IBM i
Version 7.2

Security
Secure Sockets Layer

IBM

Note

Before using this information and the product it supports, read the information in "Notices" on page 35.

Contents

Secure Sockets Layer

This topic describes how to use Secure Sockets Layer (SSL) on your server.

Secure Sockets Layer (SSL) is an industry standard for enabling applications for secure communication sessions over an unprotected network, such as the Internet.

What's new for IBM i 7.2

Read about new or significantly changed information for Secure Sockets Layer.

What's new as of November 2016

The 3DES cipher suites should not be used due to the Sweet32 vulnerability. PTF MF62778 removes the 3DES cipher suite from the System SSL default eligible cipher suite list. For more information, see "SSL Cipher Suites" on page 6.

What's new as of July 2015

The SSLv3 protocol and RC4 cipher suites should not be used due to the POODLE and Bar Mitzvah vulnerabilities. PTF SI57320 removes SSLv3 and RC4 cipher suites from the System SSL defaults. For more information, see "SSL Protocols" on page 5 and "SSL Cipher Suites" on page 6.

- Added support for Transport Layer Security (TLS) protocols: TLS Version 1.2 and TLS Version 1.1
- New SSL protocols and cipher suites added to "System SSL Properties" on page 5 and introduced new properties for signature algorithms and handshake renegotiation.
- The ability to configure multiple certificates for a secure environment was added. Multiple Certificate Selection allows a secure environment to enable Elliptic Curve Digital Signature Algorithm (ECDSA) certificates while still allowing RSA certificates.
- Added support to "DCM Application Definitions" on page 12 for some System SSL attributes.
- Added client support for "Online Certificate Status Protocol" on page 16.

How to see what's new or changed

To help you see where technical changes have been made, the information center uses:

- The » image to mark where new or changed information begins.
- The « image to mark where new or changed information ends.

In PDF files, you might see revision bars (|) in the left margin of new and changed information.

To find other information about what's new or changed this release, see the Memo to users.

PDF file for SSL

You can view and print a PDF file of this information.

To view or download the PDF version of this document, select Secure Sockets Layer (SSL).

Saving PDF files

To save a PDF on your workstation for viewing or printing:

1. Right-click the PDF link in your browser.
2. Click the option that saves the PDF locally.
3. Navigate to the directory in which you want to save the PDF.

4. Click **Save**.

Downloading Adobe Reader

You need Adobe Reader installed on your system to view or print these PDFs. You can download a free copy from the Adobe Web site (www.adobe.com/products/acrobat/readstep.html).

SSL concepts

SSL concepts includes supplemental information, providing some basic building blocks for the Secure Sockets Layer (SSL) protocols.

With the SSL protocol, you can establish secure connections between clients and server applications which provide authentication of one or both endpoints of the communication session. SSL also provides privacy and integrity of the data that client and server applications exchange.

How SSL works

SSL is actually two protocols. The protocols are the record protocol and the handshake protocol. The record protocol controls the flow of the data between the two endpoints of an SSL session.

The handshake protocol authenticates one or both endpoints of the SSL session and establishes a unique symmetric key used to generate keys to encrypt and decrypt data for that SSL session. SSL uses asymmetric cryptography, digital certificates, and SSL handshake flows, to authenticate one or both endpoints of an SSL session. Typically, SSL authenticates the server. Optionally, SSL authenticates the client. A digital certificate, issued by a Certificate Authority, can be assigned to each of the endpoints or to the applications using SSL on each endpoint of the connection.

The digital certificate is comprised of a public key and some identifying information that a trusted Certificate Authority (CA) has digitally signed. Each public key has an associated private key. The private key is not stored with or as part of the certificate. In both server and client authentication, the endpoint which is being authenticated must prove that it has access to the private key associated with the public key contained within the digital certificate.

SSL handshakes are performance intensive operations because of the cryptographic operations using the public and private keys. After an initial SSL session has been established between two endpoints, the SSL session information for these two endpoints and applications can be cached in secure memory to speed up subsequent SSL session enablements. When an SSL session is resumed, the two endpoints use an abbreviated handshake flow to authenticate that each has access to unique information without using the public or private keys. If both can prove that they have access to this unique information, then new symmetric keys are established and the SSL session resumes. For TLS Version 1.2, 1.1, 1.0, and SSL Version 3.0 sessions, cached information does not remain in the secure memory for greater than 24 hours. You can minimize SSL handshake performance impacts on the main CPU by using cryptographic hardware.

Related information

Digital certificate concepts
Cryptographic hardware

Supported SSL and Transport Layer Security protocols

This topic describes which versions of the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols the IBM i implementation supports.

There are several versions of the SSL protocol defined. The IBM i implementation supports the following versions of the SSL and TLS protocols:

- TLS Version 1.2
- TLS Version 1.1
- TLS Version 1.0

- SSL Version 3.0
- SSL Version 2.0

Note:

1. Specifying more than one protocol at the same time is compatibility mode. Compatibility means that the highest specified protocol is negotiated if possible and if that is not possible then the next highest specified protocol is negotiated. If none of the specified protocols can be negotiated, the SSL handshake fails.
2. In compatibility mode, it is recommended to specify all protocols between the highest and lowest protocols enabled. Unpredictable results might happen if TLS Version 1.2 and TLS Version 1.0 are specified and TLS Version 1.1 is not.

TLS Version 1.2 versus TLS Version 1.1

The latest industry standard SSL protocol is Transport Layer Security (TLS) Version 1.2. Its specifications are defined by the Internet Engineering Task Force (IETF) in RFC 5246, The TLS Protocol Version 1.2.

TLS version 1.2 provides these enhancements over TLS version 1.1:

- All ciphers that are negotiated with TLSv1.2 must use at least SHA256. The existing ciphers that have SHA(1) in their name use SHA256.
- The MD5/SHA-1 combination in the digitally signed element is replaced with a single hash. Signed elements now include a field that explicitly specifies the hash algorithm used.
- Extension support is merged into the RFC rather than being defined separately.
- The DES cipher is not allowed. That means this cipher suite cannot be negotiated for TLSv1.2.
  - 09 = *RSA_DES_CBC_SHA

TLS Version 1.1 versus TLS Version 1.0

The second most recent industry standard SSL protocol is Transport Layer Security (TLS) Version 1.1. Its specifications are defined by the Internet Engineering Task Force (IETF) in RFC 4346, The TLS Protocol Version 1.1.

TLS version 1.1 provides these enhancements over TLS version 1.0:

- The implicit Initialization Vector (IV) is replaced with an explicit IV to protect against Cipher Block Chaining (CBC) attacks. The explicit IV changes the inner workings for the AES and DES ciphers.
- The export ciphers are not allowed. That means these two currently supported cipher suites cannot be negotiated for TLSv1.1.
  - 03 = *RSA_EXPORT_RC4_40_MD5
  - 06 = *RSA_EXPORT_RC2_CBC_40_MD5
- Miscellaneous internal improvements, see RFC 4346 for details

TLS Version 1.0 versus SSL Version 3.0

The first industry standard SSL protocol to be based on SSL version 3.0 was Transport Layer Security (TLS) Version 1.0. Its specifications are defined by the Internet Engineering Task Force (IETF) in RFC 2246, *The TLS Protocol.*

The major goal of TLS is to make SSL more secure and to make the specification of the protocol more precise and complete. TLS provides these enhancements over SSL Version 3.0:

- A more secure MAC algorithm
- More granular alerts
- Clearer definitions of "gray area" specifications

TLS provides the following security improvements:

- **Key-Hashing for Message Authentication** TLS uses Key-Hashing for Message Authentication Code (HMAC), which ensures that a record cannot be altered while traveling over an open network such as the Internet. SSL Version 3.0 also provides keyed message authentication, but HMAC is more secure than the (Message Authentication Code) MAC function that SSL Version 3.0 uses.
- **Enhanced Pseudorandom Function (PRF)** PRF generates key data. In TLS, the HMAC defines the PRF. The PRF uses two hash algorithms in a way which guarantees its security. If either algorithm is exposed, the data will remain secure as long as the second algorithm is not exposed.
- **Improved finished message verification** Both TLS Version 1.0 and SSL Version 3.0 provide a finished message to both endpoints that authenticates that the exchanged messages were not altered. However, TLS bases this finished message on the PRF and HMAC values, which again is more secure than SSL Version 3.0.
- **Consistent certificate handling** Unlike SSL Version 3.0, TLS attempts to specify the type of certificate which must be exchanged between TLS implementations.
- **Specific alert messages** TLS provides more specific and additional alerts to indicate problems that either session endpoint detects. TLS also documents when certain alerts should be sent.

SSL Version 3.0 versus SSL Version 2.0

SSL version 3.0 is an almost totally different protocol compared to SSL Version 2.0. Some of the major differences between the two protocols include:

- SSL Version 3.0 handshake protocol flows are different than SSL Version 2.0 handshake flows.
- SSL Version 3.0 includes a number of timing attack fixes and the SHA-1 hashing algorithm. The SHA-1 hashing algorithm is considered to be more secure than the MD5 hashing algorithm. SHA-1 allows SSL Version 3.0 to support additional cipher suites which use SHA-1 instead of MD5.
- SSL Version 3.0 protocol reduces man-in-the-middle (MITM) type of attacks from occurring during SSL handshake processing. In SSL Version 2.0, it was possible, though unlikely, that a MITM attack might accomplish cipher specification weakening. Weakening the cipher can allow an unauthorized person to break the SSL session key.

Related information

RFC 5246: "The Transport Layer Security (TLS) Protocol Version 1.2"

RFC 4346: "The Transport Layer Security (TLS) Protocol Version 1.1"

RFC 2246: "The TLS Protocol Version 1.0"

System SSL

System SSL is a set of generic services provided in the IBM i Licensed Internal Code (LIC) to protect TCP/IP communications using the SSL/TLS protocol. System SSL is tightly coupled with the operating system and the sockets code specifically providing extra performance and security.

System SSL is accessible to application developers from the following programming interfaces and JSSE implementation:

- Global Security Kit (GSKit) APIs
  - These ILE C APIs are accessible from other ILE languages
- Integrated IBM i SSL_ APIs
  - These ILE C APIs are accessible from other ILE languages
  - The use of this API set is not recommended. GSKit is the recommended C interface.
- Integrated IBM i JSSE implementation
  - The IBM i JSSE implementation is available for JDK 1.6 and JDK 7.

