DevNote #105 –Testing HTTPS with the SIFWorks® ADK for .Net

The Schools Interoperability Framework 1.1 and higher specifications require that all agents be able to communicate with zone integration servers over the HTTPS transport protocol. The SIFWorks® Agent Developer Kit (ADK™) for .Net offers both HTTP and HTTPS implementations that are easily selected by changing an agent's properties at runtime. This Technical Note shows how to set up an HTTPS test environment with the SIFWorks Enterprise ZIS and the GetZoneStatus example agent.

The following software is assumed:

- SIFWorks Enterprise ZIS 0.9.19 or later
- Java 2 Standard Edition SDK version 1.4 or later (required for the SIFWorks Enterprise ZIS)
- Microsoft .Net Framework version 1.1 or later
- SIFWorks ADK for .NET 1.5.1.25 or later
- The Microsoft makecert.exe utility, version 5.131.3617.0 or greater

HTTPS Certificates

1. Creating a Keystore and an HTTPS Certificate for the SIFWorks Enterprise ZIS

Agents that connect to the ZIS over an HTTPS connection expect a valid certificate to be presented by the server. This is done as part of the normal handshaking process performed when the agent and server initiate a connection. When SIFWorks establishes an HTTPS socket to listen for incoming traffic, it reads its key material and certificate from a Java keystore file located in the SIFWorks \certs directory named Server.ks. By default this file does not exist, so it must be created and a valid certificate imported into it. There are several ways to obtain a certificate. You can purchase one from a third-party Certificate Authority (CA) such as VeriSign; you can establish your own Certificate Authority using software from Microsoft, Netscape, and others; or you can create a "self-signed" certificate using the keytool program included with Java. This Developer Note assumes you will be creating your own self-signed certificate. Follow the steps below to create a keystore with a private key and self-signed public key certificate for the zone integration server.

1. Open a command prompt to the SIFWorks `\certs` directory

1. Execute the following line to create a private key in a new keystore file named `Server.ks`

   ```
   keytool -genkey -v -keystore Server.ks -alias SIFWorks -keyalg RSA -keysize 1024
   ```

2. When prompted for a password, enter "changeit"

   ```
   Enter keystore password: changeit
   ```

3. When prompted for First and Last name, enter the IP address of the SIFWorks server:

   ```
   What is your first and last name?
   [unknown]: localhost
   ```

4. Fill in the remainder of the fields:

```
What is the name of your organizational unit?

[unknown]: SIFWorks ZIS

What is the name of your organization?

[unknown]: Edustructures

What is the name of your City or Locality?

[unknown]: Salt Lake City

What is the name of your State or Province?

[unknown]: UT

What is the two-letter country code for this unit?

[unknown]: US
```

5. When prompted for a password, press Enter to use the same password as the keystore

6. The `Server.ks` keystore should now be created and should contain a private key and self-signed public key certificate. You can view the contents of the keystore by executing the following command

   ```
   keytool -list -v -keystore Server.ks
   ```

7. Next, export the public key certificate to a file by executing the following command:

   ```
   keytool -export -keystore Server.ks -alias SIFWorks -file ZIS.cer
   ```

At this point we have successfully created the `Server.ks` keystore with a private key and self-signed public key certificate for the ZIS, and have exported the certificate to a file named `ZIS.cer`. In Step 3 below we'll import this certificate into the SIFWorks keystore so it will be presented to agents that establish HTTPS connections with the server.

Note that the certificate you created was for the reserved internet address "localhost", which represents the local computer. This certificate will work if the ZIS and GetZoneStatus agent are running on the same computer. If they are running on seperate computers, you can use the computer name or IP address, or a fully-qualified DNS name, if desired. The name used must match the host name used to connect to the ZIS from the agent.

