Encryption in SAS® Viya® 3.4: Data in Motion

Overview

Encryption Coverage

SAS Viya provides encryption in two contexts:

- Data in motion is data that is being transmitted to another location. Data is most vulnerable while in transit. Sensitive data in transit should be encrypted. TLS is used as the mechanism to provide encryption for data in motion. This document covers encrypting data in motion.

  Note: All discussion of TLS is also applicable to the predecessor protocol, Secure Sockets Layer (SSL).

- Data at rest is data stored in databases, file servers, endpoint devices, and various storage networks. This data can be on-premises, virtual, or in the cloud. This data is usually protected in conventional ways by access controls. Numerous layers of defense are needed, and encrypting sensitive data is another layer. See "Encryption for Data at Rest: Overview" in *Encryption in SAS Viya: Data at Rest*.

Encryption in a SAS Viya Full Deployment on Linux

In a full deployment of SAS Viya on Linux, almost all external network connections are secured by default. SAS Viya is deployed with Transport Layer Security (TLS) to secure network connections and is fully compliant with SAS security standards. You can harden the full Linux deployment by blocking external connections to port 80, by adding custom certificates on all machines in the deployment, and by upgrading the security protocol and ciphers that are enabled by default. You can also configure TLS- encrypted connections between CAS workers and take additional steps to secure SAS Embedded Process. See "Tasks to Harden Security for Your Linux Full Deployment" on page 3.

Encryption in a SAS Viya Programming-Only Deployment on Linux

In a SAS Viya programming-only deployment, the basic framework for security is included by default, but is not enabled by default. However, you can harden the deployment and enable TLS by following the tasks outlined in "Tasks to Harden Security for Your Linux Programming-Only Deployment" on page 4.

Encryption in a SAS Viya Deployment on Windows

In a Windows deployment, the deployment provides a default level of encryption for data in motion. You can increase the level of security for the deployment by blocking external connections to port 80, by adding custom certificates on Apache httpd, and by upgrading the security protocol and ciphers that are enabled by default. You can upgrade to custom certificates on CAS and SAS/CONNECT. You can also configure TLS encrypted connections between your LDAP provider and SAS Viya. See "Tasks to Harden Security for Your Windows Deployment" on page 5.

Terminology

Various security strategies are used to maintain data usability and data confidentiality, as well as to validate the integrity of content. Various encryption, hashing, and encoding algorithms are used by SAS to protect your data in motion and data at rest. SAS highly recommends using TLS for protecting data that is exchanged in a networked environment.

encoding
: Encoding transforms data into another format using a scheme that is publicly available so that it can easily be reversed. It does not require a key. The only thing required to decode it is the algorithm that was used to encode it. PROC PWENCODE, for example, encodes passwords.

encryption
: Encryption is a process of protecting data. Encryption transforms data into another format in such a way that only specific individuals can reverse the transformation. It uses a key that is kept secret, in conjunction with the plaintext and the algorithm, in order to perform the encryption operation. As such, the ciphertext, algorithm, and key are all required to return to the plaintext. Example encryption algorithms are AES and RSA. TLS is an encryption technology.

hashing
: Hashes are commonly used to validate passwords without having to store or record the password itself. Hash algorithms are one-way functions. They turn any amount of data into a fixed-length "fingerprint" that cannot be reversed. If the input changes by even a tiny bit, the resulting hash is completely different. When passwords are hashed, only the hash is kept. To verify a password, you hash the password and check to see whether the password matches the stored hash. SHA-256 is a hashing algorithm.

salting
: Salt is data used as an additional input to the algorithm that encrypts data. The salt is randomly generated and is used to increase the difficulty of brute-force decryption attacks on the data.

How To

This section provides tasks that can be performed to strengthen (harden) the security of your SAS Viya deployment and tasks to use the default security provided by the SAS Viya deployment. There are also tasks that help you manage truststores, generate new certificates, manage tokens, enable and disable TLS using port families and more.

SAS recommends that you harden the security of your SAS Viya deployment by completing the tasks described in “Harden TLS Security for Your SAS Viya Deployment” on page 3.

Harden TLS Security for Your SAS Viya Deployment

This section provides a roadmap of the tasks that SAS recommends being performed post-deployment to harden security for your SAS Viya deployment. SAS Viya provides a level of security during the deployment process that differs depending on the type of deployment (full or programming-only) and on the platform that you are using (Windows or Linux).

You can choose to enable the default security provided by SAS Viya. However, SAS recommends that you strengthen security by following the tasks to harden security for your deployment.

Tasks to Harden Security for Your Linux Full Deployment

The SAS Viya deployment on Linux provides default security at deployment. SAS recommends that the following additional tasks be performed to increase the level of security and secure any points of entry that are not secured by default.

1. Secure the Apache HTTP Server by adding certificates that conform to the policies at your enterprise and strengthen the default cryptography.

   See “Update Apache HTTP Server TLS Certificates and Cryptography ” on page 5 and “Update the Default Ciphers and TLS Protocol on the Apache HTTP Server” on page 16.

2. Enforce HTTPS for access to SAS Viya using options for Port 80 on page 7.

3 Enable TLS and configure CAS TLS to use your custom certificates. End users can access CAS from outside the Visual Interfaces, either from a third-party language like Python, Java, or Lua, or directly from SAS 9.4M5. For end-user access, SAS recommends that you use sanctioned certificates for the entry point to CAS and update the private key and server certificate used by CAS.

   See "Configure CAS TLS to Use Custom Certificates (Linux Full Deployment)" on page 17.

4 If you are using SAS/CONNECT to access data from older versions of SAS 9.4, enable TLS for SAS/CONNECT.

   See "Use SAS/CONNECT with TLS Enabled to Import Data" on page 44.

5 Configure and secure the connection from the SAS Viya environment to your LDAP Provider.

   See "Encrypt LDAP Connections" on page 37.

In addition to the tasks listed above, you can take additional steps to secure SAS Embedded Process and CAS Inter-node communication. You can enable and disable TLS based on port families using SAS Environment Manager.

- SAS Embedded Process is not secure by default. SAS Viya supports TLS encryption between the data provider (Hadoop, Teradata) and the CAS server, and you can take steps to enable that encryption. If you are using a SAS Data Connect Accelerator, the data that is transferred between the data provider and the CAS server is not encrypted by default. The configuration on the CAS side is complete by default.

  See "Encrypt Data Transfer When Using the SAS Data Connect Accelerator (Linux Full Deployment)" on page 50.

- SAS CAS inter-node communication is not secured by default. There is a large performance impact for enabling the CAS inter-node encryption. You can take additional steps to configure CAS inter-node encryption.

  See "Configure CAS Internode TLS (Linux Full Deployment)" on page 35.

- Enable and Disable TLS on a port family basis.

  See "Disable and Enable TLS (Linux Full Deployment)" on page 61.

Tasks to Harden Security for Your Linux Programming-Only Deployment

In a SAS Viya programming-only deployment, the basic framework for security is included by default, but is not enabled by default. In particular, the SAS Viya deployment provides the following default framework to secure data in motion.

1 Secure the Apache HTTP Server by adding certificates that conform to the policies at your enterprise and strengthen the default cryptography. On the Apache HTTP Server (reverse proxy server), the module called mod_ssl provides TLS support. This module relies on OpenSSL to provide the cryptography engine.

   See "Update Apache HTTP Server TLS Certificates and Cryptography " on page 5.

2 Enforce HTTPS for access to SAS Viya using options for Port 80 on page 7.

3 Enable TLS and configure CAS TLS to use your custom certificates. End users can access CAS directly from a third-party language like Python, Java, or Lua, or directly starting in SAS 9.4M5. For end-user access, SAS recommends that you use your own signed certificates for the entry point to CAS and update the private key and server certificate used by CAS.

   See "Configure CAS TLS to Use Custom Certificates (Linux Programming-Only Deployment)" on page 23.

4 Enable TLS for SAS/CONNECT.

   See "Use SAS/CONNECT with TLS Enabled to Import Data" on page 44.

5 Configure and secure the connection from the SAS Viya environment to your LDAP Provider.

See “Encrypt LDAP Connections” on page 37.

Tasks to Harden Security for Your Windows Deployment

SAS recommends that you enhance the default security that is applied by the deployment script. As a best practice, follow these steps as soon as the deployment process has completed:

1. Secure the Apache HTTP Server by adding certificates that conform to the policies at your enterprise.

   See “Update Apache HTTP Server TLS Certificates and Cryptography ” on page 5.

2. Enable TLS for the CAS server.

   See “Update Certificates and Configure TLS on CAS” on page 17.

3. If you are using SAS/CONNECT to access data from older versions of SAS 9.4, enable TLS for SAS/CONNECT.

   See “Use SAS/CONNECT with TLS Enabled to Import Data” on page 44.

4. Enforce HTTPS for access to SAS Viya using options for Port 80 on page 7.

5. Import the certificate file (sas.crt) and the private key file (sas.key) that are configured automatically by the deployment script into your Windows certificate stores.

   See “Import CA Certificates into the Windows Trusted Root Certificate Authorities Store” on page 72.

6. If you are using LDAP, encrypt the connections between LDAP servers and the SAS Viya deployment.

   See “Encrypt LDAP Connections” on page 37.

7. Prevent administrators from altering the default permissions on subdirectories of **`Program Files\SAS\Viya`** and **`ProgramData\SAS`**. Use your preferred network monitoring or security tool to monitor permissions on subdirectories of Program **`Files\SAS\Viya`** and **`ProgramData\SAS`** after the deployment has completed.

Configure and Update TLS and HTTPS

Update Apache HTTP Server TLS Certificates and Cryptography

Overview

The Apache HTTP Server (acting as a reverse proxy server) is the main entry point for end users in a SAS Viya deployment. Ensuring that TLS certificates are trusted by clients is critical to the SAS Viya deployment. The TLS trust affects browsers, mobile devices, and REST API clients.

SAS Viya uses an Apache HTTP Server to act as a reverse proxy server to secure your environment. You can have the SAS Viya deployment provide default-level security. The deployment enables TLS on connections to the Apache HTTP Server, installs Apache httpd, and provides a self-signed certificate for use across the deployment.

SAS highly recommends that you replace the default certificates with your own custom certificates that comply with the security policies at your enterprise. This task can be accomplished pre-deployment on Linux or post-deployment on Windows and Linux.

CAUTION! SAS Viya self-signed certificates prior to the July 2019 release of SAS Viya are valid for only one year. In the July 2019 release of SAS Viya, the self-signed certificates provided by SAS Viya have a seven-year expiration time. Prior to the July 2019 release of SAS Viya, the self-signed certificates expire in only one year. Perform

the task to updated SAS Viya default self-signed certificate to extend the expiration date, or contact SAS Technical Support for help.

If you have an Apache server that has httpd configured prior to the SAS Viya deployment, you need to provide the location on your system that contains the certificates and key files that will be used for TLS. The HTTPD_CERT_PATH variable in the vars.yml file is set with the location.

On Windows, the Apache HTTP Server is secured with a self-signed certificate (sas.crt) and a private key that the SAS Viya deployment process generates. SAS recommends that you enhance the security by replacing this certificate provided by SAS with a custom certificate that is generated according to the security standards at your enterprise. See "Replace Self-Signed Certificates with Custom Certificates (Post-deployment)" on page 10.

On a Linux deployment, the Ansible playbook can install Apache httpd and mod_ssl automatically. This option uses default Apache security settings. The installation of mod_ssl creates a private key and self-signed certificates. These settings are reasonably secure. However, SAS recommends that you replace the default certificates with your own custom certificates that comply with the security policies at your enterprise.

On a Linux full deployment, you can replace the certificates pre-deployment or post-deployment. During the deployment, SAS Viya determines whether the CA certificates installed on the Apache HTTP Server meet the SAS Security standards. For more details, see "How SAS Viya Determines If Certificates Meet the SAS Security Standards on an Installed Linux HTTP Server" on page 100.

Whether you replace the certificates on Linux pre-deployment or post-deployment on Linux or Windows, SAS recommends replacing the certificates before giving end users access to SAS Viya.

By default, HTTPS access to SAS Drive is enabled in a SAS Viya full deployment. The URL to access SAS Drive after installing Apache httpd and installing SAS Viya is https://*reverse-proxy-server*/SASDrive/.

Note: In a programming-only deployment, there is no SAS Drive. SAS Viya end users connect to SAS Studio and to CAS Server Monitor using Apache HTTP Server to support HTTPS.

- For SAS Studio 4.4: http://*hostname*/SASStudio
- For CAS Server Monitor: http://*reverse-proxy-server*/cas-shared-default-http/

The Apache HTTP Server is configured with the mod_ssl security module enabled. The mod_ssl module relies on OpenSSL to provide strong cryptography for the Apache server using TLS cryptographic protocols. SAS recommends strengthening the default cryptography using the sas-ssl.conf file on Linux or the httpd-ssl.conf file on Windows. See "Update the Default Ciphers and TLS Protocol on the Apache HTTP Server" on page 16. You can read more about mod_ssl at Apache TLS Encryption.

You can strengthen security on the Apache HTTP Server by performing the tasks in this section. These tasks can be performed at any time after your initial deployment. The task for replacing your certificates pre-deployment on Linux is the exception.

Apache httpd That Is Not Deployed by SAS

If you have an Apache server that has httpd configured prior to the SAS Viya deployment, you need to provide the location on your system that contains the certificates and key files that will be used for TLS. The HTTPD_CERT_PATH variable in the vars.yml file is set with the location.

To harden your deployment, you will also need to add the following directives to the httpd.conf file.

- On Red Hat Enterprise Linux and equivalent distributions, add the following directives to /etc/httpd/conf/httpd.conf:

```
UseCanonicalName On
ServerName http://hostname:port
```

- On SUSE Linux Enterprise Server, add the following directives to /etc/apache2/httpd.conf:

```
UseCanonicalName On
ServerName http://hostname:port
```

For more details, see UseCanonicalName and ServerName.

Options for Port 80

You do not have to force secure web access or to restrict unsecure web access to port 80 to use that port securely. Here are a few options to use port 80:

- If you want access to port 80, you do not have to block port 80 or redirect to port 443 (HTTPS). However, this is an unsecure port.
- If you want access to port 80 using HTTP, you can redirect port 80 to port 443 (HTTPS).
- The most secure method is to block port 80 internally and externally. However, if you block port 80 internally and externally, when you try to access port 80 using HTTP, you will get a `connection refused` error message in the browser.

On the Windows machine where the SAS Viya deployment is installed, SAS recommends that you block port 80 externally and leave port 443 (HTTPS) open externally to provide the most secure access to the SAS Viya software. After deployment, port 80 should be open internally. See the Windows Defender Firewall with Advanced Security documents for information about how to block and open ports in the Windows Defender Firewall.

On a Linux full deployment, in order to secure web access to your SAS Viya software, you can block port 80 internally and externally. Port 80 is not required internally to be open for microservices access. Port 443 (HTTPS) is used for external communications. For Linux, refer to the Red Hat Enterprise Linux Reference and Security Guides for information about best practices for securing ports.

For information about enabling ports on Linux and Windows, see "Enable Required Ports" in *SAS Viya for Linux: Deployment Guide* and "Enable Required Ports" in *SAS Viya for Windows: Deployment Guide*.

Secure Consul by Default (Linux Full Deployment)

In a SAS Viya 3.4 Linux full deployment, Consul is secure by default on port 8501. The HTTP port 8500 is disabled by default. Therefore, Consul communicates only over HTTPS (port 8501). The following settings are set in the vars.yml file by default:

```
SECURE_CONSUL: true

DISABLE_CONSUL_HTTP_PORT: true
```

Note: See "Modify the vars.yml File" in *SAS Viya for Linux: Deployment Guide* for more details.

Replace Self-Signed Certificates with Custom Certificates (Linux Pre-deployment)

Note: The Windows deployment of SAS Viya does not support this pre-deployment task.

The SAS Viya deployment can install Apache httpd with mod_ssl and self-signed certificates. These settings are reasonably secure, but they are not compliant with SAS security standards. SAS recommends replacing these self-signed certificates with custom certificates that comply with the security policies at your enterprise.

Note: SAS recommends that you install Apache httpd and replace the self-signed certificates before you start the deployment process. When you perform this task before installing SAS Viya, the Ansible playbook used to deploy SAS Viya distributes your custom certificates across the deployment and adds them to the truststore. This process avoids the brief outage necessary to replace the certificates after SAS Viya has been deployed.

For information about deploying Apache httpd and default deployment settings, see "Security Requirements" in *SAS Viya for Linux: Deployment Guide*

During the deployment, the playbook inspects existing certificates and the CA chain to determine whether they comply with SAS security requirements. See "How SAS Viya Determines If Certificates Meet the SAS Security Standards on an Installed Linux HTTP Server" on page 100.

If you do not add compliant certificates and instead keep the default security settings and certificates provided by Apache, end users will see a standard web browser warning message. SAS recommends replacing the

default certificates before giving end users access to SAS Viya. Adding your own certificates post-deployment requires a brief outage. See "Replace Self-Signed Certificates with Custom Certificates (Post-deployment)" on page 10.

To pre-configure Apache httpd, a user must first install httpd and enable the TLS packages (mod_ssl, a2enmodssl) on the desired machines. Afterward, configure httpd to use the custom certificates. Lastly, update vars.yml and run the full Ansible playbook as in a regular deployment.

1 Install httpd and enable TLS on the desired machines.

   Note: Even though you are advised to follow the instructions in the Ansible documentation, streamlined instructions are provided here as a convenience. Before performing these instructions, ensure that they are appropriate for your site and that they comply with the IT policies in your organization.

   a On Red Hat Enterprise Linux and equivalent distributions, enter the following command to install the httpd service and enable TLS with mod_ssl.

      Note: These steps assume that you have sudo access to the machine where you are installing Ansible.

      ```
      sudo yum install -y httpd mod_ssl
      ```

   b On SUSE Linux Enterprise Server, enter the following commands. In this code, `apache2` is the package for installing httpd, and the `a2enmod ssl` command enables TLS.

      ```
      zypper update
      zypper install apache2
      a2enmod ssl
      ```

      For more information about the zypper commands, see Update from the Command Line with zypper.

   c On SUSE Linux Enterprise Server, edit the apache2 file at **`/etc/sysconfig/apache2`**. Add "SSL" as a value to the APACHE_SERVER_FLAGS line.

      ```
      APACHE_SERVER_FLAGS='SSL'
      ```

   For more information, see "Install Ansible" in *SAS Viya for Linux: Deployment Guide*.

2 When generating new certificates, provide the following information for the certificate signing request.

   - Provide fixed host names (required by the SAS Viya environment).
   - Provide fully qualified domain names (FQDN).
   - Provide subject alternative names (SAN), including IP addresses.
   - For multi-tenancy, ensure that the certificates contain subject alternate names for each tenant or use a wildcard for the subdomain. For more information about multi-tenant DNS naming, see "Additional Requirements for Multi-tenancy" in *SAS Viya for Linux: Deployment Guide*. For an example of using OpenSSL to generate a new certificate where wildcards are specified for multi-tenancy, see the certificate signing request (CSR) conf file at "Create Certificates with SANs Using OpenSSL" on page 76.
   - On Linux, the default server identity certificate is named localhost.crt. It is recommended that when using your own certificates you name your certificate and key files something other than localhost. In this example, we are using file names customer.crt and customer.key.