SSL applications created by IBM, IBM business partners, independent software vendors (ISV), or customers that use one of the three System SSL interfaces listed above will use System SSL. For example, FTP and Telnet are IBM applications that use System SSL. Not all SSL enabled applications running on IBM i use System SSL.

System SSL Properties
A subset of the System SSL attributes can have their properties modified at a system level. This subset of attributes is called System SSL Properties.

System SSL has many attributes that determine how secure environments or secure sessions are created. When an application is developed, the designer makes a choice for the value of each of the attributes. In some cases, the choice is to explicitly set an attribute value to a fixed value. For some attributes, the designer provides a user interface to allow the application administrator to control the attribute value. For a majority of attributes, the designer uses the System SSL default value for an attribute by not having code that changes it. The default attribute value is also used anytime new System SSL attributes are added after the application was compiled.

Like all attributes, the System SSL properties are limited by supported values and default values. Supported values limit the System SSL options for the attribute functionality. Default values determine what happens when the designer does not explicitly set the attribute.

The following System SSL properties can have their default values, supported values, or both changed. Use system values or the System Service Tools (SST) Advanced Analysis Command SSLCONFIG as specified.

Related concepts
The SSLCONFIG macro
The SSLCONFIG macro allows viewing or altering system-wide System SSL default properties.

Related information
SSL system value: QSSLPCL
SSL system value: QSSLCSLCTL
SSL system value: QSSLCSL

SSL Protocols
System SSL has the infrastructure to support multiple protocols.

The following protocols can be supported by System SSL:

- Transport Layer Security version 1.2 protocol (TLSv1.2)
- Transport Layer Security version 1.1 protocol (TLSv1.1)
- Transport Layer Security version 1.0 protocol (TLSv1.0)
- Secure Sockets Layer version 3.0 protocol (SSLv3)
- Secure Sockets Layer version 2.0 protocol (SSLv2)
  - SSLv2 cannot be used if TLSv1.2 is supported.

Shipped SSL Supported Protocols

System SSL is shipped with the following supported protocols:

- Transport Layer Security version 1.0 protocol (TLSv1.0)
- Transport Layer Security version 1.1 protocol (TLSv1.1)
- Transport Layer Security version 1.2 protocol (TLSv1.2)

Note: SSLv3 and SSLv2 are shipped as disabled for System SSL. The QSSLPCL system value can be used to disable or enable any of the protocols.

Shipped SSL Default Protocols

The following default protocols are used by System SSL when requested by an application:

- Transport Layer Security version 1.0 protocol (TLSv1)
- Transport Layer Security version 1.1 protocol (TLSv1.1)
- Transport Layer Security version 1.2 protocol (TLSv1.2)

The shipped default protocols can be changed by using System Service Tools (SST) Advanced Analysis Command SSLCONFIG.

Note: Removing a default protocol from the supported protocol list also removes it from the default protocol list.

Related concepts

The SSLCONFIG macro
The SSLCONFIG macro allows viewing or altering system-wide System SSL default properties.

Related information

SSL system value: QSSLPCL

SSL Cipher Suites

System SSL has the infrastructure to support multiple cipher suites.

The cipher suites are specified in different ways for each programming interface. The following cipher suites that are shown with the system value format, can be supported by System SSL:

- *RSA_AES_128_GCM_SHA256
- *RSA_AES_256_GCM_SHA384
- *ECDHE_ECDSA_NULL_SHA
- *ECDHE_ECDSA_RC4_128_SHA
- *ECDHE_ECDSA_3DES_EDE_CBC_SHA
- *ECDHE_RSA_NULL_SHA
- *ECDHE_RSA_RC4_128_SHA
- *ECDHE_RSA_3DES_EDE_CBC_SHA
- *ECDHE_ECDSA_AES_128_CBC_SHA256
- *ECDHE_ECDSA_AES_256_CBC_SHA384
- *ECDHE_RSA_AES_128_CBC_SHA256
- *ECDHE_RSA_AES_256_CBC_SHA384
- *ECDHE_ECDSA_AES_128_GCM_SHA256
- *ECDHE_ECDSA_AES_256_GCM_SHA384
- *ECDHE_RSA_AES_128_GCM_SHA256
- *ECDHE_RSA_AES_256_GCM_SHA384
- *RSA_AES_128_CBC_SHA256
- *RSA_AES_256_CBC_SHA256
- *RSA_NULL_SHA256
- *RSA_NULL_MD5
- *RSA_NULL_SHA
- *RSA_EXPORT_RC4_40_MD5
- *RSA_RC4_128_MD5
- *RSA_RC4_128_SHA
- *RSA_EXPORT_RC2_CBC_40_MD5
- *RSA_DES_CBC_SHA
- *RSA_3DES_EDE_CBC_SHA
- *RSA_AES_128_CBC_SHA

- *RSA_AES_256_CBC_SHA
- *RSA_RC2_CBC_128_MD5
- *RSA_DES_CBC_MD5
- *RSA_3DES_EDE_CBC_MD5

Shipped SSL supported cipher specification list

A cipher specification list contains a list of cipher suites. System SSL ships with 29 cipher suites supported. Administrators can control the ciphers that are supported by System SSL with system values QSSLCSL and QSSLCSLCTL. A cipher suite cannot be supported if the SSL protocol it requires is not also supported.

The following cipher suites are shipped as supported by System SSL:

- *ECDHE_ECDSA_AES_128_CBC_SHA256
- *ECDHE_ECDSA_AES_256_CBC_SHA384
- *ECDHE_ECDSA_AES_128_GCM_SHA256
- *ECDHE_ECDSA_AES_256_GCM_SHA384
- *RSA_AES_128_CBC_SHA256
- *RSA_AES_128_CBC_SHA
- *RSA_AES_256_CBC_SHA256
- *RSA_AES_256_CBC_SHA
- *RSA_AES_128_GCM_SHA256
- *RSA_AES_256_GCM_SHA384
- *ECDHE_RSA_AES_128_CBC_SHA256
- *ECDHE_RSA_AES_256_CBC_SHA384
- *ECDHE_RSA_AES_128_GCM_SHA256
- *ECDHE_RSA_AES_256_GCM_SHA384
- *ECDHE_ECDSA_3DES_EDE_CBC_SHA
- *ECDHE_RSA_3DES_EDE_CBC_SHA
- *RSA_3DES_EDE_CBC_SHA
- *ECDHE_ECDSA_RC4_128_SHA
- *ECDHE_RSA_RC4_128_SHA
- *RSA_RC4_128_SHA
- *RSA_RC4_128_MD5
- *RSA_DES_CBC_SHA
- *RSA_EXPORT_RC4_40_MD5
- *RSA_EXPORT_RC2_CBC_40_MD5
- *ECDHE_ECDSA_NULL_SHA
- *ECDHE_RSA_NULL_SHA
- *RSA_NULL_SHA256
- *RSA_NULL_SHA
- *RSA_NULL_MD5

The supported cipher specification list is affected by the SSL protocols that are supported by the system and by changes that are made to the system value QSSLCSL. You can display the value of QSSLCSL to see the cipher specification list on your system.

Shipped SSL default cipher specification list

The following displays the order of the shipped default cipher specification list:

- *ECDHE_ECDSA_AES_128_CBC_SHA256
- *ECDHE_ECDSA_AES_256_CBC_SHA384
- *ECDHE_ECDSA_AES_128_GCM_SHA256
- *ECDHE_ECDSA_AES_256_GCM_SHA384
- *RSA_AES_128_CBC_SHA256
- *RSA_AES_128_CBC_SHA
- *RSA_AES_256_CBC_SHA256
- *RSA_AES_256_CBC_SHA
- *RSA_AES_128_GCM_SHA256
- *RSA_AES_256_GCM_SHA384
- *ECDHE_RSA_AES_128_CBC_SHA256
- *ECDHE_RSA_AES_256_CBC_SHA384
- *ECDHE_RSA_AES_128_GCM_SHA256
- *ECDHE_RSA_AES_256_GCM_SHA384

The shipped default cipher specification list can be reduced and reordered by changing the QSSLCSL system value. The shipped default cipher specification list values, but not order, can also be changed by using System Service Tools (SST) Advanced Analysis Command SSLCONFIG.

The following table shows the cipher specifications that are supported for each protocol version. The supported cipher specifications for each protocol are indicated by the "X" in the appropriate column.

Table 1. Supported Cipher Specifications for TLS and SSL Protocols

QSSLCSL System Value Representation	**TLSv.2 1**	**TLSv.1 1**	**TLSv.0 1**	**SSLv 3**	**SSLv 2**
	X				
	X				
	X				
	X				
	X				
	X				
	X				
	X				
	X				
	X				
	X				
	X				
	X				
	X				
	X				

Table 1. Supported Cipher Specifications for TLS and SSL Protocols (continued)

QSSLCSL System Value Representation	TLSv.2 1	TLSv.1 1	TLSv.0 1	SSLv 3	SSLv 2
	X				
	X				
	X				
*RSA_AES_256_CBC _SHA	X	X	X		
*RSA_AES_128_CBC _SHA	X	X	X		
*RSA_3DES_EDE_CB C_SHA	X	X	X	X	
*RSA_RC4_128_SHA	X	X	X	X	
*RSA_RC4_128_MD5	X	X	X	X	X
*RSA_DES_CBC_SHA		X	X	X	
*RSA_EXPORT_RC4_ 40_MD5			X	X	X
*RSA_EXPORT_RC2_ CBC_40_MD5			X	X	X
*RSA_NULL_SHA256	X				
*RSA_NULL_SHA	X	X	X	X	
*RSA_NULL_MD5	X	X	X	X	
*RSA_RC2_CBC_128 _MD5					X
*RSA_3DES_EDE_CB C_MD5					X
*RSA_DES_CBC_MD5					X

Related concepts

The SSLCONFIG macro
The SSLCONFIG macro allows viewing or altering system-wide System SSL default properties.

Related information

SSL system value: QSSLCSLCTL

SSL system value: QSSLCSL

Signature Algorithms

The TLSv1.2 protocol made the signature algorithm and the hash algorithm that are used for digital signatures an independent attribute. Previously the negotiated cipher suite determined these algorithms. System SSL has the infrastructure to support multiple signature algorithms.

