

SSL Configuration Best Practices for SAS® Visual Analytics 7.1 Web Applications and SAS® LASR™ Authorization Service

Heesun Park and Jerome Hughes, SAS Institute Inc., Cary, NC

ABSTRACT

One of the challenges in Secure Socket Layer (SSL) configuration for any web configuration is the SSL certificate management for client and server side. The SSL overview covers the structure of the x.509 certificate and SSL handshake process for the client and server components. There are three distinctive SSL client/server combinations within the SAS® Visual Analytics 7.1 web application configuration. The most common one is the browser accessing the web application. The second one is the internal SAS® web application accessing another SAS web application. The third one is a SAS Workspace Server executing a PROC or LIBNAME statement that accesses the SAS® LASR™ Authorization Service web application. Each SSL client/server scenario in the configuration is explained in terms of SSL handshake and certificate arrangement. Server identity certificate generation using Microsoft Active Directory Certificate Services (ADCS) for enterprise level organization is showcased. The certificates, in proper format, need to be supplied to the SAS® Deployment Wizard during the configuration process. The prerequisites and configuration steps are shown with examples.

INTRODUCTION TO X.509 CERTIFICATE AND SSL PROTOCOL

The backbone of the SSL protocol is the use of x.509 certificates. These certificates are based on public key cryptography. Public key cryptography is an asymmetric encryption scheme that uses a public key and a private key. The public key and its matching private key are calculated based on prime number theory. The content encrypted with the public key can only be decrypted with the matching private key, and vice versa. The delivery mechanism of the public key is the X.509 (SSL) certificate. The matching private key is stored in a safe place and is typically protected with a passphrase.

SSL uses the widely used X.509 certificate standard that defines the fields in a certificate and the uses of those fields. There are three types of X.509 (SSL) certificates: server identity certificates, client, or personal certificates (we use these terms interchangeably), and Certificate Authority (CA) certificates. To ensure the authenticity of the certificates, server identity certificates and client certificates must be “signed” by a Certificate Authority. Signing the certificate means that a one-way hash of the data in the certificate is encrypted with the CA’s private key. To decrypt and validate the certificate, the consumer of the certificate needs the CA’s certificate (that contains its public key) in its “trusted signer” area. These days, most browsers come with certificates by well-established CAs.

Among other things, an X.509 certificate contains the following fields. The “subject” field is used to identify the owner of the certificate. The “issuer” field indicates the name of the CA who has signed the certificate. The “Subject Public Key Info” field contains the public key and the public key algorithm used. . The “signature” is the message digest (or the hash value of the certificate) encrypted with the CA’s private key. The “subject” field is used for user identification in the case that the client certificate is required for authentication for two-way SSL. It contains an LDAP like tree structure and typically works well when the application server’s user registry is an LDAP server.

SSL certificates have become relatively easy to create using open-source tools like OpenSSL from Apache and the Java based keytool from Oracle. Microsofts Active Directory Domain Services (flavor of LDAP server) for Windows comes with Public Key Infrastructure (PKI) that provides certificate management for the servers and the users as well. In this paper, we will use Active Directory Certificate Service (AD CS) to generate server identity certificates for a SAS 9.4 middle-tier machine that houses a SAS Visual Analytics 7.1 web application.

The addition of the SSL protocol to the HTTP protocol has made secure communication on the web possible, and to a large extent made today’s e-commerce a reality. The SSL protocol works in two stages. The first stage is the public

key cryptography-based handshake, during which the two parties that want to communicate, agree upon a traditional symmetric encryption algorithm to use in subsequent communication. The parties also exchange session specific information (a “pre-master secret”) in order to generate the symmetric encryption key (“session key”) that each side uses with the mutually agreed upon encryption algorithm. Because the session key is dynamically generated for each session and is destroyed once the session is terminated, the security exposure of the session key is minimal. The second stage is the use of symmetric encryption with the dynamically generated session key for the rest of the session.

