Platform SDK Developer's Guide

Configuring TLS Parameters in Configuration Manager

4/8/2026

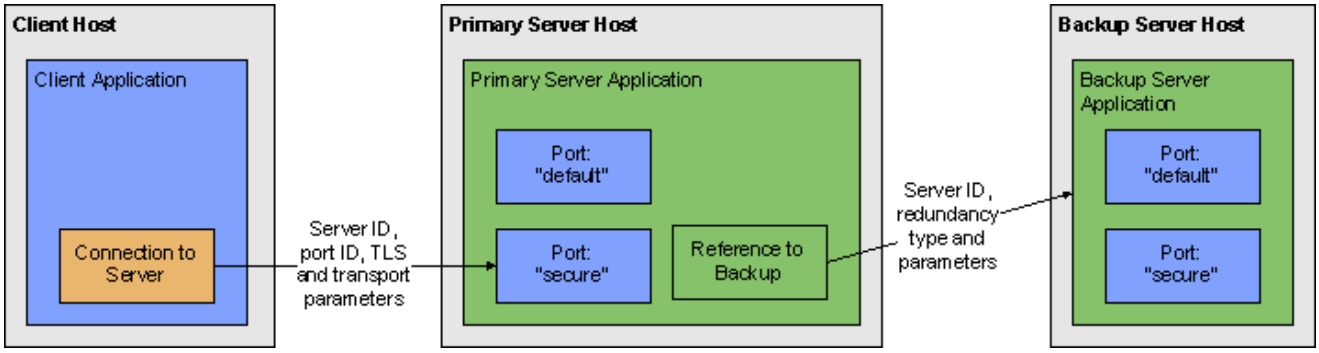
Contents

Configuring TLS Parameters in Configuration Manager

Introduction

As described earlier, the Platform SDK Application Template Application Block allows both client and server applications to read TLS parameters from configuration objects. This page describes how to set TLS parameters correctly in those configuration objects.

Configuration objects that will be used, and their relations, are shown in the diagram below:



To edit TLS-related parameters for these objects, you will need to have access to the Annex tab in Configuration Manager.

Precedence of Configuration Objects

Platform SDK uses different sets of configuration objects to configure client- and server-side TLS settings. For TLS parameters, these objects are searched from the most specific object to the most general one. Parameters found in specific objects take precedence over those in more general objects.

Note: This search occurs independently for each supported TLS parameter.

Location of specific TLS parameters can differ for each object, but is detailed in the appropriate section on this page.

Configuration Object Precedence

Application type	Configuration Objects Used, in Order of Precedence
Client	1. Connection from the client application to the server.

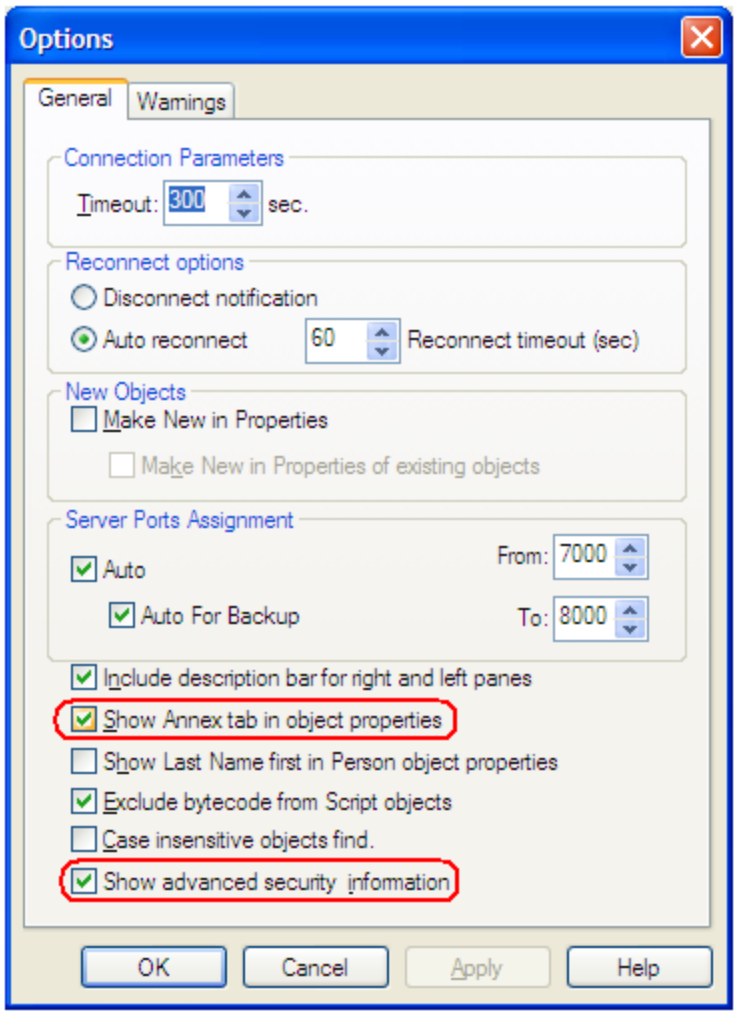
Application type	Configuration Objects Used, in Order of Precedence
	2. Application of the client. 3. Host where client application resides. 4. Port of the target server that client connects to.[1]
Server	1. Port of the server application. 2. Application of the server. 3. Host where the server application resides.

1. If the `tls` parameter is not set to 1 in both the client Application and Connection objects, then the client application will look to the Port object for the target server to determine if TLS should be turned on. Configuration Manager does not automatically add the `tls`=1 parameter to Connection Transport parameters when it is linked to a server's secure Port. This is the only case when a client application considers settings in the server's configuration objects.

Displaying the Annex Tab in Configuration Manager

By default, Configuration Manager does not show Annex tab in Object Properties windows. This tab can contain TLS parameters for Host and Application objects.

To show the Annex tab, select *View > Options...* from the main menu and ensure the *Show Annex tab in object properties* and *Show Advanced Security Information* options are selected.



