Paper SAS3441-2016

Tips and Techniques for Using Site-Signed HTTPS with SAS® 9.4

Stuart J Rogers, SAS Institute Inc.

ABSTRACT

Are you going to enable HTTPS for your SAS® environment? Looking to improve the security of your SAS deployment? Do you need more details about how to efficiently configure HTTPS? This paper guides you through the configuration of SAS® 9.4 with HTTPS for the SAS middle tier. We examine how best to implement site-signed Transport Layer Security (TLS) certificates and explore how far you can take the encryption. This paper presents tips and proven practices that can help you be successful.

INTRODUCTION

In this paper we review the process involved in establishing an HTTPS connection and identify the key components that impact establishing a secure session. The contents of an X.509 Certificate are examined and we show how the elements are used in the HTTPS handshake. We present that HTTPS is more than just encryption and describe the method of establishing trust. For establishing trust we present the importance of the signing Certificate Authority and how the chain of trust is used by a client to establish which X.509 Certificates to trust. We are going to review how different clients are able to establish trust and the mechanisms that you can use to add new Certificate Authorities to these different truststores.

Next we discuss how far to take the HTTPS connection. We will see that you can configure HTTPS for the SAS® Web Server, SAS® Web Application Server, and SAS® Environment Manager Server. For each of these SAS components we review how you configure HTTPS. Only the SAS® Web Application Server should be configured manually, and the SAS® Deployment Wizard should be used for the SAS® Web Server and SAS® Environment Manager Server.

We examine in detail what types of private keys and X.509 Certificates that need to be provided to the three components. We present some example commands that you can use to convert files between different formats to ensure that the correct format is available when you configure your SAS environment.

Finally, we list the correct order in which to complete the configuration steps, both automatic with the SAS® Deployment Wizard, and manual changes. We provide the detail of the configuration settings and which configuration files you need to change.

So long as you follow the information presented on the prerequisites, obtain the files in the correct format, and follow the order of steps we've discussed you will be successful in your deployment.

HTTPS: OVERVIEW

Adding Transport Layer Security (TLS) on top of Hyper Text Transfer Protocol (HTTP) provides you with the commonly recognized HTTPS protocol. The HTTPS protocol has made secure communications on the Internet possible. The HTTPS protocol provides encryption between the client and the server protecting the content of messages sent across the Internet.

The TLS protocol operates in two distinct stages. The first stage is a TLS Handshake using public/private key cryptography. This is an asymmetric encryption scheme that uses a public and private key pair. Content encrypted with one of the keys can only be decrypted using the other key. The TLS Handshake Protocol is responsible for the authentication and key exchange necessary to establish or resume secure sessions. The second stage of the TLS protocol is the TLS Record protocol that secures application data using the symmetric keys created during the Handshake. The Record Protocol is responsible for securing application data and verifying its integrity and origin.

CERTIFICATES

The mechanism to distribute the public key used for the TLS Handshake is the X.509 certificate. X.509 is the TLS certificate standard that defines the fields within the TLS certificate. RFC 5280 (Cooper, et al.

2008) defines the version 3 structure of a TLS certificate. The structure of an X.509 certificate is as follows:

- Certificate
  - Version Number
  - Serial Number
  - Signature Algorithm ID
  - Issuer Name
  - Validity period
    - Not Before
    - Not After
  - Subject name
  - Subject Public Key Info
    - Public Key Algorithm
    - Subject Public Key
  - Issuer Unique Identifier (optional)
  - Subject Unique Identifier (optional)
  - Extensions (optional)
    - ...
- Certificate Signature Algorithm
- Certificate Signature

As we can see from the structure of an X.509 certificate listed above the public key information only forms a small part of the X.509 certificate. Other key parts of the X.509 certificate are the Issuer Name, Validity period, and Subject name. Essentially, the X.509 certificate enables the binding of a public key to a particular subject name, where the subject name is normally a distinguished name for a website. For example, such a distinguished name might be CN = www.google.co.uk, O = Google Inc, L = Mountain View, S = California, C = US. The most important part of the subject name is the “common name” (CN), which must match the DNS name of the host or website.

SIGNING CERTIFICATES

The Issuer Name identifies the party that has digitally signed the X.509 certificate. The signer of the X.509 certificate is the entity that is vouching that the public key in the X.509 certificate really does identify the subject. The signer signs the certificate by using a Hash algorithm to create a one-way hash of the X.509 certificate, which is then encrypted using the signer’s private key, as illustrated in Figure 1.

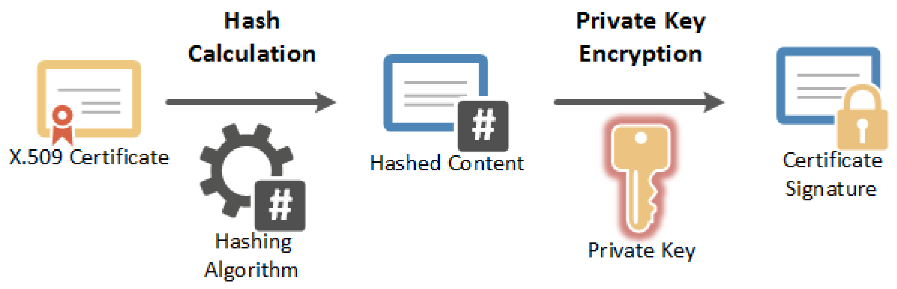

Figure 1. Creating an X.509 Certificate Signature

When the client validates the signature, it performs two actions and compares the results. First, the client calculates a one-way hash of the X.509 certificate using the Hash algorithm provided in the Certificate Signature Algorithm field. This produces the same hash value used by the signer to create the signature. The second action is to decrypt the Certificate Signature using the public key of the signer. This results in the hashed value that was originally produced by the signer being decrypted. This then allows the hash value calculated by the signer and the client to be compared. The two hashes are compared and if they are the same the X.509 certificate is validated. This is illustrated in Figure 2.

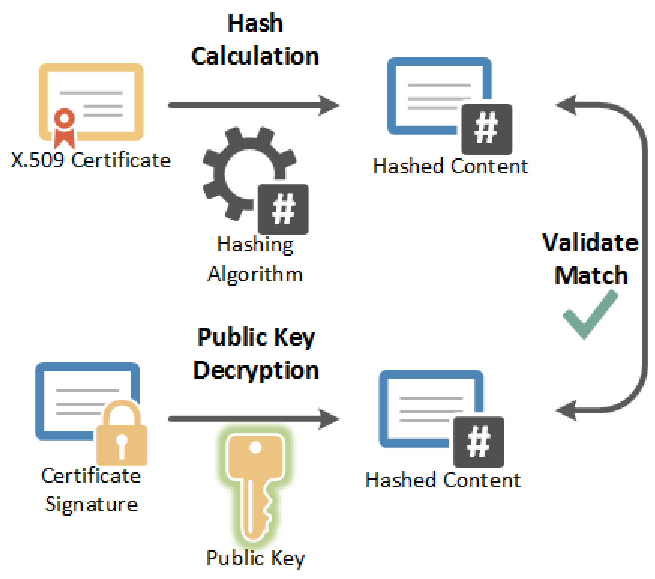

Figure 2. Validating an X.509 Certificate Signature

This illustration presents one of the major considerations for the effective use of Transport Layer Security (TLS) and HTTPS; that is, how the client is provided with the public key for the entity signing the X.509 certificate. We can classify the X.509 Certificate based on who the entity is that has performed the signing. If the X.509 Certificate is signed using the same private key that corresponds to the public key contained in the X.509 Certificate, this is called self-signed. With a self-signed certificate the Issuer Name and Subject Name in the X.509 Certificate will be the same.