The first step in the SSL handshake is the server sending its certificate to the client side. The client needs to verify that the server identity certificate is authentic and trustworthy. The server identity certificate carries the hash value (called the signature) of the certificate that is encrypted by the CA’s private key. That hash value should be decrypted with the CA’s public key and compared to the calculated hash value from the certificate itself. If they match, the server identity certificate is considered authentic. This is why the client side should have the CA certificate that signed server identity certificate.

Once we identify the client-side of the SSL handshake, the next step is to understand the component structure and how it stores the CA certificate. Unfortunately, the structure and format of certificate storage is not consistent among the client components. Therefore, each storage mechanism will be explained for the specific client / server combination.

CREATION OF SERVER IDENTITY CERTIFICATE WITH AD CS

There are a number of ways to create server identity certificates for the SAS Web Server within the SAS Visual Analytics 7.1 middle-tier configuration. If the SAS Visual Analytics 7.1 web application is external facing, the server identity certificate request should be generated (by tools like OpenSSL, KeyTool, or Microsoft Active Directory Certificate Service) and submitted to a well-known CA and signed by them. When the SAS Visual Analytics 7.1 web application is used primarily by the enterprise organization -, the server identity certificate can be signed by the enterprise level CAs. We will use the latter approach for SSL certificate generation.

As mentioned earlier, Microsoft has a certificate management tool (ADCS) that we can use to generate many types of certificates. Included below is a screenshot that shows the creation of a server identity certificate using the “Web Server with Private Key Export” template. Once you submit the request, you get a “Certificate Issued” page with a link to “Install this certificate”. Click the “Install” link and the certificate is placed in the “personal” certificate area of your browser.

Note that the AD CS tool works well with IE, but has some issues with other browsers.

From IE, locate the certificate from the Internet options window. Select Content -> Certificates. Double click on the certificate that you generated. The Certificate window then appears. Select the “Details” tab, select “Copy to File”,

and save the certificate to a file location. During the process, select the “export private key” option and enter the password for the private key (of your choice).

Ironically, you will need to remove the password from the private key before you can use it for the SAS Web Server.

Next, from the Certificate window, select “Certification Path”. You will see the CA certificate chains. Double click on each CA certificate in the chain and export it as you did above.

In this configuration example, we now have a server identity certificate (bctlax15.unx.sas.com), an intermediate CA (sasca01), and a Root CA (SASRootCA) certificate in our file location.

The screenshot shows the 'Advanced Certificate Request' page in Internet Explorer. The browser title is 'Microsoft Active Directory Certificate Services - Microsoft Internet Explorer provided by SAS'. The URL is 'https://certificates.sas.com/certsrv/certreqna.asp'. The page header is 'Microsoft Active Directory Certificate Services - sasca01' with a 'Home' link. The main content area is titled 'Advanced Certificate Request' and contains several sections: 'Certificate Template:' with a dropdown menu set to 'Web Server with Private key Export'; 'Identifying Information For Offline Template:' with input fields for Name (va7mt.unx.sas.com), E-Mail (my_email@sas.com), Company (SAS), Department (BIRD), City (Cary), State (NC), and Country/Region (US); 'Key Options:' with radio buttons for 'Create new key set' (selected) and 'Use existing key set', a CSP dropdown (Microsoft RSA SChannel Cryptographic Provider), 'Key Usage' set to 'Exchange', 'Key Size' set to 1024, and checkboxes for 'Automatic key container name' (selected), 'Mark keys as exportable' (checked), and 'Enable strong private key protection' (unchecked); 'Additional Options:' with radio buttons for 'Request Format' (CMC selected, PKCS10 unselected), 'Hash Algorithm' set to sha1, and a 'Save request' checkbox (unchecked). There is also an 'Attributes' section with a list box and a 'Friendly Name' input field. A 'Submit >' button is located at the bottom right of the form.