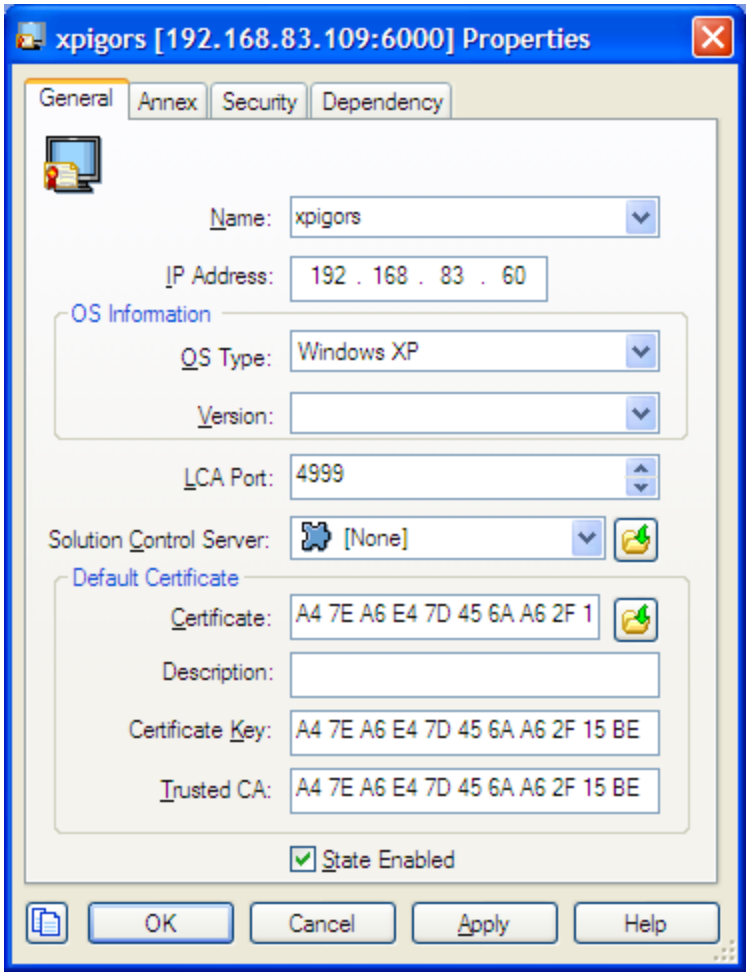
Application Objects

Host Object

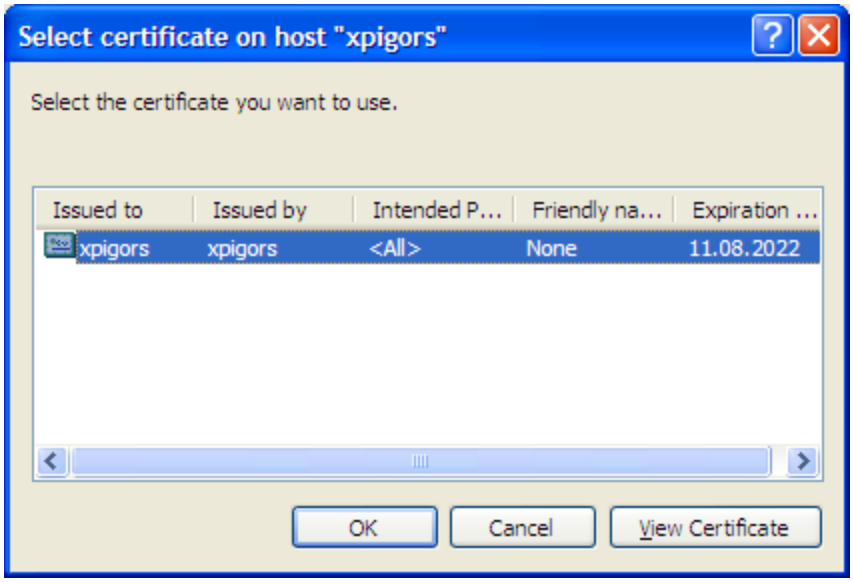
The properties window for a Host object includes most common TLS parameters on the General tab:

- Certificate
- Certificate Key
- Trusted CA

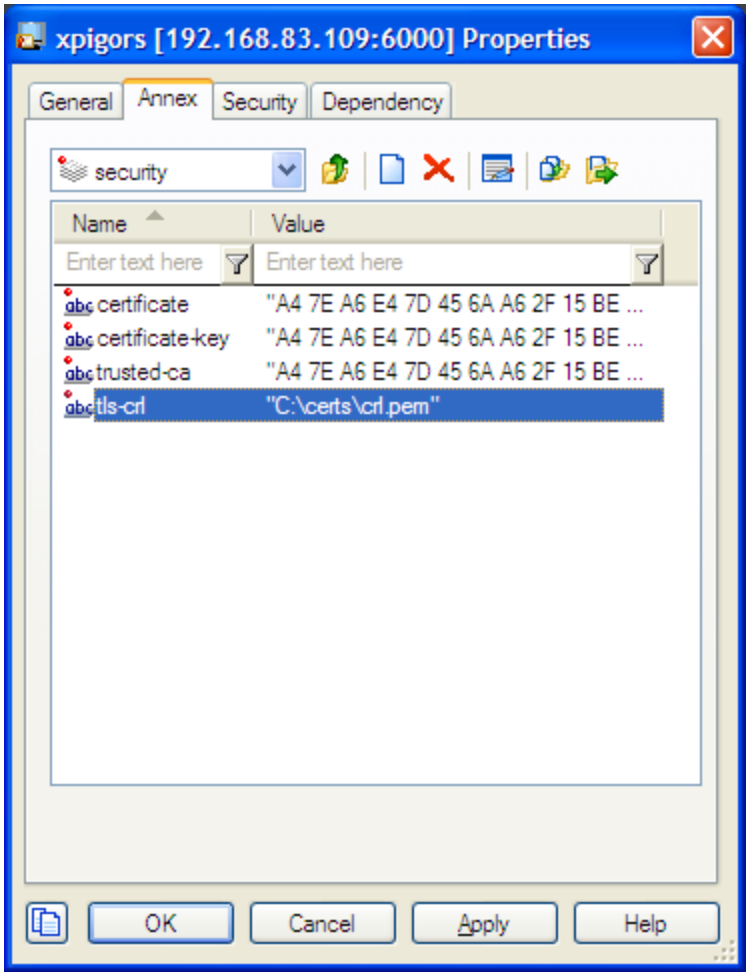
These fields allow copy/paste operations, so they can be set manually by copying and pasting the "Thumbprint" field values from certificates in Windows Certificate Services (WCS) into the related field in Configuration Manager.