If an alternative entity has signed the X.509 Certificate, so the private key used to sign the X.509 Certificate does not correspond to the public key contained in the X.509 Certificate, then we refer to the entity performing the signing as a Certificate Authority (CA). With an X.509 Certificate signed by a CA the Issuer Name and Subject Name will be different. As part of the TLS Handshake the client validates the X.509 Certificate. If the Issuer Name and Subject Name are different in the X.509 Certificate, then the client looks for an X.509 Certificate for the Issuer. The X.509 certificate for the Issuer will be validated, as well as the X.509 Certificate for the server.

The Issuer X.509 Certificate could also be signed by an alternative entity. So for the Issuer X.509 Certificate the Subject Name could be different to the Issuer Name. This introduces the concept of a Certificate Chain. A number of X.509 Certificates can be linked together by their respective Subject Names and Issue Names, as illustrated in Figure 3.

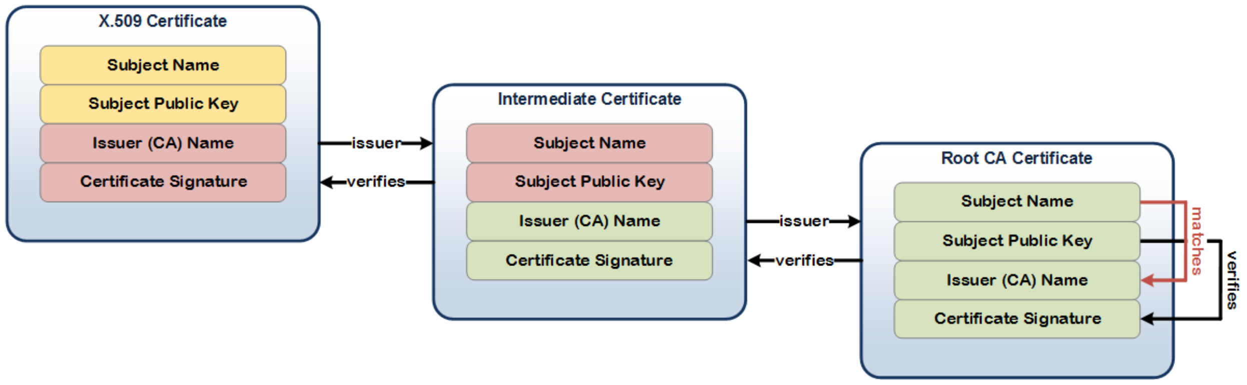

Figure 3. X.509 Certificate Chain

The Certificate Chain is followed by the client until the Subject Name and Issuer Name match. This final X.509 Certificate in the chain is called the Root CA Certificate. Any X.509 Certificates in the chain between the server X.509 Certificate and the Root CA Certificate are called Intermediate Certificates. There can be multiple Intermediate Certificates in the chain. For simplicity, Figure 3 only show a single Intermediate Certificate. Full details of the certification path validation algorithm can be found in RFC 5280 (Cooper, et al. 2008).

TLS HANDSHAKE

Within the TLS Handshake you can define either one-way or two-way certificate validation. For one-way certificate validation the server provides the certificate to the client and the client validates the server certificate. Two-way certificate validation adds a certificate to the client, the client sends their certificate to the server, and the server validates the client certificate. Throughout this paper we only consider the case of one-way certificate validation.

For the TLS Handshake to successfully validate the X.509 Certificate, either the X.509 Certificate (for self-signed certificates) or the Root CA Certificate must be "trusted". The validation algorithm checks to see if the certificate of the issuing CA was issued by a trusted CA, and so on, until either a trusted CA is found, at which point a trusted, secure connection is established. Or no trusted CA can be found at which point the validation algorithm produces an error. Each client holds a list of CAs it will trust. The list of trusted CAs might be specific to the client application or held by the operating system of the client application.

CERTIFICATE FILES AND CONTAINERS

X.509 Certificates and the associated private keys can be stored in different formats and placed into different containers. Table 1 lists the common file formats for X.509 Certificates and private keys. The main difference between the two types of files is whether the content is in binary (.DER) or ASCII (.PEM) format. ASCII format certificates and private keys can be viewed and manipulated with a text editor; while binary format files require an application to manipulate them.

File Extension	Comment
.DER	The DER extension is used for binary Distinguished Encoding Rules (DER) encoded certificates.
.PEM	The PEM extension is used for Privacy-enhanced Electronic Mail (PEM) Base64 encoded certificate, enclosed between "-----BEGIN …" and "-----END …".
.CRT	The CRT extension is used for certificates. The certificates can be encoded as binary DER or as ASCII PEM.
.CER	Alternate form of .crt (Microsoft Convention).
.KEY	The KEY extension is used both for public and private keys. The keys can be encoded as binary DER or as ASCII PEM.

Table 1. Certificate File Formats

Both binary and ASCII format files can be placed into containers. The containers are files that will contain multiple X.509 Certificates and possibly the associated private keys as well. The two most common types of containers are Java keystores and PKCS#12 files. Table 2 lists the common file extensions for these X.509 Certificate containers. PKCS#12 files are very common on Microsoft Windows systems. Java keystores might not have any file extension associated with them.

File Extension	Comment
.JKS	Java keystores can contain certificate(s) and private keys (password protected).
.P7B, .P7C	PKCS#7 SignedData structure without data, just certificate(s) or Certificate Revocation Lists(s).

File Extension	Comment
.JKS	Java keystores can contain certificate(s) and private keys (password protected).
.P12	PKCS#12, can contain certificate(s) and private keys (password protected).
.PFX	PFX, predecessor of PKCS#12.

Table 2. Certificate Containers

CERTIFICATES AND SIGNATURES

There are three ways to sign certificates. In the rest of this paper we focus on the use of Site-Signed X.509 Certificates because they are common throughout many organizations. Many of the points we raise in the rest of this paper can also be applied to Self-Signed X.509 Certificates. While we are not addressing the use of Third-Party Signed X.509 Certificates in the rest of this paper, it is helpful for you to understand some of the differences between Third-Party Signed, Site-Signed, and Self-Signed X.509 Certificates.

As stated above, we can classify the X.509 Certificate based on the CA that signed the certificate. We can split the classification into three types of X.509 Certificate:

- Self-Signed, where the X.509 Certificate is signed by itself.
- Site-Signed, where your organization maintains a central Certificate Authority responsible for signing X.509 Certificates for use only within the organization.
- Third-Party-Signed, where an independent third-party organization is used to sign the X.509 Certificate. Normally Third-Party-Signed X.509 Certificates are used for public facing applications.

THIRD-PARTY SIGNED: A DEFINITION

A Third-Party CA signs X.509 Certificates after verifying the identity of the organization that requests the X.509 Certificate. The Third-Party CA typically charges per certificate, per year, and the costs vary greatly depending on the type of X.509 Certificate requested. The level of verification carried out by the Third-Party CA depends on the type of X.509 Certificate requested as well.