Notice that the server identity certificate generated above is in .PFX format that includes the private key. To be used in the SAS Web Server, the private key needs to be separated from the certificate. Here is the process and the OpenSSL commands to use to accomplish this task

Export the private key file from the pfx file

```
openssl pkcs12 -in bctlax15.pfx -nocerts -out privatekey.pem
```

Remove the passphrase from the private key

```
openssl rsa -in privatekey.pem -out bctlax15.unx.sas.com.key
```

Export the certificate file from the pfx file

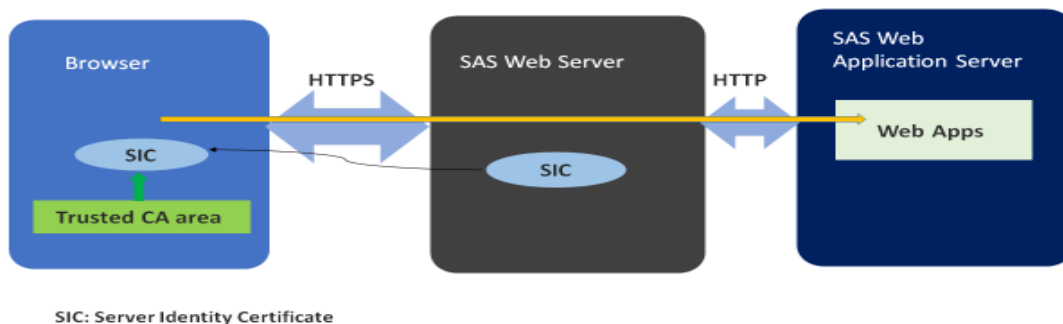
```
openssl pkcs12 -in bctlax15.pfx -clcerts -nokeys -out bctlax15.unx.sas.com.crt
```

SSL CLIENT/SERVER SCENARIO 1 - WEB BROWSER ACCESSING SAS VISUAL ANALYTICS 7.1 WEB APPLICATION THROUGH SAS® WEB SERVER

The mostly commonly used SSL client/server environment is where the web browser accesses the SAS Visual Analytics 7.1 web application through the SAS Web Server. When you supply the certificate and private key to the the SAS Deployment Wizard process for the SAS Visual Analytics 7.1 configuration, the SAS Web Server's httpd-ssl.conf file located in conf/extra includes the following certificate definition for the SSL server side:

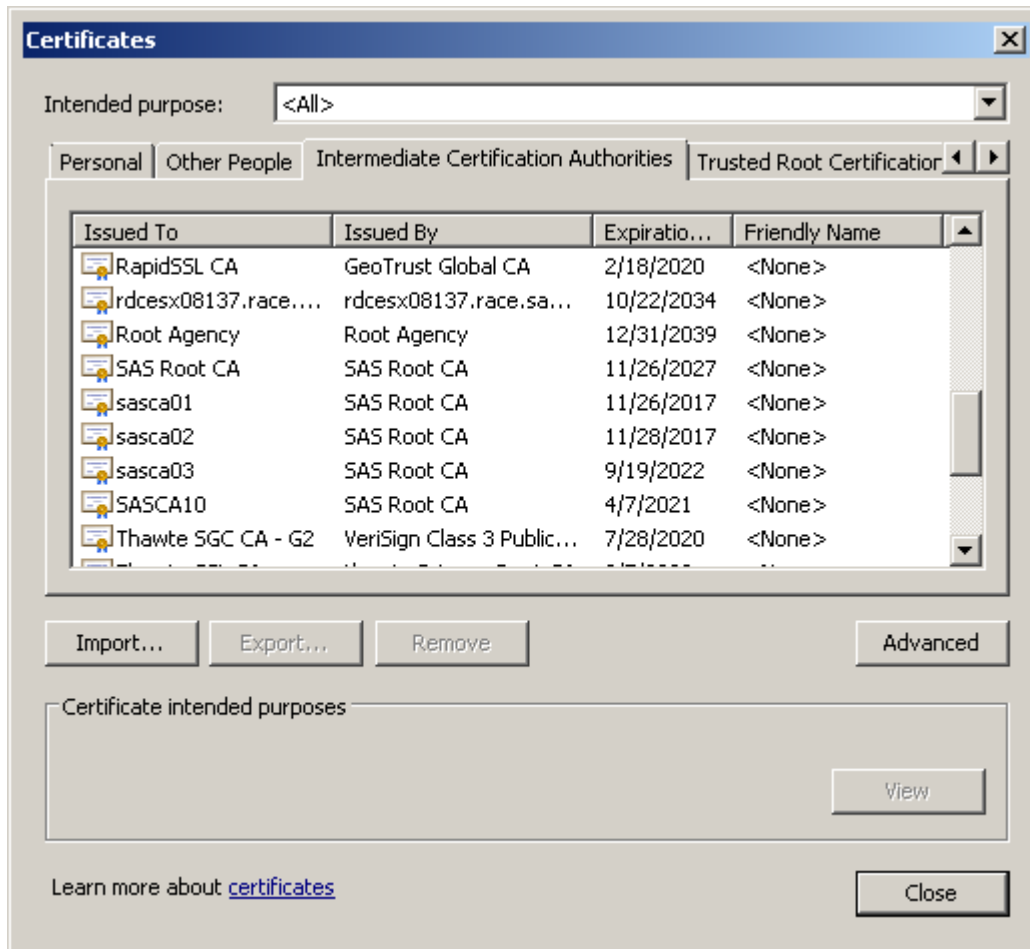
```
# Server Certificate:  
SSLCertificateFile "ssl/bctlax15.unx.sas.com.crt"  
# Server Private Key:  
SSLCertificateKeyFile "ssl/bctlax15.unx.sas.com.key"
```