2. Creating an HTTPS Certificate for the GetZoneStatus Agent

Just as SIFWorks must present a valid certificate to agents that connect to it, agents must also present a valid certificate to the server. This technote describes how to create certificates using the Microsoft makecert utility. Unfortunately, the version of makecert that is installed as part of the Microsoft .Net Framework SDK 1.1, is unable to create an SSL server certificate for use with the .Net ADK. This article assumes that a newer version of makecert is installed, such as version 5.131.3617.0 or greater. Newer versions of makecert can be obtained from one of the following sources:

- The Microsoft .Net Framework SDK 2.0 (in beta at the time of this writing)
- The current Microsoft Platform SDK
- Downloading the file directly from the Microsoft Platform SDK update site. At the time of this writing, the file is available at the following locations
  - http://tinyurl.com/2qnw5
  - http://download.microsoft.com/download/platformsdk/Update/5.131.3617.0/NT45XP/EN-US/makecert.exe (The entire URL needs to be entered as a single line in the browser).

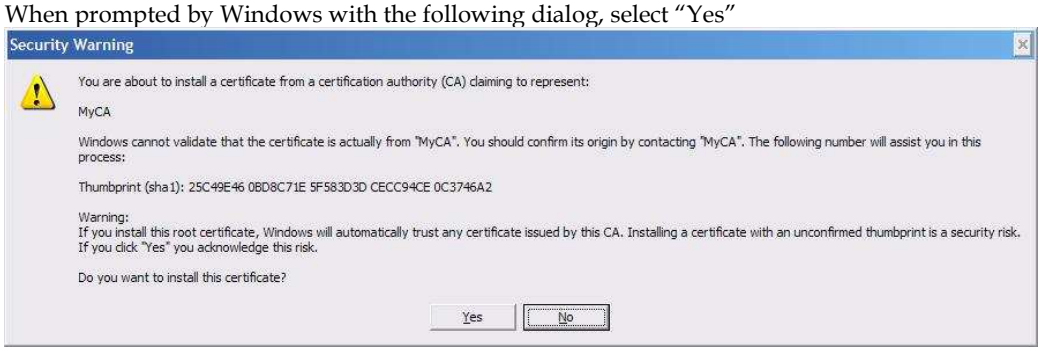
Follow the steps below to create a signing certificate and a test SSL certificate using makecert. The SSL certificate will be created in the system "Current User" certificate store.

1. Open a command prompt to the directory containing the makecert executable

2. Execute the following line to create a custom self-signed certificate to be used as a Certificate Authority for issuing further certificates.

   ```
   Makecert -pe -r -n "CN=MyCA" -ss root
   ```

3. When prompted by Windows with the following dialog, select "Yes"



   Execute the following line to create a server SSL certificate, signed by the the "MyCA" certificate authority you created in step 2.

   ```
   Makecert -pe -n "CN=localhost" -ss my -is root -in MyCa -sky exchange
   ```

4. If the command prompt returns the text "Succeeded", the certificate has been successfully created and stored in your personal certificate store.

Note that the certificate you created was for the reserved internet address "localhost", which represents the local computer. This certificate will work if the ZIS and GetZoneStatus agent are running on the same computer. If they are running on seperate computers, you can use the computer name or IP address, or a fully-qualified DNS name, if desired. The name used must match the host name used by the ZIS to connect to the agent.

3. Viewing and managing the certificates you created

The certificates you create in this manner with makecert are store in the windows certificate store. You can access the certificate manager by using the following steps on Windows XP, Windows 2000, or Windows 2003.

1. From Windows Explorer, click on Start -> Run -> and enter "mmc" and press enter.

2. A blank Windows Management Console window will appear as shown below

3. The first main menu in the window will be named either "File" or "Console", depending on the OS you are running. Click on it and select the "Add/Remove Snap-in..." option.