   You can use OpenSSL to create your new private key or to generate a new certificate signing request (CSR) with a new private key as follows:

   ```
   openssl genrsa-out /etc/pki/tls/private/web_server_key.pem 2048
   ```

   ```
   openssl req -new -key /etc/pki/tls/private/web_server_key.pem
   -out /etc/pki/tls/certs/web_server_csr.pem
   ```

3 Create your certificate chain of trust file(s). SAS recommends two ways to create a certificate chain of trust for Apache httpd on page 101. How you create these files determines how the SSLCertificateFile and the

SSLCertificateChainFile directives are set in the configuration file. It is important to understand these recommendations and the problems that can occur if there is not a complete chain of trust.

4 Copy your certificate chain file to the following directory:

- On Red Hat Enterprise Linux and equivalent distributions, place the server certificate in **/etc/pki/tls/certs**.
- On SUSE Linux Enterprise Server, place your server certificate in **/etc/apache2/ssl.crt**.

Note: The certificate file needs to be a Base64 PEM encoded file.

5 Copy your key file to the following locations:

Note: The key file needs to be a Base64 PEM encoded file.

- On Red Hat Enterprise Linux and equivalent distributions, copy the key file to **/etc/pki/tls/private**.
- On SUSE Linux Enterprise Server, place your key file in **/etc/apache2/ssl.key**.

6 Change the permissions on your certificate file and your chain file to 644. Change permissions on the key file to 600. Use *chmod* or *sudo* commands to change the permissions.

Note: In the following example, we are using the name *customer* for the customer-provided key and certificate files. Note that you will need a certificate chain file if you are a customer providing your own certificates. If you are using the self-signed certificates provided by SAS Viya, this file contains only one certificate file.

```
chmod 600 customer.key
chmod 644 customer-chain.crt
chmod 644 customer.crt
```

When you list the files, you see the permissions are Read/Write only for the root account: -rw-r--r-- for the certificate files and -rw------- for the key file.

7 Update the Apache server certificate and key file directives. Set the directives appropriate for your version of Apache HTTP Server.

Note: In Apache HTTP Server version 2.4.8, the SSLCertificateFile directive was extended to load intermediate CA certificates from the server certificate file. This change enables you to use the SSLCertificateFile directive for chained certificates instead of the SSLCertificateChainFile directive. See an explanation at SSLCertificateChainFile Directive.

Note: These directives must be specified on one line, without line breaks.

- On Red Hat Enterprise Linux and equivalent distributions, update the ssl.conf file in **/etc/httpd/conf.d** to point to your certificates and key.

  ```
  SSLCertificateFile /etc/pki/tls/certs/customer-chain.crt
  SSLCertificateKeyFile /etc/pki/tls/private/customer.key
  ```

  If your Apache HTTP Server version is earlier than 2.4.8, use the SSLCertificateChainFile directive.

  ```
  SSLCertificateChainFile /etc/pki/tls/certs/customer-chain.crt
  ```

- On SUSE Linux Enterprise Server, update the ssl-global.conf file in **/etc/apache2** to point to your new certificates and key.

  ```
  SSLCertificateFile /etc/apache2/ssl.crt/customer-chain.crt
  SSLCertificateKeyFile /etc/apache2/ssl.key/customer.key
  ```

  If your Apache HTTP Server version is earlier than 2.4.8, use the SSLCertificateChainFile directive.

  ```
  SSLCertificateChainFile /etc/apache2/ssl.crt/customer-chain.crt
  ```

  Also update the vhost-ssl.conf file in **/etc/apache2/vhosts.d/**. Update the server name and new certificates and key.

```
SSLCertificateFile /etc/apache2/ssl.crt/customer-chain.crt
SSLCertificateKeyFile /etc/apache2/ssl.key/customer.key
```

If your Apache HTTP Server version is earlier than 2.4.8, use the SSLCertificateChainFile directive.

```
SSLCertificateChainFile /etc/apache2/ssl.crt/customer-chain.crt
```

8 On a Linux deployment, update the value of HTTPD_CERT_PATH in vars.yml file.

Note: Review the information in "Create a Certificate Chain of Trust for Apache httpd" on page 101 to ensure that the certificate file that you have provided contains a complete chain of trusted certificates.

a If the certificate file contains the full chain of trusted CA certificates (root and all intermediate CA certificates) and the server identity certificate, set the HTTPD_CERT_PATH value as follows:

```
HTTPD_CERT_PATH:
```

b Otherwise, set the HTTPD_CERT_PATH to point to the file that contains the CA certificate chain of trust (the root CA Certificate and all intermediate CA certificates) as follows:

- On Red Hat Enterprise Linux and equivalent distributions, add the certificate that you used in the previous step to the vars.yml file.

```
HTTPD_CERT_PATH: '/etc/pki/tls/certs/customer-chain.crt'
```

- On SUSE Linux Enterprise Server, add the certificate that you used in the previous step to the vars.yml file.

```
HTTPD_CERT_PATH: '/etc/apache2/ssl.crt/customer-chain.crt'
```

9 On a Linux deployment, run the Ansible playbook to install the SAS Viya deployment. During deployment, the following occurs:

- New custom certificates are distributed to all hosts in your SAS Viya environment. The certificates are placed in **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts/** as file httpproxy-*host definition*-ca.crt. The *host definition* value is taken from the inventory.ini.
- New custom certificates are added to the truststores.
- The Apache HTTP Server certificate is persisted in the SAS Configuration Server, either by Ansible or by the httpproxy start script, to ensure that the Apache HTTP Server certificate is added to all truststores.

  Note: Because the Apache HTTP Server certificate is persisted in the SAS Configuration Server, it must be removed when the certificate is changed.

See "Installation" in *SAS Viya for Linux: Deployment Guide* for details.

Replace Self-Signed Certificates with Custom Certificates (Post-deployment)

The SAS Viya deployment on Linux and Windows can install Apache httpd with mod_ssl and Apache self-signed certificates. These settings are reasonably secure, but they are not compliant with SAS security standards. SAS recommends replacing the self-signed certificates with custom certificates that comply with the security policies at your enterprise. SAS also recommends that you upgrade the security protocol and ciphers on the Apache HTTP Server. For more information, see "Update the Default Ciphers and TLS Protocol on the Apache HTTP Server" on page 16.

Note: On Linux, SAS recommends that you install Apache httpd and replace the self-signed certificates before you start the deployment process. When you perform this task before installing SAS Viya, the Ansible playbook used to deploy SAS Viya distributes your custom certificates and adds them to the truststore. This process avoids the brief outage necessary to replace the certificates after SAS Viya has been deployed. See "Replace Self-Signed Certificates with Custom Certificates (Linux Pre-deployment)" on page 7.

On a Linux deployment, the playbook inspects certificates and the CA chain on the installed Apache HTTP Server to determine whether the certificates comply with SAS security requirements. If you do not add compliant certificates and instead keep the default security settings and certificates provided by Apache, end users will see

a standard web browser warning message. SAS recommends replacing the default certificates before giving end users access to SAS Viya. See "How SAS Viya Determines If Certificates Meet the SAS Security Standards on an Installed Linux HTTP Server" on page 100.

On Windows, SAS installs the Apache HTTP Server and provides a SAS self-signed certificate and key file at deployment. SAS recommends replacing the SAS self-signed certificates before giving end users access to SAS Viya.

Configure httpd to use your custom certificates.

Note: On Linux, update vars.yml, and run the full Ansible playbook as in a regular deployment.

1. Log on to the Apache HTTP Server.
   - Log on to the Linux Apache HTTP Server as a user with root or sudo privileges.
   - Open a Windows PowerShell prompt as an Administrator on the Windows Apache HTTP Server.
2. When generating new certificates, see "Use Best Practices to Create and Manage Certificates" on page 74.
3. Download your server identity certificate files.
4. Copy your new server certificate file to the following directory:
   - On Red Hat Enterprise Linux and equivalent distributions, place the server certificate in **`/etc/pki/tls/certs`**.
   - On SUSE Linux Enterprise Server, place your server certificate in **`/etc/apache2/ssl.crt`**.
   - On Windows, place your server certificate in **`C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\tls\certs`**

   If you are also downloading the root and intermediate certificates, you need to copy the chain file that includes the root and the intermediate certificates to this location.

   Note: The certificate file needs to be a Base64 PEM encoded file.
5. Copy your new key file to the following locations.

   Note: The key file needs to be a Base64 PEM encoded file.
   - On Red Hat Enterprise Linux and equivalent distributions, copy the key file to **`/etc/pki/tls/private`**.
   - On SUSE Linux Enterprise Server, copy the key file to **`/etc/apache2/ssl.key`**.
   - On Windows, place the key file in **`C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\private\`**.
6. On Linux, change the permissions on your certificate file and your chain file to 644. Change the permissions on the key file to 600. Use *chmod* or *sudo* commands to change the permissions.

   Note: In the following example, we are using the name *customer* for our newly created key and certificate files.

   ```
   chmod 600 customer.key
   chmod 644 customer.crt
   chmod 644 customer-chain.crt
   ```

   When you list the files, you see the permissions are Read/Write only for the root account: -rw-r--r-- for the certificate files and -rw------- for the key file.
7. Update the certificate and key file directives.

   Note: These directives must be specified on one line, without line breaks.
   - On Red Hat Enterprise Linux and equivalent distributions, update the ssl.conf file in **`/etc/httpd/conf.d`** to point to your new certificates and key.

```
SSLCertificateFile /etc/pki/tls/certs/customer.crt
SSLCertificateKeyFile /etc/pki/tls/private/customer.key
```

If you are using a chained certificate, specify the SSLCertificateChainFile directive instead of the SSLCertificateFile directive.

```
SSLCertificateChainFile /etc/pki/tls/certs/customer-chain.crt
```

- On SUSE Linux Enterprise Server, update the ssl-global.conf file in **/etc/apache2** to point to your new certificates and key.

```
SSLCertificateFile /etc/apache2/ssl.crt/customer.crt
SSLCertificateKeyFile /etc/apache2/ssl.key/customer.key
```

Also update the vhost-ssl.conf file in **/etc/apache2/vhosts.d/**. Update the server name and new certificates and key.

```
SSLCertificateFile /etc/apache2/ssl.crt/customer.crt
SSLCertificateKeyFile /etc/apache2/ssl.key/customer.key
```

If you are using a chained certificate, specify the SSLCertificateChainFile directive instead of the SSLCertificateFile directive.

```
SSLCertificateChainFile /etc/apache2/ssl.crt/customer-chain.crt
```

- On SUSE Linux Enterprise Server, also update the vhost-ssl.conf file in **/etc/apache2/vhosts.d/**. Update the server name and new certificates and key.

```
SSLCertificateFile /etc/apache2/ssl.crt/customer.crt
SSLCertificateKeyFile /etc/apache2/ssl.key/customer.key
```

If you are using a chained certificate, specify the SSLCertificateChainFile directive instead of the SSLCertificateFile directive.

```
SSLCertificateChainFile /etc/apache2/ssl.crt/customer-chain.crt
```

- On SUSE Linux Enterprise Server, also edit the apache2 file at **/etc/sysconfig**. Add "SSL" as a value to the APACHE_SERVER_FLAGS=.

```
APACHE_SERVER_FLAGS='SSL'
```

- On Windows, update the httpd-ssl.conf file in **C:\ProgramData\SAS\Viya\etc\httpd\conf\extra** to point to your new certificates and key.

```
SSLCertificateFile C:\ProgramData\SAS\Viya\etc\
   SASSecurityCertificateFramework\tls\certs\customer.crt
SSLCertificateKeyFile C:\ProgramData\SAS\Viya\etc\
   SASSecurityCertificateFramework\private\customer.key
```

If you are using a chained certificate, specify the SSLCertificateChainFile directive instead of the SSLCertificateFile directive.

```
SSLCertificateChainFile C:\ProgramData\SAS\Viya\etc\
   SASSecurityCertificateFramework\tls\certs\customer-chain.crt
```

8 Restart the httpproxy service. Restart of sas-httpproxy updates the certificates stored in Consul. If a certificate has been removed from the chain, it is removed from Consul automatically when restarting the SAS Viya httpproxy service.

How you run the following command depends on your operating system.

Note: On a Linux multiple-machine deployment, there is an important sequence to follow for starting and stopping SAS Viya servers and services. See "General Servers and Services: Operate (Linux)" in *SAS Viya Administration: General Servers and Services*.

- Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

```
sudo systemctl restart sas-viya-httpproxy-default
```

- Red Hat Enterprise Linux 6.x (or an equivalent distribution):

  ```
  sudo service sas-viya-httpproxy-default restart
  ```

- On Windows, restart the SAS HTTP Proxy Server using the Services snap-in in the Microsoft Management Console. See "Start and Stop a Specific Server or Service" in *SAS Viya Administration: General Servers and Services*.

9 Restart Apache HTTP Server.

```
sudo apachectl restart
```

10 On a Linux deployment, update the value of HTTPD_CERT_PATH in the vars.yml file to point to the new custom CA certificate chain.

Note: Review the information in "Create a Certificate Chain of Trust for Apache httpd" on page 101 to ensure that the certificate file that you have provided contains a complete chain of trusted certificates.

Set the HTTPD_CERT_PATH to point to the file that contains the CA certificate chain of trust (the root CA certificate and all intermediate CA certificates) as follows.

Note: If you have more than one CAS controller, you must update this information on each of your CAS controllers.

- On Red Hat Enterprise Linux and equivalent distributions, add the certificate that you used in the previous step to the vars.yml file.

  ```
  HTTPD_CERT_PATH: '/etc/pki/tls/certs/customer-chain.crt'
  ```

- On SUSE Linux Enterprise Server, add the certificate that you used in the previous step to the vars.yml file.

  ```
  HTTPD_CERT_PATH: '/etc/apache2/ssl.crt/customer-chain.crt'
  ```

11 Add the certificates to the truststores.

- In a Linux full deployment, you can run the distribute-httpd-certs.yml Ansible play to distribute the certificate to the CA Certificate directory and rebuild the truststores. On the Ansible controller machine and for every inventory file that you maintain, run the distribute-httpd-certs.yml play, which is located in the **/viya/sas_viya_playbook** directory.

  ```
  ansible-playbook -i inventory.ini utility/distribute-httpd-certs.yml
  ```

  To distribute the certificates to the second CAS controller, run the distribute-httpd-certs.yml play for the added CAS controller. In this example, the second inventory file is named inventory_addcas.ini.

  ```
  ansible-playbook -i inventory_addcas.ini utility/distribute-httpd-certs.yml -e
  "@vars_addcas.yml"
  ```

  This play adds your new custom certificate to **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts**. The play distributes copies of the certificate file to all machines with a name of httpproxy-**inventory name**-ca.crt. The play then rebuilds the trustedcerts.pem and trustedcerts.jks files and includes the CA certificates from customer.crt in the trustedcerts.pem and trustedcerts.jks file on every machine in the deployment.

- In a Linux programming-only deployment, update the SAS Truststore manually on page 68.

- On Windows, add the CA root certificate and intermediate certificates to the SAS Viya truststores (trustedcerts.pem and trustedcerts.jks). To add the certificates to the truststore, see "Add Certificates to the Trustedcerts Files" on page 68.

  Add the CA root certificate and all of the intermediate certificates to the Windows certificates stores. Import the CA certificates into the Windows Trusted Root Certification Authorities local machine store. See "Import CA Certificates into the Windows Trusted Root Certificate Authorities Store" on page 72.

12 Restart all services on all machines.

Note: On a Linux multiple-machine deployment, there is a sequence for starting and stopping SAS Viya servers and services. See "General Servers and Services: Operate (Linux)" in *SAS Viya Administration: General Servers and Services*.

- On Linux, stop and then start all servers using the following commands:

```
sudo /etc/init.d/sas-viya-all-services stop
sudo /etc/init.d/sas-viya-all-services start
```

- On Windows, stop and then start all services (SAS Services Manager) using the Microsoft Management Console (MMC) Services snap-in. See "Start and Stop All Servers and Services" in *SAS Viya Administration: General Servers and Services*.

Update SAS Viya Default Self-Signed Certificate to Extend the Expiration Date

SAS highly recommends that you replace the SAS Viya provided default certificates with your own custom certificates that comply with the security policies at your enterprise either pre-deployment or post deployment. A best practice is to update this certificate immediately after deploying SAS Viya. See "Replace Self-Signed Certificates with Custom Certificates (Linux Pre-deployment)" on page 7 and "Replace Self-Signed Certificates with Custom Certificates (Post-deployment)" on page 10.

If you are upgrading your deployment from an earlier version of SAS Viya, you might need to extend the expiration date on the self-signed certificate that is provided by SAS. Perform the following tasks to renew this certificate before it expires.

CAUTION! SAS Viya self-signed certificates prior to the July 2019 release of SAS Viya are valid for only one year. In the July 2019 release of SAS Viya, the self-signed certificates provided by SAS Viya have a seven-year expiration time. Prior to the July 2019 release of SAS Viya, the self-signed certificates expire in only one year. Use the following instructions or contact SAS Technical Support to renew the self-signed certificates before they expire.

1. You can check the expiration date of the certificate using the following OpenSSL command on Linux:

```
openssl x509 -noout -in /etc/pki/tls/certs/localhost.crt -enddate
```

   You can check the expiration date of the certificate using the following OpenSSL command on Windows:

```
"C:\Program Files\SAS\Viya\httpd\bin\openssl" x509 -noout -in
C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\tls\certs\
localhost.crt -enddate
```

2. Modify replace_httpd_default_cert.sh located in **/opt/sas/viya/home/bin/**. In the code, locate the comment, # Self-sign CSR and add the number of days (in the code, this is -days *number of days*) that you want this self-signed certificate to be valid.

```
# Self-sign CSR
${SASCRYPTOHOME}/sas-crypto-management \
   selfsign \
   -days 256 \
   -csr ${DEFAULTCSRPATH}/localhost.csr \
   -out-file  ${DEFAULTCERTPATH}/localhost.crt \
   -out-form  pem \
   -signing-key  ${DEFAULTKEYPATH}/localhost.key
```

   On Linux, execute the following command with the force option:

```
/opt/sas/viya/home/bin/replace_httpd_default_cert.sh --force
```

3. The updated certificate replaces the existing certificate named localhost. This certificate is as follows:
   - On Red Hat Enterprise Linux and equivalent distributions, the localhost certificate is in **/etc/pki/tls/certs/localhost.crt**.
   - On SUSE Linux Enterprise Server, the localhost certificate is in **/etc/apache2/ssl.crt/locahost.crt**.

- On Windows, the localhost certificate is in `C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\tls\certs\localhost.crt`.

4 Restart the sas-viya-httpproxy service. In a SAS Viya full deployment, a restart of sas-viya-httpproxy also updates the certificates stored in Consul. If a certificate has been removed from the certificate chain, it is removed from Consul automatically when restarting the sas-viya-httpproxy service.

How you run the following command depends on your operating system as follows.

Note: On a Linux multiple-machine deployment, there is an important sequence to follow for starting and stopping SAS Viya servers and services. See "General Servers and Services: Operate (Linux)" in *SAS Viya Administration: General Servers and Services*.

- Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x;

```
sudo systemctl restart sas-viya-httpproxy-default
```

- Red Hat Enterprise Linux 6.x (or an equivalent distribution):

```
sudo service sas-viya-httpproxy-default restart
```

- On Windows, restart the SAS HTTP Proxy Server using the Services snap-in in the Microsoft Management Console. See "Start and Stop a Specific Server or Service" in *SAS Viya Administration: General Servers and Services*.

5 Restart httpd as follows:

```
sudo apachectl restart
```

6 Add the certificates to the truststores based on your deployment type.

- In a Linux full deployment, you can run the distribute-httpd-certs.yml Ansible play to distribute the certificate to the CA Certificate directory and rebuild the truststores. On the Ansible controller machine and for every inventory file that you maintain, run the distribute-httpd-certs.yml play located in the **`/viya/sas_viya_playbook`** directory.

```
ansible-playbook -i inventory.ini utility/distribute-httpd-certs.yml
```

To distribute the certificates to the second CAS controller, run the distribute-httpd-certs.yml play for the added CAS controller. In this example, the second inventory file is named inventory_addcas.ini.

```
ansible-playbook -i inventory_addcas.ini utility/distribute-httpd-certs.yml -e
"@vars_addcas.yml"
```

This play adds your new custom certificate to **`/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts`**. The play distributes copies of the certificate file to all machines with a name of httpproxy-**`inventory name`**-ca.crt. The play then rebuilds the trustedcerts.pem and trustedcerts.jks files and includes the CA certificates from customer.crt in the trustedcerts.pem and trustedcerts.jks file on every machine in the deployment.

- In a Linux programming-only deployment, update the SAS truststore manually on page 68.

- On Windows, add the CA root certificate and intermediate certificates to the SAS Viya truststores (trustedcerts.pem and trustedcerts.jks). To add the certificates to the truststore, see "Add Certificates to the Trustedcerts Files" on page 68.

Add the CA root certificate and all of the intermediate certificates to the Windows certificates stores. Import the CA certificates into the Windows Trusted Root Certification Authorities local machine store. See "Import CA Certificates into the Windows Trusted Root Certificate Authorities Store" on page 72.

7 Restart all services on all machines.

Note: On a Linux multiple-machine deployment, there is a sequence for starting and stopping SAS Viya servers and services. See "General Servers and Services: Operate (Linux)" in *SAS Viya Administration: General Servers and Services*.

- On Linux, stop and then start all servers using the following commands:

```
sudo /etc/init.d/sas-viya-all-services stop
sudo /etc/init.d/sas-viya-all-services start
```

- On Windows, stop and then start all services (SAS Services Manager) using the Microsoft Management Console (MMC) Services snap-in. See "Start and Stop All Servers and Services" in *SAS Viya Administration: General Servers and Services*.

8 Remove the expired locahost.crt certificate from the truststores. See "Add Certificates to or Remove Certificates from the Truststore or a Certificate Chain Manually" on page 68.

- For a Linux full deployment, see "Remove Certificates from the Truststores Using Ansible (Linux Full Deployment)" on page 66.
- On Windows or on a Linux programming-only deployment, see "Add Certificates to or Remove Certificates from the Truststore or a Certificate Chain Manually" on page 68.

9 On any client that connects to the deployment and performs a TLS handshake, if the expired self-signed certificate was imported into the client truststore, remove it and import the new one. For Windows, see "Import CA Certificates into the Windows Trusted Root Certificate Authorities Store" on page 72,

Update the Default Ciphers and TLS Protocol on the Apache HTTP Server

In the SAS Viya deployment, the Apache HTTP Server is configured with the mod_ssl security module enabled. The mod_ssl module provides strong cryptography for the Apache server using SSL and TLS cryptographic protocols. You can read more about what mod_ssl does at Apache SSL/TLS Encryption.

SAS recommends that you update your Apache HTTP Server to not only use your own custom certificates, but to also upgrade the security protocol and ciphers being used by default with the ones contained in the sas-ssl.conf file on Linux or the httpd-ssl.conf file on Windows. The TLS protocols and ciphers are recommended by SAS to meet the highest data-in-motion standard for cryptography.

1 Locate the sas-ssl.conf file and the ssl.conf on Linux.

Note: On Windows, the same changes as shown below for Linux are already populated to the Apache HTTP Server. Therefore, there is nothing else that needs to be done. Windows customers can still look at applying their own configuration in the httpd.conf and the httpd-ssl.conf files.

- On RHEL and equivalent distributions of Linux, the ssl.conf file is located here.

  `/etc/httpd/conf.d/`

  The sas-ssl.conf file is located here.

  `/opt/sas/viya/config/etc/httpd/conf.d/`

- On SUSE Linux Enterprise Server 12.x, ssl.conf and sas-ssl.conf are located here.

  `/etc/apache2/conf.d`

- On Windows, the httpd-ssl.conf file is located here.

  `C:\ProgramData\SAS\Viya\etc\httpd\conf\extra\`

  On Windows, the httpd.conf file is located here.

  `C:\ProgramData\SAS\Viya\etc\httpd\conf`

2 On Linux, edit the sas-ssl.conf file. If you do not find the sas-ssl.conf file, create your own sas-ssl.conf file using the following example code.

Note: On Windows, the same changes as shown below for Linux are already populated to the Apache HTTP Server.

Note: The following code is shown on more than one line for display purposes only. The SSLCipherSuite variable plus the ciphers must be on one line and must not contain line breaks.

```
Header set Strict-Transport-Security 'max-age=31536000'
SSLProtocol TLSv1.2
SSLHonorCipherOrder On
# The line containing variable SSLCipherSuite and values
# must not include line breaks
SSLCipherSuite
ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:
ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:
ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA256:
ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA256:
AES256-GCM-SHA384:AES128-GCM-SHA256:AES256-SHA256:
AES128-SHA256
```

3 On Linux, edit the ssl.conf file to include the sas-ssl.conf file. In the ssl.conf file,locate the `<VirtualHost_default_:443>` block of code. Just before the `</VirtualHost>` line, add the following line of code:

- On RHEL and equivalent distributions, edit the ssl.conf file and include the location of the sas-ssl.conf file.

  ```
  Include /opt/sas/viya/config/etc/httpd/conf.d/sas-ssl.conf
  ```

- On SUSE Linux Enterprise Server 12.x, edit the ssl.conf file and include the location of the sas-ssl.conf file.

  ```
  Include /etc/apache2/conf.d/sas-ssl.conf
  ```

On Windows, in the httpd.conf, file, you should see code that includes httpd-ssl.conf. In the httpd-ssl.conf file, near the end of the file, locate comment "#Secure (SSL/TLS) connections" . Ensure that the following Include statement exists and is uncommented.

```
Include C:\ProgramData\SAS\Viya\etc\httpd\conf\extra\httpd-ssl.conf
```

4 Restart the HTTPD service on Linux.

Note: On a Linux multiple-machine deployment, there is an important sequence to follow for starting and stopping SAS Viya servers and services. See “General Servers and Services: Operate (Linux)” in *SAS Viya Administration: General Servers and Services*.

- On Linux deployments:

  ```
  sudo apachectl restart
  ```

- On Windows, restart the SAS HTTP Proxy Server using the Services snap-in in the Microsoft Management Console. See “Start and Stop a Specific Server or Service” in *SAS Viya Administration: General Servers and Services*.

Update Certificates and Configure TLS on CAS

Configure CAS TLS to Use Custom Certificates (Linux Full Deployment)

Note: The following instructions are for adding custom certificates to a SAS Viya full deployment.

By default, in a full deployment of SAS Viya, SAS Secrets Manager issues certificates and keys that are used to secure the deployment. These certificates issued by SAS Secrets Manager are provided for each CAS machine and are added to the Mozilla bundle of trusted CA certificates by default.

Table 1 *Security Certificates and Keys Provided for CAS in a SAS Viya Full Deployment*

Security Artifact	Default Certificate and Key Files	Location	Permissions/User Access	Description
Trusted CA certificates	trustedcerts.pem trustedcerts.jks	**/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts**	This file should be world readable (for example, 644 or -rw-r--r--.)	CA certificates issued by SAS Secrets Manager. The trusted list of CA certificates includes the Mozilla bundle of trusted CA certificates, the root CA certificates issued by SAS Secrets Manager, the Apache httpd certificates, and the chain of trust certificates.
Certificate file	sas_encrypted.crt	/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tls/certs/cas/shared/default For multi-tenancy, the file is located in **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tls/certs/cas/*<tenant-id>*/default**	This file should be world readable (for example, 644 or -rw-r--r--.) The file should be owned by the CAS service account. In multi-tenant environments this is the tenant admin user.	Certificates issued by SAS Secrets Manager. This file contains the CAS server certificate.
Private key file	sas_encrypted.key	**/opt/sas/viya/config/etc/SASSecurityCertificateFramework/private/cas/shared/default** For multi-tenancy, the file is located in **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/private/cas/*<tenant-id>*/default**	This file should be world readable (for example, 644 or -rw-r--r--.) The file should be owned by the CAS service account. In multi-tenant environments this is the tenant admin user.	Encrypted key file issued by SAS Secrets Manager. This key is the CAS server private key.

Security Artifact	Default Certificate and Key Files	Location	Permissions/User Access	Description
Certificate private key passphrase file	encryption.key (optional)	**/opt/sas/viya/config/etc/SASSecurityCertificateFramework/private/cas/shared/default** For multi-tenancy, the file is located in **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/private/cas/*<tenant-id>*/default**	This file has permission 600 -rw-----. This file needs to be readable by the CAS service account user. In multi-tenant environments, this is the tenant admin user. Only service account users should be allowed to read the passphrase.	The CAS server private key password file. If you are replacing the certificate and key files provided by SAS Viya with your own custom files, it is highly recommended that you encrypt your key file and provide a passphrase to protect the file that contains the key.

You can use your own custom certificates instead of the certificates provided by SAS. Best practices for managing certificates and securing your private keys should be followed. See "Use Best Practices to Create and Manage Certificates" on page 74.

The following instructions are provided to configure TLS for the CAS client with your own custom certificates. In a full deployment of SAS Viya, because the SAS Configuration Server (Consul) handles most configuration tasks, you need to configure Consul as well as the CAS controllers to enable CAS client TLS.

1 Perform the following steps on Consul to configure CAS client TLS and add your own custom certificates.

   a On the main Consul machine (the machine listed within the [consul] host group in the inventory.ini file), edit the sitedefault.yml file located at **/opt/sas/viya/config/etc/consul.d/default/** and place the certificate strings for your custom CA certificates (in PEM format) in the file.

   Note: If you do not have a sitedefault.yml file at this location, see the instructions for creating one at "Operations" in *SAS Viya Administration: Configuration Properties*.

   ```
   sudo vi /opt/sas/viya/config/etc/consul.d/default/sitedefault.yml
   ```

   Here is an example of what the sitedefault.yml file might look like after you add your custom CA bundle to this file. First add the identifier (node) named `cacerts`. Beneath that node, add nodes that identify each of the CA root certificates that are being used. In this example, certificate identifiers named `sascaroot` and `sassh2rootca` and their certificates were already added to this file. Using an editor, we added another node to identify the customer certificates (`customer_ca_chain`) that we added.

   Note: Keep the indention of this file.

   ```
   cacerts:
       sascaroot: |
           -----BEGIN CERTIFICATE-----
           certificate string
           -----END CERTIFICATE-------
       sassha2rootca: |
           -----BEGIN CERTIFICATE-----
            certificate string
           -----END CERTIFICATE-------
       customer_ca_chain: |
           -----BEGIN CERTIFICATE-----
           certificate string
   ```

```
        -----END CERTIFICATE-------
        -----BEGIN CERTIFICATE-----
        certificate string
        -----END CERTIFICATE-------
```

b On the main Consul machine, restart Consul to pick up the changes to the sitedefault.yml file. Only new key-value pairs are added to the Consul key-value (KV) store. Existing key-value pairs are not updated. How you run the following command depends on your operating system.

- Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

```
sudo systemctl restart sas-viya-consul-default
```

- Red Hat Enterprise Linux 6.x (or an equivalent distribution):

```
sudo service sas-viya-consul-default restart
```

c Use the Ansible playbook to rebuild the truststores across all the hosts in the SAS Viya environment. On an Ansible controller machine, run the **./utility/rebuild-trust-stores.yml** Ansible play.

```
ansible-playbook -i inventory.ini ./utility/rebuild-trust-stores.yml
```

Note: Use the same admin user that you used during the initial SAS Viya deployment.

For more information, see "User and Group Requirements" in *SAS Viya for Linux: Deployment Guide*.

d Add the sitedefault.yml content from step 1 to the sitedefault.yml that forms part of the Ansible playbooks. This sitedefault.yml file is located in **/roles/consul/files/**.

2 To configure TLS between the CAS client and CAS controllers, perform the following steps on the primary and secondary CAS controllers. If you are also using the same custom certificates on the worker nodes, perform the following steps to add the certificates, encrypted key files, and the passphrase-protected key file to the worker nodes.

Note: Do not name your custom certificates and key files the same names as the default certificate and key files (sas_encrypted.crt, sas_encrypted.key, encryption.key). The default certificates and keys are renewed every time the primary controller is restarted. Therefore, the custom certificate and key files are overwritten if stored using the same name as the default.

a Log on to the CAS controller machine as a user with root or sudo privileges.

b If you have a CAS session running, stop the CAS server.

- Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

```
sudo systemctl status stop sas-viya-cascontroller-default
```

- Red Hat Enterprise Linux 6.x (or an equivalent distribution):

```
sudo service sas-viya-cascontroller-default stop
```

c Place your custom certificate in **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tls/certs/cas/shared/default**. The certificate file provided by SAS Viya is named sas_encrypted.crt. Do not overwrite this file. Add your certificate to the directory with a unique name. In our example, we named the file customer.crt.

Note: Intermediate certificates need to be added to the server identity certificate in a certificate chain. The file needs to include the server identity certificate first, and then the signing intermediate CA certificates in the order in which they were signed. The root CA does not need to be included in this chain file.

Note: Ensure that your certificate files have file system permissions 644: -rw-r--r--. Also, ensure that the file has appropriate file system ownership and permissions for the cas service account. For more information, see "User and Group Requirements" in *SAS Viya for Linux: Deployment Guide*.

d Protect your certificate private key file with a passphrase. In this example, the key file is named customer.key. We use OpenSSL to encrypt the customer.key file and name the encrypted version customer_encrypted.key. The default key file provided by SAS is named encryption.key. Do not overwrite the encryption.key file.

Place your encrypted private key file in **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/private/cas/shared/default**. For multi-tenancy, see Table 1 on page 18.

i Use the following OpenSSL command to password-protect the file named customer.key file.

```
openssl rsa -aes128 -in /opt/sas/viya/config/etc/
SASSecurityCertificateFramework/private/cas/shared/default/customer.key
-out /opt/sas/viya/config/etc/SASSecurityCertificateFramework/
private/cas/shared/default/customer_encrypted.key -passout pass:password
```

ii Ensure that your files have file system permissions 644: -rw-r--r--. Also, ensure that the file has appropriate file system ownership and permissions for CAS service account. Use chmod to change the permissions:

Note: For multi-tenancy, the file should be owned by the tenant admin user, not the CAS service account.

```
chmod 644 customer_encrypted.key
```

e Create a customer-supplied certificate private key passphrase file. Place the private key passphrase file in **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/private/cas/shared/default**. For multi-tenancy, see Table 1 on page 18.

Use the `echo` command to create the private key passphrase file. In this example, the private key passphrase filename is customer_encrypted.encryption.key. The user needs to be the cas service account. See "User and Group Requirements" in *SAS Viya for Linux: Deployment Guide*.

```
sudo bash -c 'echo -n 'password' > customer_encrypted.encryption.key'
sudo chown cas:sas customer_encrypted.encryption.key
sudo chmod 0600 customer_encrypted.encryption.key
```

f You can remove the original customer.key file. You now have an encrypted key file (customer_encrypted.key) and passphrase-protected key file (customer_encrypted.encryption.key).

g Configure CAS to use the customer-supplied certificates and key.

i On every CAS controller (the primary controller and secondary controller, as well as CAS worker nodes if you have CAS Internode TLS enabled) in your deployment, edit the node_usermods.lua file. The node_usermods.lua file is located by default in **/opt/sas/viya/config/etc/cas/default**. The node_usermods.lua file is located by default in **/opt/sas/viya/config/etc/cas/default.** For multi-tenancy, the node_usermods.lua file is located by default in **/opt/sas/<*tenant-id*>/config/etc/cas/default**.

Note: Configuration changes that should apply only to specific CAS nodes must be set in node_usermods.lua on that host. For information about when to use the various configuration files, see "Configuration File Options" in *SAS Viya Administration: SAS Cloud Analytic Services*.

ii Change the required CAS_CLIENT_SSL environment variables. Specify the names of your custom certificate (customer.crt), the custom encrypted certificate private key file (customer_encrypted.key), and the customer-supplied certificate private key passphrase file (customer_encrypted.encryption.key). As a best practice, use the same names on the primary and secondary controllers for the certificate and key files.

For multi-tenancy, see Table 1 on page 18.

Note: Ensure that the permissions on the node_usermods.lua file are readable only by the CAS account (-r--------).

```
env.CAS_CLIENT_SSL_REQUIRED=true
env.CAS_CLIENT_SSL_CERT='/opt/sas/viya/config/etc/
SASSecurityCertificateFramework/tls/certs/cas/shared/default/customer.crt'
env.CAS_CLIENT_SSL_KEY='/opt/sas/viya/config/etc/
SASSecurityCertificateFramework/private/cas/shared/default/customer_encrypted.key'
env.CAS_CLIENT_SSL_KEYPWLOC = '/opt/sas/viya/config/etc/
SASSecurityCertificateFramework/private/cas/shared/default/
customer_encrypted.encryption.key'
```

When setting the CAS client environment variables, consider the following information.

Note: See "Modify the vars.yml File" in *SAS Viya for Linux: Deployment Guide* for more details.

- If you are using an intermediate CA certificate, then a certificate chain file needs to be specified for the CAS_CLIENT_SSL_CERT= environment variable. The file needs to include the server identity certificate first, and then the signing intermediate CA certificates in the order in which they were signed. The root CA does not need to be included in this chain file.
- If you are using your own custom certificate and key, you should copy the changes made to CAS_CLIENT_SSL_CERT= and CAS_CLIENT_SSL_KEY= environment variables to the vars.yml file. This change ensures that your settings are not changed when upgrades are made to the deployment.
- If you are setting the CAS_CLIENT_SSL_REQUIRED= environment variable to **true**, you should copy the change made to this environment variable to the vars.yml file. This change ensures that your settings are not changed when upgrades are made to the deployment.

h Start the cascontroller service on the primary controller. How you run the following command depends on your operating system.

For multi-tenant environments, use the following commands.

- Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

```
sudo systemctl start sas-<tenant-id>-cascontroller-default
```

- Red Hat Enterprise Linux 6.x (or an equivalent distribution):

```
sudo service sas-<tenant-id>-cascontroller-default start
```

Otherwise, use the following commands.

- Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

```
sudo systemctl start sas-viya-cascontroller-default
```

- Red Hat Enterprise Linux 6.x (or an equivalent distribution):

```
sudo service sas-viya-cascontroller-default start
```

3 Add the root certificate and any intermediate certificates that might have been used to sign the identity/server certificate to the truststore on the client. This is necessary because each CAS client must be able to trust the server certificate. In SAS Viya deployments, the file that contains these certificates is the trustedcerts.pem file located at **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts/trustedcerts**.

