Article
Katherine Reid · Dec 6, 2016 7m read

Configuring Caché client applications for SSL/TLS

When using Studio, ODBC or a terminal connection to Caché or Ensemble, you may have wondered how to secure the connection. One option is to add TLS (aka SSL) to your connection. The Caché client applications - TELNET, ODBC and Studio - all understand how to add TLS to the connection. They just need to be configured to do it.

Configuring these clients is easier in 2015.1 and later. I'm going to be discussing this new method. If you're already using the old, legacy method, it will continue to work, but I would recommend you consider switching to the new one.

Background

These client applications can be installed on a machine which does not have the server install. They can't depend on having access to the normal places to store settings, such as the CACHESYS database or cpf file. Instead, their settings for which certificates or protocols to accept are stored in a text file. Many of the settings in this file are similar to settings in an SSL/TLS configuration in the management portal.

Where is the settings file?

You will have to create your own file. The client installer doesn't create one for you.

By default the settings file is called SSLDefs.ini and should be put in the InterSystems\Cache directory under the directory for 32-bit common program files. This directory is found in the Windows environment variable *CommonProgramFiles(x86)* on 64-bit Windows or *CommonProgramFiles* on 32-bit Windows.

For example, on Windows 8.1, the default file would be:

```
C:\Program Files (x86)\Common Files\InterSystems\Cache\SSLdefs.ini
```

If you would like to change this, you will have to tell the client executables where to find the settings file. You can do this by defining the environment variable *ISC_SSLconfigurations* and setting it to the entire path and file name of your file. You may need administrator permissions to do this.

What's in the settings file?

The file has two types of sections. The first type matches connections with TLS configurations. For example, it might tell Studio to use the section named "Default Settings" to find its TLS parameters when connecting to development.intersystems.com.

The second type defines the TLS settings to use for the connection. For example, these would define what Certificate Authority to expect the server's certificate to be signed by. The settings in these sections are very similar to the settings in an SSL/TLS configuration on a Caché or Ensemble server.

The first type of section looks like this:

```
[Development Server]
Address=10.100.0.17
Port=1972
TelnetPort=23?
SSLConfig=DefaultSettings?
```

The name in brackets can be anything you want. It's only there to make it easier for you to keep track of which connection this is.

The Address, Port, and TelnetPort settings are used to decide which connections should match this section. Either IP addresses or DNS names can be used for the address on 2016.1 or later clients. Both the address and either the Port or TelnetPort must match where the client application is connecting to in order for the configuration to be used.

The final parameter (SSLConfig) is the name of the configuration to get TLS settings from. It needs to match the name of one of the configurations in the file.

The second type of section looks like this:

```
[DefaultSettings]
VerifyPeer=2
VerifyHost=1
CAfile=c:\InterSystems\certificates\CAcert.pem
CertFile=c:\InterSystems\certificates\ClientCert.pem
KeyFile=c:\InterSystems\certificates\ClientKey.key
Password=
KeyType=2
Protocols=24
CipherList=ALL:!aNULL:!eNULL:!EXP:!SSLv2
```

The name of the section is listed on the first line: [DefaultSettings] and matches the name listed in the SSLConfig parameter of the example first section above. Therefore, this config will be used for connections to the 10.100.0.17 server on port 1972 or port 23.

Using copy and paste on the example above often causes non-printing characters in your text file. Please make sure you have removed any extra characters, for example, by saving the file as text only and re-opening it.

Here's a description of what the parameters mean:

- **VerifyPeer**

  Options for this are 0=none, 1=request, and 2=require. Require is the recommended value. If you choose none, it is possible for a malicious server to pretend to be the server you mean to connect to. If you choose require, you'll need to fill in a Certificate Authority that you trust to verify certificates for the CAFile value. This is the equivalent of "Server certificate verification" in the portal. (Note: request doesn't make sense for a client configuration, but I've included it here so you can understand why the options are 0 and 2.)

- **VerifyHost**

  Options for this are 0=none, 1=required. This option checks that the server's certificate lists the host name or IP you've asked to connect to in the Subject's Common Name or subjectAlternativeName fields. This field does not have an equivalent in the portal, but is the same type of check as the SSLCheckServerIdentity property of the %Net.HttpRequest class. It is only configurable if your client is using Caché / Ensemble

2018.1 or later, or any version of InterSystems IRIS Data Platform.

- **CAfile**

  The path to the trusted Certificate Authority (CA) file. This should be the CA that signed the certificate of the *other* side (the server), not your own certificate. This should be filled in if you have picked a VerifyPeer value of 2. This is the equivalent of "File containing trusted Certificate Authority certificate(s)" in the portal. Certificates must be in PEM format.

- **CertFile**

  The path to your own certificate. This should be blank if your client doesn't have one. This is the equivalent of "File containing this client's certificate" in the portal. Certificates must be in PEM format.

- **KeyFile**

  The path to the matching private key for CertFile. This should be filled in if you have a CertFile, and blank if you don't. This is the equivalent of "File containing associated private key" in the portal.

- **Password**

  The password needed to decrypt your private key. This should be blank if you're not using a certificate for this client, or if the certificate's private key is not encrypted on disk.

- **KeyType**

  Is your private key RSA (2) or DSA (1)? The value is only relevant for configurations which have CertFile and KeyFile set. If you're not sure which it is, your key is probably RSA.

- **Protocols**

  This is a decimal representation of bit values for the versions of SSL/TLS supported. The options are: 1=SSLv2, 2=SSLv3, 4=TLSv1, 8=TLSv1.1, 16=TLSv1.2. SSLv2 and SSLv3 have known problems and are not recommended. More than one version may be specfied by adding numbers. For example, 24 is TLSv1.1 and TLSv1.2. This the equivalent of the "Protocols" checkboxes in the portal. (Note: the 8 and 16 bits aren't in 2015.1. If you want to use them, you need to upgrade to 2015.2 or higher.)

- **CipherList**

  This is the equivalent of "Enabled ciphersuites" in the portal. This controls exactly which types of encryption and hashing will be acceptable to this client. **ALL:!aNULL:!eNULL:!EXP:!SSLv2** is the default value for this setting in the management portal. If you're having trouble with your connection, it's probably not this. Changing this can make your connection less secure by allowing weak encryption. You can find more information about this value on the openssl website.

Final notes

That's all you need to do! If you create your file and put it in the known location, it will automatically be used if the name or IP address and port you're connecting to match one of the connections listed in the file.

Server setup

This article is about how to configure the client side of your connection to use SSL, but don't forget that the server you're connecting to also needs to understand how to accept SSL. The documentation on setting the SuperServer to use SSL can be found here:

http://docs.intersystems.com/latest/csp/docbook/DocBook.UI.Page.cls?KEY=...

And the documentation configuring the Telnet service is here:

http://docs.intersystems.com/latest/csp/docbook/DocBook.UI.Page.cls?KEY=...

The $SYSTEM.Security.Users.SetTelnetSSLSetting() method allows you to control whether the Telnet server allows or requires SSL to be used. It is available in 2016.1 and later.

DSN configuration

You do not need to change the DSN for an ODBC connection as long as you have a matching connection address and port in your settings file. SSL will be used even if Password is selected for the authentication method in the DSN. The **Password with SSL/TLS** and **SSL/TLS server name** options were for the pre-2015.1 style of configuring SSL for ODBC.

Documentation link

Documentation on TLS for client applications is now available on the IRIS docs site:

https://irisdocs.intersystems.com/irislatest/csp/docbook/DocBook.UI.Page...

#Best Practices #Security #SSL #Caché