The following diagram depicts the scenario:



The browser side should contain CA certificate chains (in our example, sasca01.crt and sasRootCA.crt) with a signed server identity certificate (bctlax15.unx.sas.com.crt). For browsers, CA Certificate chains are stored in the "Intermediate" and "Root" trusted CA certificate area. Authorized CA certificates can be easily imported to the "trusted CA certificate" area. For large organizations, the IT department usually controls how this is done.

The following window shows the SAS CA certificates in the trusted CA area (in IE).



Here is a summary of the steps involved in the SSL handshake for this scenario:

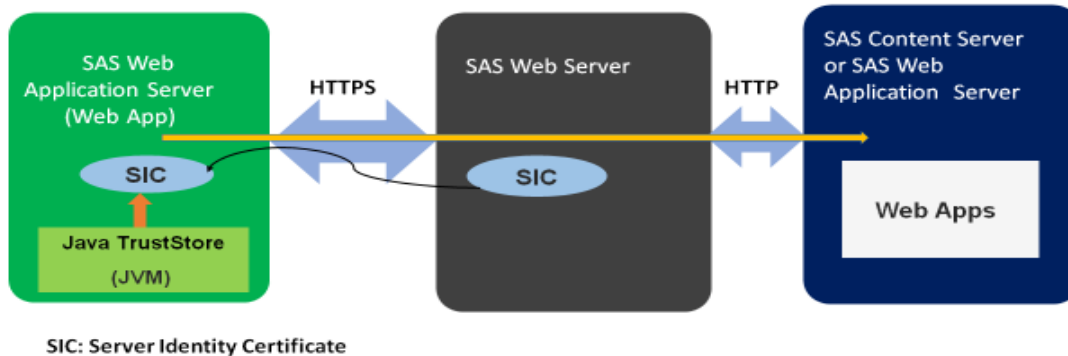
- The browser sends a request to SAS Visual Analytics 7.1 web application through HTTPS. For example, <https://bctlax15.unx.sas.com/SASVisualAnalyticsHub/>
- The SAS Web Server sends back its "server identity certificate (SIC)" to the browser
- The browser verifies the authenticity of the SIC using the CA certificate chains found in the Trusted CA certificate area of the browser.
- Once verified, the browser and the SAS Web Server agree on the symmetric encryption algorithm to use and create the symmetric encryption session key independently using the shared "pre-master" secrets.
- All traffic on the session gets encrypted and decrypted using the session key produced.