The ordered list of allowed signature/hash algorithm pairs serves two purposes in TLSv1.2 and has no meaning for prior protocols:

Certificate Selection

The ordered signature algorithm list is sent to the peer when System SSL requests a certificate during the handshake. The peer uses the received list to guide the certificate selection process. The peer should select a certificate that conforms to the list however that is not true for all implementations and configurations. System SSL treats a received certificate with an undesired signature algorithm as a session error unless optional client authentication is configured.

When System SSL receives a certificate request and is unable to select a conforming certificate, it sends an available nonconforming RSA certificate. The peer determines whether this certificate results in a session error. Refer to "Multiple Certificate Selection" on page 19 for more details on the System SSL certificate selection logic.

Message Signature

The list of algorithm pairs restricts which signature and hash algorithms can be used for handshake message digital signatures. A TLSv1.2 handshake message signature can be different from the signature of the certificate that is used for the session. For instance, the handshake message can be protected by SHA512 even though an MD5 certificate is selected for the session.

System SSL has the infrastructure to support the following signature algorithms:

- ECDSA_SHA512
- ECDSA_SHA384
- ECDSA_SHA256
- ECDSA_SHA224
- ECDSA_SHA1
- RSA_SHA512
- RSA_SHA384
- RSA_SHA256
- RSA_SHA224
- RSA_SHA1
- RSA_MD5

Shipped SSL Supported Signature Algorithms

System SSL is shipped with the following list of supported signature algorithms:

- ECDSA_SHA512
- ECDSA_SHA384
- ECDSA_SHA256
- ECDSA_SHA224
- ECDSA_SHA1
- RSA_SHA512
- RSA_SHA384
- RSA_SHA256
- RSA_SHA224
- RSA_SHA1
- RSA_MD5

Shipped SSL Default Signature Algorithms

The following displays the order of the shipped default signature algorithm list:

- ECDSA_SHA512

- ECDSA_SHA384
- ECDSA_SHA256
- ECDSA_SHA224
- ECDSA_SHA1
- RSA_SHA512
- RSA_SHA384
- RSA_SHA256
- RSA_SHA224
- RSA_SHA1
- RSA_MD5

The shipped default signature algorithm list can be changed by using System Service Tools (SST) Advanced Analysis Command SSLCONFIG.

Related concepts

The SSLCONFIG macro
The SSLCONFIG macro allows viewing or altering system-wide System SSL default properties.

Renegotiation

Starting a new handshake negotiation inside of an existing secure session is called renegotiation. There are two properties that determine System SSL renegotiation characteristics.

Multiple reasons exist for an application to use renegotiation. Renegotiation can be started by either the client or server. The application layer might not be aware that a secure session is renegotiated at the request of a peer. `gsk_secure_soc_misc()` is used by a GSKit System SSL application to initiate renegotiation.

The SSL and TLS protocol architecture as defined by their base RFCs contain a flaw with renegotiation. The protocols fail to provide cryptographic verification that a session renegotiation is linked to the existing secure session. Supplemental RFC 5746 defines an optional extension to the base protocols to correct the issue.

Since RFC 5746 is a recent addition to a previously defined protocol, not all SSL implementations currently support it. Some SSL implementations have not or cannot be updated to support RFC 5746. To allow for business continuity/interoperability during various stages of the transition, two renegotiation properties exist.

SSL Renegotiation Mode

The System SSL default requires RFC 5746 semantics are used for all renegotiated handshakes. The default mode can be changed with the System Service Tools (SST) Advanced Analysis Command SSLCONFIG.

The mode can be set to allow all unsecure renegotiations or to allow only abbreviated unsecure renegotiations. Use these modes only after careful consideration.

A mode exists to disable all peer initiated handshake renegotiation. This mode prevents secure (RFC 5746 semantics) and unsecure renegotiation. This mode can result in interoperability issues for applications that require the use of renegotiation. Locally initiated secure renegotiation such as `gsk_secure_soc_misc()` is still allowed in this mode.

SSL Extended Renegotiation Critical Mode

Extended Renegotiation Critical Mode determines when System SSL requires all peers provide the RFC 5746 renegotiation indication during initial session negotiation.

To completely protect both sides of the secure session against the renegotiation weakness, all initial negotiations must indicate support for RFC 5746. This indication can be in the form of the "renegotiation_info" TLS Extension or the Signaling Cipher Suite Value (SCSV) as defined by RFC 5746.

Critical mode is disabled by default to maintain interoperability with SSL implementations that do not implement RFC 5746. When critical mode is enabled, System SSL is restricted to negotiating with systems that implemented RFC 5746. The restriction exists even if neither side supports or uses renegotiation. If it is determined that all System SSL peers support RFC 5746, then this mode can be changed to be enabled.

The default extended renegotiation critical mode can be changed with System Service Tools (SST) Advanced Analysis Command SSLCONFIG. There is a property for client applications and a separate property for server applications.

System SSL always sends the “renegotiation_info” TLS Extension or the SCSV in ClientHello. SCSV is sent only if no other extensions are part of the ClientHello.

Related concepts

The SSLCONFIG macro
The SSLCONFIG macro allows viewing or altering system-wide System SSL default properties.

Related information

RFC 5746: "Transport Layer Security (TLS) Renegotiation Indication Extension"

DCM Application Definitions

Digital Certificate Manager (DCM) manages an application database that contains application definitions. Each application definition encapsulates certificate processing information for a specific application. As of the IBM i 7.1 release, the application definition also encapsulates some System SSL attributes for the application. System SSL users know this application definition as an “Application ID.”

Many of the IBM i provided applications use application definitions to configure certificate information for their application. Any application developer can design an application to use application definitions.

The DCM application definition contains two fields that are used to identify it. The **Application description** field is used to find and interact with the application definition in DCM. The **Application ID** field is used by System SSL to identify the application definition that holds the configuration information.

Each of the following System SSL programming interfaces has a method for identifying the “Application ID” to use.

- Global Security Kit (GSKit) APIs
  - gsk_attribute_set_buffer(with attribute GSK_IBMI_APPLICATION_ID)
- Integrated IBM i SSL_ APIs
  - SSL_Init_Application(set value in struct SSLInitAppStr)
- Integrated IBM i JSSE implementation
  - Set the Java™ system property os400.secureApplication

The following DCM application definition fields can be used to control the corresponding System SSL attributes of an application:

SSL Protocols

The **SSL protocols** application definition field determines which SSL protocol versions are supported by the application.

The default value is *PGM which means the program that uses this "application ID" set the SSL protocol attribute to the appropriate value. All System SSL programs have a protocol attribute value that is set explicitly through an API call, or implicitly by allowing the system default to be used. Use *PGM unless it is known that the required attribute value is not set by the program.

If *PGM does not result in the appropriate protocols, this application definition field can override the protocols that are supported by this application. If at least one of the protocols that are identified here are enabled on the system by the QSSLPCL system value, protocols that are not enabled on the system are silently ignored. Where possible follow the configuration steps that are included in the application documentation to set the protocols instead of using the application definition field. An administrator can configure weaker security properties for an IBM application than was previously possible by using this field.

Related information

SSL system value: QSSLPCL

SSL Cipher Specification Options

The **SSL cipher specification options** application definition field determines which SSL cipher suites are supported by the application.

The default value is *PGM which means the program that uses this "application ID" set the supported cipher suites attribute to the appropriate value. The program might set the value explicitly through an API call or implicitly by allowing the system default to be used.

If *PGM results in the incorrect value, this field can define the SSL cipher suites that are supported by this application. Cipher suites that are disabled by the QSSLCSL system value are ignored when at least one cipher suite is enabled.

The server application controls the cipher suites that are supported by a prioritized list. A combination of security policy, performance, and interoperability considerations is used by the administrator to determine the appropriate configuration. Use caution when you consider changes to the list. The flexibility of the user-defined list allows for a weaker security configuration than might be possible with *PGM. Security can be weakened in several ways:

- Selecting a higher priority for a relatively weak encryption algorithm
- Disabling a relatively strong encryption algorithm
- Enabling a relatively weak encryption algorithm

A server is only as secure as the weakest cipher suite it allows regardless of its position in the ordered list.

Related information

SSL system value: QSSLCSL

Extended Renegotiation Critical Mode

This application definition field determines whether the application requires the peer provide the RFC 5746 renegotiation indication during the initial handshake.

See SSL Extended Renegotiation Critical Mode in the "Renegotiation" on page 11 topic under System SSL Properties for details on this concept.

The default value is *PGM which means the program that uses this "application ID" already set the mode to the appropriate value. The program is either using the System SSL default value or a value that is set explicitly by the `gsk_attribute_set_enum()` API call for this attribute.

Set to "Enable" to require the RFC 5746 renegotiation indication is included in the initial handshake for the initial handshake to be successful. By design, this application is no longer able to handshake with peers that have not or cannot be updated to support RFC 5746.

Set this application definition field to "Disable" if the application does not require RFC 5746 renegotiation indication from the peer on initial handshake. The RFC 5746 renegotiation indication is still required for all renegotiated handshakes.

Note: The application always provides the RFC 5746 renegotiation indication information to the peer regardless of the value of this setting.

Related information

RFC 5746: "Transport Layer Security (TLS) Renegotiation Indication Extension"

Server Name Indication

The **Server Name Indication (SNI)** application definition field is used to tell System SSL to provide limited SNI support for this application.

SNI, as defined by RFC 6066, allows TLS clients to provide to the TLS server the name of the server they are contacting. This function is used to facilitate secure connections to servers that host multiple 'virtual' servers at a single underlying network address. To use SNI in this way, as either the client or the server, gsk_attribute_set_buffer() must be used to configure SNI in the application.

If limited SNI support is needed, enter the fully qualified domain name (FQDN) for the server application. If not required, accept the default value of *NONE for the limited SNI support which does not override existing SNI application configuration.

If the application configures SNI information with gsk_attribute_set_buffer(), then the value that is set for this application definition field is appended to the end of that existing information. If the existing information is configured to be critical, then this value would also be critical. Critical means that if the client FQDN does not match a name in the server list, then the server generates a failure for the session negotiation. If no existing information exists, then the limited SNI support is not critical.

A use case for setting the limited support SNI field: Your Company is contacted by a user of your services. The user has a new security requirement that all TLS servers they communicate with provide the server SNI acknowledgement. Your server is a simple server that is used for just one service with no need for virtual server configuration.