At the basic level is a domain-validated X.509 Certificate where the CA validates the WHOIS record for the domain name and ensures that it matches the information submitted in the certificate request. Alternatively, an organization-validated X.509 Certificate will involve the CA validating the company or organization requesting the X.509 certificate. The validation of the organization typically involves checking government databases to check company registration details.

The highest level X.509 Certificate is an Extended Validation Certificate (CA/Browser Forum. 2014). An Extended Validation (EV) Certificate is designed to prevent phishing attacks more effectively than standard X.509 Certificates. For an EV certificate the CA must complete extensive validation of the organization requesting the certificate, including the following:

- Verifying that the organization is legally registered and active.
- Verifying the address and phone number of the organization.
- Verifying that the organization has exclusive right to use the domain specified in the EV Certificate.
- Verifying the person ordering the certificate has been authorized by the organization.
- Verifying that the organization is not on any government blacklists.

Because of the extensive validation, an EV certificate can take a few days to a few weeks to receive.

SITE-SIGNED: A DEFINITION

Given the costs and the time requirements for obtaining an X.509 Certificate from a Third-Party CA many organizations set up an internal CA. These organizations use their internal CA to provide Site-Signed X.509 Certificates for any applications that will only be accessed within the organization. Typically, for these Site-Signed certificate requests, the mechanism to sign the X.509 Certificates is automated making this a simple and fast approach for internal applications.

Using Site-Signed X.509 Certificates presents some additional challenges for the correct configuration of the SAS environment. Many of the points that we raise in the rest of this paper can also be applied to Self-Signed X.509 Certificates but do not apply to Third-Party Signed Certificates.

HTTPS: WHO DO YOU TRUST?

The signer of the certificate has a central role in establishing trust. As we have seen in the previous sections the question of trust is key to the TLS Handshake. As part of the TLS Handshake the X.509 Certificate is validated. When the validation algorithm cannot establish trust, the validation fails and the TLS Handshake issues an error. Without a successful TLS Handshake the TLS session is not established and the connection fails.

Previously we stated that the client holds the list of trusted CAs or that this list might be held by the operating system of the client. Now we will look at the providers of trust in more detail.

PROVIDERS OF TRUST

To be able to enumerate the providers of trust we need to examine the types of clients. Different types of clients will have different providers of trust. In the third maintenance release of SAS® 9.4 the provider of trust changed for several SAS clients; Table 3 provides a summary of the providers of trust.

TLS Client	Provider of Trust
Web Browser – Internet Explorer / Google Chrome	Microsoft Windows Certificate Store
Web Browser – Mozilla Firefox	Mozilla Firefox
Java Desktop Client	SAS® Security Certificate Framework
Java Web Start Client	Java Runtime Environment
.Net Desktop Client	Microsoft Windows Certificate Store
Mobile Application – iOS & Android	Mobile Operating Systems
SAS® Server Process – Windows	Microsoft Windows Certificate Store
SAS® Server Process – UNIX	SAS® Security Certificate Framework
SAS® Web Application Server	SAS® Security Certificate Framework

Table 3. TLS Clients and Providers of Trust

For releases of SAS® 9.4 prior to the third maintenance release, see the discussions of the required configuration of certificates within a SAS deployment in papers by Heesun Park (Park. 2014) and (Park & Hughes. 2015).

Microsoft Windows Certificate Store

Both Internet Explorer and Google Chrome use the Microsoft Windows Certificate Store as a repository of trusted CA certificates. In addition, .Net clients such as SAS® Enterprise Guide and SAS® Add-in for Microsoft Office, as well as SAS® Foundation processes on Microsoft Windows use the same Certificate Store. The Microsoft Windows Certificate Store is the default repository for X.509 Certificates on Microsoft Windows.

SAS® Security Certificate Framework

Introduced with the third maintenance release of SAS® 9.4, the SAS® Security Certificate Framework enables SAS to provide the Mozilla bundle of trusted CA Certificates (Mozilla Wiki. 2015). Starting with the third maintenance release of SAS® 9.4 any installation that includes the SAS® Private Java Runtime Environment includes the SAS® Security Certificate Framework. In addition, any SAS® installed Java process will use the SAS® Security Certificate Framework as its provider of trust. This framework supersedes the default provider included with the SAS® Private Java Runtime Environment. If you upgrade an existing deployment to the third maintenance release of SAS® 9.4 (or higher), then the deployment tools will prompt you to create the new framework and move certificates as needed.

The SAS® Security Certificate Framework is also included with SAS® Foundation installations on UNIX platforms. This provides SAS® Foundation processes on UNIX platforms with a standard provider for trust, which was not available in previous releases of SAS®.

Java Runtime Environment

For some deployments, users access SAS applications using a Java Web Start process. These Java Web Start clients use the version of the Java Runtime Environment installed on the client machine. These non-installed SAS® Java clients, such as SAS® Enterprise Miner™, do not use the SAS® Security Certificate Framework because no SAS® installation is completed on the client machine. Instead, these clients use the default truststore provided by the Java distribution.

Mozilla Firefox

The Mozilla Firefox browser behaves differently to the other SAS® supported browsers. Mozilla Firefox maintains a separate provider of trust. The truststore used by Mozilla Firefox contains similar content to the SAS® Security Certificate Framework but is managed separately.

Mobile Operating Systems

Both Apple iOS and Google Android maintain their own trust providers in the form of an operating system specific truststore.

ESTABLISHING TRUST: ADDING THE ROOT CERTIFICATE TO THE PROVIDER OF TRUST

To be able to establish trust for a Site-Signed or Self-Signed X.509 Certificate the Root CA Certificate must be added to the provider of trust. Each of the different providers of trust have a different mechanism to add the Root CA Certificate. Table 4 presents a summary of the different mechanisms for each provider of trust.

Provider of Trust	Mechanism to Add Root CA Certificate
Microsoft Windows Certificate Store	Individually to a single host using Microsoft Windows Certificate Manager in Microsoft Management Console. Distributed centrally to multiple hosts using Group Policy within an Active Directory Domain.
SAS® Security Certificate Framework	Individually to a single host using the SAS® Deployment Manager.
Java Runtime Environment	Individually to a single host using the Java keytool command-line interface.
Mozilla Firefox	Individually using the Mozilla Certificate Manager within Mozilla Firefox.
Mobile Operating Systems	Individually to devices using email or web distribution.

Table 4. Mechanisms to Add Root CA Certificate

We focus here on the SAS® Security Certificate Framework. Information about most of the providers of trust is widely available online and is covered in detail in the *SAS® 9.4 Intelligence Platform: Installation and Configuration Guide, Second Edition*. To add a Root CA Certificate to the SAS® Security Certificate Framework you must use the SAS® Deployment Manager. This means that you can only add a certificate after SAS® software has been installed on the host.

The following figures are taken from the *SAS® 9.4 Intelligence Platform: Installation and Configuration Guide, Second Edition*, and summarize the process of:

- Installing the SAS® Security Certificate Framework.
- Subsequently adding a certificate to the trusted bundle.

Figure 4 illustrates the installation of the SAS® Security Certificate Framework and calls out the two important files. The cacerts.pem and cacerts.jks both contain the Mozilla CA Certificate List, but in two different formats. The .pem file contains the Certificate List in BASE-64 encoded ASCII text; while the .jks file contains the Certificate List in a Java keystore.