For the various clients, the following are considerations for clients connecting to TLS-enabled CAS:

- In a SAS Viya full deployment, the CA certificate files that CAS is using can be found in the vault-ca.crt file or the trustedcerts.pem file, located at **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts**. The vault-ca.crt file contains two certificates issued by SAS Secrets Manager: the SAS Viya root CA certificate and the SAS Viya intermediate CA certificate. The trustedcerts files contain the trusted CA certificates and the intermediate certificates.
- If your Python, SWAT, and Lua clients are Linux clients, you need to export the CAS_CLIENT_SSL_CA_LIST environment variable and point to a PEM file that contains the root CA

certificate for the deployment. This file can be the vault-ca.crt or the trustedcerts.pem file provided by the SAS Viya deployment. In the following example, we are using the vault-ca.crt file:

```
export CAS_CLIENT_SSL_CA_LIST=
'/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts/vault-ca.crt'
```

- On Linux, if the root CA is already in the OpenSSL trusted certificate store, most clients should work without having to set the CAS_CLIENT_SSL_CA_LIST= environment variable.
- On Linux, if the root CA is not in the OpenSSL trusted certificate store, set the CAS_CLIENT_SSL_CA_LIST= environment variable to point to the location of your certificate chain. Root CA certificates at a minimum are needed in the certificate chain. You can use the vault-ca.crt or the trustedcerts.pem files. For example, use the following command:

  ```
  export CAS_CLIENT_SSL_CA_LIST='/opt/sas/viya/config/etc/
  SASSecurityCertificateFramework/cacerts/trustedcerts.pem'
  ```

- The SAS Workspace Server and the SAS Compute Server are configured by default to use the trusted CA certificates that SAS Viya provides in the **`SASASSecurityCertificateFramework`** directory.

  ```
  export CAS_CLIENT_SSL_CA_LIST=
  '/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts/trustedcerts.pem'
  ```

- For SAS client-side connections on Linux, SAS should automatically find the trustedcerts.pem file that is located in **`/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts`** either through the SAS Workspace Server or SAS Compute Server EXPORT statement or the SSLCALISTLOC= system option that is set during installation.
- On a Windows client, before you can import the certificates from the vault-ca.crt file or the trustedcerts.pem files, you must create files that contain only one CA certificate each. The Windows certificate store allows only one certificate at a time to be imported. Because the vault-ca.crt file contains only two CA files, it is easier to split out the two CA certificates into their own files.

  Use a text editor to cut and paste each certificate into its own unique CA certificate file, one file that contains the SAS Viya root CA certificate, and another file that contains the intermediate CA certificate. Each certificate in the vault-ca.crt file is denoted with a `-----BEGIN CERTIFICATE-----` and an `-----END CERTIFICATE-----` pair. Include the `-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----` header and footer in each of the two new files.

  Save these two files on your Windows machine and then add those certificate files to the Windows CA store. You need to import the root certificate first and then the intermediate certificate. See “Import CA Certificates into the Windows Trusted Root Certificate Authorities Store” on page 72.
- For SAS 9.4 client-side connections, see “Configure SAS 9.4 Clients to Work with SAS Viya” on page 59.

Configure CAS TLS to Use Custom Certificates (Linux Programming-Only Deployment)

Note: The following instructions are for adding custom certificates and configuring CAS to use these in a SAS Viya programming-only deployment.

CAS supports encrypted connections between the server and clients. Use TLS to secure communications between the server and clients. The certificates used for client/server communication need to be signed by a certificate authority (CA) that is trusted by all potential clients.

The SAS Viya programming-only deployment provides self-signed certificates and keys at installation that can be used to configure and secure the deployment. These SAS Viya self-signed certificates are provided for each CAS machine in the deployment and are added to the truststore by default in a SAS Viya programming-only deployment. These SAS Viya default certificates are listed in Table 3 on page 29.

You can use the self-signed certificates provided by the SAS Viya deployment as described in “Configure CAS TLS to Use SAS Viya Default Certificates (Linux Programming-Only Deployment)” on page 28. However, SAS recommends that you provide your own custom certificates and configure the SAS Viya deployment to use them.

Here is an overview of the types of custom certificates that you will need to provide to configure CAS on a Linux programming-only deployment.

1 To configure SAS Cloud Analytic Services (CAS) to use custom certificates, you will need server and intermediate CA signed certificates (to create the chain of trust) and key files for the CAS controller. The following certificate and key files are needed.

   - A private key protected with a password in Base-64 PEM encoded format. Because the private key file needs to be accessible to anyone running a CAS session, the key file should be protected with a password.
   - A signed server certificate in Base-64 PEM encoded format. If the Certificate Authority that signed the server certificate is an intermediary, the CA signed intermediate certificate should be added to the server certificate to form a certificate chain. These are added to the same certificate file.

     Note: This certificate file should have the file extension of .crt.
   - A private key password file. This password file contains the password of the private key in plain text. Therefore, the password file needs to have access controls that allow only the CAS Service account to access the file.

2 You will need the root CA certificate for this Linux programming-only deployment. The root CA certificate will be added to the following:

   - The root CA certificate needs to be added to the SAS Viya truststores. This action ensures that all components in the SAS Viya environment can trust the server certificate presented by SAS Cloud Analytic Services.
   - The root CA certificate needs to be provided to CAS clients. Clients include SAS 9.4M5 clients that can directly access CAS, SWAT (Java, R, Lua), Python, and REST clients.

     Note: In the certificate, the host name used when connecting to CAS must match the host name in the certificate.

Table 2 *Custom Security Certificates and Keys Needed for CAS in SAS Viya Programming-Only Deployment*

Security Artifact	Customer-Provided Security files	Location	Permissions/ User Access	Description
Trusted CA certificates	trustedcerts.pem trustedcerts.jks	`/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts`	This file should be world readable (for example, 644 or -rw-r--r--).	Contains the trusted list of CA certificates. The trusted list of CA certificates includes the Mozilla bundle of trusted CA certificates, the root CA certificates issued by SAS Viya, the Apache httpd certificates, and the chain of trust certificates. Add customer-provided CA signed certificates to trustedcerts files. **Important:** Do not remove these files. Add CA certificates to the trustedcerts files.

Security Artifact	Customer-Provided Security files	Location	Permissions/ User Access	Description
Certificate or certificate chain file	Customer-provided server certificate file or chain certificate file (for example, *customer_encrypted*.crt).	**/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tls/certs**	This file should be world readable (for example, 644 or -rw-r--r--).	Add the customer provided server certificate to this directory. Intermediate certificates need to be added to the server certificate in a certificate chain file. The file needs to include the server certificate first, and then the intermediate CA certificates used to sign the server certificates. These certificates need to be added in the order in which they were signed. The root CA does not need to be included in this chain file.
Certificate private key file	Customer-provided key file (for example, *customer_encrypted*.key).	**/opt/sas/viya/config/etc/SASSecurityCertificateFramework/private**	This file should be world readable (for example, 644 or -rw-r--r--).	Add the customer provided encrypted key file to this directory.
Certificate private key passphrase file	Customer-provided private key passphrase file (for example, *customer_encryption*.key). (Optional)	**/opt/sas/viya/config/etc/SASSecurityCertificateFramework/private**	This file needs to have permission 640 -rw-r-----prior to starting the CAS server using TLS. This file needs to be readable by the CAS service account user (cas:sas).	When you provide customer certificate and key files, it is highly recommended that you encrypt your key file and provide a passphrase to protect the file that contains the key.

Note: Ensure that the files have appropriate file system ownership and permissions for CAS ADMIN user. For more information, see "User and Group Requirements" in *SAS Viya for Linux: Deployment Guide*.

To configure TLS between the CAS client and server using your custom certificates, perform the following steps on the primary and secondary CAS controllers.

Note: Do not name your custom certificates and key files the same names as the default certificate and key files (sas_encrypted.crt, sas_encrypted.key, encryption.key).

1. Log on to the CAS controller machine as a user with root or sudo privileges.
2. If you have a CAS session running, stop the CAS server.
   - Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

     ```
     sudo systemctl status stop sas-viya-cascontroller-default
     ```

   - Red Hat Enterprise Linux 6.x (or an equivalent distribution):

     ```
     sudo service sas-viya-cascontroller-default stop
     ```

3. On the CAS Controller, place your custom certificate (signed server certificate) in **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tls/certs**. The certificate file provided by SAS Viya is named sas_encrypted.crt. Do not overwrite this file. Add your certificate to the directory with a unique name. In our example, we named the file customer.crt.

Note: Intermediate certificates need to be added to the server certificate file to create a certificate chain. The file needs to include the server identity certificate first, then the signing intermediate CA certificates in the order in which they were signed. The root CA does not need to be included in this chain file.

4 Protect your certificate private key file with a passphrase. In this example, the key file is named customer.key. We use OpenSSL to encrypt the customer.key file and name the encrypted version customer_encrypted.key. The default key file provided by SAS Viya is named sas_encrypted.key. Do not overwrite the sas_encrypted.key file.

Place your encrypted private key file in **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/private**.

a Use the following OpenSSL command to password-protect the file named customer.key file.

```
openssl rsa -aes128 -in /opt/sas/viya/config/etc/
SASSecurityCertificateFramework/private/customer.key
-out /opt/sas/viya/config/etc/SASSecurityCertificateFramework/
private/customer_encrypted.key -passout pass:password
```

b Ensure that your encrypted key file has file system permissions 644: -rw-r--r--. Also, ensure that the file has appropriate file system ownership and permissions for the CAS ADMIN user. Use chmod to change the permissions:

```
chmod 644 customer_encrypted.key
```

5 Create a customer-supplied certificate private key passphrase file. Place the private key passphrase file in **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/private**.

Use the `echo` command to create the private key passphrase file. In this example, the private key passphrase filename is customer_encrypted.encryption.key. The user needs to be the cas service account. See "User and Group Requirements" in *SAS Viya for Linux: Deployment Guide*.

```
sudo bash -c 'echo -n 'password' > customer_encrypted.encryption.key'
sudo chown cas:sas customer_encrypted.encryption.key
sudo chmod 0640 customer_encrypted.encryption.key
sudo cat customer_encrypted.encryption.key;
echo password
```

6 You can remove the original customer.key file. You now have an encrypted key file (customer_encrypted.key) and passphrase-protected key file (customer_encrypted.encryption.key).

7 Update the truststore.

- If your CA certificate is not already included in the Mozilla bundle of CA certificates, append the root certificate to the trustedcerts files in the **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts/** directory.

  Note: Add your root certificate to the trustedcerts.pem and trustedcerts.jks files on every machine in the deployment.

  - To add the root certificate to the trustedcerts.pem file, just include the root certificate at the end of the trustedcerts.pem file.
  - To add the root certificate to the trustedcerts.jks file, you need to import the file using a keytool command.

  See "Add Certificates to the Trustedcerts Files" on page 68 for information about adding your certificates to the truststore.

  Note: Do not delete the trustedcerts files.

8 Configure CAS to use the customer-supplied certificates and key by editing the casconfig_usermods.lua file or the node_usermods.lua file. Configuration changes that should apply only to specific CAS nodes must be

set in node_usermods.lua on that host. For CAS REST port (8777), edit the casconfig_usermods.lua file. For information about when to use the various configuration files, see "Configuration File Options" in *SAS Viya Administration: SAS Cloud Analytic Services*.

The node_usermods.lua file is located by default in **/opt/sas/viya/config/etc/cas/default**.

Change the required CAS_CLIENT_SSL environment variables. Specify the names of your custom certificate (customer.crt), the custom encrypted certificate private key file (customer_encrypted.key), and the customer-supplied certificate private key passphrase file (customer_encrypted.encryption.key). As a best practice, use the same names on the primary and secondary controllers for the certificate and key files.

The following environment variables are set for the CAS binary port 5570 in the node_usermods.lua file. Add this information to every CAS controller (primary controller and secondary controller, as well as CAS worker nodes if you have CAS Internode TLS enabled) in your deployment.

```
env.CAS_CLIENT_SSL_REQUIRED=true
env.CAS_CLIENT_SSL_CERT=
'/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tls/certs/customer.crt'
env.CAS_CLIENT_SSL_KEY=
'/opt/sas/viya/config/etc/SASSecurityCertificateFramework/private/customer_encrypted.key'
env.CAS_CLIENT_SSL_KEYPWLOC=
'/opt/sas/viya/config/etc/SASSecurityCertificateFramework/private/customer_encrypted.encryption.key'
```

The following environment variables are set for the CAS REST port 8777. Set the following environment variables to look for the certificates in the same location on every node. Add this information to every CAS controller (primary controller and secondary controller, as well as CAS worker nodes if you have CAS Internode TLS enabled) in your deployment.

```
env.CAS_USE_HTTPS_ALL = 'true'
env.CAS_CERTLOC=
'/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tls/certs/customer.crt'
env.CAS_PVTKEYLOC=
'/opt/sas/viya/config/etc/SASSecurityCertificateFramework/private/customer_encrypted.key'
env.CAS_PVTKEYPASSLOC=
'/opt/sas/viya/config/etc/SASSecurityCertificateFramework/private/customer_encrypted.encryption.key'
```

When setting the CAS client environment variables, consider the following information.

Note: See "Modify the vars.yml File" in *SAS Viya for Linux: Deployment Guide* for more details.

- If you are using an intermediate CA certificate, a certificate chain file needs to be specified for the CAS_CLIENT_SSL_CERT= environment variable. The file needs to include the server identity certificate first, and then the signing intermediate CA certificates in the order in which they were signed. The root CA does not need to be included in this chain file.
- If you are using your own custom certificate and key, you should copy the changes made to CAS_CLIENT_SSL_CERT= and CAS_CLIENT_SSL_KEY= environment variables to the vars.yml file. This change ensures that your settings are not changed when upgrades are made to the deployment.
- If you are setting the CAS_CLIENT_SSL_REQUIRED= environment variable to **true**, you should copy the change made to this environment variable to the vars.yml file. This change ensures that your settings are not changed when upgrades are made to the deployment.

9 Restart the cascontroller service on the primary controller. How you run the following command depends on your operating system.

- Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

  ```
  sudo systemctl restart sas-viya-cascontroller-default
  ```

- Red Hat Enterprise Linux 6.x (or an equivalent distribution):

  ```
  sudo service sas-viya-cascontroller-default restart
  ```

10 Each CAS client must be able to trust the server certificate. Therefore, the client needs to add the root certificate and any intermediate certificates that might have been used to sign the identity/server certificate to the truststore on the client. In SAS Viya deployments, the file that contains these certificates is the trustedcerts.pem file located at **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts/trustedcerts**.

For the various clients, you also need to set the following environment variables:

- On Linux, the SAS Workspace Server and the SAS Compute Server are configured by default to use the trusted CA certificates that SAS Viya provides in the **SASASSecurityCertificateFramework** directory. In the sasenv_local file, the following environment variable is set:

  ```
  export CAS_CLIENT_SSL_CA_LIST=
  '/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts/trustedcerts.pem'
  ```

- On Linux, open programming clients using the SWAT package (Python, Lua, R) require that the CAS_CLIENT_SSL_CA_LIST= environment variable be set. This environment variable points to a Base-64 PEM encoded text file containing the root CA certificate. Specify a path to the certificate that is a local path. The certificate is copied from the server.

  Note: If the root CA is already in the trusted certificate store, open programming clients using the SWAT package (Python, Lua, R) should work without having to set the CAS_CLIENT_SSL_CA_LIST= environment variable.

- Otherwise, on Linux, set the CAS_CLIENT_SSL_CA_LIST= environment variable to point to the location of your certificate chain. Root CA certificates at a minimum are needed in the certificate chain.

  ```
  export CAS_CLIENT_SSL_CA_LIST="/opt/sas/viya/config/etc/
  SASSecurityCertificateFramework/cacerts/trustedcerts.pem"
  ```

- On Windows clients connecting to a Linux programming-only deployment, you need the root CA certificate (at minimum) from the SAS Viya deployment. From the SAS Viya deployment, copy the signing certificate to the Windows client machine and import that certificate into the Windows Certificate store. See Adding certificates to the Trusted Root Certification Authorities store for a local computer.

- For SAS 9.4 client-side connections, see “Configure SAS 9.4 Clients to Work with SAS Viya” on page 59.

Configure CAS TLS to Use SAS Viya Default Certificates (Linux Programming-Only Deployment)

Use TLS to secure communications between the CAS server and clients on Linux. The certificate used for client and server communication needs to be signed by a certificate authority (CA) that is trusted by all potential clients.

At installation, SAS Viya provides self-signed certificates that can be used to secure the deployment. You can use these certificates and activate TLS security. However, SAS recommends that you provide your own custom certificates to configure CAS client TLS. See “Configure CAS TLS to Use Custom Certificates (Linux Programming-Only Deployment)” on page 23.

Here are the certificates that are added to the CAS machines in a SAS Viya programming-only deployment. Before completing the task of turning on TLS using the following security artifacts, ensure that files sas_encrypted.crt, sas_encrypted.key, and encryption.key have the proper permissions and user access shown in Table 3 on page 29.

Table 3 *Security Certificates and Keys for CAS in a Programming-Only SAS Viya Deployment*

Security Artifact	Deployment File Name	Location	Permissions/User Access	Description
Certificate truststore	trustedcerts.pem trustedcerts.jks	`/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts/`	This file should be world readable (for example, 644 or -rw-r--r--).	Contains the trusted list of CA certificates. These include the Mozilla bundle of trusted CA certificates, the SAS Viya self-signed certificates, the Apache httpd certificates, and the chain of trust certificates.
Certificate file	sas_encrypted.crt	`/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tls/certs/`	This file should be world readable (for example, 644 or -rw-r--r--).	Certificates issued by SAS Viya. Contains the SAS Viya self-signed certificates.
Certificate private key file	sas_encrypted.key	`/opt/sas/viya/config/etc/SASSecurityCertificateFramework/private`	This file should be world readable (for example, 644 or -rw-r--r--).	The key file issued by SAS Viya. Contains the private key generated by SAS Viya.
Certificate private key passphrase file	encryption.key (optional)	`/opt/sas/viya/config/etc/SASSecurityCertificateFramework/private`	This file needs to have permission 640 -rw-r----- set prior to starting the CAS server using TLS. This file needs to be readable by the CAS service account user (cas:sas) or the SAS account can have group Read permissions (sas:sas). Only service account users should be allowed to read the passphrase.	Contains the encrypted passphrase file provided by SAS Viya.

Note: Ensure that the files have appropriate file system ownership and permissions for the CAS ADMIN user. For more information, see "User and Group Requirements" in *SAS Viya for Linux: Deployment Guide*.

Perform the following tasks to use the SAS Viya self-signed certificates that are provided at installation and enable TLS between the CAS servers and CAS client.

1 Log on to the CAS controller machine as a user with root or sudo privileges.

2 If you have a CAS session running, stop the CAS server.

- Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

  ```
  sudo systemctl status stop sas-viya-cascontroller-default
  ```

- Red Hat Enterprise Linux 6.x (or an equivalent distribution):

```
sudo service sas-viya-cascontroller-default stop
```

3 Add certificates to the truststore on page 68.

4 Configure CAS to use the certificates and key by editing the casconfig_usermods.lua file. Configuration changes that should apply only to specific CAS nodes must be set in node_usermods.lua on that host. For information about when to use the various configuration files, see "Configuration File Options" in *SAS Viya Administration: SAS Cloud Analytic Services*.