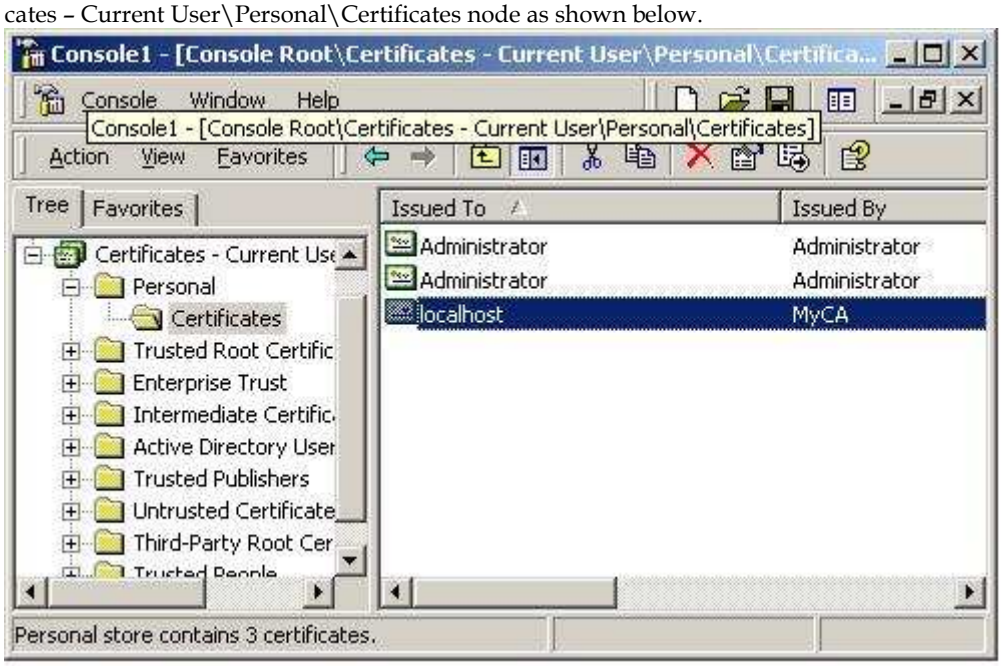
4. Click on the "Add" button

5. Select "Certificates" and click "Add"

6. Select "My User Account" and click "Finish"

7. Close the "Add Standalone Snap-in" dialog by clicking "Close" and the "Add/Remove Snapin" dialog by clicking "OK"

8. The certificate you created for SSL can be found in the certificates manager by expanding the Certificates – Current User\Personal\Certificates node as shown below.



You will need to export the certificate to a file in order for it to be imported into the SIFWorks Enterprise ZIS. We will configure the ZIS in a later step in this document. To export the certificate, follow these steps:

9. Right-Click on the "localhost" certificate you created and choose All Tasks -> Export.

10. The Certificate Export Wizard will walk you through the steps of exporting the certificate. You do not need to export the private key, when asked. You may choose either the DER or Base-64 formats.

If you wish to test client authentication between the GetZoneStatus agent and the ZIS, you must export the root signing certificate instead of the SSL certificate. The root signing certificate will be located in the Trusted Root Certicate Authories/Certificates node in the certificate manager. Export this file and remember the location for step 4 below.

3. Import the ZIS Certificate into the SIFWorks Keystore

The SIFWorks keystore file—`Server.ks`—was created in Step 1 with a private key and a self-signed public key certificate was exported to the `ZIS.cer` file. In this step we'll use the SIFWorks Certificates Manager to import that file into the server's keystore as a *trusted* certificate. Note if you obtained your certificate from a third-party CA, you would follow this same procedure to import it into the SIFWorks keystore. Agents that connect to the zone integration server will be presented with this certificate.

1. Run the `certsmgr.cmd` batch file from the SIFWorks directory

2. Click the Import button and browse to the `ZIS.cer` file created in Step 1.

Leave the Certificates Manager running as we'll use it in the next step.

4. Import the Agent Certificate into the SIFWorks Truststore

Now that both the ZIS and agent have a private key and self-signed public key certificate, it's time to configure each to *trust* one another's certificates. With SIFWorks, this is done by using the Certificates Manager to import the agent's certificate into the `Trusted.ks` "truststore" located in the `\certs` directory (the file is automatically created by the Certificates Manager if it doesn't exist.)

