificate.

      After the CA issues the requested digital certificate, the digital certificate is automatically installed in the Certificate Store. The installed digital certificate is highlighted, as shown in Display A2.1 on page 41.

Display A2.1 Digital Certificate Installation in the Certificate Store


Certificates
? ×
Intermediate Certification Authorities
Trusted Root Certification Authorities
Import...
Export...
Remove
Advanced...

Intended purpose: <All>

Issued To	Issued By	Expiratio...	Friendly Name
First Data Digital Ce...	First Data Digital Certi...	7/3/2019	First Data Digital...
FNMT Clase 2 CA	FNMT Clase 2 CA	3/18/2019	Fabrica Nacional ...
Glenn's CA	Glenn's CA	7/17/2005	<None>
GlobalSign Root CA	GlobalSign Root CA	1/28/2014	GlobalSign Root CA
GTE CyberTrust Glo...	GTE CyberTrust Globa...	8/13/2018	GTE CyberTrust ...
GTE CyberTrust Root	GTE CyberTrust Root	4/3/2004	GTE CyberTrust ...
GTE CyberTrust Root	GTE CyberTrust Root	2/23/2006	GTE CyberTrust ...
http://www.valicert...	http://www.valicert.c...	6/25/2019	ValiCert Class 1 ...
http://www.valicert...	http://www.valicert.c...	6/25/2019	ValiCert Class 3 ...

Request a Digital Certificate from a Certification Authority That Is Not Microsoft

Users, perform the following tasks to request digital certificates that are not issued by the Microsoft CA:

1 Request a digital certificate from a CA.

2 Import the digital certificate to a Certificate Store by using the Certificate Manager Import wizard application from a Web browser.

A digital certificate can be generated by using the Certificate Request wizard or any third-party application that generates digital certificates.

Note: The Windows operating environment can import digital certificates that were generated in the UNIX operating environment. To convert from UNIX (PEM format) to Windows (DER format) before importing, see "Converting between PEM and DER File Formats for SSL" on page 43. △

For details about importing existing digital certificates, see "Import a Digital Certificate to a Certificate Store" on page 41.

Import a Digital Certificate to a Certificate Store

Digital certificates that were issued by a Certification Authority that is not Microsoft can be imported to an appropriate Certificate Store, as follows:

Certificate Type	Certificate Storage Location
Client	Personal Certificate Store
Server	Personal Certificate Store
CA (self-signed)	Trusted Root Certification Authorities

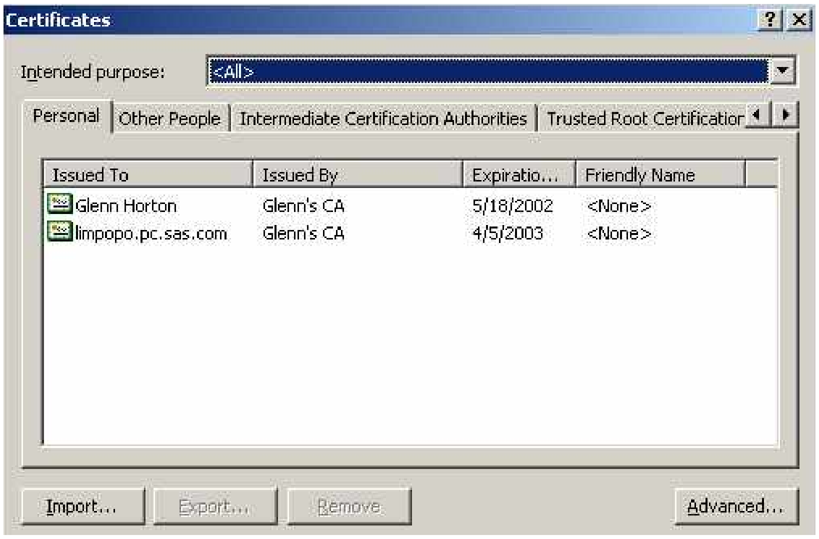
Perform the following tasks to import a digital certificate to a Certificate Store:

1 Access the Certificate Manager Import wizard application from your Web browser. From the **`Tools`** drop-down menu, select

```
Internet Options → Content tab → Certificates button
```

Specify the digital certificate to import to a Certificate Store by selecting the Personal tab in the Certificates window, as shown in Display A2.2 on page 42.

Display A2.2 Digital Certificate Selections for a Personal Certificate Store



Intended purpose: <All>

Issued To	Issued By	Expiratio...	Friendly Name
Glenn Horton	Glenn's CA	5/18/2002	<None>
limpopo.pc.sas.com	Glenn's CA	4/5/2003	<None>