By setting the FQDN of the server, System SSL can send the SNI server acknowledgement if asked. The server certificate that is sent is the one assigned to the application definition. Nothing is changed from the server perspective, yet the peer client satisfied their requirement.

If the client requested FQDN does not match (because this field was not set or had a different value), no server SNI acknowledgment is sent. The server continues with handshake negotiation as if no SNI request was made. The client determines whether this condition is a critical error for the session negotiation.

Related information

RFC 6066: "Transport Layer Security (TLS) Extensions: Extension Definitions"

Online Certificate Status Protocol Attributes

The Online Certificate Status Protocol (OSCP) attribute fields in the application definitions are used to control OCSP enablement.

OSCP is a mechanism for determining the revocation status of a certificate. See the detailed OCSP description to understand how that processing works. The GSKit APIs allow for configuring numerous attributes that determine OCSP processing.

The application definition has two OCSP attribute fields that are used to control OCSP enablement. The other OCSP attributes cannot be changed by the application definition. Those other attribute values are determined by GSKit API settings or by internal default values.

Related concepts

Online Certificate Status Protocol

Online Certificate Status Protocol (OCSP) provides applications a way to determine the revocation status for a digital certificate. Certificate revocation status that is checked via OCSP provides more up-to-date status information than is available through CRLs.

Related information

RFC 2560: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP

OCSP URL

The **Online Certificate Status Protocol (OCSP) URL** application definition field determines whether this application uses a general OCSP responder to send requests during certificate validation for end entity certificates.

When a URL is present, the specified OCSP responder is contacted for all end entity certificates to determine revocation status.

The default value for the field is *PGM meaning the program that uses this "application ID" set the attribute to the appropriate value. All System SSL attributes have an initial default value which for this attribute is no URL value. Programs can also call gsk_attribute_set_buffer() to explicitly set a URL value.

If *PGM does not result in the required OCSP responder, enter the appropriate OCSP responder URL in this field. HTTP is the only supported URL protocol; therefore, this value must begin with "http://". Setting this value overrides the configuration that is set internally by the program for the URL destination. However, the other configured OCSP attributes continue to be used as appropriate.

If *PGM results in the application that uses an OCSP responder, yet no general OCSP responder processing is wanted, set this field to “Disable.” This setting overrides a URL internally configured by using gsk_attribute_set_buffer(). Disabling OCSP weakens the security model for the application, so use due diligence before you make this choice.

Related concepts

Certificate Revocation
Certificate revocation checking is one phase of certificate validation that is done as part of session negotiation. The certificate chain is validated to ensure that the certificate is not revoked.

OCSP AIA Processing

The **Online Certificate Status Protocol (OCSP) Authority Information Access (AIA) processing** application definition field determines whether OCSP certificate revocation checking is done with AIA certificate extension information.

Revocation checking is done with AIA certificate extension information if OCSP revocation status is undetermined and both of the following conditions are true:

- OCSP AIA checking is enabled.
- The certificate to be validated has an AIA extension with a PKIK_AD_OCSP access method that contains a URI of the HTTP location of the OCSP responder.

  Note: The first OCSP responder that is identified in the AIA extension is queried for revocation status.

The default value for the field is *PGM meaning the program that uses this "application ID" set the attribute to the appropriate value. All System SSL attributes have an initial default value. For this attribute the default value is disabled. Programs can explicitly enable or disable OCSP AIA with gsk_attribute_set_enum().

If *PGM does not result in OCSP AIA validation and revocation checking is wanted, set this field to “Enable.” The internal setting is overridden and OCSP checking happens when AIA information is available.

If *PGM results in OCSP AIA validation, yet revocation checking is not wanted, set this field to “Disable.” Disabling OCSP weakens the security model for the application, so use due diligence before you make this choice.

Related concepts

Certificate Revocation
Certificate revocation checking is one phase of certificate validation that is done as part of session negotiation. The certificate chain is validated to ensure that the certificate is not revoked.

SSL Signature Algorithms

The **SSL signature algorithms** application definition field determines which algorithms are supported by the application.

For an overview of what signature algorithms are and how they are used, see the “Signature Algorithms” on page 9 topic under System SSL Properties.

The default value is *PGM which means the program that uses this "application ID" set the SSL signature algorithms to the appropriate value. All System SSL attributes have an initial default value. Programs can explicitly define the list by using gsk_attribute_set_buffer().

If *PGM results in the inappropriate configuration, set this field to contain the required ordered list of supported signature algorithms.

Online Certificate Status Protocol
Online Certificate Status Protocol (OCSP) provides applications a way to determine the revocation status for a digital certificate. Certificate revocation status that is checked via OCSP provides more up-to-date status information than is available through CRLs.

The implementation of OCSP revocation status checking is done in accordance with RFC 2560. OCSP certificate revocation status checking is available for the end entity certificate. Protocol version 1 over HTTP and the basic response type are supported.

Certificate revocation status is checked by an application via OCSP when at least one of the following conditions are true:

- A URL address of an OCSP responder is configured.
- Authority Information Access (AIA) checking is enabled and the certificate to be validated has an AIA extension. The AIA extension must contain a PKIK_AD_OCSP access method with a URI that indicates the HTTP location of the OCSP responder.

  Note: Only the first OCSP responder that is identified in the AIA extension is queried for revocation status.

When URL and AIA checking are enabled, AIA checking is only done if the query sent to the URL responder results in undetermined revocation status.

Related concepts
Certificate Revocation
Certificate revocation checking is one phase of certificate validation that is done as part of session negotiation. The certificate chain is validated to ensure that the certificate is not revoked.

Related information
RFC 2560: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP

OCSP Configuration
In addition to enabling Online Certificate Status Protocol (OCSP), there are a number of properties that can be configured by an application to customize the OCSP client behavior.

When OCSP revocation checking is enabled, an HTTP request is sent to an OCSP responder. The request contains information to identify the certificate for which revocation status is being queried and an optional signature. The optional signature on the request is used to help the responder verify valid requests that are received from clients. Request signatures are disabled by default. The request is sent to the responder over HTTP by the GET or the POST method. Requests that are sent by the GET method enable HTTP caching. If configuration indicates that the GET method is preferred and the request is smaller than 255 bytes, the request is sent by the GET method. Otherwise, the request is sent by the POST method. The GET method is preferred by default.

After a request is sent, OCSP revocation checking blocks until a response is received from the responder or it times out. Revocation checking happens as part of the session negotiation; therefore, the session negotiation blocks while revocation checking is done. If the timeout set on the session negotiation is smaller than the OCSP timeout configured, the smaller value is used for the OCSP timeout. The OCSP timeout value defaults to 10 seconds, but can be configured by an application.

A valid response is signed and includes data to indicate the revocation status of the certificate queried. The revocation status of the certificate can be good, unknown, or revoked. The response must be signed by a certificate that meets at least one of the following requirements:

- The signing certificate is trusted by the local certificate store.
- The signing certificate is the certificate authority (CA) that issued the certificate to be validated.
- The signing certificate includes a value of id-ad-ocspSigning in an ExtendedKeyUsage extension and is issued by the CA that issued the certificate in question.

The size of a response can vary. It is up to the application to determine the maximum response size allowed. By default the maximum response size that is allowed is 20480 bytes. If a response is larger than the maximum size allowed, the response is ignored and treated the same as a response that indicates unknown certificate status.

A cryptographic nonce value is a security mechanism that can be used to verify that the response received is a reply to a particular request. The nonce value, which is a random generated bit string, is computed and included as part of both the request and response. If nonce checking is enabled, the nonce value included on the response is verified with the value that is sent in the request. If the nonce values do not match, the response is ignored. Nonce checking is disabled by default.

Revocation checking can slow down session negotiation. However, caching OCSP responses allows the client to obtain revocation status from previous requests without sending the same request again. The OCSP response cache is enabled by default, but can be disabled for an application.

A proxy HTTP server can be used as an intermediate server to handle OCSP requests from cached responses, or forward requests to the configured responder. If a proxy server is configured for an application, all the OCSP requests for the application are sent to the configured server. The default proxy port is 80. A proxy server is not configured by default.

The Global Security Kit (GSKit) APIs, `gsk_attribute_set_buffer()`, `gsk_attribute_set_numeric_value()`, and `gsk_attribute_set_enum()`, can be used to configure OCSP with the following API attributes:

- GSK_OCSP_URL - URL of the OCSP responder to which OCSP requests are sent
- GSK_OCSP_ENABLE - Enable AIA checking
- GSK_OCSP_REQUEST_SIGKEYLABEL - Certificate label of the certificate that is used to sign the OCSP request
- GSK_OCSP_REQUEST_SIGALG - Signature algorithm that is used to generate the signature of the OCSP request
- GSK_OCSP_RETRIEVE_VIA_GET - Method with which the OCSP request is sent
- GSK_OCSP_TIMEOUT - Number of seconds to wait for a response from the OCSP responder
- GSK_OCSP_MAX_RESPONSE_SIZE - Maximum response size in bytes to be accepted from the OCSP responder
- GSK_OCSP_CLIENT_CACHE_SIZE - Enable or disable the OCSP client response cache
- GSK_OCSP_NONCE_GENERATION_ENABLE - Send a nonce extension as part of the OCSP request
- GSK_OCSP_NONCE_CHECK_ENABLE - Verify that the nonce extension in the OCSP response matches the one sent in the OCSP request
- GSK_OCSP_NONCE_SIZE - Number of bytes to be used to generate the nonce value
- GSK_OCSP_PROXY_SERVER_NAME - Server name of the proxy server to which OCSP requests are sent
- GSK_OCSP_PROXY_SERVER_PORT - Port number of the proxy server to which OCSP requests are sent

Applications that use the integrated IBM i SSL_ APIs or IBM i JSSE do not have an interface to configure OCSP. However, any programs that use an "application ID" can enable or disable OCSP revocation checking through DCM. The default values are used for all other OCSP configuration options.

Related concepts

Online Certificate Status Protocol Attributes
The Online Certificate Status Protocol (OSCP) attribute fields in the application definitions are used to control OCSP enablement.

Related information

Global Security Kit (GSKit) APIs

OCSP Revocation Status
OCSP revocation status is determined by the OCSP response sent in reply to an OCSP request. Two types of responses can be received. One indicates that the OCSP responder sent a valid response; the other signals that the responder encountered an issue as it processed the prior request.