1. Run the `certsmgr.cmd` batch file from the SIFWorks directory if not already running

2. Click on the second tab, "Trusted Agent Certificates"

3. Click the Import button and browse to the certificate file you exported from the Certificate Manager in Step 2.

4. Click OK to close the Certificates Manager

5. Running SIFWorks and verifying the HTTPS transport

Follow these steps to configure and start the SIFWorks ZIS:

1. If SIFWorks if already running, stop it

2. Start SIFWorks

3. Open a web browser and navigate to the Console at `http://localhost:7003/index.jsp`

4. Login to the SIFWorks Console by typing your login name and password

5. Navigate to the `Settings > Transports > HTTPS` node in the tree. Verify that the HTTPS transport is enabled on port 7443 and is listed as Active. (NOTE: If the word "Active" is displayed in red, one or more errors have occurred. Click the text to view the error messages.)

Once you've verified that an HTTPS transport is enabled, create a new zone for testing:

6. Create a new zone named "`TEST`".

7. Click on the Zone Settings tab. Under Messaging Options, enable "Can Request SIF_ZoneStatus". The GetZoneStatus example agent requires this permission.

8. Add an Agent called SIFWorksExample to the `TEST` zone. This is the default source id used by the GetZoneStatus agent.

6. Import the ZIS Certificate into the Agent's Truststore

In the previous step, we configured SIFWorks to *trust* the agent's certificate. It's now time to configure the agent to trust the certificate used by the SIFWorks ZIS. Follow these steps.

1. Open Internet Explorer and type the HTTPS URL of the `TEST` zone into the address bar. The URL can be obtained by looking at the zone page in the SIFWorks Zis, but should be something like this by default: `https://localhost:7443/Test`

2. Internet Explorer will open a security alert dialog as shown below

3. Click on "View Certificate". This brings up the Certificate details dialog as shown below.

4. Click on "Install Certificate" and allow Windows to automatically select the location. This will cause the certificate to be trusted by Windows.

Running GetZoneStatus

7. Compile and Run the GetZoneStatus Example Agent

Follow these steps to compile and run the ADK's GetZoneStatus example agent:

1. Open the GetZoneStatus.csproj in Visual Studio and compile

2. Create a response file from which the agent will read its command-line arguments. This is necessary-when running the example agents in HTTPS mode because of the number of necessary arguments that are required to configure HTTPS. By using a response file, you can include an unlimited number of the arguments to the GetZoneStatus program. (An added benefit is that you don't have to retype the command-line each time you run the agent.)

3. Use Notepad to create a file named `https.txt` with the following lines

```
/zone TEST
/url https://localhost:7443/TEST
/https
/push
/port 15000
/sslcert cn=localhost
```

4. Run the agent by executing the following command:

```
GetZoneStatus.exe https.txt
```

5. You should see output similar to the following:

```
Reading command-line arguments from https.txt:

/zone TEST /url https://localhost:7443/TEST /https /port 15000 /sslcert CN=localhost
```

The agent will log a number of lines of debug information and then print the contents of the SIF_ZoneStatus to the console

8. Testing Client Authentication

If you wish to test client authentication, both the ZIS and the GetZoneStatus agent must be configured to require client authentication. To require client authentication for all connections to the SIFWorks Zis, follow these steps.

1. Start SIFWorks

2. Open a web browser and navigate to the Console at `http://localhost:7003/index.jsp`

3. Login to the SIFWorks Console by typing your login name and password

4. Navigate to the `Settings > Transports > HTTPS` node in the tree.

5. Click on the "All Available" link

6. Check the "Require Client Authentication" checkbox

7. Restart the ZIS

To require client authentication in the GetZoneStatus agent, add the following lines to the https.txt file created in step 7

```
/clientAuth 3
/clientcert cn=localhost
```

Run the GetZoneStatus example agent again to verify connectivity using client certificates. If you have trouble connecting, add the following line to https.txt

```
/debug 5
```

This will enable extended debugging and log a lot more information to the console, including the certificates used for server and client authentication. Verify that the agent is finding and using the proper certificates.