SSL CLIENT/SERVER SCENARIO 2 - SAS VISUAL ANALYTICS 7.1 INTERNAL WEB APPLICATION ACCESSING SAS® CONTENT SERVER OR INTERNAL SAS WEB APPLICATIONS THROUGH SAS WEB SERVER

For this scenario, the client side is represented by the Java Virtual Machine (JVM) that houses a web application. For JVM, the trusted CA certificates are stored in the trust keystore. "Keystore" is the place where we keep certificates for Java implementation. The location of the keystore for server identity certificates (called keyStore) and

the keystore for the CA certificates (called trustStore) are associated with the JVM using the following JVM parameters:

```
-Djavax.net.ssl.trustStore=  
-Djavax.net.ssl.trustStorePassword=  
-Djavax.net.ssl.keyStore=  
-Djavax.net.ssl.keyStorePassword=
```



Clearly, when JVM becomes an SSL client, the trustStore should contain the CA certificates that signed the incoming server identity certificate. The default trustStore location for the SAS 9.4 middle-tier configuration is `$SASHome/SASPrivateJavaRuntimeEnvironment/9.4/jre/lib/security/cacerts`. The `cacerts` file contains many well-known CA certificates. When enterprise level CAs are used to sign the server identity certificate, like our case, those CA chains should be inserted into this `cacerts` file.

Here are the keytool commands used to add the SAS CA certificates into the trustStore:

```
keytool -importcert -keystore $SASHome/SASPrivateJavaRuntimeEnvironment/9.4/jre/lib/security/cacerts -storepass  
changeit -file sasrootca.cer -alias sasrootca  
keytool -importcert -keystore $SASHome/SASPrivateJavaRuntimeEnvironment/9.4/jre/lib/security/cacerts -storepass  
changeit -file sasca01.cer -alias sasca01
```

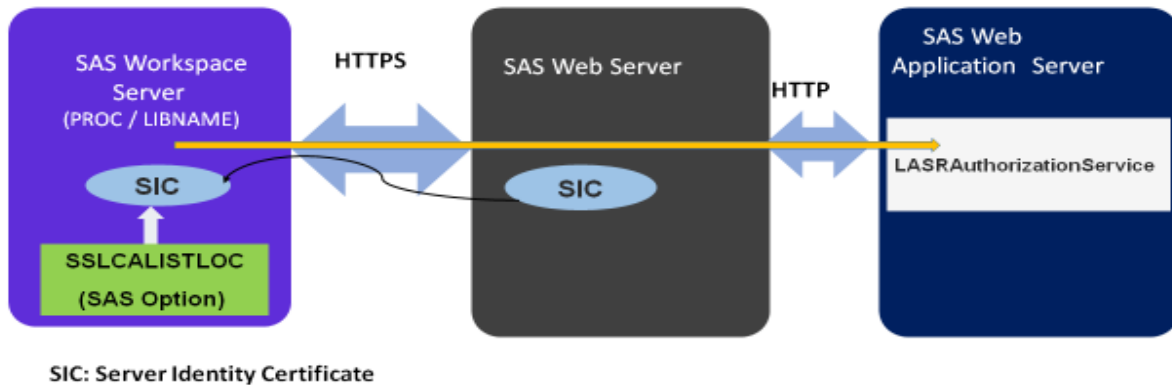
Here is a summary of the steps involved in the SSL handshake for this scenario:

- An internal web application deployed on a middle tier server (JVM) sends a request for the SAS Visual Analytics 7.1 web application using HTTPS. For example, <https://bctlax15.unx.sas.com/SASWIPClientAccess/>
- The SAS Web Server sends back its server identity certificate (SIC) to the browser
- The middle tier server (JVM) verifies the authenticity of the SIC using the CA certificate chains in the Java truststore (`$SASHome/SASPrivateJavaRuntimeEnvironment/9.4/jre/lib/security/cacerts`) file.
- Once certificates are verified, JVM and the SAS Web Server agree on the symmetric encryption algorithm to use and independently create the symmetric encryption session key independently using the shared "pre-master" secrets.
- All traffic on the session gets encrypted and decrypted using the session key produced.