To select a certificate, use the button next to *Certificate* field. This opens the *Select certificate* window, displaying a list of certificates installed in WCS under the Local Computer account for the local machine.

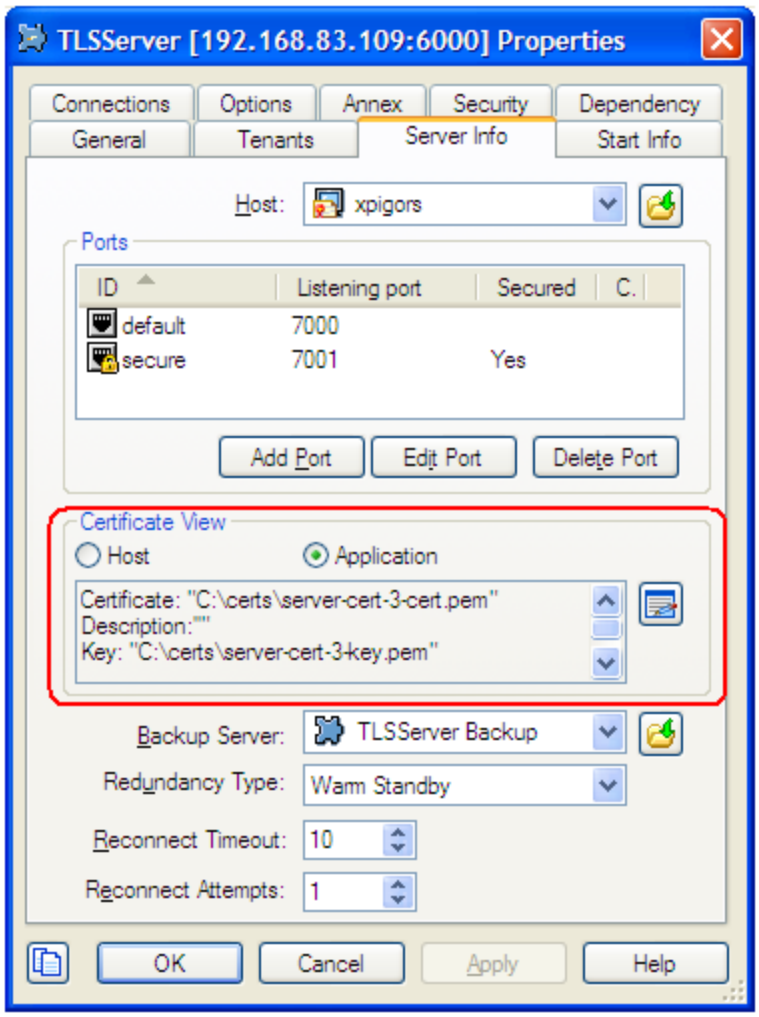


The Annex tab contains a security section that holds TLS settings for this object. Any change made to TLS-related fields on the General tab are mirrored between the Annex tab automatically. You can also specify additional TLS parameters here that aren't reflected on the General tab.

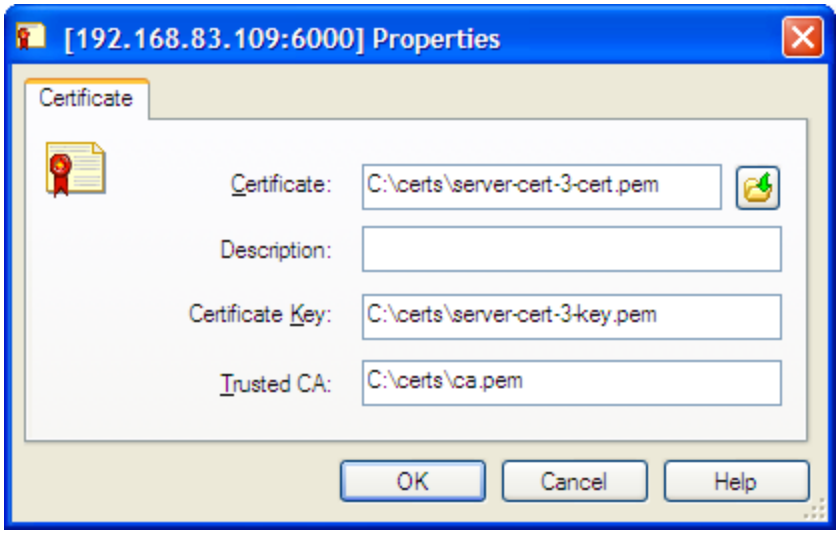


Server Application Object

For the server Application object, TLS-related fields are located on the *Server Info* tab of the properties window. Note the *Certificate View* controls group, where the server can be set to use Host TLS parameters (generally recommended for Genesys Framework) or application-specific ones.