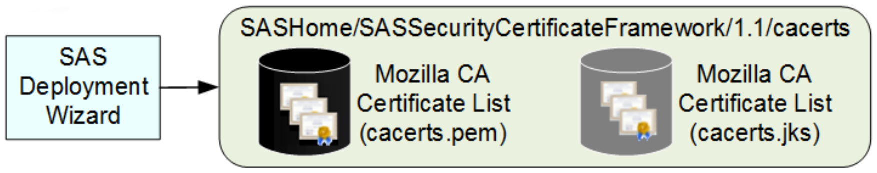
SAS Deployment Wizard
SASHome/SASSecurityCertificateFramework/1.1/cacerts
Mozilla CA Certificate List (cacerts.pem)
Mozilla CA Certificate List (cacerts.jks)

Figure 4. SAS® Deployment Wizard Installs Mozilla Bundle

Figure 5 demonstrates as part of the installation the two initial cacerts files are directly copied to create the trustedcerts.pem and trustedcerts.jks files. At this stage of the installation all four files have exactly the same content. The content in all four files is the Mozilla CA Certificate bundle.

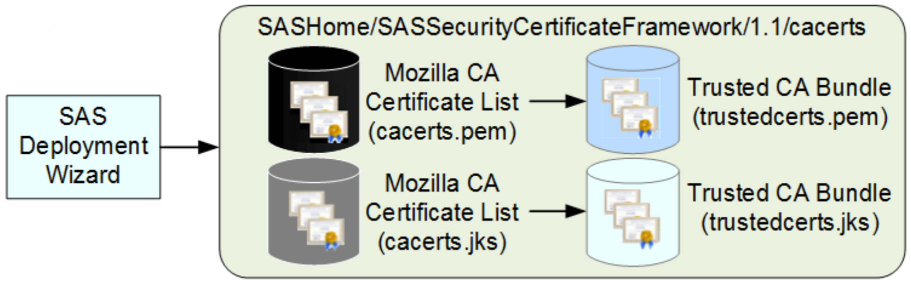
SAS Deployment Wizard
SASHome/SASSecurityCertificateFramework/1.1/cacerts
Mozilla CA Certificate List (cacerts.pem)
Trusted CA Bundle (trustedcerts.pem)
Mozilla CA Certificate List (cacerts.jks)
Trusted CA Bundle (trustedcerts.jks)

Figure 5. SAS® Deployment Wizard Creates Trusted CA Bundle

The final stage of the SAS® Deployment Wizard tasks is shown in Figure 6, where the Java keystore version of the trustedcerts file is copied to the SAS® Private Java Runtime Environment. As per the Java standard, the jssecacerts file takes precedence over any existing cacerts files located within the Java Runtime Environment file structure. Therefore, any Java process using this installation will use the jssecacerts truststore without the need for any additional Java Virtual Machine (JVM) options.

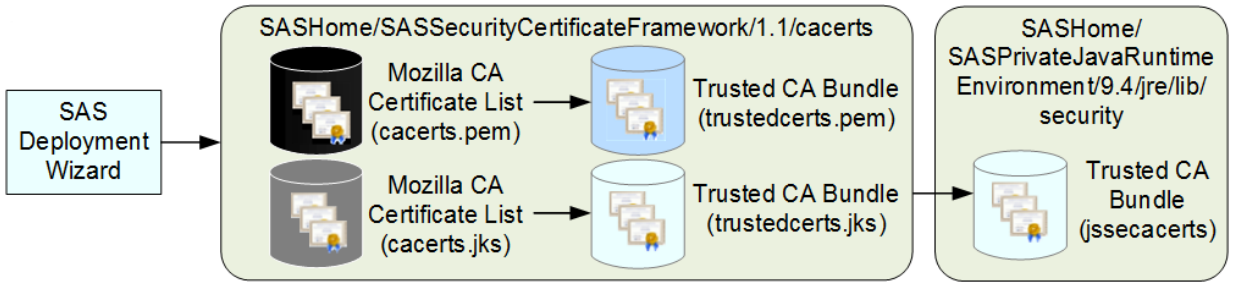
SAS Deployment Wizard
SASHome/SASSecurityCertificateFramework/1.1/cacerts
Mozilla CA Certificate List (cacerts.pem)
Trusted CA Bundle (trustedcerts.pem)
Mozilla CA Certificate List (cacerts.jks)
Trusted CA Bundle (trustedcerts.jks)
SASHome/ SASPrivateJavaRuntime Environment/9.4/jre/lib/ security
Trusted CA Bundle (jssecacerts)

Figure 6. SAS® Deployment Wizard Creates Trusted CA Bundle in the SAS® Private Java Runtime Environment

The key process for you to add any site-signing Root CA Certificates is depicted in Figure 7. The SAS® Deployment Manager task is used to add Root CA Certificates to the trustedcacerts files within the SAS® Security Certificate Framework installation. As with the initial installation shown in Figure 6, the Java keystore version of the updated trustedcacerts file is again copied to the SAS® Private Java Runtime Environment.

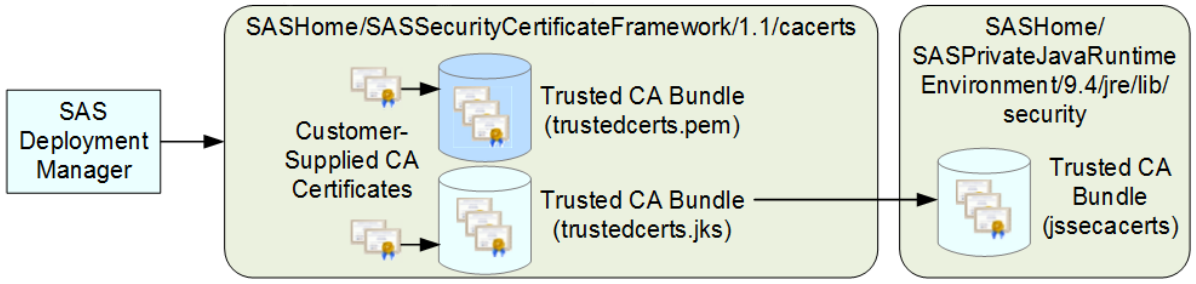
SAS Deployment Manager
SASHome/SASSecurityCertificateFramework/1.1/cacerts
Customer-Supplied CA Certificates
Trusted CA Bundle (trustedcerts.pem)
Trusted CA Bundle (trustedcerts.jks)
SASHome/SASPrivateJavaRuntimeEnvironment/9.4/jre/lib/security
Trusted CA Bundle (jssecacerts)

Figure 7. SAS® Deployment Manager Adds Certificates to the Trusted Bundle

You can review a YouTube video (Rogers. 2015) covering the addition of certificates in the SAS® Technical Insights and Expertise channel.

HTTPS: HOW FAR DO YOU GO?

We have looked at the method of establishing trust in an HTTPS connection. Now let's look at encryption options. One important question that we must address when considering the use of HTTPS with SAS 9.4 is what you want the encryption to cover, as illustrated by Figure 8.