How to Enable HTTPS in an ADK Agent

Because HTTP is so much easier to use during development and initial deployment of SIF Agent software, the default transport protocol enabled by the ADK is HTTP instead of HTTPS. Turning on HTTPS support is as easy as setting a few agent properties in your code at agent startup time. In a commercial agent, these properties are usually read from a configuration file or a database.

Enabling HTTPS for a Push mode agent involves setting up HTTPS-specific properties in your agent's `AgentProperties` object. For an example, take a look at the `parseCL` function of the `ADKExamples.cs` source included with any of the ADK example agents like GetZoneStatus. This should be done at agent startup time before the agent connects to any zones.

8.1. Programmatically enabling HTTPS in the SIFWorks ADK for .Net

Follow these steps in the startup code of your agent to enable HTTPS:

1. Obtain the default `HttpsProperties` object from your `Agent` object:

```
// Get the Agent's default HttpsProperties object
HttpsProperties props = myAgent.DefaultHttpsProperties;
```

2. Set any of the methods on the `HttpsProperties` class

```
// Set the Server SSL Certificate to use
props.SSLCertName = "cn=localhost"

// Set the Client Certificate to use for mutual authentication
props.ClientCertName = "cn=localhost"

// Require client authentication
props.ClientAuthLevel = 3;
```

3. Call the `HttpsProperties.Port property` to assign a port number for the listening socket. This is the port your Push-mode agent will listen on for incoming traffic from the ZIS. If the agent will run in Pull mode, this step is not necessary.

```
props.Port = 15000;
```

4. Finally, call the `AgentProperties.setTransportProtocol` method to inform the agent that it should use HTTPS instead of HTTP as its transport protocol:

```
props.TransportProtocol = "https";
```

That's all there is to it. When you start up your agent it will establish an HTTPS listening socket instead of an HTTP socket.

8.2. Https Property Reference

The .Net version of the ADK supports the following configuration properties for HTTPS. Note that these properties can be set using APIs on the HttpsProperties class, setting properties in a <transport> node in the config file, or by setting a system property.

- `sslCertFile`– gets and sets the name of a certificate file that should be used instead of opening the Windows certificate store. This file must contain a private key in order to be used for SSL, so the file must in the PFX/P12 format and have a ".pfx" extension.
- `sslCertFilePassword` – gets and sets the password used to open the private key portion of the certificate file
- `certStoreLocation`– gets and sets the location of the certificate store to open. Options are to open it from the local user's account, the computer account, and a special store that is created for each service. The possible values for this property are shown below. The default is to open the certificate

store pertaining to the current user. If the agent is running as a service, it might be desirable to change this to the certificate store pertaining to the current service.

Property Value	Description
CurrentService	The certificate store for the current service.
CurrentUser	The certificate store for the currently logged-on user. This is the default value for this property if it is not explicitly specified.
CurrentUserGroupPolicy	The certificate store for the currently logged-on group.
LocalMachine	The certificate store for the local computer.
LocalMachineEnterprise	The certificate store for the local machine enterprise downloaded from a network setting.
LocalMachineGroupPolicy	The certificate store for the local machine group policy downloaded from a network setting.

- `certStore` – gets and sets the store from which to load certificates. Defaults to "My", which corresponds to the "Personal" folder in the Windows certificate manager.
- `sslCertName` – Gets or sets the certificate that the agent should use for SSL server authentication when running in Push mode. If not specified, the first applicable certificate in the store that is valid for Server authentication and not out of date will be used. If not found, an exception will be thrown and the agent will not be able to start the https transport.
- `clientCertName` – Gets or sets the certificate that the agent should use for client authentication when connecting to the ZIS. This is only used if the agent is already connecting using SSL. If not specified, the first applicable certificate in the store that is valid for client authentication and not out of date will be used. If specified and not found, an exception will be thrown and the agent will not be able to connect to the ZIS. If not specified and not found, the agent will not attempt to send a client certificates, but will still connect using SSL.
- `clientAuthLevel` – Gets or sets the SIF Authentication level that the agent requires for incoming connections from a server.