On every CAS controller (the primary controller and secondary controller, as well as CAS worker nodes if you have CAS Internode TLS enabled) in your deployment, edit the node_usermods.lua file. The node_usermods.lua file is located by default in **/opt/sas/viya/config/etc/cas/default**.

Turn on TLS for the CAS client on port 5570. Set the CAS_CLIENT_SSL_REQUIRED= environment variable to **true**. The other environment variables are already set to point to the self-signed certificates and keys that were provided at installation.

```
env.CAS_CLIENT_SSL_REQUIRED  = true
env.CAS_CLIENT_SSL_CERT = '/opt/sas/viya/config/etc/
SASSecurityCertificateFramework/tls/certs/sas_encrypted.crt'
env.CAS_CLIENT_SSL_KEY = '/opt/sas/viya/config/etc/
SASSecurityCertificateFramework/private/sas_encrypted.key'
env.CAS_CLIENT_SSL_KEYPWLOC = '/opt/sas/viya/config/etc/
SASSecurityCertificateFramework/private/encryption.key'
```

Note: By default, SAS Viya self-signed certificates are generated using the fully qualified domain name for the Common Name. Make sure that the CAS host name in programs submitted to CAS match the Common Name used in the SAS Viya self-signed certificates.

Note: If you are setting the CAS_CLIENT_SSL_REQUIRED= environment variable to **true**, you should copy the change made to this environment variable to the vars.yml file. This change ensures that your settings are not changed when upgrades are made to the deployment. See "Modify the vars.yml File" in *SAS Viya for Linux: Deployment Guide* for more details.

5 Set the permissions and user access for the encryption.key file before starting the CAS server that is using TLS. The encryption.key file needs to have permission 640 -rw-r-----and be readable by the CAS service account user (cas:sas). The SAS account with group Read permissions (sas:sas) can also be used. Only service account users should be allowed to read the passphrase.

Ensure that files sas_encrypted.crt, sas_encrypted.key, and encryption.key have the proper permissions and user access shown in Table 3 on page 29.

6 Restart the cascontroller service on each controller and node. How you run the following command depends on your operating system.

- Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

```
sudo systemctl restart sas-viya-cascontroller-default
```

- Red Hat Enterprise Linux 6.x (or an equivalent distribution):

```
sudo service sas-viya-cascontroller-default restart
```

7 Each CAS client must be able to trust the server certificate. Therefore, the client needs to add the root certificate and any intermediate certificates that might have been used to sign the identity/server certificate to the truststore on the client. In SAS Viya deployments, the file that contains these certificates is the trustedcerts.pem file located at **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts/trustedcerts**.

Other considerations for configuring CAS to use the self-signed certificates provided by SAS Viya are as follows:

- On Linux, open programming clients using the SWAT package (Python, Lua, R) require that the CAS_CLIENT_SSL_CA_LIST= environment variable be set. This environment variable points to a

Base-64 PEM encoded text file containing the root CA certificate. Specify a path to the certificate that is a local path. The certificate is copied from the server.

Note: If the root CA is already in the trusted certificate store, open programming clients using the SWAT package (Python, Lua, R) should work without having to set the CAS_CLIENT_SSL_CA_LIST= environment variable.

- In SAS Viya, the workspace server exports the trustedcerts.pem file by default.
- Otherwise, on Linux, set the CAS_CLIENT_SSL_CA_LIST= environment variable to point to the location of your certificate chain. Root CA certificates at a minimum are needed in the certificate chain.

```
export CAS_CLIENT_SSL_CA_LIST='/opt/sas/viya/config/etc/
SASSecurityCertificateFramework/cacerts/trustedcerts.pem'
```

Note: For SAS client-side connections, SAS Viya should automatically find the trustedcerts.pem file that is located in **`/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts`** through the workspace server EXPORT statement.

- On Windows clients connecting to a Linux programming-only deployment, you need the root CA certificate (at minimum) from the SAS Viya deployment. From the SAS Viya deployment, copy the signing certificate to the Windows client machine and import that certificate into the Windows Certificate store. See Adding certificates to the Trusted Root Certification Authorities store for a local computer.
- For SAS 9.4 client-side connections, see “Configure SAS 9.4 Clients to Work with SAS Viya” on page 59.

Configure CAS TLS to Use SAS Viya Default Certificates (Windows)

Use TLS to secure communications between the CAS server and clients on Windows. The certificate used for client and server communication needs to be signed by a certificate authority (CA) that is trusted by all potential clients.

At installation, SAS Viya provides self-signed certificates that can be used to secure the deployment. You can use these certificates and activate TLS security. However, SAS recommends that you provide your own custom certificates to configure CAS client TLS. See “Configure CAS TLS to Use Custom Certificates (Windows)” on page 32.

Here are the certificates that are added to the CAS machines in a SAS Viya Windows deployment.

Table 4 *Security Artifacts Provided at Installation for SAS Viya Windows Deployment*

Security Artifact	Deployment File Name	Location	Description
Certificate truststore	trustedcerts.pem trustedcerts.jks	**`C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\cacerts\`**	Contains the trusted list of CA certificates. These include the Mozilla bundle of trusted CA certificates, the SAS Viya self-signed certificates, the Apache httpd certificates, and the chain of trust certificates.
Certificate file	sas_encrypted.crt	**`C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\tls\certs`**	Contains the certificate generated by SAS Viya. These are self-signed certificates.

Security Artifact	Deployment File Name	Location	Description
Certificate private key file	sas_encrypted.key	**C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\private**	Contains the private key generated by SAS.
Certificate private key passphrase file	encryption.key (optional)	**C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\private**	Contains the encrypted passphrase file provided by SAS Viya.

To use the SAS Viya self-signed certificates that are provided at installation, perform the following tasks:

1. Open a Windows PowerShell prompt as an Administrator.
2. Change to directory **C:\Program Files\SAS\Viya\SASFoundation\utilities\bin**.
3. Run the Enable-CAS-TLS.ps1 PowerShell script. This script reads the certificate serial number, the certificate thumbprint, and the certificate issuer of the sas_encrypted.crt certificate provided by SAS. It then creates a temporary .pfx file and imports that file into the Windows Certificate store on the Local machine.

   From the PowerShell prompt, enter the following command:

   ```
   .\Enable-CAS-TLS.ps1
   ```

   Observe the output of the PowerShell script. Confirm that no errors are printed.

   This script also turns on TLS for the CAS client on port 5570, setting the CAS_CLIENT_SSL_REQUIRED= environment variable to **true** in the casconfig.lua file.

   Note: By default, SAS Viya self-signed certificates are generated using the fully qualified domain name for the Common Name. Make sure that the CAS host name in programs submitted to CAS match the Common Name used in the SAS Viya self-signed certificates.
4. Restart CAS using the Services snap-in in the Microsoft Management Console. Restart SAS Cloud Analytic Services. See "Operate (Windows)" in *SAS Viya Administration: SAS Cloud Analytic Services*.
5. Copy the trustedcerts.pem and trustedcerts.jks from your Windows deployment to a client machine. You need these certificates on your client machine to create a chain of trust between client and the SAS Viya deployment on Windows.

   Client considerations when configuring CAS to use the self-signed certificates provided in SAS Viya deployment on Windows are as follows:

   - For client-side connections from Linux (Lua, Python), you need the root CA certificate from the SAS Viya Windows deployment on the Linux client.
   - For client-side connections from a Windows client, you need the root CA certificate from the SAS Viya Windows deployment on the Windows client. From the SAS Viya Windows deployment, copy the SAS Viya provided root CA certificate (sas_encrypted.crt) to your client. Then import your CA certificate into your Windows certificate store. See Import your CA certificate into the Windows Trusted Root Certification Authorities store on page 72.

Configure CAS TLS to Use Custom Certificates (Windows)

CAS supports TLS encrypted connections between the server and the clients. The certificate used for client server communication needs to be signed by a certificate authority (CA) that is trusted by all potential clients.

The SAS Viya deployment provides certificates and keys at installation that secure the deployment. SAS Viya also adds self-signed certificates created for each CAS machine in the deployment to the Mozilla bundle of trusted certificates.

You can use the default self-signed certificates provided by the SAS Viya deployment (shown in Table 4 on page 31). However, SAS recommends that you provide and configure your own custom certificates

The following instructions are provided to configure TLS for CAS in order for a client to access CAS directly using their own custom certificates. These instructions show how to configure your Windows deployment to use your own custom certificates instead of the certificates provided by SAS Viya.

Your custom certificates need to be in a PFX file that contains the following:

- A private key is embedded in the PFX file.
- The private key within the PFX file is protected with a password.
- The PFX file contains all certificates in the certification path (the PFX file contains the certificates that make up the CA chain).
- Your certificates need to follow best practices for creating and managing certificates on page 74.

You can generate your own custom certificates using OpenSSL and Keytool. See "Manage Certificates and Generate New Certificates" on page 74 for some basic instructions.

To configure TLS between the CAS client and SAS Viya server, perform the following steps.

1. Log on to the SAS Viya Windows machine as an administrator.
2. On the SAS Viya Windows desktop, search for the *Command Prompt* app.
   a. In the Search Windows box, type *Command Prompt* .
   b. From the list of apps displayed, Right-click on **Command Prompt Desktop App**.

   Select **Run as administrator**.
3. Set the path where the OpenSSL executable lives. If you are using an OpenSSL.conf file, set the path to that file.

```
C:\>set PATH=%PATH%;
'C:\Program Files\SAS\Viya\SASSecurityCertificateFramework\bin'
C:\>set OPENSSL_CONF=C:\Program Files\SAS\Viya\httpd\conf\openssl.cnf
```

4. Verify that OpenSSL can inspect your custom certificate that is in PFX format by using the following command. This is the file that you will import into the Windows truststore. If you password-protected this certificate file, you will be asked to provide the password.

```
C:\>openssl.exe pkcs12 -info -in customerCert.pfx
```

   Note: You set the PATH variable in an earlier step to the path where the OpenSSL executable resides. This certificate file contains the custom certificate and the private key.

5. Extract the CA certificates from the PFX file and create a temporary PEM file. This is the file that you will import into the Windows truststore. If you password-protected this certificate file, you will be asked to provide the password.

```
C:\>openssl.exe pkcs12 -in customerCert.pfx -out customerCA.pem -cacerts -nokeys
```

   Note: The certificate file provided by SAS Viya is named sas_encrypted.crt. Do not overwrite this file. Add your certificate to the directory with a unique name. In our example, we named the file customerCA.crt.

6. Extract the client certificate from the PFX file and create a temporary PEM file. If you password-protected this certificate file, you will be asked to provide the password.

```
C:\>openssl.exe pkcs12 -in customerCert.pfx -out customerClient.pem -clcerts -nokeys
```

7 Add the CA root certificate (customerCa.pem) and intermediate certificates (customerClient.pem) to the truststores (trustedcerts.pem and trustedcerts.jks). To add the certificates to the truststore, see see "Add Certificates to the Trustedcerts Files" on page 68.

8 Verify that your certificates are valid.

9 Import the client certificate into the Windows Personal local machine store. See "Import the Client Certificate into the Windows Personal Machine Store" on page 73.

10 Grant Read permission to authenticated users who will need Read access to the client certificate's private key. see "Grant Read Permission to Authenticated Users for the Client Certificate's Private Key" on page 74.

11 Import the CA certificates into the Windows Trusted Root Certification Authorities local machine store. See "Import CA Certificates into the Windows Trusted Root Certificate Authorities Store" on page 72.

12 You need to extract information from the client certificate that can be used for environment variables specified in the casconfig_usermods.lua file. From the client certificate, you will need the serial number and the issuer of the certificate.

From the command prompt where you are running as an administrator, run the following commands and note the output.

Note: For information about running as an administrator, see Step 2 on page 33.

a Print the serial number of the client certificate.

```
C:\>openssl.exe x509 -in customerClient.pem -serial -noout
```

Note the value of serial= in the output. Save that numerical value for use with environment variable CAS_CLIENT_SSL_CERTSERIAL.

b Print the issuer of the client certificate.

```
C:\>openssl.exe x509 -in customerClient.pem -issuer -noout
```

Note the issuer of the certificate. For example, if `issuer= /DC=com/DC=Company/CN=Company SHA2 Issuing CA02`, you will need the value of `CN=`, which is `Company SHA2 Issuing CA02`. Save that value for use with environment variable CAS_CLIENT_SSL_CERTISS.

13 Edit casconfig_usermods.lua. This file is found in **`C:\ProgramData\SAS\Viya\etc\cas\default`**

a Use a text editor to edit casconfig_usermods.lua.

b Add the following environment variables and their values to the end of the file.

Note: The values for CAS_CLIENT_SSL_CERTSERIAL and CAS_CLIENT_SSL_CERTISS were retrieved in were retrieved in Step 12 on page 34.

```
env.CAS_CLIENT_SSL_REQUIRED=true
env.CAS_CLIENT_SSL_CERTSERIAL='190000AB8122B4DEC1D0AD1A7800000000AB57'
env.CAS_CLIENT_SSL_CERTISS='Company SHA2 Issuing CA02'
```

14 You can now remove the temporary files that you created named customerCA.pem and customerClient.pem.

15 Restart all of the SAS Viya services using the Services snap-in in the Microsoft Management Console. Wait for all of the services to stop before starting all of the services again. See "Start and Stop All Servers and Services" in *SAS Viya Administration: General Servers and Services*.

16 Programming clients (Python, Lua, Java, SAS) that connect directly to CAS on TCP port 5570 must now trust the CA that was used to issue the client certificate that you just configured CAS to use.

Client considerations when configuring CAS to use the customer provided certificates in a SAS Viya deployment on Windows are as follows:

- For client-side connections from Linux, you need the root CA certificate from the SAS Viya Windows deployment on the Linux client.
- For client-side connections from a Windows Client, you need the root CA certificate from the SAS Viya Windows deployment on the Windows client. From the SAS Viya Windows deployment, copy the signing certificate to the Windows client machine and import that certificate into the Windows Certificate store. See Adding certificates to the Trusted Root Certification Authorities store for a local computer.

Configure CAS Internode TLS (Linux Full Deployment)

Note: The following instructions are for configuring internode TLS in a SAS Viya full deployment.

CAS supports TLS encrypted connections between the worker nodes. When configured, any data sent between worker nodes is sent over a TLS connection. The CAS internode communication is not secured by default. This is due to the large performance impact of enabling CAS internode encryption.

Items of note when configuring CAS internode TLS are as follows:

- There is a significant performance impact to using internode encryption
- The TLS certificates and private keys are deployed by default.
- On the CAS controller node, the env.CAS_INTERNODE_DATA_SSL is set to FALSE by default. This option is set originally in the **`/opt/sas/viya/config/etc/cas/default/casconfig_deployment.lua`** file.

CAUTION! Encryption has performance costs. Encryption will degrade your performance and increase the amount of CPU time that is required to complete any action. Actions that move large amounts of data are penalized the most. Session start-up time is also impacted negatively. On tests that move large blocks of data between nodes, elapsed times can increase by a factor of ten.

Configure CAS internode encryption using the SAS Bootstrap Config CLI on SAS Configuration Server (Consul).

1. Turn on CAS internode TLS.

   Note: The following commands should be run as a root or sudo user.

   First, set the Consul access token in the CONSUL_HTTP_TOKEN environment variable. This command needs to be performed before executing any utilities or services that might access Consul.

   ```
   . /opt/sas/viya/config/consul.conf
   export CONSUL_HTTP_TOKEN=$(sudo cat /opt/sas/viya/config/etc/SASSecurityCertificateFramework
   /tokens/consul/default/client.token)
   ```

   Use SAS Bootstrap Config CLI to turn on CAS Internode TLS.

   Note: The SAS Bootstrap Config CLI must establish trust for the TLS handshake to proceed and allow secure communication. To establish trust, the truststore must be specified as an environment variable. Sourcing the consul.conf file sets the SSL_CERT_FILE environment variable to the trusted certificates. After this trust is established, you can communicate using the SAS Bootstrap Config CLI.

   ```
   . /opt/sas/viya/config/consul.conf
   /opt/sas/viya/home/bin/sas-bootstrap-config kv write --force
   config/cas-shared-default/sas.security/network.casInternode.enabled true
   ```

   Note: The preceding code is shown on more than one line for display purposes only. Place these commands on one line without line breaks.

2. Restart the cascontroller service on each controller and node. How you run the following command depends on your operating system.

   - Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

     ```
     sudo systemctl restart sas-viya-cascontroller-default
     ```

   - Red Hat Enterprise Linux 6.x (or an equivalent distribution):

```
sudo service sas-viya-cascontroller-default restart
```

To validate that CAS is using internode encryption, access SAS Environment Manager and verify that the CAS_INTERNODE_SSL environment variables are set.

1 From the applications menu (☰), under **Administration** , select **Manage Environment**.

2 From the side menu, click .

Note: The tasks described in this section are performed from the Servers page and most can be performed only by SAS Administrators.

3 You can view CAS server configuration values and identify how they are set. Click .

4 Make sure that the **Nodes** tab is selected. Click . The following CAS internode environment variables should be set as follows if you are using the certificates provided by SAS Viya.

Environment Variable	Value
CAS_INTERNODE_DATA_SSL	TRUE
CAS_INTERNODE_SSL_CERT	/opt/sas/viya/config/etc/ SASSecurityCertificateFramework/tls/ certs/cas/sharted/default/sas_encrypted.crt
CAS_INTERNODE_SSL_KEY	/opt/sas/viya/config/etc/ SASSecurityCertificateFramework/ private/cas/sharted/default/sas_encrypted.key
CAS_INTERNODE_SSL_KEYPW	*********
CAS_INTERNODE_SSL_KEYPWLOC	/opt/sas/viya/config/etc/ SASSecurityCertificateFramework/ private/cas/sharted/default/encryption.key
CAS_INTERNODE_SSL_CA_LIST	/opt/sas/viya/config /etc/SASSecurityCertificateFramework/ cacerts/trustedcerts.pem

5 To return to the Servers page, in the top left corner of the window, click .

Access CAS HTTP and HTTPS

Overview

CAS HTTP provides the following:

- a REST interface to the CAS Server. See CAS REST API.
- CAS Server Monitor in a programming-only deployment.

CAS Server Monitor is available only if you are using a SAS Viya programming-only environment. In a full deployment, the monitoring services are provided using SAS Environment Manager. See “Monitoring: How to (SAS Environment Manager)” in *SAS Viya Administration: Monitoring*.

For direct URL access on Linux and Windows, open a web browser and enter the following URL in the address field:

```
http://reverse-proxy-server/cas-shared-default-http
```

For multi-tenant access on Linux:

```
https://tenant.reverse-proxy-server/cas-tenant-instance-http
```

Log on using one of the SAS Administrator users that were established during deployment. See "Set Up Administrative Users" in *SAS Viya for Linux: Deployment Guide* and "Set Up Administrative Users" in *SAS Viya for Windows: Deployment Guide*.

Access CAS Server Monitor from SAS Studio

From within SAS Studio, complete these steps to access CAS Server Monitor (CAS Administration). CAS Server Monitor is accessed using the HTTPS protocol by default.