The issues a responder might encounter while it processes a request include:

- malformedRequest - The request was malformed.
- internalError - An internal error occurred on the OCSP responder.
- tryLater - The OCSP responder is temporarily unable to respond; try the request again later.
- sigRequired - The OCSP responder requires that the request must be signed.
- unauthorized - The OCSP client is not authorized to query the OCSP responder.

A valid response contains certificate revocation status for the queried certificate. Certificate status values include good, revoked, or unknown. OCSP revocation status checking is complete if it receives good or revoked revocation status for the certificate. Good status allows the handshake to continue and revoked status causes the handshake to fail.

If revocation status from both URL and AIA checking is undetermined, validation continues as if the status is not revoked. Information about the certificate with undetermined revocation status can be retrieved with `gsk_attribute_get_buffer()` and attribute GSK_UNKNOWNREVOCATIONSTATUS_SUBJECT.

Undetermined Revocation Status

The following responses result in undetermined revocation status:

- No response received within the specified timeout
- OCSP response status that indicates the responder encountered an issue
- Valid response with one of the following conditions:
  - Unknown response type (only PKIX_AD_OCSP_basic response type supported)
  - Unknown response version (only version 1 supported)
  - Invalid signature or invalid signing certificate
    - The signing certificate must meet one of the following criteria:
      - Is trusted by the local keystore
      - Is the certificate of the certificate authority (CA) that issued the certificate in question
      - Includes a value of id-ad-ocspSigning in an ExtendedKeyUsage extension and is issued by the CA that issued the certificate in question
  - No nonce included on the response when nonce checking is required
  - Bad nonce value on the response when nonce checking is required
  - Invalid or expired nextUpdate value is specified on the response
  - Unknown certificate status value that indicates the responder does not know the status of the certificate

Certificate Revocation
Certificate revocation checking is one phase of certificate validation that is done as part of session negotiation. The certificate chain is validated to ensure that the certificate is not revoked.

The following steps apply for certificate revocation checking:

1. Check revocation status with a Certificate Revocation List (CRL) location.
   a. When a CRL location is configured through the Digital Certificate Manager (DCM), a CRL database (LDAP) server is queried for CRLs containing the revocation status of the certificate.
      - If the certificate is revoked, the certificate revocation phase of certificate validation is complete and the session negotiation fails.

   - Otherwise, continue with certificate revocation processing.

   Note: CRL locations are configured individually for each certificate authority (CA) in a local certificate store.

2. Check revocation status with Online Certificate Status Protocol (OCSP).

   a. When an OCSP URL responder address is configured, query the responder.

      - If the certificate is revoked, the certificate revocation phase of certificate validation is complete and the session negotiation fails.
      - If the certificate is good, the certificate revocation phase of certificate validation is complete and certificate validation continues.
      - If the certificate revocation status is undetermined, continue with certificate revocation processing.

   b. When AIA checking is enabled and the certificate has a PKIK_AD_OCSP access method with a URI that indicates the HTTP location, query the responder.

      - If the certificate is revoked, the certificate revocation phase of certificate validation is complete and the session negotiation fails.
      - If the certificate is good, the certificate revocation phase of certificate validation is complete and certificate validation continues.
      - If the certificate revocation status is undetermined, the certificate revocation phase of certificate validation is complete and certificate validation continues.

   Note: If revocation status is undetermined, GSKit stores information about the certificate for which revocation status is undetermined and continues as if the status is not revoked. The application can retrieve undetermined certificate status information with `gsk_attribute_get_buffer()` and attribute GSK_UNKNOWNREVOCATIONSTATUS_SUBJECT and make a policy decision on whether to continue or end the connection.

Note: An application definition that is configured in DCM can override CRL and OCSP revocation checking that is configured by an application that uses the application definition.

Related concepts

Online Certificate Status Protocol
Online Certificate Status Protocol (OCSP) provides applications a way to determine the revocation status for a digital certificate. Certificate revocation status that is checked via OCSP provides more up-to-date status information than is available through CRLs.

Related information

Managing CRL locations

Multiple Certificate Selection

System SSL allows up to four certificates to be assigned to a secure environment. The purpose of multiple certificates is to enable Elliptic Curve Digital Signature Algorithm (ECDSA) certificates while still allowing RSA certificates to be used with clients that require RSA.

For maximum interoperability, a server must be able to negotiate with a wide range of clients with varying SSL capabilities. When one client does not support TLSv1.2 or ECDSA certificates, the server must retain the ability to negotiate with an RSA certificate.

Two methods exist to configure multiple certificates for an environment:

- Use the Digital Certificate Manager (DCM) to assign multiple certificates to the application definition configured for the secure environment.
- Use the GSKit API gsk_attribute_set_buffer(GSK_KEYRING_LABEL_EX) to configure a list of certificate labels to be used.

During each SSL handshake, the appropriate certificate is selected for use by the session based on the attributes for the session. The selection process uses the SSL attributes from both the client and server to make the decision. It is possible that no certificate is usable for a combination of attributes.

Selection Considerations

When the negotiated protocol is TLSv1.1, TLSv1.0, SSLv3, or SSLv2 the first RSA certificate found that also has an RSA signature is used.

When the negotiated protocol is TLSv1.2, the selection process involves several steps.

1. The key type of the server's most preferred cipher suite that is supported by the client starts the selection process. A certificate has either an ECDSA or RSA key. If the cipher suite name contains "ECDSA", then an ECDSA key/certificate must be used. If the cipher suite name contains "RSA", then an RSA key/certificate must be used. If a matching certificate does not exist, then selection moves to the next cipher suite in the list until it finds a cipher suite that works with one of the certificates.

   When one or more certificate key type matches are found, additional criteria are checked to determine which if any can be used based on the rest of the handshake attributes.
2. When ECDSA is the key type that is being considered, the named curve (which equates to key size) must be supported by the peer. If the ECDSA certificates under consideration have different named curves, the peer's preferred order of named curves is used to determine the preferred certificate. One or more certificates can be tied for selection after this step. If no certificates remain eligible after this step, revert to the previous step.
3. Both RSA and ECDSA certificates have a signature from the certificate's issuing certificate authority. The signature key type can be different from the server certificate key type. When multiple certificates remain eligible for selection in this phase, the one with the signature most preferred by the peer is selected.

   An ECDSA certificate must have a signature algorithm that is allowed by the peer's ordered signature algorithm list. If no ECDSA certificates in this step have a supported signature algorithm, revert to the previous step where a less preferred named curve can be considered for additional selection processing.

   If no RSA certificates in this step have a supported signature algorithm, revert to the first step where a different key type can be considered for additional selection processing.

   If there are multiple certificates still equivalent after this step, the first one processed is selected. The other equivalent certificates are never selected as they are cryptographically identical to the first certificate. Remove the additional certificates from the configuration to eliminate the extra selection processing.
4. For compatibility with previous releases, there is one final consideration for RSA certificates. When processing completes with no certificate that meets the preceding selection criteria, the first RSA certificate that is processed is selected provided an RSA cipher suite is in the list.

The SSLCONFIG macro

The SSLCONFIG macro allows viewing or altering system-wide System SSL default properties.

To use the SSL configuration IBM-supplied macro support, follow these steps:

1. Access System Service Tools by using SST.
2. Select **Start a service tool**.
3. Select **Display/Alter/Dump**.
4. Select **Display/Alter storage**.
5. Select **Licensed Internal Code (LIC) data**.
6. Select **Advanced analysis**. (You must page down to see this option.)
7. Page down until you find the SSLCONFIG option. Then, place a 1 (Select) next to the option and press Enter. You are now on the Specify Advanced Analysis Options window. The command shows as SSLCONFIG.
8. Enter '`-h`' without the quotation marks and press Enter to display the available options.

Server authentication

With server authentication, the client will ensure that the server certificate is valid and that it is signed by a certificate authority (CA) which the client trusts.

SSL will use asymmetric cryptography and handshake protocol flows to generate a symmetric key which will be used only for this unique SSL session. This key is used to generate a set of keys which are used for encrypting and decrypting data which will flow over the SSL session. Subsequently, when an SSL handshake has completed, one or both ends of the communication link will have been authenticated. Additionally, a unique key will have been generated to encrypt and decrypt the data. Once the handshake is completed then application layer data will flow encrypted across that SSL session.

Client authentication

Many applications allow the option to enable client authentication. With client authentication, the server will ensure that the client certificate is valid and that it is signed by a Certificate Authority which the server trusts.

The following IBM i applications support client authentication:

- IBM HTTP Server for i
- FTP server
- Telnet server
- Management Central endpoint system
- IBM Tivoli® Directory Server for IBM i

Related information

Secure Sockets Layer (SSL) and Transport Layer Security (TLS) with the Directory Server
Securing FTP clients with Transport Layer Security or Secure Sockets Laye
Securing Telnet with SSL
Setting up SSL for the administration (ADMIN) server for HTTP Server

SSL prerequisites

This topic describes the prerequisites for System SSL on the IBM i, as well as some helpful tips.

Verify you have the following options installed before using SSL:

- IBM Digital Certificate Manager (DCM) (5770-SS1 Option 34)

  Note: IBM Java Secure Socket Extension (JSSE) and OpenSSL do not require DCM.
- IBM TCP/IP Connectivity Utilities for i (5770-TC1)
- IBM HTTP Server for i (5770-DG1)
- If you are trying to use the HTTP server to use the DCM, ensure that you have the IBM Developer Kit for Java (5770-JV1) installed. Otherwise, the HTTP admin server will not start.
- You may also want to install cryptographic hardware to use with SSL to speed up the SSL handshake processing. If you want to install cryptographic hardware, you must also install the IBM CCA Service Provider (5770-SS1 Option 35) and IBM Cryptographic Device Manager (5733-CY3).

Related concepts

Troubleshooting SSL
This very basic troubleshooting information is intended to help you reduce the list of possible problems that the IBM i platform can encounter with SSL.

Related information

Cryptographic hardware
Public certificates versus private certificates
Configure DCM

Application security with SSL

A number of applications on IBM i can be secured with SSL. Refer to each application's documentation for details.

Related concepts

Scenario: Securing all connections to your Management Central server with SSL
This scenario explains how to use SSL to secure all connections with an IBM i that is acting as a central system by using the System i Navigator Management Central system.

Related information

Enterprise Identity Mapping
Use SSL to secure the FTP server
HTTP server
Secure Sockets Layer administration (iSeries Access for Windows topic)
Telnet scenario: Secure Telnet with SSL
Secure Sockets API

Scenarios: SSL

The SSL scenarios are designed to help you maximize the benefits of enabling SSL on your IBM i.