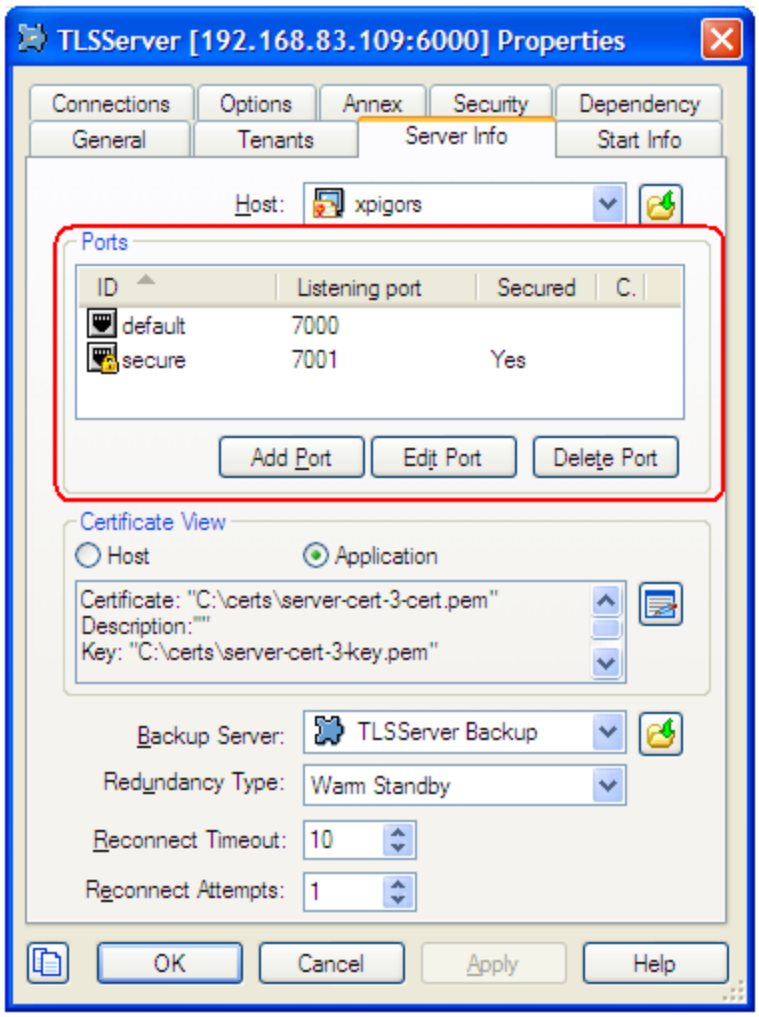


If using application-specific TLS parameters, use the button next to the certificate information field to open a certificate selection window where you can choose from a list of certificates installed for the Local Computer account or manually enter certificate information:



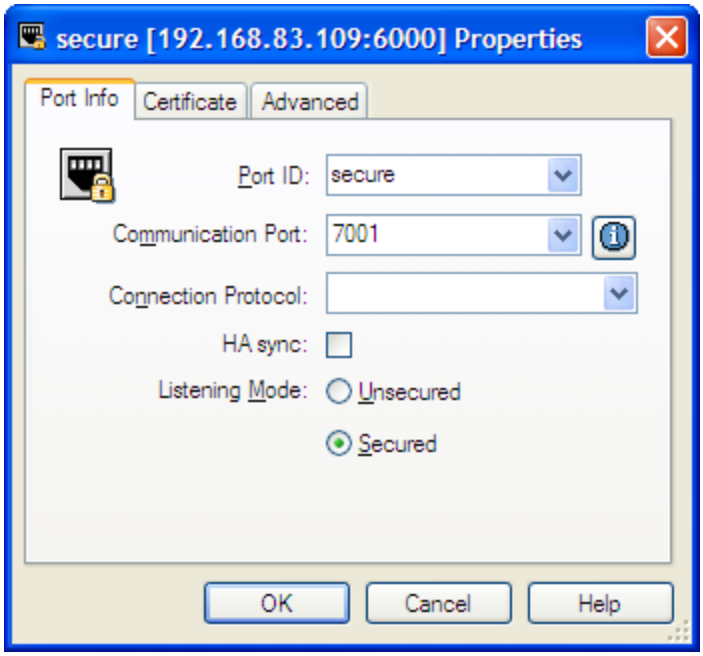
Port Object

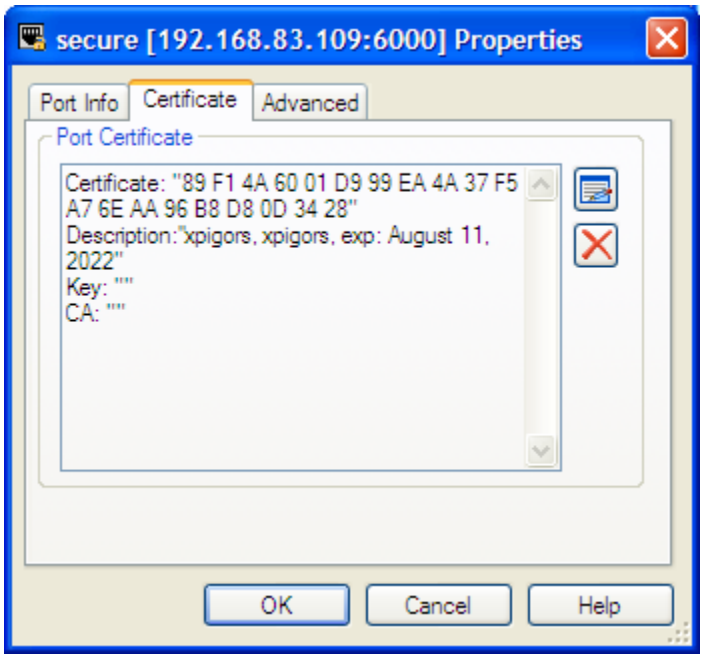
For port objects, TLS-related fields are located on the *Server Info* tab of the properties window. You can see here whether a port is secured (TLS-enabled) or not, and have the option to edit existing ports to update TLS parameters or to add new ports.

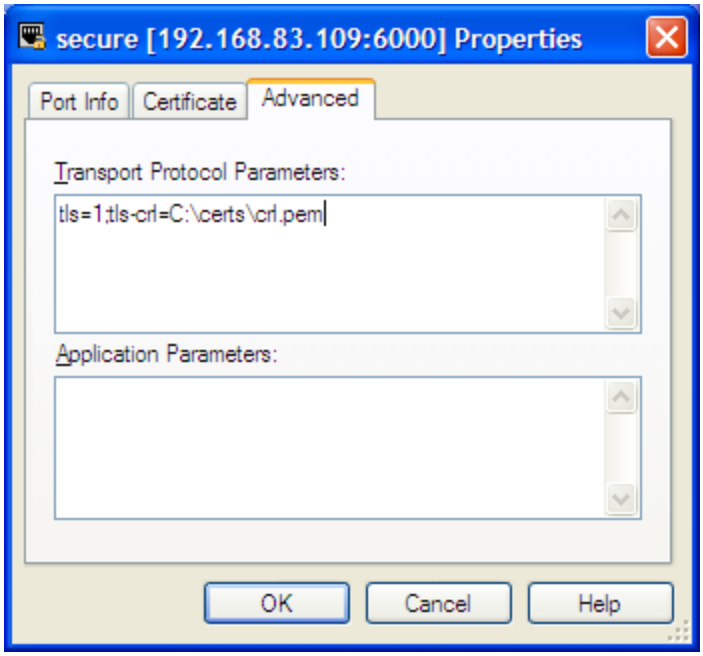


When adding or editing a port, TLS parameters are specified on the following tabs:

- *Port Info* — Turn on *Secured* listening mode for the port (the same as adding the *tls=1* string to transport parameters).
- *Certificate* — Show certificate information, open a certificate selection window, or delete the current certificate information.
- *Advanced* — Manually edit the *Transport Protocol Parameters* field. TLS parameters not reflected on the *Certificate* tab can be added here.