1. In the banner of SAS Studio, click ☰ and select **CAS Administration**.
2. In the sign-in window, enter your operating system credentials.
3. To sign out, at the right edge of the banner, click your user name, and select **Sign Out**.

Note: To access CAS Server Monitor, the password must be set for the CAS user ID or other administrative account. See "Access CAS Server Monitor" in *SAS Viya for Linux: Deployment Guide* and "Set Up Administrative Users" in *SAS Viya for Windows: Deployment Guide* for information about password access setup during deployment.

If you receive a "Connection Not Secure" message because of HTTPS access, you need to take one of the following actions:

- Import the Certificate Authority certificates used by the Apache HTTP Server into your browser.
- Change the HTTP protocol to HTTPS by changing the variables in the casconfig_usermods.lua file that control the protocol and port used in the CAS Server Monitor link accessed from within SAS Studio. Change the following variables:

  ```
  env.CAS_VIRTUAL_HOST = 'external.mycompany.com'
  env.CAS_VIRTUAL_PORT = 443
  env.CAS_VIRTUAL_PROTO = 'https'
  ```

- Make sure the host name for the CAS_VIRTUAL_HOST variable is the same as the Common Name used in the server identity certificate that the Apache HTTP Server is using.

Encrypt LDAP Connections

Configure the LDAPS (Secure LDAP) Connection

Lightweight Directory Access Protocol (LDAP) connections can be established in a TLS session so that all data that is sent between the LDAP client and LDAP server is encrypted. LDAP over TLS is known as LDAPS.

To securely connect to an LDAP provider, SAS Viya needs access to the CA certificate used by the LDAP provider. To configure TLS between SAS Viya and the LDAP provider, use the following instructions to add the CA certificates to the trustedcerts files on every machine in the deployment. See "Use Best Practices to Create and Manage Certificates" on page 74.

Note: Only LDAP-based identity providers are supported. These instructions assume that you have basic familiarity with LDAP administration.

1. Log on to your machine as install user or administrator. On Linux, log in as a user with root, SAS Admin, or sudo privileges.
2. On Windows, if your LDAPS provider's CA certificate is not already included in the Mozilla bundle of trusted CA certificates, append the root certificate to the truststores (trustedcert.pem and trustedcerts.jks files). You manually add the CA certificates to the truststore.

Note: On Linux, you can also use Ansible to update the truststores. See Step 3 on page 38 for instructions.

- To add the root certificate to the trustedcerts.pem file, just include the root certificate at the end of the trustedcerts.pem file.
- To add the root certificate to the trustedcerts.jks file, import the file using the keytool command.

Manually add the CA certificates to the truststore. See "Add Certificates to or Remove Certificates from the Truststore or a Certificate Chain Manually" on page 68.

3 On Linux, you can use Ansible to add the LDAPS provider's CA certificate to the truststore if it is not already included in the Mozilla bundle of trusted CA certificates. Add the new certificates to the sitedefault.yml file located in **`/opt/sas/viya/config/etc/consul.d/default`**.

For our example, we are adding the erca certificate to sitedefault.yml.

Note: Keep the indention of this file. If you do not have an identifier node named `cacerts`, add it to the file. Beneath that node, add nodes that identify each of the CA root certificates that are being used. In this example, the certificate identifier named `sascaroot` and the certificate file was already added to this file. Using an editor, we added another customer certificate (`customerca`).

```
cacerts:
    sascaroot: |
        -----BEGIN CERTIFICATE-----
        certificate string
        -----END CERTIFICATE-----
    customerca: |
        -----BEGIN CERTIFICATE-----
         certificate string
        -----END CERTIFICATE-----
```

To manually add the certificates, see "Add Certificates to or Remove Certificates from the Truststore or a Certificate Chain Manually" on page 68 for information about adding your certificates to the trustedcerts files.

4 On Linux, restart the SAS Configuration Server to pick up the changes to the sitedefault.yml file. Only new key-value pairs are added. Existing key-values are not updated. How you run the following command depends on your operating system.

- Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

  ```
  sudo systemctl restart sas-viya-consul-default
  ```

- Red Hat Enterprise Linux 6.x (or an equivalent distribution):

  ```
  sudo service sas-viya-consul-default restart
  ```

5 On Linux, using Ansible, run the utility play rebuild-trust-stores.yml to rebuild the truststores. This play adds the certificates to all truststores on all hosts.

```
ansible-playbook -i inventory.ini ./utility/rebuild-trust-stores.yml
```

6 On Linux, add the certificates that you added in step 1 to the sitedefault.yml file located in **`/roles/consul/files`**. This file forms part of the Ansible playbook.

7 Use the SAS Environment Manager to set the configuration property sas.identities.providers.ldap.connection. Specify an LDAPS port number (by default LDAPS is 636) and specify *LDAPS* in the **url** field. You can also use the port value 3269 (Global Catalog) for LDAPS.

a If you are not already in SAS Environment Manager, select **Manage Environment** from the applications menu (☰).

b From the side menu, click .

The Configuration page is an advanced interface. It is available only to SAS Administrators.

c Select **Basic Services** from the list, and then select the **Identities service** from the list of services.

d In the **sas.identities.providers.ldap.connection** section, click . In the Edit sas.identities.providers.ldap.connection Configuration window, do the following:

i Update values for the **port** field, adding an LDAPS port value. Update the **url** field to specify *LDAPS*. For the remaining fields, review the default values and make changes as necessary. The default values are appropriate for most sites.

ii Click **Save**.

For information about how to configure the connection to your identity provider, see "Configure Security" in *SAS Viya for Linux: Deployment Guide*. For details about the sas.identities.providers.ldap.connection property, see "Configuration Properties: Reference (Services)" in *SAS Viya Administration: Configuration Properties*.

8 Restart the SAS Logon Manager service. How you run the following command depends on your operating system.

- Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

```
sudo systemctl restart sas-viya-saslogon-default
```

- Red Hat Enterprise Linux 6.x (or an equivalent distribution):

```
sudo service sas-viya-saslogon-default restart
```

- Restart the SAS Logon Manager service by using the Services snap-in from the Microsoft Management Console. Use the search box to search for the Services App. See "Operate (Windows)" in *SAS Viya Administration: SAS Cloud Analytic Services*.

Note: It might take several minutes to restart SAS Logon Manager.

9 Restart the Identifies service. How you run the following command depends on your operating system.

- Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

```
sudo systemctl restart sas-viya-identities-default
```

- Red Hat Enterprise Linux 6.x (or an equivalent distribution):

```
sudo service sas-viya-identities-default restart
```

- Restart the Identities service by using the Services snap-in from the Microsoft Management Console. Use the search box to search for the Services App. See "Operate (Windows)" in *SAS Viya Administration: SAS Cloud Analytic Services*.

Configure the Secure LDAP Connection Using STARTTLS

Lightweight Directory Access Protocol (LDAP) connections can be established in a TLS session so that all data that is sent between the LDAP client and LDAP server is encrypted. In SAS Viya 3.4, SAS supports encrypting connections to LDAP using STARTTLS.

STARTTLS upgrades a connection that is not encrypted by wrapping it with TLS during the connection process. This allows unencrypted and encrypted connections to be handled by the same port.

To connect to a STARTTLS provider, SAS Viya needs access to the CA certificate used by the STARTTLS provider. SAS recommends that the customer configure SAS Viya with their own CA certificates. It is recommended that this certificate also be the same CA certificate that is used by the STARTTLS server.

To configure TLS between SAS Viya and the STARTTLS provider, use the following instructions to add the CA certificates to the trustedcerts files on every machine in the deployment. See "Use Best Practices to Create and Manage Certificates" on page 74.

Note: Only LDAP-based identity providers are supported. These instructions assume that you have basic familiarity with LDAP administration.

1 Log on to your machine as a user with root, SAS Admin, or sudo privileges.

2 On Windows, if your STARTTLS provider's CA certificate is not already included in the Mozilla bundle of trusted CA certificates, append the root certificate to the truststores (trustedcert.pem and trustedcerts.jks files).

   Note: On Linux, you can also use Ansible to update the truststores. See Step 3 on page 40 for instructions.

   - To add the root certificate to the trustedcerts.pem file, just include the root certificate at the end of the trustedcerts.pem file.
   - To add the root certificate to the trustedcerts.jks file, import the file using the keytool command.

   Manually add the CA certificates to the truststore. See "Add Certificates to or Remove Certificates from the Truststore or a Certificate Chain Manually" on page 68.

3 On Linux, if your STARTTLS provider's CA certificate is not already included in the Mozilla bundle of trusted CA certificates, you can use Ansible to add those certificates. Add the new certificates to the sitedefault.yml file located in **`/opt/sas/viya/config/etc/consul.d/default`**.

   In the following example, we are adding a new customer CA certificate below the customerca label in the sitedefault.yml file.

   Note: Keep the indention of this file. If you do not have an identifier node named `cacerts`, add it to the file. Beneath that node, add nodes that identify each of the CA root certificates that are being used. In this example, the certificate identifier named `sascaroot` and the certificate file was already added to this file. Using an editor, we added another customer certificate (`customerca`).

```
cacerts:
   sascaroot: |
       -----BEGIN CERTIFICATE-----
       certificate string
       -----END CERTIFICATE-----
   customerca: |
       -----BEGIN CERTIFICATE-----
        customer CA certificate string
       -----END CERTIFICATE-----
```

   To manually add certificates to the truststore, see "Add Certificates to or Remove Certificates from the Truststore or a Certificate Chain Manually" on page 68.

4 On Linux, restart the SAS Configuration Server to pick up the changes to the sitedefault.yml file. Only new key-value pairs are added. Existing key-values are not updated. How you run the following command depends on your operating system.

   - Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

     ```
     sudo systemctl restart sas-viya-consul-default
     ```

   - Red Hat Enterprise Linux 6.x (or an equivalent distribution):

     ```
     sudo service sas-viya-consul-default restart
     ```

5 On Linux, using Ansible, run the utility play rebuild-trust-stores.yml to rebuild the truststores. This play adds the certificates to all truststores on all hosts.

```
ansible-playbook -i inventory.ini ./utility/rebuild-trust-stores.yml
```

6 On Linux, add the certificates that you added in step 1 to the sitedefault.yml file located in **/roles/consul/files**. This file forms part of the Ansible playbook.

7 Use the SAS Environment Manager to set the configuration property sas.identities.providers.ldap.connection.

   a If you are not already in SAS Environment Manager, select **Manage Environment** from the applications menu ( ).

   b From the side menu, click

   The Configuration page is an advanced interface. It is available only to SAS Administrators.

   c Select **Basic Services** from the list, and then select the **Identities service** from the list of services.

   d In the **sas.identities.providers.ldap.connection** section, click . In the Edit sas.identities.providers.ldap.connection Configuration window, do the following:

      i Update values for the **port** field, adding a STARTTLS port value (389 or 3268). Set startTLS.mode to *simple*. Update the **url** field. Prefix this url with *ldap*. For the remaining fields, review the default values and make changes as necessary. The default values are appropriate for most sites.

      The url field might look something like the following:

      ```
      ldap://${sas.identities.providers.ldap.connection.host}:
      ${sas.identities.providers.ldap.connection.port}
      ```

      ii Click **Save**.

   For information about how to configure the connection to your identity provider, see "Configure Security" in *SAS Viya for Linux: Deployment Guide*. For details about the sas.identities.providers.ldap.connection properties, see "Configuration Properties: Reference (Services)" in *SAS Viya Administration: Configuration Properties*.

8 Restart the SAS Logon Manager service. How you run the following command depends on your operating system.

   - Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

     ```
     sudo systemctl restart sas-viya-saslogon-default
     ```

   - Red Hat Enterprise Linux 6.x (or an equivalent distribution):

     ```
     sudo service sas-viya-saslogon-default restart
     ```

   - Restart the SAS Logon Manager service by using the Services snap-in from the Microsoft Management Console. Use the search box to search for the Services App. See "Operate (Windows)" in *SAS Viya Administration: SAS Cloud Analytic Services*.

   Note: It might take several minutes to restart SAS Logon Manager.

9 Restart the Identifies service. How you run the following command depends on your operating system.

   - Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

     ```
     sudo systemctl restart sas-viya-identities-default
     ```

   - Red Hat Enterprise Linux 6.x (or an equivalent distribution):

     ```
     sudo service sas-viya-identities-default restart
     ```

   - Restart the Identities service by using the Services snap-in from the Microsoft Management Console. Use the search box to search for the Services App. See "Operate (Windows)" in *SAS Viya Administration: SAS Cloud Analytic Services*.

Replace Certificates in the Truststore for Secure LDAP (Linux)

By default, the Apache HTTP Server has been configured to serve as a reverse proxy to connect to the SAS Viya web application. SAS Viya installation provides certificates and configures TLS options in the Apache HTTP Server configuration. However, SAS recommends replacing the default certificates with custom certificates. After the certificates have been replaced in the Apache HTTP Server, the truststore can be rebuilt using the following instructions.

To connect to a STARTTLS or LDAPS provider, SAS Viya needs access to the CA certificate used by the STARTTLS or LDAPS provider. SAS recommends that the customer configure SAS Viya with their own CA certificates. SAS also recommends that this certificate be the same CA certificate that is used by the STARTTLS or LDAPS server.

Note: For information about TLS on Apache, see "Update Apache HTTP Server TLS Certificates and Cryptography " on page 5.

1 Locate the certificates that you would like to remove.

   If they were added to the sitedefault.yml file, you will need to remove them. There are two locations for the sitedefault.yml file.

   - **`/opt/sas/viya/config/etc/consul.d/default`**
   - **`/roles/consul/files`**

   For our example, we are removing the customerca certificate from sitedefault.yml. In the following example, beneath the `cacerts` node, use a text editor and remove the customer certificate (`customerca`). The certificates left in the cacerts node are the certificate strings for sascaroot and the sassha2rootca.

   Note: Keep the indention of this file.

```
cacerts:
    sascaroot: |
        -----BEGIN CERTIFICATE-----
        certificate string
        -----END CERTIFICATE-----
    sassha2rootca: |
        -----BEGIN CERTIFICATE-----
         certificate string
        -----END CERTIFICATE-----
```

2 To update the certificates, you must first remove the certificates and keys from Consul that you want to replace or update. The SAS Bootstrap Config CLI must establish trust for the TLS handshake to proceed and allow secure communication. To establish trust, the truststore must be specified as an environment variable. Sourcing the consul.conf sets the SSL_CERT_FILE environment variable to the trusted certificates.

   Note: The following commands should be run as a root or sudo user.

   The code is shown on more lines for display purposes only. This command should be on one line and should not contain line breaks.

   a Set the Consul access token in the CONSUL_HTTP_TOKEN environment variable. This command needs to be performed before executing any utilities or services that might access Consul.

```
. /opt/sas/viya/config/consul.conf
export CONSUL_HTTP_TOKEN=$(sudo cat /opt/sas/viya/config/etc/SASSecurityCertificateFramework
/tokens/consul/default/client.token)
```

   b Remove the existing key-value pair from the cacerts tree using the following sas-bootstrap-config command.

```
. /opt/sas/viya/config/consul.conf
```

```
/opt/sas/viya/home/bin/sas-bootstrap-config kv delete cacerts/sassha2rootca
```

3 Ensure that the certificates have been removed.

```
/opt/sas/viya/home/bin/sas-bootstrap-config --consul
https://your-configuration-server:8501
--token 75bef370-cc93-44bb-a290-82833f6c4ddf kv read
--recurse cacerts
```

4 Stop all services.

Note: On a Linux multiple-machine deployment, there is a sequence for starting and stopping SAS Viya servers and services. See "General Servers and Services: Operate (Linux)" in *SAS Viya Administration: General Servers and Services*.

```
sudo /etc/init.d/sas-viya-all-services stop
```

5 Using Ansible, run the utility play rebuild-trust-stores.yml to rebuild the truststores.

```
ansible-playbook -i inventory.ini ./utility/rebuild-trust-stores.yml
```

6 Restart all the services so that they now reference the new truststores. Restart the SAS Configuration Server to pick up the changes to the sitedefault.yml file. Remember only new key-value pairs are added. Existing key-value pairs are not updated.

Note: On a multiple-machine deployment, there is a sequence for starting and stopping SAS Viya servers and services. See "General Servers and Services: Operate (Linux)" in *SAS Viya Administration: General Servers and Services*.

```
sudo /etc/init.d/sas-viya-all-services start
```

7 The next time that you sign in, application will authenticate using a secured LDAP connection.

Access SAS Studio via HTTPS

SAS Studio is set up for HTTP and HTTPS during initial deployment. In SAS Viya, SAS Studio is configured to work with the Apache HTTP Server.

The version of SAS Studio that you are using depends on which type of deployment you performed. If you deployed a full environment, then your environment contains both SAS Studio 4.4 and SAS Studio 5.1.

- To access SAS Studio 4.4, open a web browser and enter the following URL in the address field: https://*reverse-proxy-server*/SASStudio. By default, SAS Studio is secured.
- To access SAS Studio 5.1, open a web browser and enter the following URL in the address field: https://*reverse-proxy-server*/SASStudioV. By default, SAS Studio is secured.

SAS Studio 5.1 is available in the full visual deployment. The new HTML 5 interface integrates with other SAS applications and is available as part of SAS Drive. SAS Studio 5.1 also uses microservices and has common authentication with other SAS applications.

If you deployed a programming-only environment, your environment contains SAS Studio 4.4. To access SAS Studio in a programming-only deployment, open a web browser and enter the following URL in the address field: https://*reverse-proxy-server*/SASStudio/.

Access SAS Message Broker via HTTPS and HTTP (Linux Full Deployment)

Note: This section is applicable only if you have a Linux SAS Viya full deployment. If you have a Linux SAS Viya programming-only deployment, skip this section.

By default, SAS provides self-signed certificates and keys to secure the deployment. The URLs available to access SAS Message Broker for HTTP and HTTPS post installation are as follows:

- https://*RabbitMQ-IP-address*:15672/#/ (Communication via this URL is encrypted.)

- http://*RabbitMQ-IP-address*:15672/#/ (Communication via this URL is not secured.)

If you receive a certificate error after clicking the preceding HTTPS link to access the SAS Message Broker, you might need to import the SAS Viya trusted CA certificate into the browser truststore. Here is one way to perform this task:

1. From your browser, enter https://*RabbitMQ-IP-address*:15672/#/
2. Import your SAS Viya intermediate CA.
   a. In the RabbitMQ login window, click **Certificate error**.
   b. In the Untrusted Certificate dialog box, select **View Certificates**.
   c. In the Certificate window, click the **Install Certificate** button.
   d. From the Certificate Import Wizard dialog box, for **Store Location**, select **Current User**. Select **Next**.
   e. For **Certificate Store**, select **Place all certificates in the following store**. Browse and select **Trusted Root Certification Authorities**. Click **OK**.
   f. Click **Next**. Then select **Finish**. You should receive the following message: `Import was successful.`

Use SAS/CONNECT with TLS Enabled to Import Data

Overview

SAS programming clients in a 9.4M5 environment can call procedures that are enabled in SAS Viya and submit DATA step code, operating directly on CAS data sources. As a result, SAS/CONNECT is no longer required as a separate product in order to transfer data from SAS 9.4M5 and later to a SAS Viya deployment. The built-in CAS integration capabilities of SAS 9.4M5 enable the user to communicate directly from a CAS client. For more information, see "SAS 9 and SAS Viya" in *SAS Viya: Overview*.