We can configure the encryption of traffic in three key places:

1. Between web browsers or mobile clients to the SAS® Web Server. The SAS® Web Server operates as a proxy server for the SAS® Web Application Server instances actually running the SAS® web applications.
2. Between the SAS® Web Server and the SAS® Web Application Server instances.
3. For the SAS® Environment Manager Server, which impacts both browser-based clients and the SAS® Environment Manager Agents.

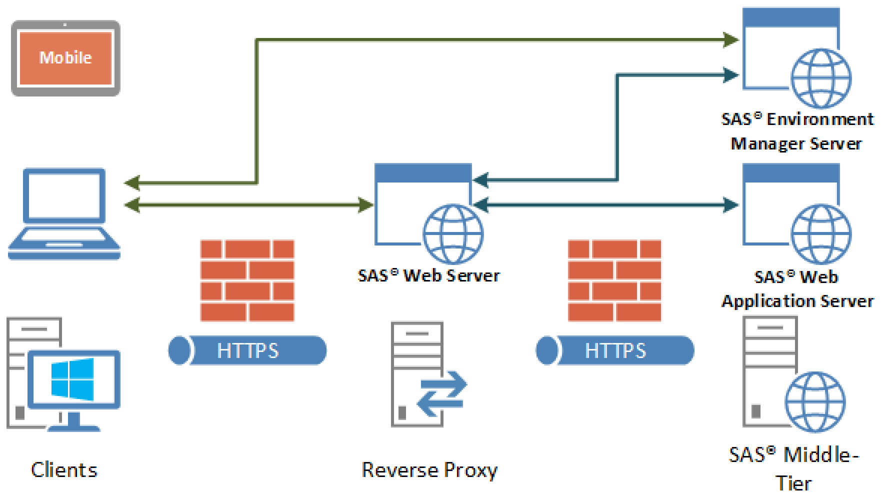

Figure 8. HTTPS to Each Middle-Tier Application

HTTPS TO THE SAS® WEB SERVER

Enabling HTTPS for an environment is becoming the default configuration for many customers since there is an industry-wide drive to use HTTPS by default (Grigorik and Far 2014). When you consider using HTTPS for your SAS environment, setting up the SAS® Web Server to use HTTPS is probably your first consideration. This is the recommended configuration for any Internet-facing environment, and SAS deployment tools make it easy to setup security for the SAS® Web Server.

HTTPS TO THE SAS® WEB APPLICATION SERVER

A less common configuration is to include HTTPS to the SAS® Web Application Server. Since the SAS® Web Server, acting as a proxy, needs to examine the content of the packets sent from the client to the Middle-Tier, setting up HTTPS for the SAS® Web Application Server introduces an additional encryption step. This means that the encrypted transmission from the client to the SAS® Web Server is decrypted at the SAS® Web Server and then a separate HTTPS connection is made between the SAS® Web Server and SAS® Web Application Server.

You might want to consider HTTPS to the SAS® Web Application Server if you need to prove the security of the network between the SAS® Web Server host and the SAS® Web Application Server host.

Protecting cookies is another reason you might want to enable HTTPS for the SAS® Web Application Server. Enabling secure cookies ensures that cookies are only sent by the browser when the request is going to an HTTPS site. This prevents the content of cookies from being observed by unauthorized parties due to the transmission of the cookie in clear text. Even if both the SAS® Web Server and SAS® Web Application Server processes are running on the same host, you will need to enable HTTPS for the SAS® Web Application Server if you want to enable secure cookies.

HTTPS TO THE SAS® ENVIRONMENT MANAGER

The SAS® Environment Manager includes both a Server component and Agents that are deployed to each of the server machines in your deployment. For the configuration of HTTPS we are concerned with the SAS® Environment Manager Server. The SAS® Environment Manager Server includes an encapsulated Java Application Server that is separate from the SAS® Web Application Server. For more information about SAS® Environment Manager, see the paper “Monitoring 101: New Features in SAS 9.4 for Monitoring Your SAS Intelligence Platform” (Peters. et al. 2013).

By default, the SAS® Deployment Wizard configures HTTPS to the SAS® Environment Manager. However, this is configured with a self-signed X.509 Certificate generated by the SAS® Environment Manager during configuration. In addition, the HTTP listener is not disabled, so a default configuration has both HTTP & HTTPS for the SAS® Environment Manager.

Since the default HTTPS connection uses a self-signed X.509 certificate, you will receive warnings in your browser about the certificate trust when trying to connect to the SAS® Environment Manager. As such, updating the HTTPS connection with a site-signed or third-party signed certificate is recommended.

HTTPS: WHEN DO YOU CONFIGURE IT?

The SAS® Web Server and the SAS® Environment Manager Server should be configured for HTTPS during the initial deployment, using the SAS® Deployment Wizard. You must manually configure HTTPS for the SAS® Web Application Server if you want to add that layer of security to your deployment.

You can choose to manually configure the SAS® Web Server for HTTPS. This is not a recommended practice. The *SAS® 9.4 Intelligence Platform: Middle-Tier Administration Guide, Third Edition* contains instructions for manually configuring the SAS® Web Server for HTTPS; we recommend that you avoid manual configuration. The process for manual configuration of HTTPS with the SAS® Web Server contains multiple steps where errors can be introduced. Also, the manual configuration steps will need to be reverted before applying any future maintenance to the SAS environment.

For the SAS® Web Application Server, only manual configuration of HTTPS is supported. If you want to enable HTTPS for the SAS® Web Application Server, you make changes after the SAS deployment has been installed and configured. Some administration functions you carry out on your SAS deployment will automatically revert parts of the configuration. Anytime a SAS web application is redeployed the SAS® Web Server configuration is rebuilt. Rebuilding the SAS® Web Server configuration removes the manual changes required for HTTPS to the SAS® Web Application Server. When applying an upgrade or maintenance to the environment, you will need to revert the manual configuration changes to remove the HTTPS settings: you can reapply them after the upgrade.

HTTPS: HOW DO YOU CONFIGURE IT?

In this next section we examine in detail the process for configuring HTTPS with a site-signed X.509 Certificate. We list out the prerequisites and step through the configuration changes that you need to make. We split the next section into two main areas; the first deals with the automatic configuration completed with the SAS® Deployment Wizard and the second deals with the manual configuration steps.

AUTOMATIC CONFIGURATION WITH THE SAS® DEPLOYMENT WIZARD

We use the SAS® Deployment Wizard to configure HTTPS for both the SAS® Web Server and the SAS® Environment Manager Server.