Read the SSL scenarios to increase your understanding of SSL on IBM i by providing possible examples of how SSL can work for you.

Related information

Scenario: Secure FTP with SSL
Scenario: Secure Telnet with SSL
Scenario: Protect private keys with cryptographic hardware

Scenario: Securing a client connection to your Management Central server with SSL

This scenario explains how to use SSL to secure the connection between a remote client and an IBM i that is acting as a central system by using the System i Navigator Management Central server.

Situation:

A company has a local area network (LAN) that includes several IBM i systems in their office. This company's system administrator, Bob, has specified one of the IBM i systems as the central system (hereafter referred to as System A) for the LAN. Bob uses the Management Central server on System A to manage all of the other endpoints on his LAN.

Bob is concerned about connecting to the Management Central server on System A from a network connection that is external to his company's LAN. Bob travels for work a lot, and requires a secure connection to the Management Central server while he is away. He wants to ensure the connection between his PC and the Management Central server is secure when he is not in the company office. Bob decides to enable SSL on his PC and on the System A's Management Central server. With SSL enabled in this way, Bob can be certain that his connection to the Management Central server is secure when he is traveling.

Objectives:

Bob wants to secure the connection between his PC and the Management Central server. Bob does not require additional security for the connection between the Management Central server on System A and the endpoints that are on the LAN. Other employees that work from the company office do not need additional security for their connections to the Management Central server, either. Bob's plan is to configure his PC and the Management Central server on System A, so that his connection uses server authentication. Connections to the Management Central server from other PCs or IBM i systems on the LAN are not secured with SSL.

Details:

The following table illustrates the types of authentication used, based on the enabling or disabling of SSL on a PC client:

Table 2. Required elements for an SSL-secured connection between a client and the Management Central server

SSL status on Bob's PC	Specified authentication level for the Management Central server on System A	SSL connection enabled?
SSL off	Any	No
SSL on	Any	Yes (server authentication)

Server authentication means that Bob's PC authenticates the Management Central server's certificate. Bob's PC acts as an SSL client when connecting to the Management Central server. The Management Central server acts as an SSL server and must prove its identity. The Management Central server does this by providing a certificate issued by a Certificate Authority (CA) that Bob's PC trusts.

Prerequisites and assumptions

Bob must perform these administration and configuration tasks in order to secure the connection between his PC and the Management Central server on System A:

1. System A meets the prerequisites for SSL.
2. System A is running i5/OS V5R3, or a later.
3. The PC client is running IBM i Access for Windows V5R3, or later.
4. Get a Certificate Authority (CA) for IBM i systems.
5. Create a certificate that is signed by the CA, for System A.
6. Send the CA and a certificate to System A, and import them into the key database.
7. Assign the certificate with the Management Central server identification, and the application identifications for all of the IBM i systems. The TCP central server, database server, data queue server, file server, network print server, remote command server and signon server are all IBM i systems.
   a. On System A, Start IBM Digital Certificate Manager. Bob obtains or create certificates, or otherwise sets up or changes his certificate system now.
   b. Click **Select a Certificate Store**.
   c. Select ***SYSTEM** and click **Continue**.
   d. Enter the *SYSTEM *Certificate Store password*, and click **Continue**. When the menu reloads, expand **Manage Applications**.
   e. Click **Update certificate assignment**.
   f. Select **Server** and click **Continue**.
   g. Select the **Management Central Server**, and click **Update certificate assignment**. This assigns a certificate to the Management Central server to use.
   h. Click **Assign New Certificate**. DCM reloads to the **Update certificate assignment** page with a confirmation message.
   i. Click **Done**.
   j. Assign the certificate to all of the client access servers.
8. Download the CA to the PC client.

Before Bob can enable SSL on the Management Central server, he must install the SSL Prerequisites and set up digital certificates on the system. Once he has met the prerequisites, he can complete the following procedures to enable SSL for the Management Central server.

Configuration steps

Bob needs to complete the following steps in order to secure his PC client connection to the Management Central server on System A, with SSL:

1. "Step 1: Deactivate SSL for the System i Navigator client" on page 24
2. "Step 2: Set the authentication level for the Management Central server" on page 25
3. "Step 3: Restart the Management Central system on the central system" on page 25
4. "Step 4: Activate SSL for the System i Navigator client" on page 25
5. "Optional step: Deactivate SSL for the System i Navigator client" on page 25

Related concepts

Scenario: Securing all connections to your Management Central server with SSL
This scenario explains how to use SSL to secure all connections with an IBM i that is acting as a central system by using the System i Navigator Management Central system.

SSL prerequisites
This topic describes the prerequisites for System SSL on the IBM i, as well as some helpful tips.

Related information

Configure DCM
Start Digital Certificate Manager

Configuration details: Secure a client connection to your Management Central system with SSL

This topic shows the expanded configurations steps for using SSL to secure a client connection to your Management Central server.

The following information assumes you have read through the Scenario: Secure a client connection to your Management Central server with SSL.

In this scenario, an IBM i is specified as the central system in a company's local area network (LAN). Bob uses the Management Central server on the central system (referred to here as System A) to manage the endpoints on the company network. The following information explains how to perform the steps required to secure an external client connection to the Management Central server. Follow along as Bob completes the scenario configuration steps.

Related concepts

SSL prerequisites
This topic describes the prerequisites for System SSL on the IBM i, as well as some helpful tips.

Scenario: Securing all connections to your Management Central server with SSL
This scenario explains how to use SSL to secure all connections with an IBM i that is acting as a central system by using the System i Navigator Management Central system.

Related information

Set up certificates for the first time

Step 1: Deactivate SSL for the System i Navigator client

This step is only necessary if you have already enabled SSL for the System i Navigator client.

1. In System i Navigator, expand **My Connections**.
2. Right-click System A and select **Properties**.
3. Click the **Secure Sockets** tab and deselect **Use Secure Sockets Layer (SSL) for connection**.
4. Exit System i Navigator and restart it.

The padlock disappears from the Management Central container in System i Navigator, indicating an unsecured connection. This indicates to Bob that he no longer has an SSL-secured connection between his client and the central system of his company.

Step 2: Set the authentication level for the Management Central server

1. In System i Navigator, right-click **Management Central**, and select **Properties**.
2. Click the **Security** tab, and select **Use Secure Sockets Layer (SSL)**.
3. Select **Any** for the authentication level (available on IBM i Access for Windows).
4. Click **OK** to set this value on the central system.

Step 3: Restart the Management Central system on the central system

1. In System i Navigator, expand **My Connections**.
2. On System A, expand **Network-->Servers** and select **TCP/IP**.
3. Right-click **Management Central** and select **Stop**. The central system view collapses, and a message displays, explaining you are not connected to the server.
4. After the Management Central server has stopped, click **Start** to restart it.

Step 4: Activate SSL for the System i Navigator client

1. In System i Navigator, expand **My Connections**.
2. Right-click System A and select **Properties**.
3. Click the **Secure Sockets** tab and select **Use Secure Sockets Layer (SSL) for connection**.
4. Exit System i Navigator and restart it.

A padlock appears next to the Management Central server in System i Navigator, indicating an SSL-secured connection. This indicates to Bob that he has successfully activated an SSL-secured connection between his client and the central system of his company.

Note: This procedure only secures the connection between one PC and the Management Central system. Other client connections with the Management Central server, as well as connections from endpoints to the Management Central server, will not be secure. To secure other clients, ensure they meet the prerequisites and repeat “Step 4: Activate SSL for the System i Navigator client” on page 25. To secure other connections with the Management Central server, see Scenario: Secure all connections to your Management Central server with SSL.

Optional step: Deactivate SSL for the System i Navigator client

If Bob wants to work from the company office and does not want an SSL connection affecting the performance of his PC, he can easily deactivate it by performing the following steps:

1. In System i Navigator, expand **My Connections**.
2. Right-click System A and select **Properties**.
3. Click the **Secure Sockets** tab and deselect **Use Secure Sockets Layer (SSL) for connection**.
4. Exit System i Navigator and restart it.

Scenario: Securing all connections to your Management Central server with SSL

This scenario explains how to use SSL to secure all connections with an IBM i that is acting as a central system by using the System i Navigator Management Central system.

Situation:

A company has just set up a wide area network (WAN) that includes several IBM i systems in remote locations (endpoints). The endpoints are centrally managed by one system (the central system), located at the main office. Tom is the company's security specialist. Tom wants use Secure Sockets Layer (SSL) to secure all of the connections between the Management Central server on the company's central system and all IBM i systems and clients.

Details:

Tom can manage all connections to the Management Central server **securely**, with SSL. To use SSL with the Management Central server, Tom needs to secure System i Navigator on the PC that he uses to access the central system.

Tom chooses from two authentication levels for the Management Central server:

Server authentication
: Provides authentication of the server certificate. The client must validate the server, whether the client is System i Navigator on a PC, or the Management Central server on the central system. When System i Navigator connects to the central system, the PC is the SSL Client and the Management Central server running on the central system is the SSL Server. The central system acts as an SSL client when connecting to an endpoint system. The endpoint system acts as an SSL server. The server must prove its identity to the client by providing a certificate that was issued by a Certificate Authority that the client trusts. There must be a valid certificate issued by a trusted CA for every SSL server.