Note: SAS 9.4M6 contains all of the most current integration updates.

However, SAS/CONNECT is required to transfer data from SAS deployments prior to SAS 9.4M5 to SAS Viya. When you use SAS/CONNECT to import data from releases prior to SAS 9.4M5 to SAS Viya, SAS Viya acts as the SAS/CONNECT client. The SAS/CONNECT server must be running on the SAS 9.4 machine, and the connection to SAS 9.4 must be initiated from SAS Viya. SAS/CONNECT must be available in both environments if you intend to upload, download, and analyze data within CAS from a SAS 9.4 client.

You can use TLS to secure the SAS/CONNECT bridge when you sign on to the SAS/CONNECT spawner from a SAS/CONNECT client. The sign-on command starts a SAS/CONNECT server.

This section discusses the following topics:

- How to configure TLS for SAS/CONNECT on a Windows deployment.
- How to configure TLS for SAS/CONNECT on a Linux programming-only deployment.

TLS Certificates

You need to use the certificates provided by SAS, add custom certificates, or generate your own certificates to use TLS to secure SAS/CONNECT.

- If you are using a certificate whose root CA is not already in the Mozilla Trusted CA Certificate bundle, you need to add the root CA certificate to the truststore by editing the trustedcerts.pem file. See "Add Certificates to or Remove Certificates from the Truststore or a Certificate Chain Manually" on page 68.
- To create site-signed or third-party-signed certificates using OpenSSL,see "Generate Site-Signed or Third-Party-Signed Certificates in PEM Format" on page 77.

- To create a self-signed CA certificate with SANS using OpenSSL, see "Create Certificates with SANs Using OpenSSL" on page 76.

Note: If you are using custom certificates or generating your own certificates, use Best Practices on page 74 for securing your certificates and keys.

If you are using SAS 9.4 to start a SAS/CONNECT session to a SAS Viya CAS server, the server certificate needs subject alternative name (SAN) extension entries in the server certificate for each name the host can be known by. The certificate needs the physical host name and DNS alias listed in the SAN. If you need to update the default certificates provided by SAS to include the SANs, contact SAS Technical Support. They can use the sas-crypto-management tool to re-create these default certificates.

Enable and Disable TLS for SAS/CONNECT (Linux Full Deployment)

You can enable and disable TLS for SAS/CONNECT in a Linux full deployment by port family. By default, the sasData port family is turned on. The sasData port family controls TLS for SAS/CONNECT server and SAS/CONNECT spawner. For information about how to enable and disable TLS by port families, see "Enable or Disable TLS Using Port Families" on page 61.

Configure SAS/CONNECT to Use TLS (Linux programming-only deployment)

To configure SAS/CONNECT in a programming-only deployment of SAS Viya, perform the following steps:

1. Sign in with administrator privileges to the machine containing the SAS/CONNECT spawner.
2. In the connect_usermods.sh file located in **/opt/sas/viya/config/etc/connect/default**, set up TLS by adding the TLS encryption options. Edit the connect_usermods.sh file, and add the following encryption options to the USERMODS= line to encrypt the connection for the SAS/CONNECT spawner. Note that this file needs to have global Read permissions: -rw-r--r--

   In the following code example, the names of the certificate file and the private key file are just example names. These would be the names of the files that you placed in the **/opt/sas/viya/config** directories.

   Note: The options are enclosed in double quotation marks.

   ```
   USERMODS="-netencrypt            /* a */
   -netencryptalgorithm ssl         /* b */
   -sslcertloc /opt/sas/viya/config/etc/SASSecurityCertificateFramework/
   tls/certs/server.crt             /* c */
   -sslpvtkeyloc /opt/sas/viya/config/etc/SASSecurityCertificateFramework/private/
   private.key                      /* d */
   -sslpvtkeypass 'password'"       /* e */"
   ```

   a The NETENCRYPT option specifies that encryption is required.

   b The NETENCRYPTALGORITHM= option specifies that the spawner is started using TLS.

   c The SSLCERTLOC= option specifies the location of a file that contains a digital certificate for the machine's public key. This is used by the server to send to clients for authentication.

   Note: If the certificate is not self-signed, the file specified by the SSLCERTLOC= option needs to be a certificate chain file that starts with the server identity certificate and includes the signing intermediate CA certificates. The root CA certificate does not need to be included in the certificate chain.

   d The SSLPVTKEYLOC= option specifies the location of the file that contains the private key that corresponds to the digital certificate that was specified by the SSLCERTLOC= option.

   e The SSLPVTKEYPASS= option specifies the password that TLS requires to decrypt the private key. The private key is stored in the file that was specified by the SSLPVTKEYLOC= option.

   Note: SAS first looks for CA certificates in a file named trustedcerts.pem, located in the **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts** directory. Therefore, the SSLCALISTLOC= system option is not required if you are storing your trusted certificates in the default

location. However, if you choose not to use the default location to store certificates, you need to specify the SSLCALISTLOC= option with a location for the certificates for the SAS/CONNECT client and spawner.

For each of the preceding examples, the default location is used.

3 Start the SAS/CONNECT spawner. How you run the following command depends on your operating system.

- Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

```
sudo systemctl restart sas-viya-connect-default
```

- Red Hat Enterprise Linux 6.x (or an equivalent distribution):

```
sudo service sas-viya-connect-default restart
```

4 The SAS/CONNECT spawner runs the connectserver.sh script, which runs the connectserver_usermods.sh script. The connectserver_usermods.sh script is located in **`/opt/sas/viya/config/etc/connectserver/default`**. Edit the connectserver_usermods.sh file, and add the following encryption options to the USERMODS_OPTIONS= line. Note that this file needs to have global Read permissions: -rw-r--r--

Note: The options are enclosed in double quotation marks.

```
USERMODS_OPTIONS="-sslcertloc /opt/sas/viya/config/etc/
SASSecurityCertificateFramework/tls/certs/customer.crt
-sslpvtkeyloc /opt/sas/viya/config/etc/SASSecurityCertificateFramework/
private/customer.key
-sslpvtkeypass 'password'"
```

Note: The certificates specified in the preceding code are your server certificates.

5 After a spawner is started on a SAS/CONNECT server, a SAS/CONNECT client can connect to it. The following example shows how to connect a client to a spawner that is running on a SAS/CONNECT server:

```
options netencryptalgorithm=SSL;
%let myserver=<myHost.myDomain.com> <port>;
SIGNON myserver user=sasdemo passwd="password";
```

6 If you need a SAS 9.4 client to work with SAS Viya, see “Configure SAS 9.4 Clients to Work with SAS Viya” on page 59.

See Also

- “SAS System Options for Encryption” on page 111
- SAS Viya: Overview

Configure SAS/CONNECT to Use TLS (Windows)

In a SAS Viya 3.4 deployment on Windows, configure TLS for SAS/CONNECT by performing the following steps.

You can use the certificates provided by SAS, add custom certificates, or request your own certificate from the Microsoft Certificate Authority to use TLS to secure SAS/CONNECT.

In a SAS Viya programming-only deployment or a deployment prior to SAS 9.4M5, you need to configure TLS for SAS/CONNECT. To configure SAS/CONNECT on Windows, perform the following steps:

1 Sign in with administrator privileges to the machine containing the SAS/CONNECT spawner.

2 After SAS Viya deployment, self-signed certificate and key files provided by SAS are located in the following directories. These files are needed to update the Windows Truststore in order to configure SAS/CONNECT.

Table 5 Security Artifacts Provided at Installation for SAS Viya Windows Deployment

Security Artifact	Deployment File Name	Location	Description
Certificate truststore	trustedcerts.pem trustedcerts.jks	**C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\cacerts**	Contains the trusted list of CA certificates. These include the Mozilla bundle of trusted CA certificates, the SAS Viya self-signed certificates, the Apache httpd certificates, and the chain of trust certificates.
Certificate file	sas_encrypted.crt	**C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\tls\certs**	Contains the certificate generated by SAS Viya. These are self-signed certificates.
Certificate private key file	sas_encrypted.key	**C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\private**	Contains the private key generated by SAS.
Certificate private key passphrase file	encryption.key (optional)	**C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\private**	Contains the encrypted passphrase file provided by SAS Viya.

3 Update the Windows Truststores.

To use the SAS Viya self-signed certificates (shown in the preceding table) that are provided at installation, perform the following tasks:

a Open a Windows PowerShell prompt as an Administrator.

b Change to directory **C:\Program Files\SAS\Viya\SASFoundation\utilities\bin**.

c Use Windows PowerShell to run the Enable-CAS-TLS.ps1 script. This script reads the certificate serial number, the certificate thumbprint, and the certificate issuer of the sas_encrypted.crt certificate provided by SAS. It then creates a temporary .pfx file and imports that file into the Windows Certificate store on the Local machine.

From the PowerShell prompt, enter the following command:

```
.\Enable-CAS-TLS.ps1
```

Observe the output of the PowerShell script. Confirm that no errors are printed.

For information about how to set up and use Windows PowerShell, see How to Open Powershell in Windows 10.

Here is a sample of the output that you should see when the Enable-CAS-TLS.ps1 script is run. Save the results of the certificate serial number and the certificate issuer for use in setting system options SSLCERTISS and SSLCERTSERIAL.

```
Sample output:
PS C:\Program Files\SAS\Viya\SASFoundation\utilities\bin> .\Enable-CAS-TLS.ps1
[INFO] Reading certificate serial number from file
C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\tls\certs\sas_encrypted.crt
[INFO] certificateSerialNumberString=4ED7DA72EBA098E7 (length 16)
[INFO] Reading certificate thumbprint from file
C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\tls\certs\sas_encrypted.crt
[INFO] certificateThumbprintString=F6:C6:CA:96:6A:78:D2:7A:A7:8E:2A:6B:
E0:32:DB:E0:12:45:E2:F8 (length 59)
[INFO] Reading certificate issuer from file
C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\tls\certs\sas_en crypted.crt
[INFO] certificateIssuerString=cntvdmml18w25w.na.sas.com (length 25)
[INFO] Creating temporary PFX file
```

d Verify that the certificates have been added to the Windows Truststores. Certificates appear in the **`Certificates`** folder, **`Personal Certificates`**, and the **`Trusted Root Certificate Authorities Certificates`** folder. Use the Windows MMC snap-in to perform this function.

   i Click the Windows command prompt, enter **`mmc`**, and click **OK**.

   ii In the Console window, select **File** ⇨ **Add/Remove Snap-in**.

   iii Double-click **Certificates** from the list of available snap-ins.

   iv In the dialog box that appears, select **Computer account**, and click **Next**.

   v In the Select Computer window, select **Local computer**, click **Finish**, and then click **OK**.

   vi In the Console window, expand **Certificates (Local Computer)** on the left.

   vii In the Console window, expand **Trusted Root Certification Authorities** to make sure that the certificates have been imported.

   viii In the Console window, expand **Personal** to make sure that the certificates have been imported.

From a Windows client machine, perform the following steps.

1 Copy certificate file sas_encrypted.crt from **`C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\tls\certs`** from the first machine to a location on your client machine.

2 Click the Windows command prompt, enter **`mmc`**, and press the Enter key.

3 In the Console window, select **File** ⇨ **Add/Remove Snap-in**.

4 Double-click **Certificates** from the list of available snap-ins.

5 In the dialog box that appears, select **Computer account**, and click **Next**.

6 In the Select Computer window, select **Local computer**, click **Finish**, and then click **OK**.

7 In the Console window, expand **Certificates (Local Computer)** on the left.

8 In the Console window, expand **Trusted Root Certification Authorities**. Right-click **Certificates All Tasks** ⇨ **Import**

9 On the Certificate Import Wizard page, click **Next**.

10 On the second wizard page, click **Browse**, and navigate to the location where you placed the sas_encrypted.crt file. Click **Next**.

11 Make sure that **Place all certificates in the following store** is selected, and click **Next**, **Finish**, and then click **OK**.

12 In the Console window, expand **Trusted Root Certification Authorities** to make sure that the certificate that you imported is listed.

To configure SAS/CONNECT:

1 On the machine where you are running SAS/CONNECT, navigate to **C:\ProgramData\SAS\Viya\etc\connect**.

```
cd C:\ProgramData\SAS\Viya\etc\connect
```

2 Stop the SAS Connect Spawner service. In Windows Services Manager, right-click the **SAS Connect Spawner** service and select the **stop** operation.

3 To encrypt the connection for the SAS/CONNECT spawner using TLS, edit the connect_usermods.bat file located in **C:\ProgramData\SAS\Viya\etc\connect** and add the following encryption options to the USERMODS= line in the file.

Copy the values of the Issuer and the serial number from the output shown above .

Note: The options are enclosed in double quotation marks.

```
USERMODS="-netencrypt                    /* a */
-netencryptalgorithm ssl                 /* b */
-sslcertiss "issuer-of-digital-certificate" /* c */
-sslcertserial "cert-serial-number"        /* d */"
```

a The NETENCRYPT option specifies that encryption is required.

b The NETENCRYPTALGORITHM= option specifies that the spawner is started using TLS.

c The SSLCERTISS= option specifies the name of the issuer of the digital certificate that should be used by TLS.

d The SSLCERTSERIAL= option specifies the serial number of the digital certificate that should be used by TLS.

4 The SAS/CONNECT spawner runs the connectserver.bat script, which runs the connectserver_usermods.bat script. The connectserver_usermods.bat is located in **C:\ProgramData\SAS\Viya\etc\connectserver**. Edit the connectserver_usermods.bat file, and add the following encryption options to the USERMODS_OPTIONS= line.

The certificates that are being referenced are the server certificates.

Note: The options are enclosed in double quotation marks.

```
USERMODS="-netencrypt
-netencryptalgorithm ssl
-sslcertiss "issuer-of-digital-certificate"
-sslcertserial "cert-serial-number"
```

5 In Windows Services Manager, right-click the **SAS Connect Spawner** service and select the install operation.

Start the SAS/CONNECT spawner. In Windows Services Manager, right-click the **SAS Connect Spawner** service and select the start operation.

6 After a spawner is started on a SAS/CONNECT server, a SAS/CONNECT client can connect to it. The following example shows how to connect a client to a spawner that is running on a SAS/CONNECT server:

```
options netencryptalgorithm=SSL;
%let myserver=<myHost.myDomain.com> <port>;
SIGNON myserver user=sasdemo passwd="password";
```

7 If you need a SAS 9.4 client to work with SAS Viya, see "Configure SAS 9.4 Clients to Work with SAS Viya" on page 59.

See Also

- "SAS System Options for Encryption" on page 111
- SAS Viya: Overview

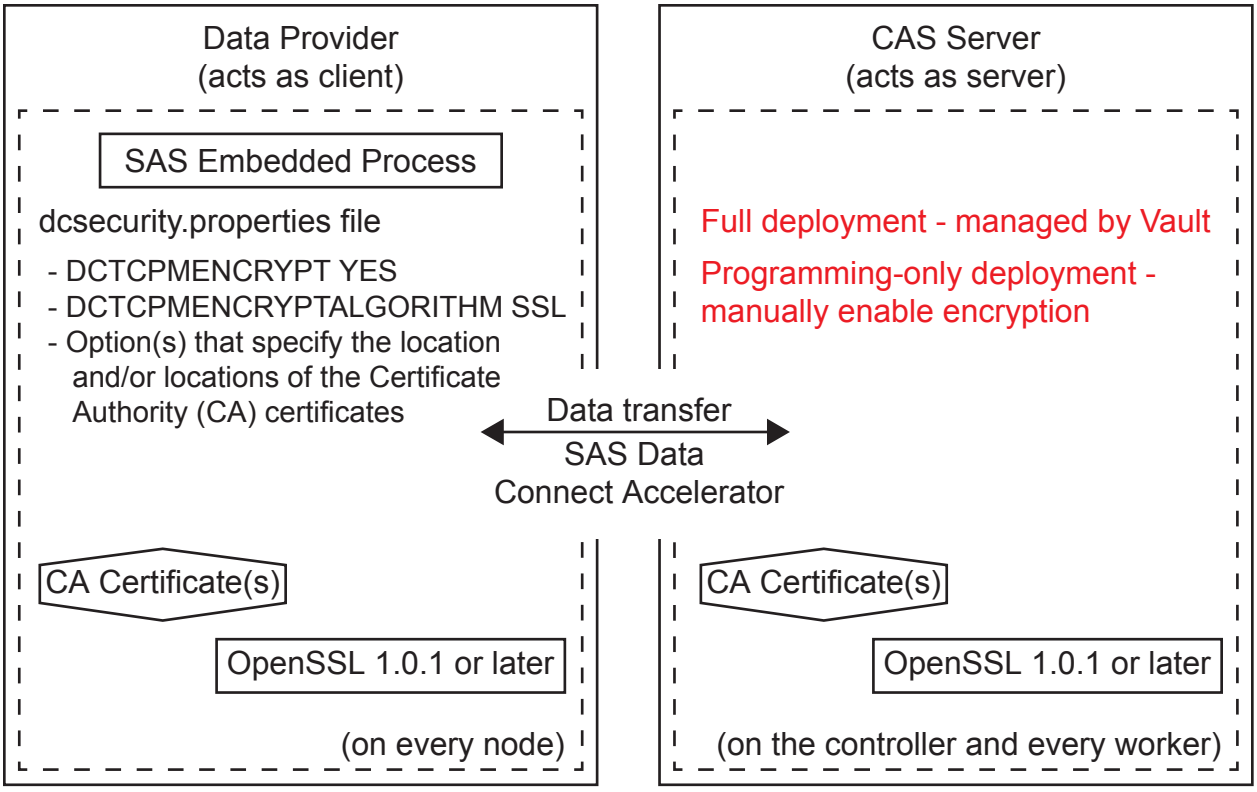
Encrypt Data Transfer When Using the SAS Data Connect Accelerator (Linux Full Deployment)

If you are using a SAS Data Connect Accelerator, the data that is transferred between the data provider and the CAS server is not encrypted by default. However, SAS Viya does support TLS encryption between the data provider and the CAS server, and you can take steps to enable that encryption. It should be noted that performance can be affected when TLS encryption is enabled and large amounts of data are being transferred.

Overview of SAS Data Connect Accelerator Encryption

When data is transferred between a data provider and CAS, the data provider acts as the client and the CAS server acts as a server.

Figure 1 *Data Connect Accelerator Encryption and the CAS Server*



When the SAS Embedded Process is deployed on the data provider, a dcsecurity.properties file and a **`certs`** directory are created, as well as the ***`SAS-Embedded-Process-home`***/**`security`** directory. The **`certs`** directory hold the TLS security certificates. The dcsecurity.properties file must be updated to enable data connector encryption.

Whether TLS is enabled and configured on the CAS server (server side) automatically depends on the type of SAS Viya deployment:

- By default, in a full deployment of SAS, TLS is enabled and configured on the CAS server. The deployment process provides a default level of encryption for data in motion. Hashicorp Vault issues certificates and keys that are used to secure the deployment. These certificates issued by Vault are provided for each CAS machine and are added to the Mozilla bundle of trusted CA certificates by default. Options are set in the vars.yml file and defined in the casconfig_deployment.lua file to enable data connector encryption and to provide the location of the TLS private key and password.
- In a programming-only deployment, you must set options on the CAS server in the casconfig_usermods.lua file to enable data connector encryption and to provide the location of the TLS private key and password.