Prerequisites for the SAS® Web Server

The SAS® Web Server requires a private key and X.509 Certificate in PEM Base-64 format. The private key should be an RSA (Rivest-Shamir-Adleman) private key with a recommended minimum length of 2048 bits. Also, the private key should not be encrypted or have a password associated to it. Tools such as OpenSSL (https://www.openssl.org) can be used to create the RSA private key. For example, the following command generated a key with a length of 2048 bits:

```
openssl genrsa -out key.pem 2048
```

The header and footer lines of the key.pem are normally:

```
-----BEGIN RSA PRIVATE KEY-----
-----END RSA PRIVATE KEY----
```

If a password, and hence encryption, has been set on the private key, it should be removed before providing the file to the SAS® Deployment Wizard. Tools such as OpenSSL can also be used to remove the encryption and password from the private key. For example, the following command using OpenSSL removes the password:

```
openssl rsa -in key.pem -out keyout.pem
```

Once the private key has been generated, the X.509 Certificate Signing Request (CSR) can be generated. This will be sent to the CA to be signed. Again OpenSSL can be used to generate the CSR:

```
openssl req -new -key key.pem -out csr.pem
```

OpenSSL will then prompt for the following information:

- Country Name (2 letter code) [XX]:
- State or Province Name (full name) []:
- Locality Name (e.g., city) [Default City]:
- Organization Name (e.g., company) [Default Company Ltd]:
- Organizational Unit Name (e.g., section) []:
- Common Name (e.g., your name or your server's host name) []:
- Email Address []:
- A challenge password []:
- An optional company name []:

It is important that you enter the fully qualified domain name for the SAS® Web Server in response to the Common Name prompt. The header and footer lines of the csr.pem file are normally:

```
-----BEGIN CERTIFICATE REQUEST-----
-----END CERTIFICATE REQUEST-----
```

The contents of the csr.pem file can then be submitted to your Certificate Authority (CA) for signing. The PEM format signed X.509 Certificate that you receive back from the CA will have header and footer lines like the following:

```
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----
```

Your CA might not provide the X.509 Certificate in PEM Base-64 format. If a different format of X.509 Certificate is provided by the CA, it will need to be converted before it can be used. OpenSSL can be used to convert the X.509 Certificate between different formats. For example, to convert from a PKCS12 format file to a PEM Base-64 format file use the following command:

```
openssl pkcs12 -in certificate.pfx -nodes -out certificate.pem
```

Alternatively, to convert from a DER Binary format file to a PEM Base-64 format file use the following command:

```
openssl x509 -inform der -in certificate.der -out certifcate.pem
```

Both the private key and X.509 Certificate are provided to the SAS® Deployment Wizard during the configuration.

Prerequisites for the SAS® Environment Manager Server

The SAS® Environment Manager Server requires a Java keystore containing the private key and X.509 Certificate. The password for the private key must be the same as the password for the Java keystore. The private key for the SAS® Environment Manager Server should use the RSA algorithm and be a minimum length of 2048 bits. The Java keytool (http://docs.oracle.com/javase/7/docs/technotes/tools/windows/keytool.html) utility, included in the SAS® Private Java Runtime Environment, can be used to create the Java keystore and generate the private key with the following command:

```
keytool -genkey -alias SASEV -keyalg RSA -keystore sasev.jks -keysize 2048
```

You will be prompted to set the password for the new Java keystore password. Then you will be prompted to provide the following details:

- What is your first and last name? [Unknown]:
- What is the name of your organizational unit? [Unknown]:
- What is the name of your organization? [Unknown]:
- What is the name of your City or Locality? [Unknown]:
- What is the name of your State or Province? [Unknown]:
- What is the two-letter country code for this unit? [Unknown]:
- Is CN=*fully.qualified.hostname*, OU=*Organizational Unit*, O=*Organization*, L=*City*, ST=*State*, C=*Country* correct? [no]:
- Enter key password for <SASEV> (RETURN if same as keystore password):

It is important that you enter the fully qualified domain name for the SAS® Environment Manager Server at the first prompt, when you are prompted for your first and last name. Just pressing ENTER for the last prompt ensures that the password for the private key matches the password for the Java keystore.

Once the private key has been generated, the X.509 Certificate Signing Request (CSR) can be generated. This will be sent to the CA to be signed. Java keytool can be used to create the CSR with the following command:

```
keytool -certreq -alias SASEV -keystore sasev.jks -file sasev.csr
```

You will need to provide the password for the Java keystore for the CSR to be created. Once your CA has signed the request, you will need to import the X.509 Certificates provided by the CA into the Java keystore. First you should import the CA Root Certificate and any intermediate certificates, then once these have been imported you can import the signed X.509 Certificate. Again, the Java keytool utility can be used to import the CA Root Certificate with the following command:

```
keytool -importcert -trustcacerts -alias CARoot -file ca_cert.pem -keystore
sasev.jks
```

To import a CA intermediate certificate use the following command:

```
keytool -importcert -trustcacerts -alias CAInt -file ca_int_cert.pem -
keystore sasev.jks
```

To import the X.509 Certificate using Java keytool use the following command:

```
keytool -importcert -trustcacerts -alias SASEV -file sasev_cert.pem -
keystore sasev.jks
```

You can validate the content of the SAS® Environment Manager Java keystore with the Java keytool utility with the following command:

```
keytool -list -keystore sasev.jks
```

Output 1 shows the contents of an example SAS® Environment Manager keystore.

```
Keystore type: JKS
Keystore provider: SUN

Your keystore contains 2 entries

sasev, May 22, 2015, PrivateKeyEntry,
Certificate fingerprint (SHA1):
D0:A8:AB:36:41:8A:3D:50:F1:8C:58:FC:96:2B:99:E6:0D:9C:C5:06
gelcaroot, May 22, 2015, trustedCertEntry,
Certificate fingerprint (SHA1):
0A:ED:15:59:C0:76:89:94:AB:6D:2A:33:41:79:CA:C5:B3:6C:D8:2D
```

Output 1. Example Contents of a Java Keystore

The Java keystore file and password for the Java keystore will need to be provided to the SAS® Deployment Wizard.

Prerequisites for the Trust Chain

To enable the different components of the SAS environment to trust the X.509 Certificates generated for the SAS® Web Server and the SAS® Environment Manager Server, the trust chain must be made available. This means that you will need to have the CA Root Certificate and any intermediate certificates available in the correct format. The SAS® Deployment Manager will require these files in BASE-64 PEM encoded format. If the files have been provided in a different format, then you will need to convert them before they can be used. OpenSSL provides a mechanism to convert from one format to another and we provided example commands in the section above on Prerequisites for the SAS® Web Server.

Configuration Options

This section lists the 10 steps that you perform to configure your SAS deployment to support HTTPS with your site-signed X.509 Certificates. The order in which different steps are completed is key to a successful implementation. In this section we outline the steps you need to complete and the correct order in which to complete them. For more details about any specific step please refer to the *SAS® 9.4 Intelligence Platform: Installation and Configuration Guide, Second Edition*. At a high level you should complete the following steps:

1. Complete the prerequisites including the additional items we've covered in detail in the preceding sections: prerequisites for SAS® Web Server, SAS® Environment Manager Server, and Trust Chain.
2. Install and configure your SAS® Metadata tier and SAS® Server tier using the SAS Deployment Wizard.
3. For both Windows and UNIX, on the SAS® Metadata tier and SAS® Server tier machines add your CA Root certificate and any intermediate certificates to the SAS® Security Certificate Framework using the SAS® Deployment Manager.
4. For Windows, on the SAS® Metadata tier and SAS® Server tier machines add your CA Root certificate and any intermediate certificates to the Windows certificate stores.
5. Install your SAS® Middle-tier using the SAS® Deployment Wizard. **Do <u>not</u> complete the configuration at this stage**.
6. For both Windows and UNIX, use the SAS® Deployment Manager to add your CA Root certificate and any intermediate certificates to the SAS® Security Certificate Framework on the SAS® Middle-tier.
7. For Windows, also add the CA Root certificate and any intermediate certificates to the Windows certificate stores.