Value	Description
0	SIF Authentication level 0: No client authentication is required
1	SIF Authentication level 1: A valid certificate must be presented.
2	SIF Authentication level 2: A valid certificate from a trusted certificate authority must be presented.
3	SIF Authentication level 3: A valid certificate from a trusted certificate authority must be presented and the CN field of the certificate's Subject entry must match the host sending the certificate.

8.3. HTTPS communication initiated by the agent

If the Https transport is enabled in the .Net ADK, the agent will only allow connections to the ZIS using SSL. Regardless of whether the agent is in push or pull mode, the agent must initiate outgoing communications with the ZIS using SSL. Whether the SSL communication can succeed is based upon the validity of the certificate that the server provides.

There is currently no configuration required in the .Net ADK for validating server certificates for outgoing communications. The agent uses the standard certificate verification process followed by Windows for all SSL outgoing communications. The ZIS that the agent connects to provides a server certificate to the agent when the HTTPS connection is initiated. The provided certificate or the Root Authority that created the certificate must be trusted by the operating system in order for SSL communications to proceed. In addition, the subject name on the certificate must match the host name used by the agent to connect to the ZIS. This equates to SIF Level 3 Authentication, currently the highest level of trust supported by the SIF Specification.

Because the agent uses the standard Windows verification process, an easy way to install an untrusted server certificate is to connect to the ZIS with Internet Explorer using the https protocol. This will bring up an Internet Explorer certificate dialog that allows the certificate to be examined and then trusted.

When the ADK initiates an outgoing connection using SSL, it also looks for a client certificate to provide to the server. If the `clientCertName` property is specified and the specified certificate is not found, the agent will throw an exception and the outgoing communication will fail. If, however, the `clientCertName` property is not specified the agent will look for an available certificate in the Windows certificate store based upon the `certStoreLocation` and `certStore` properties. If one is not found, the communication will succeed but a client certificate will not be presented to the server.

The certificate used for client authentication must be a trusted certificate, valid for client authentication and stored in the Windows certificate store. If the ADK finds a client certificate, it logs the certificate details to the debug log. If it does not, it logs "No certificate found for client authentication" to the debug log the first time an outgoing connection is made using SSL.

8.4. HTTPS communication initiated by the ZIS

If the agent has one or more zones running in push mode, an additional set of steps is taken to set up an SSL server implementation. The .Net ADK supports HTTPS in push mode, either by using a certificate file on disk or by locating and using a certificate in the Windows certificate store.

If the `sslCertFile` property is specified, the ADK loads the certificate from the file and uses it. In this case, it also needs to open the private key stored in the file using the `sslCertFilePassword` property. If the `sslCertFile` property is not specified, the ADK looks for the first valid SSL certificate in the Windows certificate store, looking in the location specified by the `certStoreLocation` and `certStore` properties.

If a valid certificate is not found an exception is thrown and the HTTPS transport will not start. If, however, one is found the details of the certificate used are logged to the ADK's debug log, similar to this

```
2005-07-01 11:36:55,890 DEBUG [ADK.Agent.transport$https] Using CERTIFICATE:
        Format:  X509
        Name:  localhost
        Issuing CA:  Root Agency
        Key Algorithm:  1.2.840.113549.1.1.4
        Serial Number:  48F1E1548B1EF28D42C0D7B54CE25603
        Key Alogrithm Parameters:  0500
        Public Key:
30818902818100B94A472F6DD4CB30734334620E6318146F6252FFE0B1DD01CDB1FFBC43B714CFE449203E5D5
DDBC729F773798634AB014459E076E2F8B482758913F1A9562538ECA1B277425CFBE9BDD43DB01CD2500EA875
E868FA53EC089E341AF6516A57802B8788D62D7F10EF7B1B43998DBE0106CD326A2CD1D08C401F875BD46C4A6
66B0203010001
```