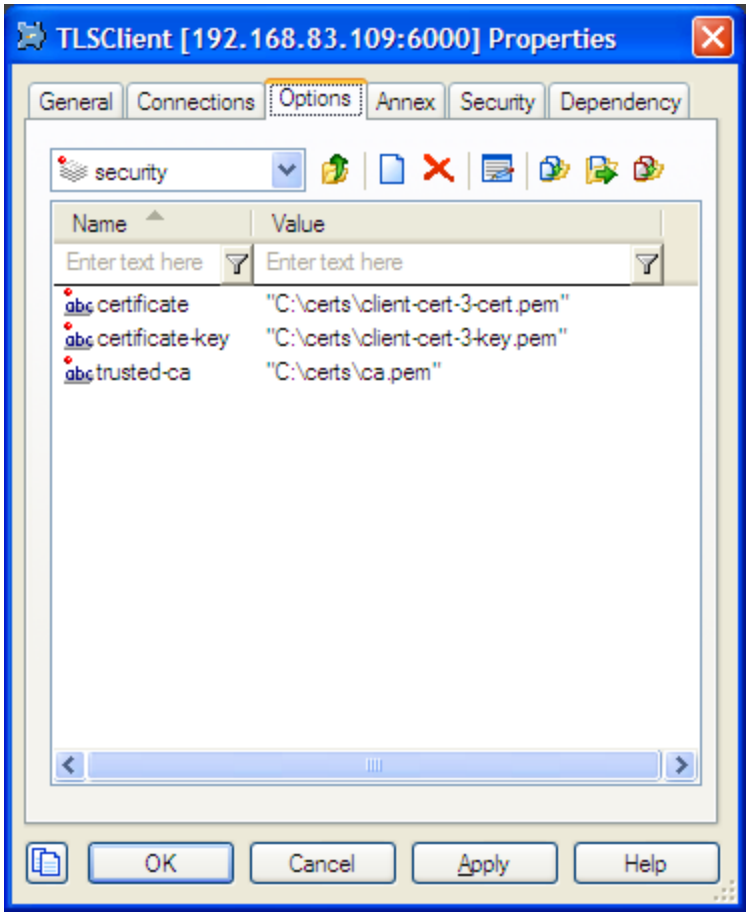

secure [192.168.83.109:6000] Properties
Port Info
Certificate
Advanced
Port ID:
secure
Communication Port:
7001
Connection Protocol:
HA sync:
Listening Mode:
Unsecured
Secured
OK
Cancel
Help

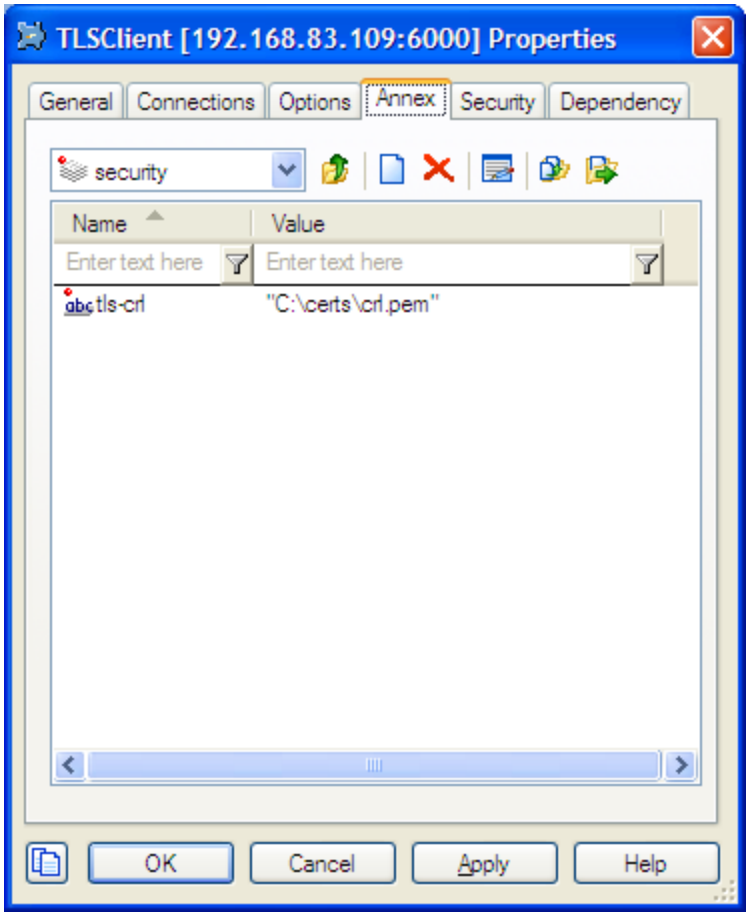

secure [192.168.83.109:6000] Properties
Port Info
Certificate
Advanced
Port Certificate
Certificate: "89 F1 4A 60 01 D9 99 EA 4A 37 F5 A7 6E AA 96 B8 D8 0D 34 28"
Description:"xpigors, xpigors, exp: August 11, 2022"
Key: ""
CA: ""
OK
Cancel
Help



Client Application Object

For client Application objects, TLS-related fields are located under the *security* sections of both the *Options* and *Annex* tabs. There is no certificate selection window provided, but TLS parameters can be configured manually in either section.