8. Configure your SAS® Middle-tier using the SAS® Deployment Wizard. Choose “Custom” prompting so that you can change the SAS® Environment Manager Server keystore options. You will need to provide or set the following items during configuration:
   - SAS® Web Server private key file.
   - SAS® Web Server X.509 Certificate file.
   - SAS® Environment Manager Server Java keystore.
   - SAS® Environment Manager Server Java keystore password.
   - Check the option for “Establish secure communication” on the screen “SAS Environment Manager Agent Communication”.
9. For client machines with SAS software installed, use the SAS® Deployment Wizard to add your CA Root certificate and any intermediate certificates to the SAS® Security Certificate Framework. Also add the same certificates to the Windows certificate stores.
10. For client machines using Java Web Start clients, add your CA Root certificate and any intermediate certificates to the Java Runtime Environment truststore using the Java keytool utility.

Completing these 10 steps will configure your SAS environment to use HTTPS with your site-signed X.509 Certificates. At the end of the SAS® Deployment Wizard configuration on your Middle-tier, you will be able to access your environment over HTTPS. In the next section we cover the manual steps you need to carry out to complete your configuration.

MANUAL CONFIGURATION

The manual configuration we need to complete covers the configuration of HTTPS for the SAS® Web Application Server and disabling the HTTP listener for the SAS® Environment Manager Server.

Prerequisites for the SAS® Web Application Server

Each of your SAS® Web Application Server instances will require a Java keystore. In the Java keystore we need to have a private key using the RSA algorithm with a minimum length of 2048 bits and a signed X.509 Certificate for the fully qualified domain name of the machine running the instance. If you are going to run a horizontal cluster, where there are multiple machines running the SAS® Web Application Server, you will need a signed X.509 Certificate for each machine. However, if you are running a vertical cluster or with just one machine you will only need a single signed X.509 Certificate.

If you are running only a single machine for the SAS® Middle-Tier, the same machine will be running the SAS® Environment Manager Server. It is up to you whether you share the same private key and signed X.509 Certificate between the SAS® Environment Manager Server and the SAS® Web Application Server. It would be considered to be more secure to have a separate private key and hence separate signed X.509 Certificate for each process.

Since all end-user connections to your SAS® Web Application Server instances are made via the SAS® Web Server, you could use a self-signed X.509 Certificate for the SAS® Web Application Server instances. Remember, from the earlier section on trust, who signs the X.509 Certificate does not impact the level of encryption used. Since only the SAS® Web Server will be a client of the SAS® Web Application Server, we can easily manage the trust between the two processes.

Manual Steps for SAS® Environment Manager

The SAS® Environment Manager Server is configured to listen on both HTTP and HTTPS. While the steps that we have completed in the SAS® Deployment Wizard have ensured that the HTTPS connection is using your X.509 Certificate, we need to disable the HTTP connection. This ensures that only secure connections to the SAS® Environment Manager Server are available.

The two connections the SAS® Environment Manager listens on are defined in the following file:

```
<SAS_CONFIG_DIR>/Web/SASEnvironmentManager/server-5.8.0-EE/hq-engine/hq-
server/conf/server.xml
```

To disable the HTTP listener you can simply comment out the first connector element in the server.xml file. To comment out an entry place “<!--“ before the entry and “-->” after the entry.

Also, to ensure that the correct ciphers are used on the HTTPS connection you update the list of ciphers in the second connector entry with the following list:

```
ciphers="TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_ECDSA_WITH_AES_2
56_GCM_SHA384,TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_
256_GCM_SHA384,TLS_RSA_WITH_AES_256_CBC_SHA,TLS_RSA_WITH_AES_128_CBC_SHA"
```

You must enter the ciphers list as a single line within the server.xml configuration file. This completes all the manual steps for the SAS® Environment Manager Server and this can be restarted.

Manual Configuration of SAS® Web Application Server

If you choose to add HTTPS for the SAS® Web Application Server, you will need to manually configure those settings. The manual changes detailed in this section need to be reverted before applying any maintenance releases or upgrades to the environment.

Steps for Self-Signed Certificates

If you have decided to use self-signed certificates for the SAS® Web Application Server instances, you will need to ensure that these X.509 Certificates are trusted by the clients of the SAS® Web Application Server. To configure the trust you need to make the X.509 Certificates available to the SAS® Web Server, SAS® Web Application Server, and SAS® Environment Manager Server. If you have a single machine for the SAS® Web Application Server you will have only one X.509 Certificate, while if you have a horizontal cluster for the SAS® Web Application Server you will have multiple X.509 Certificates.

For the SAS® Web Server you need to make either the single X.509 Certificate or the series of X.509 Certificates available in BASE-64 PEM encoded format. For the single X.509 Certificate you can simply provide the PEM file to the SAS® Web Server. If you have multiple X.509 Certificates, you should append them all to a single BASE-64 PEM encoded file and make this file available to the SAS® Web Server.

To make the self-signed Certificate or certificates trusted by the SAS® Web Application Server you must use the SAS® Deployment Manager to add them to the SAS® Security Certificate Framework. To be able to add the certificates to the SAS® Security Certificate Framework the certificates must be provided to the SAS® Deployment Manager in BASE-64 PEM encoded format. You will need to run the SAS® Deployment Manager multiple times if you have multiple certificates to add.

To make the X.509 Certificate or certificates available to the SAS® Environment Manager Server you will need to add them to the Java keystore used by the SAS® Environment Manager Server. The Java keytool utility can be used to import the certificates into the Java keystore that you provided to the SAS® Environment Manager Server.

Steps for Site-Signed Certificates

To use a site-signed X.509 Certificate or certificates for the SAS® Web Application Server instances, you need to ensure the CA Root Certificate and any intermediate certificates are made available to the SAS® Web Server and SAS® Environment Manager Server. If the same CA is used for the SAS® Web Application Server instance as you used for the SAS® Environment Manager Server, you will not need to do anything extra. If a different CA is used for the SAS® Web Application Server instances, you will need to use the Java keytool utility to import the CA Root Certificate and any intermediate certificates into the Java keystore that you provided to the SAS® Environment Manager Server.

To ensure that the site-signed X.509 Certificate or certificates are trusted by the SAS® Web Server, you must make the CA Root Certificate and any intermediate certificates available in BASE-64 PEM encoded format. You can append the certificates into a single file and use this with the SAS® Web Server.

Change SAS® Web Application Server Instances

You need to make two changes for each SAS® Web Application Server instance. First, you update the server.xml located in:

```
<SAS_CONFIG_DIR>/Web/WebAppServer/SASServern_m/conf/
```

Duplicate the existing Connector element and complete the following:

- Add the following attributes to the new connector element:

```
secure="true"
SSLEnabled="true"
sslProtocols="TLSv1,TLSv1.1,TLSv1.2"
keystoreFile="/path-to-myhost.jks"
keystorePass="changeit"
```

- Update the value port="${bio.http.port} to port="${bio.https.port}" in the new connector element.

Once you have completed the changes and confirmed that the SAS® Web Application Server is using HTTPS correctly, you can comment out the original HTTP connection.