SSL CLIENT/SERVER SCENARIO 3 - SAS WORKSPACE SERVER ACCESSING LASRAUTHORIZATIONSERVICE WEB APPLICATION THROUGH SAS WEB SERVER

This scenario is a little complicated. For some SAS Visual Analytics 7.1 administrator functions, such as starting the SAS® LASR Analytic Server or loading local data into the LASR Analytic Server, tasks are accomplished by executing PROC LASR or the LIBNAME statement within the SAS Workspace server. In the process, it needs to access the LASRAuthorizationService web application deployed in the SAS® Web Application Server. The following

diagram shows the overall process:



Note that the SAS Workspace server is a part of SAS Foundation implementation and is written in C. As you can imagine, C code handles CA certificates in a different manner. SAS provides the SAS system option, SSLCALISTLOC, that points to the file location that contains CA certificates. This option can be set as shown in the following example:

```
options sslcalistloc="<file_location>/sasca.pem";
```

In C implementation, CA certificate chains should be in .PEM format. So the CA certificates in the chain need to be converted to .PEM format. Typically, certificates are in DER format. Once converted, they need to be concatenated and stored in one file. Here are sample openssl commands on Linux:

```
OpenSSL> x509 -inform DER -outform PEM -in sasca01.cer -out sasca01.pem
```

```
OpenSSL>x509 -inform DER -outform PEM -in sasrootca.cer -out sasrootca.pem
```

(Copy the sasca01.pem to sasca.pem and append sasrootca.pem to it.)

```
cp sasca01.pem sasca.pem
```

```
cat sasrootca.pem >> sasca.pem
```

When displayed with text editor, certificates in .PEM format start with "-----BEGIN CERTIFICATE-----" tag and ends with "-----END CERTIFICATE-----" tag with encrypted certificate content in between.

SAS WEB SERVER SSL CONFIGURATION EXAMPLE USING SAS® DEPLOYMENT WIZARD (SDW)

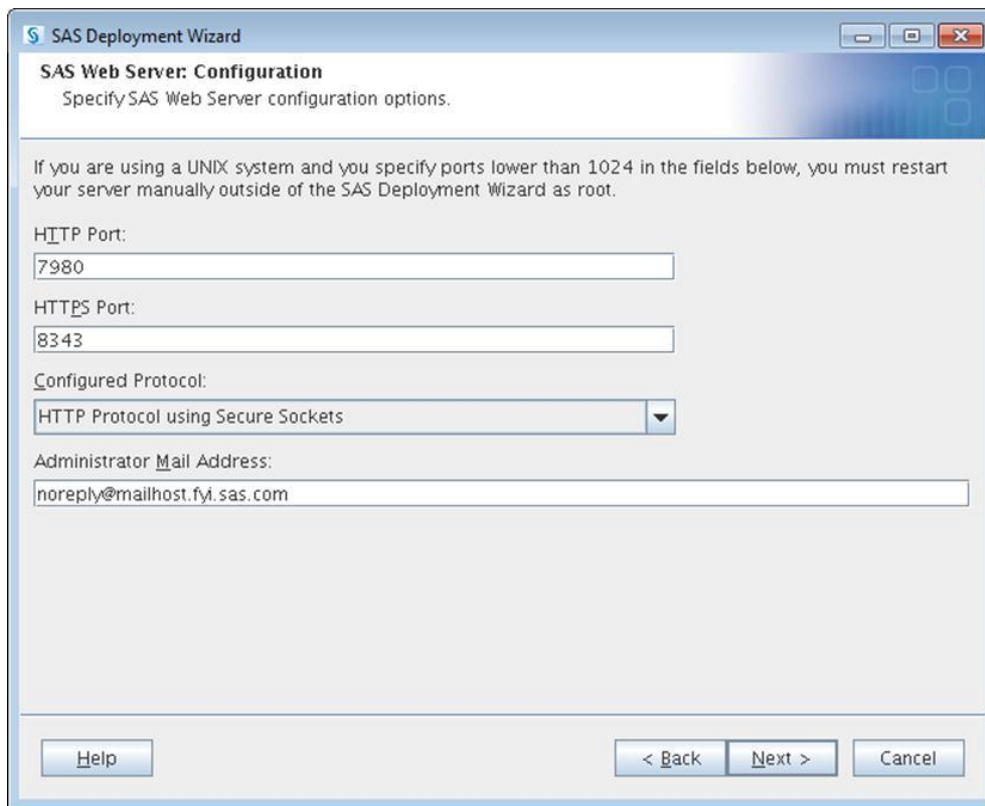
The SAS® Deployment Wizard (SDW) supports configuring SSL for the SAS Web Server from its menu. When you supply the server certificate and its private key to the SDW process for the SAS Visual Analytics 7.1 configuration, the following SSL parameters in the SAS Web Server's httpd-ssl.conf file located in the conf/extra directory are updated.

```
# Server Certificate:
SSLCertificateFile "ssl/<server_certificate>"
# Server Private Key:
SSLCertificateKeyFile "ssl/<server_private_key>"
```