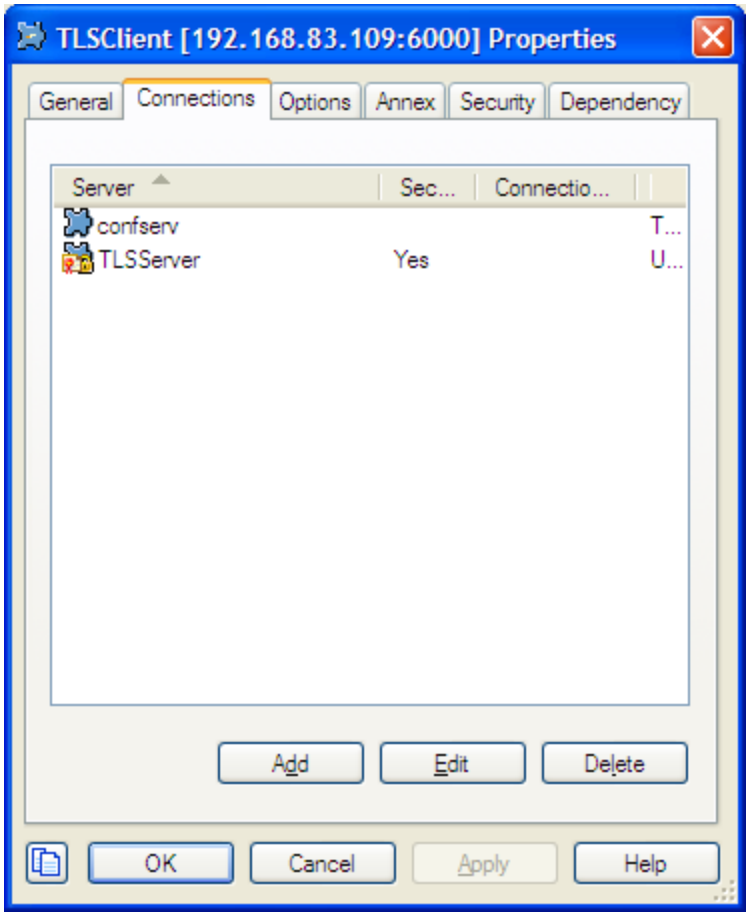

TLSClient [192.168.83.109:6000] Properties
General
Connections
Options
Annex
Security
Dependency
security
Name
Value
Enter text here
Enter text here
certificate
"C:\certs\client-cert-3-cert.pem"
certificate-key
"C:\certs\client-cert-3-key.pem"
trusted-ca
"C:\certs\ca.pem"
OK
Cancel
Apply
Help

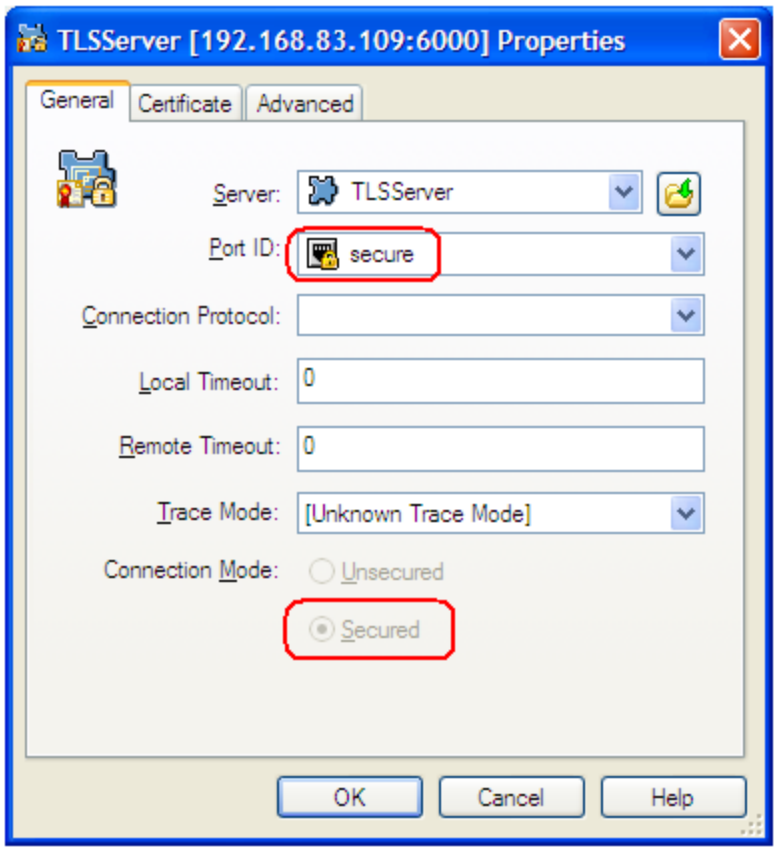


When processing a client Application object, Platform SDK looks at parameters from both sections. If any parameters are specified in both places, then the values from the *Options* tab take precedence.