Next you need to update the JVM options for the specific SAS® Web Application Server instance. For SASServer1_1 set the following options:

```
-Dsas.scs.port=<https-port>
-Dsas.scs.scheme=https
-Dsas.auto.publish.port=<https-port>
-Dsas.auto.publish.protocol=https
```

While for SASServern_m set the following options:

```
-Dsas.auto.publish.port=<https-port>
-Dsas.auto.publish.protocol=https
```

Where the <https-port> value can be found in the catalina.properties file referenced as the bio.https.port property.

To set the JVM options on Windows edit the following files:

```
<SAS_CONFIG_DIR>/Web/WebAppServer/SASServern_m/bin/setenv.bat
<SAS_CONFIG_DIR>/Web/WebAppServer/SASServern_m/conf/wrapper.conf
```

To set the JVM options on UNIX edit the following file:

```
<SAS_CONFIG_DIR>/Web/WebAppServer/SASServern_m/bin/setenv.sh
```

This completes all the changes required by the SAS® Web Application Server instances.

Manual Configuration of Secured Cookies

The final manual configuration step for the SAS® Web Application Server is to manually configure the secure attribute for cookies, which directs the web browser to only send such cookies through an encrypted HTTPS connection. To configure the SAS® Web Application Server to return the session ID with the secure attribute, edit the web.xml for each instance, located in:

```
<SAS_CONFIG_DIR>/Web/WebAppServer/SASServern_m/conf/
```

You add the following to the existing session-config within the web.xml:

```
<session-config>
        <session-timeout>30</session-timeout>
        <cookie-config>
           <secure>true</secure>
        </cookie-config>
    </session-config>
```

To configure the SAS® Logon Manager to set the secure attribute on the cookies that it sets, edit the ticketGrantingTicketCookieGenerator.xml file located in:

```
<SASHOME>/SASWebInfrastructurePlatform/9.4/Static/wars/sas.svcs.logon/
WEB-INF/spring-configuration/
```

You update the value of p:cookieSecure="false" to p:cookieSecure="true". After making the change to the SAS® Logon Manager, you must rebuild and deploy the SAS® Web Infrastructure Platform using the SAS Deployment Wizard.

Change SAS® Web Server

You update the SAS® Web Server so that connections are made to the new HTTPS connector on the SAS® Web Application Server. The configuration file that defines these connections is the sas.conf located in:

```
<SAS_CONFIG_DIR>/Web/WebServer/conf
```

There are two sets of changes that you make to the sas.conf file.

1. At the top of the sas.conf file add the following directives, before the ProxyStatus line:

   ```
   SSLProxyEngine on
   SSLProxyVerify require
   SSLProxyVerifyDepth 10
   SSLProxyCACertificateFile "/path-to/myhost.pem"
   ```

   The SSLProxyCACertificateFile is the directive used to provide the CA certificate to enable the SAS® Web Server to trust the SAS® Web Application Server instance's X.509 Certificate. As we discussed above, this could be either a single BASE-64 PEM encoded certificate file or a single file containing multiple BASE-64 encoded certificates.

2. Edit all the BalancerMember directives and change both the protocol to HTTPS and port to the HTTPS port of the SAS® Web Application Server instance. For example:

   ```
   BalancerMember https://myhost.example.com:8443
       route=myhost.example.com_SASServer1_1
   ```

This completes the changes for the SAS® Web Server. At this point you can restart the SAS® Web Server and the SAS® Web Application Server instances.

Important: each time you deploy a SAS web application you will need to update the SAS® Web Server sas.conf configuration file.

CONCLUSION

In this paper we reviewed the process involved in establishing an HTTPS connection and identified the key components that impact establishing a secure session. We examined the contents of an X.509 Certificate and showed how the elements are used in the HTTPS handshake. We discussed that HTTPS is more than just encryption and described the method of establishing trust. For establishing trust we presented the importance of the signing Certificate Authority and how the chain of trust is used by a client to establish which X.509 Certificates to trust. We reviewed how different clients are able to establish trust and the mechanisms used to add new Certificate Authorities to these different truststores.

Next we reviewed how far to take the HTTPS connection. We saw that you can configure HTTPS for the SAS® Web Server, SAS® Web Application Server, and SAS® Environment Manager Server. For each of these SAS components we then reviewed how you configure HTTPS. We stated that only the SAS® Web Application Server should be configured manually and that the SAS® Deployment Wizard should be used for the SAS® Web Server and SAS® Environment Manager Server.

We described in detail what types of private keys and X.509 Certificates need to be provided to the three components. Some example commands were presented to enable you to convert files between different formats to ensure that the correct format is available when you configure your SAS environment.

Finally, we listed the correct order in which to complete the configuration steps, both automatic with the SAS® Deployment Wizard, and manual changes. We provided the detail of the configuration settings and which configuration files you need to change.

So long as you follow the information presented on the prerequisites, obtain the files in the correct format, and follow the order of steps we've discussed you will be successful in your deployment.

REFERENCES

Cooper, et al. May 2008. "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile." RFC 5280. Available at https://tools.ietf.org/html/rfc5280.

CA/Browser Forum. October 2014. "Guidelines for the Issuance and Management of Extended Validation Certificates." Available at https://cabforum.org/wp-content/uploads/EV-V1_5_2Libre.pdf.

Grigorik, Ilya, and Pierre Far. "Google I/O 2014 – HTTPS Everywhere." YouTube. Published June 2014. Available at https://www.youtube.com/watch?v=cBhZ6S0PFCY.

Park, Heesun. 2014. "Advanced Security Configuration Best Practices for SAS® 9.4 Web Applications and Mobile Devices." *Proceedings of the SAS Global Forum 2014 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings14/SAS054-2014.pdf.

Park, Heesun, and Jerome Hughes. 2015. "SSL Configuration Best Practices for SAS® Visual Analytics 7.1 Web Applications and SAS® LASR™ Authorization Service." *Proceedings of the SAS Global Forum 2015 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings15/SAS1541-2015.pdf.

Mozilla Wiki. 2015. "Mozilla Included CA Certificate List." Accessed January 6, 2016. https://wiki.mozilla.org/CA:IncludedCAs.

Rogers, Stuart. "Managing TLS Certificates in SAS 9.4." YouTube. Published September 2015. Available at https://www.youtube.com/watch?v=eWfHhZxqogQ.

Peters, Amy, Bob Bonham and Zhiyong Li. 2013. "Monitoring 101: New Features in SAS 9.4 for Monitoring Your SAS Intelligence Platform". *Proceedings of the SAS Global Forum 2013 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings13/463-2013.pdf.

SAS Usage Note 57370. 2016. "Downloading, installing, and using the TLS/SSL Diagnostic Tool for SAS® 9.4." Available at http://support.sas.com/kb/57/370.html.

ACKNOWLEDGMENTS

I would like to thank the following people for taking the time to review and contribute to this paper:

- Adam Bullock
- Donna Bennett
- Merri Jensen

RECOMMENDED READING

- *SAS® 9.4 Intelligence Platform: Installation and Configuration Guide, Second Edition*
- *SAS® 9.4 Intelligence Platform: Security Administration Guide, Second Edition*
- *SAS® 9.4 Intelligence Platform: Middle-Tier Administration Guide, Third Edition*
- *Encryption in SAS® 9.4, Fifth Edition*

CONTACT INFORMATION

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

Stuart J Rogers
SAS Institute Inc.
stuart.rogers@sas.com
http://www.sas.com