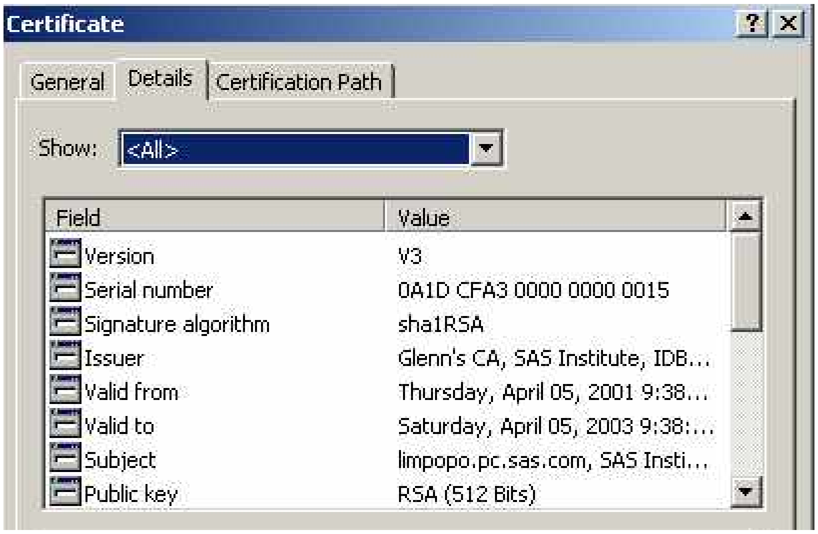
2 Click Import... and follow the instructions to import digital certificates.

Repeat this task in order to import the necessary digital certificates for the CA, the server, and the client, as appropriate.

3 After you have completed the selections for your personal Certificate Store, select the appropriate tab to view your selections.

4 To view the details about a digital certificate, select the digital certificate and click View . Typical results are shown in Display A2.3 on page 43.

Display A2.3 Digital Certificate Details Tab



Show: <All>

Field	Value
Version	V3
Serial number	0A1D CFA3 0000 0000 0015
Signature algorithm	sha1RSA
Issuer	Glenn's CA, SAS Institute, IDB...
Valid from	Thursday, April 05, 2001 9:38...
Valid to	Saturday, April 05, 2003 9:38:...
Subject	limpopo.pc.sas.com, SAS Insti...
Public key	RSA (512 Bits)

Converting between PEM and DER File Formats for SSL

By default, OpenSSL files are created in PEM (Privacy Enhanced Mail) format. SSL files that are created in Windows operating environments are created in DER (Distinguished Encoding Rules) format.

Under Windows, you can import a file that is created in either PEM or DER format. However, a digital certificate that is created in DER format must be converted to PEM format before it can be included in a trust list under UNIX.

An example of converting a server digital certificate from PEM input format to DER output format follows:

```
OpenSSL> x509 -inform PEM -outform DER -in server.pem -out server.der
```

An example of converting a server digital certificate from DER input format to PEM output format follows:

```
OpenSSL> x509 -inform DER -outform PEM -in server.der -out server.pem
```

Glossary

authentication
the process of verifying the identity of a person or process within the guidelines of a specific security policy.

block cipher
a type of encryption algorithm that divides a message into blocks and encrypts each block. See also stream cipher.

Certificate Revocation List
See CRL (Certificate Revocation List).

Certification Authority
a commercial or private organization that provides security services to the e-commerce market. A Certification Authority creates and maintains digital certificates, which help to preserve the confidentiality of an identity. Microsoft, VeriSign, and Thawte are examples of commercial Certification Authorities.

ciphertext
unintelligible data. See also encryption.

CRL (Certificate Revocation List)
a list of revoked digital certificates. CRLs are published by Certification Authorities (CAs), and a CRL contains only the revoked digital certificates that were issued by a specific CA.

cryptography
the science of encoding and decoding information to protect its confidentiality. See also encryption.

data security technologies
software features that protect data that is exchanged in client/server data transfers across a network.

DER (Distinguished Encoding Rules)
a format that is used for creating SSL files in Windows operating environments.

digital certificate
an electronic document that binds a public key to an individual or an organization. A digital certificate usually contains a public key, a user's name, an expiration date, and the name of a Certification Authority.

digital signature

a digital code that is appended to a message. The digital signature is used to verify to a recipient that the message was sent by a particular business, organization, or individual, and that the message has not been changed en route. The message can be any kind of file that is transmitted electronically.

encryption

the act of transforming intelligible data (plaintext) into an unintelligible form (ciphertext) by means of a mathematical process.

PEM (Privacy Enhanced Mail)

a format that is used for creating OpenSSL files.

plaintext

intelligible data. See also encryption, ciphertext.

private key

a number that is known only to its owner. The owner uses the private key to read (decrypt) an encrypted message. See also public key, encryption.

public key

a number that is associated with a specific entity such as an individual or an organization. A public key can be known by everyone who needs to have trusted interactions with that entity. A public key is always associated with a single private key, and can be used to verify digital signatures that were generated using that private key.

public-key cryptography

the science that uses public and private key pairs to protect confidential information. The public key can be known by anyone. The private key is known only to the owner of the key pair. The public key is used primarily for encryption, but it can also be used to verify digital signatures. The private key is used primarily for decryption, but it can also be used to generate a digital signature.

Secure Sockets Layer

See SSL (Secure Sockets Layer).

SSL (Secure Sockets Layer)

a protocol that uses encryption algorithms to provide network security and privacy.

stream cipher

a type of encryption algorithm that encrypts data one byte at a time. See also block cipher.

trust list

a file created by a user that contains the digital certificates for Certification Authorities, if more than one Certification Authority is used.

Index

Your Turn

If you have comments or suggestions about *Data Security Technologies in SAS® 9.1.2*, please send them to us on a photocopy of this page, or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Publishing
SAS Campus Drive
Cary, NC 27513
E-mail: **`yourturn@sas.com`**

For suggestions about the software, please return the photocopy to

SAS Institute Inc.
Technical Support Division
SAS Campus Drive
Cary, NC 27513
E-mail: **`suggest@sas.com`**