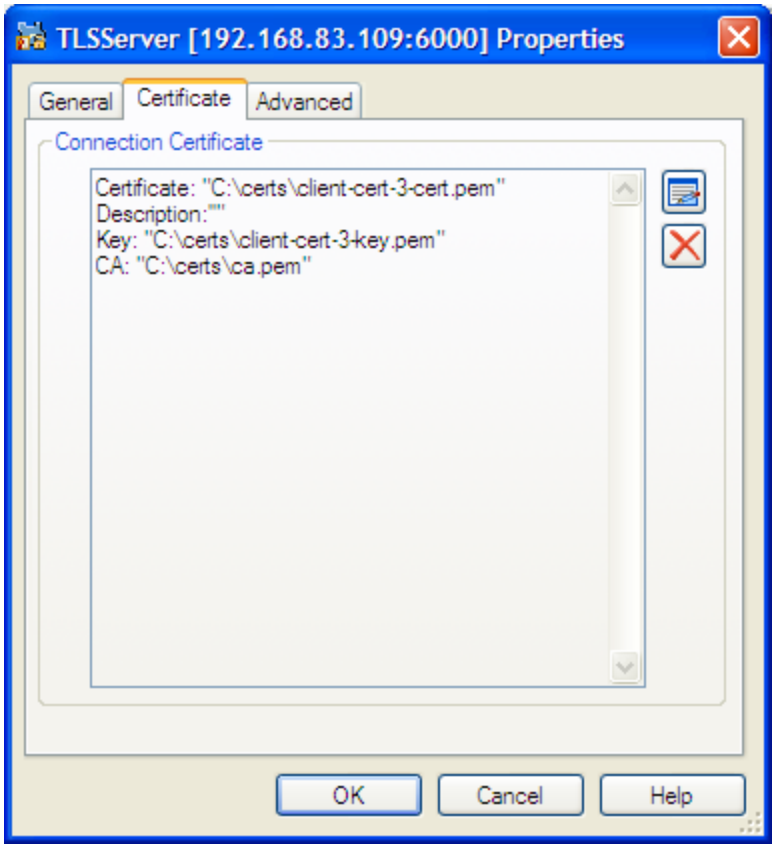
Connection Object

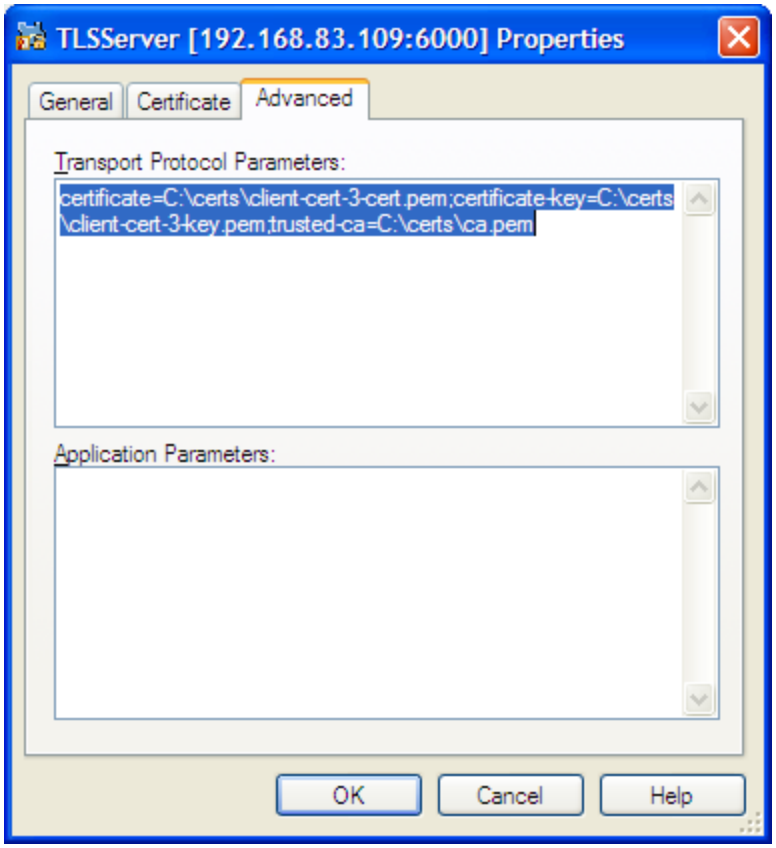
The properties window for all Application objects includes a *Connection* tab where connections to servers can be added or edited. Each connection determines if TLS mode should be enabled based on port settings for the target server.


TLSClient [192.168.83.109:6000] Properties
General
Connections
Options
Annex
Security
Dependency
Server
Sec...
Connectio...
confserv
T...
TLSServer
Yes
U...
Add
Edit
Delete
OK
Cancel
Apply
Help



Similar to the *Port* properties window, the *Certificate* tab allows you to select from a list of certificates or manually edit certificate properties. You can also use the *Advanced* tab to edit TLS settings not included with the certificate. However, the *Transport Protocol Parameters* field behaves differently for this object — which may result in lost or incorrect settings in some cases. See the Notes and Issues section for details.


TLSServer [192.168.83.109:6000] Properties
General
Certificate
Advanced
Connection Certificate
Certificate: "C:\certs\client-cert-3-cert.pem"
Description:""
Key: "C:\certs\client-cert-3-key.pem"
CA: "C:\certs\ca.pem"
OK
Cancel
Help



List of TLS Parameters

The following table lists all TLS parameters supported by Platform SDK, with their valid value ranges and purpose:

Parameter Name	Acceptable Values	Purpose	Platform
tls	Boolean value. Possible values are "1"/"0", "yes"/"no", "on"/"off", "true"/"false". Example: • "tls=1"	Client: 1 - perform TLS handshake immediately after connecting to server. 0 – do not turn on TLS immediately but autodetect can still work.	Java, .NET
provider	"PEM", "MSCAPI", "PKCS11", "JKS" Not case-sensitive. Example:	Explicit selection of security provider to be used. For example, MSCAPI and PKCS11 providers can contain all other parameters in their internal database.	Java

Parameter Name	Acceptable Values	Purpose	Platform
	• "provider=MSCAPI"	This parameter allow configuration of TLS through security provider tools.	
certificate	**Java:** PEM provider: path to a X.509 certificate file in PEM format. Path can use both forward and backward slash characters. MSCAPI provider: thumbprint of a certificate – string with hexadecimal SHA-1 hash code of the certificate. Whitespace characters are allowed anywhere within the string. PKCS11 provider: this parameter is ignored. JKS provider: path to a java key store file. If 'provider' option isn't specified implicitly then the file must have 'jks' extension. Examples: • "certificate= C:\certs\client-cert-3-cert.pem" • "certificate=A4 7E A6 E4 7D 45 6A A6 2F 15 BE 89 FD 46 F0 EE 82 1A 58 B9" • "certificate= C:\certs\ mykeystore.jks" **.NET:** Thumbprint of a certificate – string with hexadecimal SHA-1 hash code of the certificate (Whitespace characters are allowed anywhere within the string).	Specifies location of X.509 certificate to be used by application. MSCAPI provider keeps certificates in internal database and can identify them by hash code; so called thumbprint. In Java, PKCS#11 provider does not allow selection of the certificate; it must be configured using provider tools. **Note:** When using autodetect (upgrade) TLS connection, this option MUST be specified in application configuration, otherwise Configuration Server would return empty TLS parameters even if other options are set.	Java, .NET
certificate-key	PEM provider: path to a PKCS#8 private key file without password protection in PEM format. Path can use both vforward and backward slash characters. • MSCAPI provider: this	Specifies location of PKCS#8 private key to be used in pair with the certificate by application. MSCAPI provider keeps private keys paired with certificates in internal database. In Java, PKCS#11 provider does not allow selection of the private	Java

Parameter Name	Acceptable Values	Purpose	Platform
	parameter is ignored; key is taken from the entry identified by "certificate" field. • PKCS11 provider: this parameter is ignored. • JKS provider: this parameter must not be used. Examples: • "certificate-key= C:\certs\client-cert-3-key.pem"	key; it must be configured using provider tools.	
trusted-ca	PEM provider: path to a X.509 certificate file in PEM format. Path can use both forward and backward slash characters. MSCAPI provider: thumbprint of a certificate – string with hexadecimal SHA-1 hash code of the certificate. Whitespace characters are allowed anywhere within the string. PKCS11 provider: this parameter is ignored. Examples: • "trusted-ca= C:\certs\ ca.pem" • "trusted-ca=A4 7E A6 E4 7D 45 6A A6 2F 15 BE 89 FD 46 F0 EE 82 1A 58 B9"	Specifies location of a X.509 certificate to be used by application to validate remote party certificates. The certificate is designated as Trusted Certification Authority certificate and application will only trust remote party certificates signed with the CA certificate. MSCAPI provider keeps CA certificates in internal database and can identify them by hash code; so called thumbprint. In Java, PKCS#11 provider does not allow selection of the CA certificate; it must be configured using provider tools.	Java
tls-mutual	Boolean value. Possible values are "1"/"0", "yes"/"no", "on"/"off", "true"/"false". Example: • "tls-mutual=1"	Has meaning only for server application. Client applications ignore this value. When turned on, server will require connecting clients to present their certificates and validate the certificates the same way as client applications do.	Java, .NET