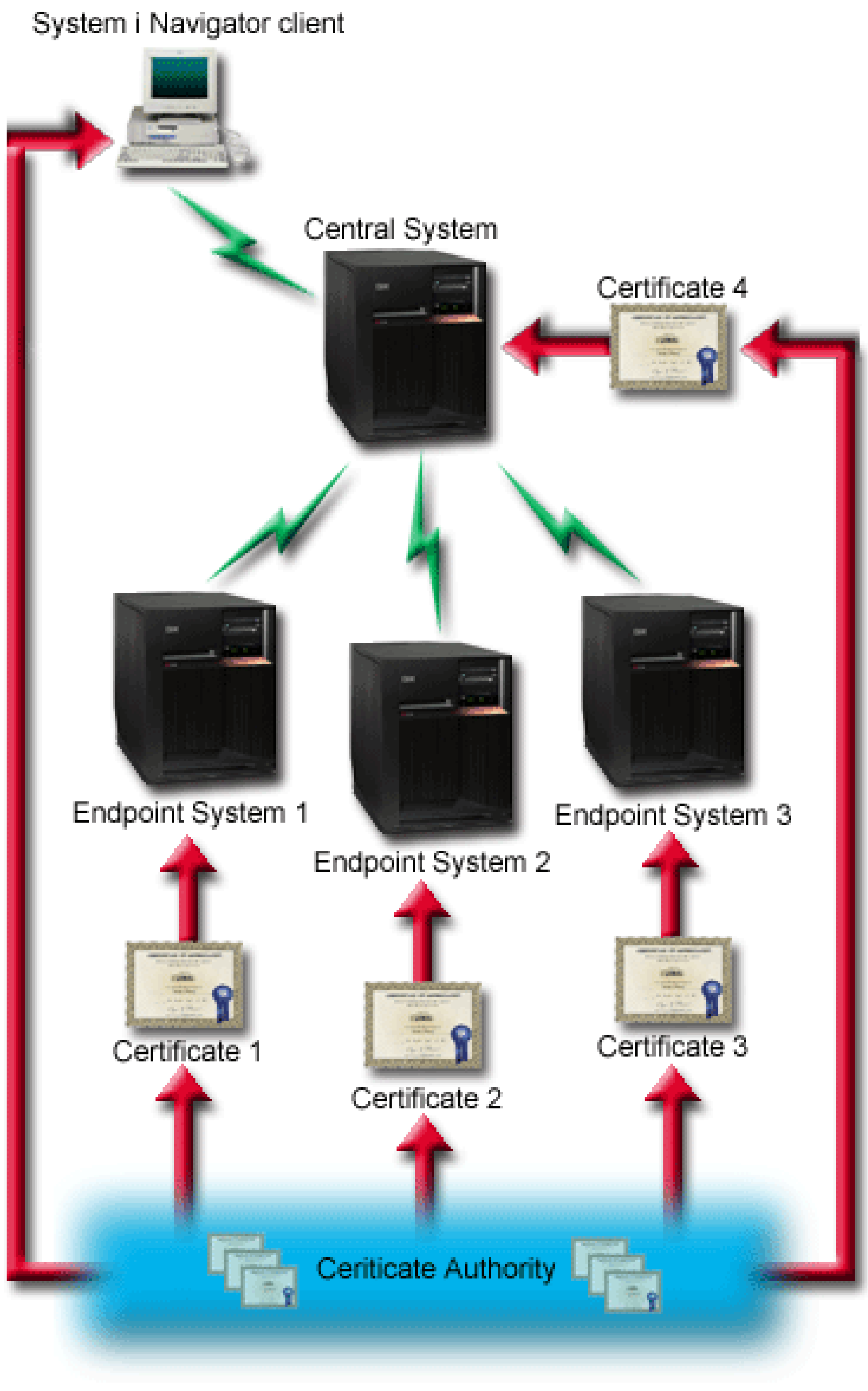
Client and server authentication
: Provides authentication of both the central system and the endpoint system certificates. This is a stronger security level than the server authentication level. In other applications, this is known as client authentication, where the client must supply a valid trusted certificate. When the central system (SSL client) attempts to establish a connection with an endpoint system (SSL server), the central system and the endpoint system authenticate each other's certificates for certificate authority authenticity.

  Note: Client and server authentication only happens between two IBM i systems. Client authentication is not performed by the server when the client is a PC.

  Unlike other applications, Management Central also provides authentication through a validation list, called Trusted Group validation list. Generally the validation list stores information that identifies the user, such as a user identification, and authentication information, such as password, personal identification number, or digital certificate. This authentication information is encrypted.

Most applications typically do not specify that you enable both server and client authentication, because server authentication almost always occurs during SSL session enablement. Many applications have client authentication configuration options. Management Central uses the term "server and client authentication" instead of client authentication because of the dual role that the central system plays in the network. When PC users connect to the central system, the central system acts as a server. However, when the central system is connecting to an endpoint system, it acts as a client. The following illustration shows how the central system operates as both a server and client in a network.

Note: In this illustration, the certificate associated with the Certificate Authority must be stored in the key database on the central system and on all of the endpoint systems. The Certificate Authority must on the central system, all the endpoints, as well as the PC.



Prerequisites and assumptions:

Tom must perform the following administration and configuration tasks, in order to secure all of the connections to the Management Central server:

1. System A meets the prerequisites for SSL.
2. The central system and all endpoint systems run OS/400® V5R2, or i5/OS V5R3, or later.

Note: IBM i 5.4, or later, connections to OS/400 V5R1 systems are not allowed.

3. The PC client is running System i Navigator for IBM i Access for Windows V5R3, or later.
4. Get a Certificate Authority (CA) for the IBM i systems.
5. Create a certificate that is signed by the CA, for System A.
6. Send the CA and a certificate to System A, and import them into the key database.
7. Assign the certificates with the Management Central application identification, and the application identifications for all of the IBM i systems. The TCP central server, database server, data queue server, file server, network print server, remote command server and signon server are all IBM i systems.
   a. Start IBM Digital Certificate Manager on the Management Central server. If Tom needs to obtain or create certificates, or otherwise set up or change his certificate system, he does so now.
   b. Click **Select a Certificate Store**.
   c. Select ***SYSTEM** and click **Continue**.
   d. Enter the *SYSTEM `Certificate Store password`, and click **Continue**. When the menu reloads, expand **Manage Applications**.
   e. Click **Update certificate assignment**.
   f. Select **Server** and click **Continue**.
   g. Select the `Management Central server`, and click **Update certificate assignment**. This assigns a certificate to the Management Central system to use.
   h. Choose the certificate you want to assign to the application, and click **Assign New Certificate**. DCM reloads to the **Update certificate assignment** page with a confirmation message.
   i. Click **Cancel** to return to the list of applications.
   j. Repeat this procedure for all IBM i systems.
8. Download the CA to the System i Navigator PC client.

Configuration steps:

Before Tom can enable SSL on the Management Central server, he must install the prerequisite programs and set up digital certificates on the central system. See the Prerequisites and assumptions for this scenario before continuing. Once he has met the prerequisites, he can complete the following procedures to secure all connections to the Management Central server:

Note: If SSL has been enabled for System i Navigator, Tom must disable it before he can enable SSL on the Management Central server. If SSL has been enabled for System i Navigator and not the Management Central server, attempts by System i Navigator to connect with the central system will fail.

1. "Step 1: Configure the central system for server authentication" on page 29
2. "Step 2: Configure endpoint systems for server authentication" on page 30
3. "Step 3: Restart the Management Central system on the central system" on page 30
4. "Step 4: Restart the Management Central system on all endpoint systems" on page 30
5. "Step 5: Activate SSL for the System i Navigator client" on page 30
6. "Step 6: Configure the central system for client authentication" on page 31
7. "Step 7: Configure endpoint systems for client authentication" on page 31
8. "Step 8: Copy the validation list to the endpoint systems" on page 32
9. "Step 9: Restart the Management Central system on the central system" on page 32
10. "Step 10: Restart the Management Central system on all endpoint systems" on page 32

Related concepts

Scenario: Securing a client connection to your Management Central server with SSL

This scenario explains how to use SSL to secure the connection between a remote client and an IBM i that is acting as a central system by using the System i Navigator Management Central server.

SSL prerequisites
This topic describes the prerequisites for System SSL on the IBM i, as well as some helpful tips.

Application security with SSL
A number of applications on IBM i can be secured with SSL. Refer to each application's documentation for details.

Related tasks

Configuration details: Secure a client connection to your Management Central system with SSL
This topic shows the expanded configurations steps for using SSL to secure a client connection to your Management Central server.

Configuration details: Secure all connections to your Management Central system with SSL
This topic shows the details for using SSL to secure all connections to your Management Central server.

Related information

Configuring DCM

Set up certificates for the first time

Configuration details: Secure all connections to your Management Central system with SSL

This topic shows the details for using SSL to secure all connections to your Management Central server.

The following information assumes that you have read through the following information: Scenario: Secure all connections to your Management Central server with SSL.

You now want to understand how to perform the steps required to secure all connections to the Management Central server. Follow along as Tom completes the scenario.

Before Tom can enable SSL on the Management Central system, he must install the prerequisite programs and set up digital certificates on the IBM i. Once he has met the prerequisites, he can complete the following procedures to secure all connections to the Management Central server.

Note: If SSL has been enabled for System i Navigator, Tom must disable it before he can enable SSL on the Management Central server. If SSL has been enabled for System i Navigator, and not the Management Central server, attempts by System i Navigator to connect with the central system will fail.

SSL allows Tom to secure transmissions between a central system and an endpoint system, as well as between the System i Navigator client and the central system. SSL provides transport and authentication of certificates and encryption of data. An SSL-connection can only occur between an SSL-enabled central system and an SSL-enabled endpoint system. Tom needs to configure server authentication before he can configure client authentication:

Related concepts

SSL prerequisites
This topic describes the prerequisites for System SSL on the IBM i, as well as some helpful tips.

Scenario: Securing all connections to your Management Central server with SSL
This scenario explains how to use SSL to secure all connections with an IBM i that is acting as a central system by using the System i Navigator Management Central system.

Related information

Set up certificates for the first time

Step 1: Configure the central system for server authentication

1. In System i Navigator, right-click **Management Central** and select **Properties**.
2. Click the **Security** tab and select **Use Secure Sockets Layer (SSL)**
3. Select **Server** as the authentication level.
4. Click **OK** to set this value on the central system.

Note: Do **NOT** restart the Management Central server until told to do so, later. If you restart the server now, you will not be able to contact your endpoint servers. You must complete more configuration tasks before the server can be restarted, activating SSL. You must propagate the SSL configuration to the endpoint systems first, with the compare and update task.

Step 2: Configure endpoint systems for server authentication

After Tom configures the central system for server authentication, he needs to configure the endpoint systems for server authentication. He completes the following tasks:

1. Expand **Management Central**.
2. Compare and update system values for the endpoint systems:
   a) Under **Endpoint Systems**, right-click the central system and select **Inventory** > **Collect**.
   b) Check the **System Values** option on the collect dialog box, in order to collect the system values inventory for the central system. Deselect any other options. Click OK and wait for the inventory task to complete.
   c) Right-click **System Groups** > **New System Group**.
   d) Define a new system group that includes all the endpoint systems to connect to, using SSL. Name this new system group 'Trusted Group.'
   e) To display the new group, 'Trusted Group,' expand the list of system groups.
   f) After the collection is complete, right-click the new system group and select **System Values** > **Compare and Update**.
   g) Verify that the central system displays in the **Model System** field.
   h) In the **Category** field, select **Management Central**.
   i) Verify that **Use Secure Sockets Layer** is set to **Yes** and select **Update** to propagate this value to the 'Trusted Group'.
   j) Verify that **SSL Authentication Level** is set to **Server** and select **Update** to propagate this value to the 'Trusted Group'.