Notice that this is how an Apache web server (such as SAS Web Server) supports the SSL configuration for the server side. Java containers such as Tomcat (SAS Web Application Server) use a keystore approach, which is a different way of keeping the server certificate for SSL server side.

Note: Before starting SDW and configuring SSL know the location of the server certificate and its private key.

From the SAS Web Server: Configuration window, change the "Configured Protocol" to "HTTP Protocol using Secure Sockets" and click "Next". Here is the screen shot of the menu:

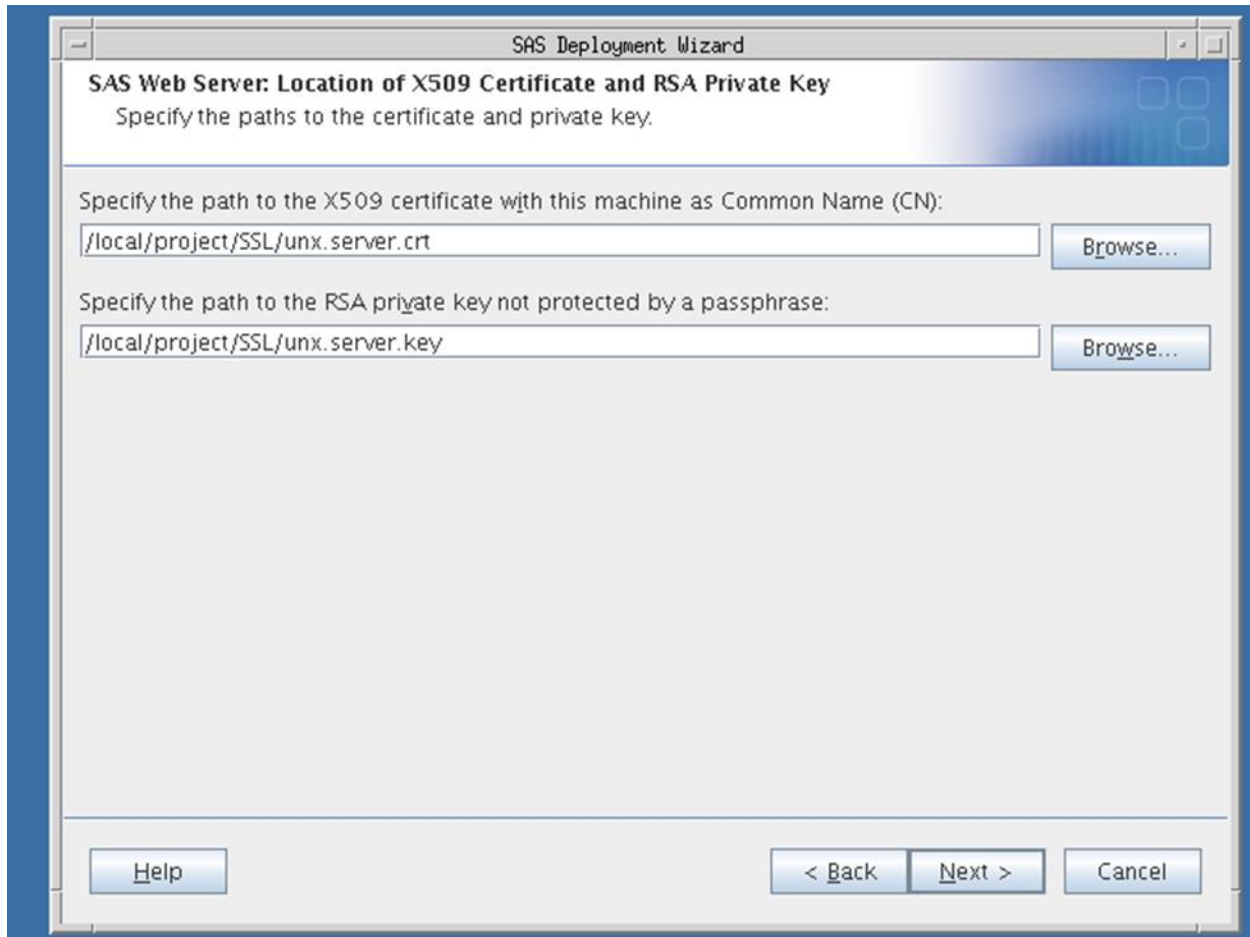


Next, specify the certificate and private key location. from the "SAS Web Server: Location of X.509 certificate and RSA Private Key" window, Our example below is from a Visual Analytics 7.1 configuration on Linux. The locations are as follows:

```
Server certificate location: /local/install/Projects/SSL/unx.server.crt
Server certificate private key location: /local/install/Projects/SSL/unx.server.key
```

It is worthwhile to double check the Common Name (CN) field of the server certificate. The CN should match the machine name where the SAS Web Server is deployed and running. During the SSL handshake, the CN of the server certificate is checked against the server machine name for certificate validation.

The screenshot below shows the server certificate location and private key location specified for our example.:



SSL SETTING FOR SAS WORKSPACE SERVER (SSL CLIENT)

We touched on this area briefly with the SSL Client and Server in scenario 3 (LASR Analytic Server start/stop through LAS Authorization service). The SAS Workspace Server must be configured for SSL so that the user can start/stop the LASR Analytic Server. Briefly, here are the steps:

- Edit <install>/config/Lev1/SASApp/WorkspaceServer/sasv9_usermods.cfg
- Add the following line to the end of the file (sasca.pem as generated in Scenario 3 above):
-sslcalistloc <install>/config/Lev1/certs/sasca.pem

CONCLUSION