Parameter Name	Acceptable Values	Purpose	Platform
tls-crl	**Java:** All providers: path to a Certificate Revocation List file in PEM format. Path can use both forward and backward slash characters. Example: • "tls-crl= C:\certs\crl.pem" **.NET:** Boolean value. Enable/disable to check of certificate's revocation. Possible values are "1"/"0", "yes"/"no", "on"/"off", "true"/"false". Example: • "tls-crl=1"	Applications will use CRL during certificate validation process to check if the (seemingly valid) certificate was revoked by CA. This option is useful to stop usage of leaked certificates by unauthorized parties.	Java, .NET
tls-target-name-check	"host" or none. Not case-sensitive. Example: • "tls-target-name-check=host"	When set to "host", enables matching of certificate's Alternative Subject Name or Subject fields against expected host name. PSDK supports DNS names and IP addresses as expected host names.	Java, .NET
cipher-list	String consisting of space-separated cipher suit names. Information on cipher names can be found online. Example: • "cipher-list= TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA"	Used to calculate enabled cipher suites. Only ciphers present in both the cipher suites supported by security provider and the cipher-list parameter will be valid.	Java
fips140-enabled	Boolean value. Possible values are "1"/"0", "yes"/"no", "on"/"off", "true"/"false". Example: • "fips140-enabled=1"	PSDK Java: when set to true, effectively is the same as setting "provider=PKCS11" since only PKCS11 provider can support FIPS-140. If set to true while using other	Java

Parameter Name	Acceptable Values	Purpose	Platform
		provider type, PSDK will throw exception.	
sec-protocol	String value. Possible values are "SSLv23", "SSLv3", "TLSv1", "TLSv11", "TLSv12". Example: • "sec-protocol= TLSv1"	Starting with PSDK release 8.5.1, an application can specify the exact protocol to send and accept secure connection requests on one or more of its connections.	Java, .NET

tls-version	String value.
	
	This value is the algorithm name used to get an instance of Java `SSLContext`. It is used to create the Java `SSLEngine` used to handle a security protocol.
	
	Standard algorithm names supported by standard java security provider are available in the official Java documentation.
	
	This parameter must be used together with the `protocol-list` option, and was a legal and powerful way to choose a provider before the `sec-protocol` parameter was available.
	
	Examples:
	
	`tls-version=TLSv1.2 protocol-list= TLSv1.2`
	
	`tls-version=TLSv1.2 protocol-list= SSLv3 TLSv1.1 TLSv1.2`
	
	`tls-version=TLSv1.2 protocol-list= TLSv1`
		Together with the `protocol-list` parameter, this value helps to specify one or more security protocols
		
		that can be used (if the Java platform supports it).
		
		This option specifies a Java security protocol name that is used to select an appropriate Java security provider. The actual list of security protocols that can be used depends on:
		
		• Which Java security provider will be selected.
		• Which security protocols are supported by the security provider. Sometimes this can be customizable, such as by using the Windows Control Panel > Java > Advanced tab.
		• Which security protocols are enabled for use. This is specified in the `protocol-list` parameter.
		
		The default value depends on version of Java in use, and its configuration.
		
		• “TLSv1” for Java 6 and 7
		• “TLSv1.2” for Java 8
			Java

Parameter Name	Acceptable Values	Purpose	Platform

protocol-list	String
	
	Java:
	
	A space-separated list of security protocols that can be used. This value is passed to the `setEnabledProtocols` method of the Java `SSLEngine`, and is the standard way to choose a security provider that supports the requested protocol version from the system providers list.
	
	This parameter must be used together with the `tls-version` option, and was a legal and powerful way to choose a provider before the `sec-protocol` parameter was available.
	
	Examples:
	
	`tls-version=TLSv1.2 protocol-list= TLSv1.2`
	
	`tls-version=TLSv1.2 protocol-list= SSLv3 TLSv1.1 TLSv1.2`
	
	`tls-version=TLSv1.2 protocol-list= TLSv1`
	
	.NET:
	
	A comma-separated list of security protocols that can be used.
		Together with the `tls-version`, helps to specify one or more security protocols that can be used (if the Java platform supports it).
		
		Restricts use of supported security protocols by the selected security provider.
			Java, .NET

Warning

Currently there is no support for options that store JKS keystore, entry and truststore passwords. Thus configuration of the passwords has to be completed programmatically using the following `TLSConfiguration` methods:

- `setKeyStoreCallBackHandler`
- `setKeyStoreEntryCallBackHandler`
- `setTrustStoreCallback`

Notes and Issues

- Key/value pairs in *Transport Protocol Parameters* fields should be separated only with a single semicolon character. Adding space characters to improve readability can cause applications, including those based on Platform SDK, unable to parse these parameters correctly.
- *Transport Protocol Parameters* fields in Configuration Manager are limited to 256 characters in length. Be sure to keep your parameter list as short as possible. For example: certificate thumbprints for MSCAPI provider take 40 characters without spaces and 49 characters with them, and long paths to certificate files can easily eat up all available space.
- The Connection properties window behaves differently from the Port properties window, as described below. Be sure to double-check TLS settings for Connection objects.
  - It does not save content of the *Transport Protocol Parameters* field unless a certificate was selected using UI controls on the *Certificate* tab.
  - If certificate information is deleted from the *Certificate* tab, then all transport protocol parameters are also erased (including those entered manually).
  - In some cases it does not save additional TLS parameters that were entered manually.
- Configuration Server reads its own TLS parameters from Application or from Host object only during startup. If you use an Application or Host object as a source of TLS parameters for Configuration Server, be sure to restart the server after any changes to the parameters.