      Note: If these values are not set, complete Step 1: Configure the central system for server authentication.
   k) Click **OK**. Wait until the **Compare and Update** completes processing before continuing to the next step.

Step 3: Restart the Management Central system on the central system

1. In System i Navigator, expand **My Connections**.
2. Expand the central system.
3. Expand **Network** > **Servers** and select **TCP/IP**.
4. Right-click **Management Central** and select **Stop**. The central system view collapses, and a message displays, explaining that you are not connected to the server.
5. Once the Management Central server has stopped, click **Start** to restart it.

Step 4: Restart the Management Central system on all endpoint systems

1. In System i Navigator, expand **My Connections**.
2. Expand the endpoint system that you are restarting.
3. Expand **Network** > **Servers** and select **TCP/IP**.
4. Right-click **Management Central** and select **Stop**.
5. Once the Management Central server has stopped, click **Start** to restart it.
6. Repeat this procedure for each endpoint system.

Step 5: Activate SSL for the System i Navigator client

1. In System i Navigator, expand **My Connections**.

2. Right-click the central system, and select **Properties**.
3. Click the **Secure Sockets** tab and select **Use Secure Sockets Layer (SSL) for connection**.
4. Exit System i Navigator and restart it.

Note: After you have completed these steps, server authentication is configured for your central and endpoint systems. You can optionally configure your central and endpoint systems for client authentication as well. Steps 6 through 10 should be completed if you want to enable client authentication on your central and endpoint systems.

Step 6: Configure the central system for client authentication

Now that Tom has completed the configuration for server authentication, he can opt to perform the following optional client authentication procedures. Client authentication provides validation of Certificate Authority and trusted group for both the endpoint systems and the central system. When the central system (SSL client) tries to use SSL to connect to an endpoint system (SSL server), the central system and the endpoint system authenticate each other's certificates through both server authentication and client authentication. This is also referred to as Certificate Authority and Trusted Group authentication.

Note: You cannot complete client authentication configuration until you have configured server authentication. If you have not configured server authentication, go back and do so, now.

1. In System i Navigator, right-click **Management Central** and select **Properties**.
2. Click the **Security** tab and select **Use Secure Sockets Layer (SSL)**.
3. Select **Client and server** for the authentication level.
4. Click **OK** to set this value on the central system.

   Note: Do **NOT** restart the Management Central server until told to do so, later. If you restart the server now, you will not be able to contact your endpoint servers. You must complete more configuration tasks before the server can be restarted, activating SSL. You must propagate the SSL configuration to the endpoint systems first, with the compare and update task.

Step 7: Configure endpoint systems for client authentication

Compare and update system values for the endpoint systems:

1. Expand **Management Central**.
2. Compare and update system values for the endpoint systems:
   a) Under **Endpoint Systems**, right-click the central system and select **Inventory** > **Collect**.
   b) Check the **System Values** option on the collect dialog box, in order to collect the system values inventory for the central system. Deselect any other options. Click OK and wait for the inventory task to complete.
   c) After the collection is complete, right-click the 'Trusted Group' and select **System Values** > **Compare and Update**.
   d) Verify that the central system displays in the **Model System** field.
   e) In the **Category** field, select **Management Central**.
   f) Verify that **Use Secure Sockets Layer** is set to **Yes** and select **Update** to propagate this value to the 'Trusted Group'.
   g) Verify that **SSL Authentication Level** is set to **Client and Server** and select **Update** to propagate this value to the 'Trusted Group'.

      Note: If these values are not set, complete Step 6: Configure the central system for client authentication..
   h) Click **OK**. Wait until the **Compare and Update** completes processing before continuing to the next step.

Step 8: Copy the validation list to the endpoint systems

This task assumes that your central system is running IBM i V5R3, or later. On OS/400 V5R2, or earlier systems, QYPSVLDL.VLDL was located in QUSRSYS.LIB, not QMGTC2.LIB. Therefore, if you have pre-V5R3 systems, you will need to send the validation list to these systems and place it in QUSRSYS.LIB, instead of QMGTC2.LIB. For V5R3 and greater systems, continue with the following steps:

1. In System i Navigator, expand **Management Central** > **Definitions**.
2. Right-click **Package**, and select **New Definition**.
3. In the **New Definition** window, work with the following:
   a) **Name:** Type the name of the definition.
   b) **Source system:** Select the name of the central system.
   c) **Selected files and folders:** Click in the field, and type `/QSYS.LIB/QMGTC2.LIB/QYPSVLDL.VLDL`.
4. Click the **Options** tab, and select **Replace existing file with the file being sent**.
5. Click **Advanced**.
6. In the **Advanced Options** window, specify **Yes** to allow object differences on restore, and change the **Target release** to be the earliest release of your endpoints.
7. Click **OK** to refresh the list of definitions and display the new package.
8. Right-click the new package, and select **Send**.
9. In the **Send** dialog box, expand **System Groups->Trusted Group**, located in the **Available Systems and Groups** list. This group is the one you defined in "Step 2: Configure endpoint systems for server authentication" on page 30.

   Note: The **Send** task will always fail on the central system, because it is always the source system. The **Send** task should complete successfully on all endpoint systems.
10. If you have any systems running IBM i V5R3, or earlier, in **Trusted Group**, you must manually go to those systems and move the `QYPSVLDL.VLDL` object from `QMGTC2.LIB` to `QUSRSYS.LIB`. If there is already a version of `QYPSVLDL.VLDL` in `QUSRSYS.LIB`, delete it and replace it with the newer one from `QMGTC2.LIB`

Step 9: Restart the Management Central system on the central system

1. In System i Navigator, expand **My Connections**.
2. Expand the central system.
3. Expand **Network** > **Servers** and select **TCP/IP**.
4. Right-click **Management Central** and select **Stop**. The central system view collapses, and a message displays, explaining that you are not connected to the server.
5. Once the Management Central server has stopped, click **Start** to restart it.

Step 10: Restart the Management Central system on all endpoint systems

Note: Repeat this procedure for each endpoint system.

1. In System i Navigator, expand **My Connections**.
2. Expand the endpoint system that you are restarting.
3. Expand **Network** > **Servers** and select **TCP/IP**.
4. Right-click **Management Central** and select **Stop**.
5. Once the Management Central server has stopped, click **Start** to restart it.

Troubleshooting SSL

This very basic troubleshooting information is intended to help you reduce the list of possible problems that the IBM i platform can encounter with SSL.

It is important to understand that this is not a comprehensive source for troubleshooting information, but rather a guide to aid in common problem resolution.

Verify that the following statements are true:

- You have met the prerequisites for SSL on the IBM i platform.
- Your certificate authority and certificates are valid and have not expired.

If you have verified that the previous statements are true for your system and you still have an SSL-related problem, try the following options:

- The SSL error code in the server job log can be cross referenced in an error table to find more information about the error. For example, this table maps the `-93` that might be seen in a server job log to the constant `SSL_ERROR_SSL_NOT_AVAILABLE`.
  - A negative return code (indicated by the dash before the code number) indicates that you are using an SSL_ API.
  - A positive return code indicates that you are using a GSKit API. Programmers can code the `gsk_strerror()or SSL_Strerror()` API in their programs to obtain a brief description of an error return code. Some applications make use of this API and print out a message to the job log containing this sentence.

  If more detailed information is required, the message id provided in the table can be displayed on an IBM i to show potential cause and recovery for this error. Additional documentation explaining these error codes may be located in the individual secure socket API that has returned the error.
- Additional information about the last certificate validation error on the current secure session can be retrieved by using the GSK_LAST_VALIDATION_ERROR attribute on `gsk_attribute_get_numeric_value()`. If `gsk_secure_soc_init()` or `gsk_secure_soc_startInit()` returned an error, this attribute might provide more error information.
- The following two header files contain the same constant names for System SSL return codes as the table, but without the message ID cross reference:
  - `QSYSINC/H.GSKSSL`
  - `QSYSINC/H.QSOSSL`

  Remember that although the names of the System SSL return codes remain constant in these two files, more than one unique error can be associated with each return code.

Related concepts

SSL prerequisites
This topic describes the prerequisites for System SSL on the IBM i, as well as some helpful tips.

Related information

Secure socket API error code messages

Related information for SSL

Use this information to learn about other resources and information relevant to using Secure Sockets Layer (SSL).

Web sites

- RFC 5246: "The Transport Layer Security (TLS) Protocol Version 1.2" (http://www.ietf.org/rfc/rfc5246.txt)

  Explains Version 1.2 of the TLS protocol in detail.
- RFC 4346: "The Transport Layer Security (TLS) Protocol Version 1.1" (http://www.ietf.org/rfc/rfc4346.txt)

  Explains Version 1.1 of the TLS protocol in detail.
- RFC 2246: "The TLS Protocol Version 1.0 " (http://www.ietf.org/rfc/rfc2246.txt)

  Explains Version 1.0 of the TLS protocol in detail.
- RFC2818: "HTTP Over TLS" (http://www.ietf.org/rfc/rfc2818.txt)

  Describes how to use TLS to secure HTTP connections over the Internet.
- RFC 5746: "Transport Layer Security (TLS) Renegotiation Indication Extension" (http://www.ietf.org/rfc/rfc5746.txt)

  Defines the TLS renegotiation indication extension.
- RFC 6066: "Transport Layer Security (TLS) Extensions: Extension Definitions" (http://www.ietf.org/rfc/rfc6066.txt)

  Defines TLS extensions.
- RFC 2560: "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP" (http://www.ietf.org/rfc/rfc2560.txt)

  Defines OCSP which is used to determine revocation status for digital certificates.
- RFC 4492: "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)" (http://www.ietf.org/rfc/rfc4492.txt)

  Defines new key exchange algorithms for TLS.

Other information

- SSL and Java Secure Socket Extension
- IBM Toolbox for Java

Notices

Trademarks

Terms and conditions