In this paper, we present best practices for configuring SSL for SAS Visual Analytics 7.1. These best practices include -SSL and certificate basics and what needs to be set up for SSL client side and server side configurations in various client/server combinations. The core piece of SSL operation is the Public Key Cryptography (PKC). X.509 certificates and the SSL handshake are just implementation and instrumentation of PKC. We have shown how to create server certificates and how to specify certificates (server certificates and CA certificates) for client and server configurations. We also point out that each component has different ways to handle certificates, but the logic is the same. Focusing on fundamentals makes everything relatively easy.

RECOMMENDED READING

[1] Heesun Park and Stan Redford. 2007. "Client Certificate and IP Address-based Multi-Factor Authentication for J2EE Web Applications." *Proceedings of the 2007 Conference of the Center for Advanced Studies on Collaborative Research (CASCON)*. New York, NY. Available at <http://dl.acm.org/citation.cfm?id=1321229>

[2] OpenSSL – Cryptography and SS/TLS Toolkit. Available at <http://www.openssl.org/>

[3] Keytool – Key and Certificate Management Tool. Available at <http://docs.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html>

[4] SAS(R) 9.4 Intelligence Platform: Middle-Tier Administration Guide. Available at <http://support.sas.com/documentation/cdl/en/bimtag/66823/HTML/default/viewer.htm#p05l4tthwfpkn12zd6wshy3wp7.htm>

[5] Heesun Park. 2011. "Building FIPS 140-2 Compliant Configuration for SAS9.3 BI Web Applications." *Annual Computer Security Applications Conference 2011 (ACSAC)*. Available at <http://www.acsac.org/2011/program/case/park.pdf?OPENCONF=377434ec97ef199f06efe94aaa42d107>.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Heesun Park
SAS Institute Inc.
E-mail: Heesun.Park@sas.com

Jerome Hughes
SAS Institute Inc.
E-mail: Jerome.Hughes@sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.