The prerequisites and process for enabling TLS encryption on the CAS server (programming-only deployment) and the data provider is different for each data provider.

Note: A TLS private key and certificate are required for each CAS host.

Prerequisites When Enabling Encryption for the SAS Data Connect Accelerator for Teradata (on SAS Viya)

Here are the prerequisites for enabling encryption for the SAS Data Connect Accelerator for Teradata (on SAS Viya).

- Upgrade the OpenSSL package on all Teradata nodes to 1.0.1g or later to support TLS.

  The 64-bit OpenSSL library package that is most likely being used at your site is `libopenssl0_9_8-0.9.8j-0.50.1`. The required version is `libopenssl1_0_0-1.0.1g-0.37.1` or later. This package update is available on the Teradata patch server. Contact Teradata Customer Services to get this package updated. If you plan to use TLS now or in the future, it is best to upgrade the OpenSSL package before you install the SAS In-Database Technologies for Teradata (on SAS Viya).

  Note: The old version, **`openssl0_`**, and new version, **`openssl1_0_0`** (or later), can coexist.

- Install SAS/ACCESS Interface to Teradata (on SAS Viya) and SAS In-Database Technologies for Teradata (on SAS Viya).

  These offerings include the SAS Embedded Process, the SAS Data Connect Accelerator for Teradata (on SAS Viya), and the SAS Embedded Process support functions. For more information, see *SAS Viya for Linux: Deployment Guide*.

- Obtain TLS identity certificates (site-signed, third-party-signed, or self-signed) from the CAS controller machine. These certificates are located in the trustedcerts.pem file. Corresponding certificate authority (CA) certificates must be installed on the Teradata nodes. If you use externally signed identity certificates in the CAS server, the Mozilla bundle of CA certificates that are provided by SAS can be deployed on the Teradata nodes.

  For more information about the location of the trustedcerts.pem file, see "Encrypt Data Transfer When Using the SAS Data Connect Accelerator (Linux Full Deployment)" on page 50.

  For more information about configuring CAS, see "Configure CAS TLS to Use Custom Certificates (Linux Programming-Only Deployment)" on page 23 and "Configure CAS TLS to Use SAS Viya Default Certificates (Linux Programming-Only Deployment)" on page 28.

  Certificates, keys, and passwords produced for authenticating to the SAS Embedded Process for Teradata might coincide with those produced for other clients of the CAS server. However, they do not need to coincide. For information about generating certificates, see the appropriate topic in "Manage Certificates and Generate New Certificates" on page 74.

- When you installed the SAS Embedded Process, the following file and directory were created:

  For SAS Embedded Process version 16.0.0 and later:

  ```
  /opt/SAS/ep/home/security/dcsecurity.properties
  /opt/SAS/ep/home/security/certs
  ```

  For SAS Embedded Process version 15.0000 and earlier:

/opt/SAS/SASTKInDatabaseServerForTeradata/*ep-version*/security/dcsecurity.properties
/opt/SAS/SASTKInDatabaseServerForTeradata/*ep-version*/security/certs

- All directories and files should have the `owner:group = tdatuser:tdatudf` setting.
- The **/opt/SAS/ep/home/security** directory (version 16.0.0 and later) or **/opt/SAS/SASTKInDatabaseServerForTeradata/*ep-version*/security** directory (version 15.0000 and earlier) should have `drwxr-xr-x` permissions.
- The **/opt/SAS/ep/home/security/certs** directory (version 16.0.0 and later) or **/opt/SAS/SASTKInDatabaseServerForTeradata/*ep-version*/security/certs** directory (version 15.0000 and earlier) should have `drwxr-xr-x` permissions.
- The dcsecurity.properties file should have `-rwxr-xr-x` permissions.

Enable Encryption for the SAS Data Connect Accelerator for Teradata (on SAS Viya)

Follow these steps to encrypt data transfer between Teradata and the CAS server using the SAS Data Connect Accelerator for Teradata (on SAS Viya).

1. On the CAS server, modify the casconfig_usermods.lua configuration file to enable SAS Data Connect Accelerator encryption.

   Note: This step is required only for programming-only deployments. If you performed a full deployment of SAS Viya, verify the current setting of cas.DCTCPMENCRYPT in the **/opt/sas/viya/config/etc/cas/default/casconfig.lua** file.

   a. Enter the following command to edit the casconfig_usermods.lua file. This example uses vi, but any text editor can be used.

   ```
   sudo vi /opt/sas/viya/config/etc/cas/default/casconfig_usermods.lua
   ```

   b. Add this value to the casconfig_usermods.lua file to enable SAS Data Connect Accelerator encryption on the CAS side.

   ```
   cas.DCTCPMENCRYPT='YES'
   ```

   CAUTION! The DCTCPMENCRYPT option is set on both the CAS server and on the data provider. How the option is set on both sides determines whether the data being transferred is encrypted or not. For more information, see "DCTCPMENCRYPT Option Setting Interaction" on page 57.

   c. Add the location of the TLS certificate(s) and private key file and password, if used, to the casconfig_usermods.lua file. The specific options that you use depend on the type of certificates.

   Here is an example. In this example, a password is not used.

   ```
   cas.DCSSLPVTKEYLOC="path-to-your-private-key"
   cas.DCSSLCERTLOC="path-to-your-id-cert"
   ```

   For more information about the options, see "CAS Configuration File Options for Data Transfer with the SAS Data Connect Accelerator" on page 135.

   d. Enter this command to restart the CAS controller. This restart is required to pick up the changes in the configuration file. How you run the following command depends on your operating system.

   - Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

   ```
   sudo systemctl restart sas-viya-cascontroller-default
   ```

   - Red Hat Enterprise Linux 6.x (or an equivalent distribution):

   ```
   sudo service sas-viya-cascontroller-default restart
   ```

2 On Teradata, modify the dcsecurity.properties file to enable SAS Data Connect Accelerator encryption.

   a Navigate to either the **/opt/SAS/ep/home/security/** directory (SAS Embedded Process version 16.0.0 or later) or the **/opt/SAS/SASTKInDatabaseServerForTeradata/*ep-version*/security/** directory (SAS Embedded Process version 15.0000 or earlier).

   b Change the DCTCPMENCRYPT option in the dcsecurity.properties file as follows:

   ```
   -DCTCPMENCRYPT YES
   ```

   CAUTION! The DCTCPMENCRYPT option is set on both the CAS server and on the data provider. How the option is set on both sides determines whether the data being transferred is encrypted or not. For more information, see "DCTCPMENCRYPT Option Setting Interaction" on page 57.

   c Add either the DCSSLCACERTDIR or DCSSLCALISTLOC option to the dcsecurity.properties file to specify either the location of the trusted certificate authorities or the public certificate(s) for trusted certificate authorities.

   Here is an example if the SAS Embedded Process is version 16.0.0 or later:

   ```
   -DCSSLCALISTLOC /opt/SAS/ep/home/security/certs/certs-filename.pem
   ```

   Here is an example if the SAS Embedded Process is version 15.0000 or earlier:

   ```
   -DCSSLCALISTLOC /opt/SAS/SASTKInDatabaseServerForTeradata/ep-version/security/certs/certs-filename.pem
   ```

   For more information about the options, see "dcsecurity.properties File Options for Data Transfer with the SAS Data Connect Accelerator" on page 138.

3 Copy the necessary TLS CA certificates to either the **/opt/SAS/ep/home/security/certs** directory (SAS Embedded Process version 16.0.0 or later) or the **/opt/SAS/SASTKInDatabaseServerForTeradata/*ep-version*/security/certs** directory (SAS Embedded Process version 15.0000 or earlier).

   - If your CA certificates already exist on the Teradata server, copy the CA certificates to this directory.
   - If your CA certificates exist on the CAS server, use a method of your choice to copy the CA certificates to this directory on the Teradata server.

   Here is an example if the SAS Embedded Process is version 16.0.0 or later:

   ```
   scp CASCA1.pem tdatuser@teramach1:/opt/SAS/ep/home/security/certs
   ```

   Here is an example if the SAS Embedded Process is version 15.0000 or earlier:

   ```
   scp CASCA1.pem tdatuser@teramach1:/opt/SAS/SASTKInDatabaseServerForTeradata/ep-version/security/certs
   ```

   For more information about the location of the CA certificates on the CAS server, see "(Optional) Deploy TLS Certificates" in *SAS Viya for Linux: Deployment Guide*.

   Note:

   - The CA certificates on the Teradata server must authorize the identity certificates that are specified on the CAS server.
   - All Teradata files and directories should have the `owner:group = tdatuser:tdatudf` setting.
   - The **/opt/SAS/ep/home/security** (version 16.0.0 or later) or the **/opt/SAS/SASTKInDatabaseServerForTeradata/*ep-version*/security** directory (version 15.0000 or earlier) should have `drwxr-xr-x` permissions.
   - The **/opt/SAS/ep/home/security/certs** (version 16.0.0 or later) or the **/opt/SAS/SASTKInDatabaseServerForTeradata/*ep-version*/security/certs** directory (version 15.0000 or earlier) should have `drwxr-xr-x` permissions.
   - The dcsecurity.properties file should have `-rwxr-xr-x` permissions.

4 Copy the contents of either the **/opt/SAS/ep/home/security/** directory (SAS Embedded Process version 16.0.0 or later) or the **/opt/SAS/SASTKInDatabaseServerForTeradata/*ep-version*/security/** directory (SAS Embedded Process version 15.0000 or earlier) to all nodes on the Teradata cluster.

   a Navigate to the **/opt/SAS** directory.

   ```
   cd /opt/SAS/
   ```

   b Create a compressed archive file of the security directory relative to the **/opt/SAS** directory.

   For SAS Embedded Process version 16.0.0 or later:

   ```
   tar cvof /tmp/sasep_security.tar ep/home/security
   ```

   For SAS Embedded Process version 15.0000 or earlier:

   ```
   tar cvof /tmp/sasep_security.tar SAS/SASTKInDatabaseServerForTeradata/ep-version/security
   ```

   c Create a script that contains the following lines:

   ```
   $ cat /tmp/sasep_extract.sh
   cd /opt/SAS/
   tar xvof /tmp/sasep_security.tar
   ```

   d Give the script a +x permission.

   ```
   $ chmod 755 /tmp/sasep_extract.sh
   ```

   e Do a parallel file transfer to push the files to all nodes.

   ```
   pcl -send /tmp/sasep_security.tar /tmp
   pcl -send /tmp/sasep_extract.sh /tmp
   ```

   f Use the parallel shell command to run the script to extract the TAR file.

   ```
   psh /tmp/sasep_extract.sh
   ```

   g Create a backup copy of the TAR file on a secured location.

   ```
   cp /tmp/sasep_security.tar /root/sasep_security.tar
   ```

   h Remove the TAR and script files from the nodes.

   ```
   psh rm /tmp/sasep_security.tar
   psh rm /tmp/sasep_extract.sh
   ```

5 Restart the SAS Embedded Process.

   a Disable the SAS Embedded Process to stop new queries from being started.

   ```
   CALL SQLJ.SERVERCONTROL ('SAS', 'disable', :A);
   ```

   b Query the status of the SAS Embedded Process until the status returns this message: `Hybrid Server is disabled with no UDFs running.`

   ```
   CALL SQLJ.SERVERCONTROL ('SAS', 'status', :A);
   ```

   c Shutdown the SAS Embedded Process.

   ```
   CALL SQLJ.SERVERCONTROL ('SAS', 'shutdown', :A);
   ```

   d Enable the SAS Embedded Process.

   ```
   CALL SQLJ.SERVERCONTROL ('SAS', 'enable', :A);
   ```

e Test the SAS Embedded Process. The SAS Embedded Process will start when the next SAS query that uses the SAS Embedded Process is sent to the database.

For more information about stopping and starting the SAS Embedded Process for Teradata, see Controlling the SAS Embedded Process.

Prerequisites When Enabling Encryption for the SAS Data Connect Accelerator for Hadoop (on SAS Viya)

Here are the prerequisites for enabling encryption for the SAS Data Connect Accelerator for Hadoop (on SAS Viya).

- Upgrade the OpenSSL package on all Hadoop nodes to 1.0.1g or later to support TLS.
- Install SAS/ACCESS Interface to Hadoop (on SAS Viya) and SAS In-Database Technologies for Hadoop (on SAS Viya).

  These offerings include the SAS Embedded Process and the SAS Data Connect Accelerator for Hadoop (on SAS Viya). For more information, see *SAS Viya for Linux: Deployment Guide*.
- Obtain TLS identity certificates (site-signed, third-party-signed, or self-signed) from the CAS controller machine. These certificates are located in the trustedcerts.pem file. Corresponding certificate authority (CA) certificates must be installed on the Hadoop nodes. If you use externally signed identity certificates in the CAS server, the Mozilla bundle of CA certificates that are provided by SAS can be deployed on the Hadoop nodes.

  For more information about the location of the trustedcerts.pem file, see "Encrypt Data Transfer When Using the SAS Data Connect Accelerator (Linux Full Deployment)" on page 50.

  For more information about configuring CAS, see "Configure CAS TLS to Use Custom Certificates (Linux Programming-Only Deployment)" on page 23 and "Configure CAS TLS to Use SAS Viya Default Certificates (Linux Programming-Only Deployment)" on page 28. Certificates, keys, and passwords produced for authenticating to the SAS Embedded Process for Hadoop might coincide with those produced for other clients of the CAS server, but they do not need to.
- When you installed the SAS Embedded Process, the following file and directory were created:

  For SAS Embedded Process version 16.0.0 and later:

  ```
  /opt/sas/ep/home/security/dcsecurity.properties
  /opt/sas/ep/home/security/certs
  ```

  For SAS Embedded Process version 15.0000 and earlier:

  ```
  /EPInstallDir/SASEPHome/security/dcsecurity.properties
  EPInstallDir/SASEPHome/security/certs
  ```

  - The **`/opt/sas/ep/home/security`** directory (version 16.0.0 or later) or the ***`/EPInstallDir/SASEPHome/security`*** directory (version 15.0000 or earlier) should have `drwxr-xr-x` permissions.
  - The **`/opt/sas/ep/home/security/certs`** directory (version 16.0.0 or later) or the ***`/EPInstallDir/SASEPHome/security/certs`*** directory (version 15.0000 or earlier) should have `drwxr-xr-x` permissions.
  - The dcsecurity.properties file should have `-rwxr-xr-x` permissions.

Enable Encryption for the SAS Data Connect Accelerator for Hadoop (on SAS Viya)

Follow these steps to encrypt data transfer between Hadoop and the CAS server using the SAS Data Connect Accelerator for Hadoop (on SAS Viya).

1. On the CAS server, modify the casconfig_usermods.lua configuration file to enable SAS Data Connect Accelerator encryption.

Note: This step is required only for programming-only deployments. If you performed a full deployment of SAS Viya, verify the current setting of cas.DCTCPMENCRYPT in the **/opt/sas/viya/config/etc/cas/default/casconfig.lua** file.

a Enter the following command to edit the casconfig_usermods.lua file.

```
sudo vi /opt/sas/viya/config/etc/cas/default/casconfig_usermods.lua
```

b Add this value to the casconfig_usermods.lua file to enable SAS Data Connect Accelerator encryption on the CAS side.

```
cas.DCTCPMENCRYPT='YES'
```

CAUTION! The DCTCPMENCRYPT option is set on both the CAS server and on the data provider. How the option is set on both sides determines whether the data being transferred is encrypted or not. For more information, see "DCTCPMENCRYPT Option Setting Interaction" on page 57.

c Add the location of the TLS certificate(s) and private key file and password, if used, to the casconfig_usermods.lua file. The specific options that you use depend on the type of certificates.

Here is an example. In this example, a password is not used.

```
cas.DCSSLPVTKEYLOC='path-to-your-private-key'
cas.DCSSLCERTLOC='path-to-id-cert'
```

For more information about the options, see "CAS Configuration File Options for Data Transfer with the SAS Data Connect Accelerator" on page 135.

d Enter this command to restart the CAS controller. This restart is required to pick up the changes in the configuration file. How you run the following command depends on your operating system.

- Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

```
sudo systemctl restart sas-viya-cascontroller-default
```

- Red Hat Enterprise Linux 6.x (or an equivalent distribution):

```
sudo service sas-viya-cascontroller-default restart
```

2 On Hadoop, modify the dcsecurity.properties file to enable SAS Data Connect Accelerator encryption.

a Navigate to either the **/opt/sas/ep/home/security/** directory (SAS Embedded Process version 16.0.0 or later) or the ***/EPInstallDir*/SASEPHome/security/** directory (SAS Embedded Process version 15.0000 or earlier).

b Change the DCTCPMENCRYPT option in the dcsecurity.properties file as follows:

```
-DCTCPMENCRYPT YES
```

CAUTION! The DCTCPMENCRYPT option is set on both the CAS server and on the data provider. How the option is set on both sides determines whether the data being transferred is encrypted or not. For more information, see "DCTCPMENCRYPT Option Setting Interaction" on page 57.

c Add either the DCSSLCACERTDIR or DCSSLCALISTLOC option to the dcsecurity.properties file to specify either the location of the trusted certificate authorities or the public certificate(s) for trusted certificate authorities.

Here is an example if the SAS Embedded Process is version 16.0.0 or later:

```
-DCSSLCALISTLOC /opt/sas/ep/home/security/certs/certs-filename.pem
```

Here is an example if the SAS Embedded Process is version 15.0000 or earlier:

```
-DCSSLCALISTLOC /EPInstallDir/SASEPHome/security/certs/certs-filename.pem
```

For more information about the options, see "dcsecurity.properties File Options for Data Transfer with the SAS Data Connect Accelerator" on page 138.

3 Copy the necessary TLS CA certificates to either the **/opt/sas/ep/home/security/certs** directory (SAS Embedded Process version 16.0.0 or later) or the ***/EPInstallDir*/SASEPHome/security/certs** directory (SAS Embedded Process version 15.0000 or earlier).

- If your CA certificates already exist on the Hadoop cluster, copy the TLS CA certificates to this directory.
- If your CA certificates exist on the CAS server, using a method of your choice, copy the CA certificates to this directory on the Hadoop cluster.

  In the following example, hdpclus1 is the name of the Hadoop cluster and the SAS Embedded Process version is 16.0.0 or later:

  ```
  scp CASCA1.pem username@hdpclus1:/opt/sas/ep/home/security/certs
  ```

  In the following example, hdpclus1 is the name of the Hadoop cluster and the SAS Embedded Process version is 15.0000 or earlier:

  ```
  scp CASCA1.pem username@hdpclus1:EPInstallDir/SASEPHome/security/certs
  ```

  For more information about the location of the trustedcerts.pem file, see "(Optional) Deploy TLS Certificates" in *SAS Viya for Linux: Deployment Guide*.

  Note:

  - The CA certificates on the Hadoop cluster must authorize the identity certificates that are specified on the CAS server.
  - The **/opt/sas/ep/home/security** directory (version 16.0.0 or later) or the ***/EPInstallDir/*SASEPHome/security** directory (version 15.0000 or earlier) should have `drwxr-xr-x` permissions.
  - The **/opt/sas/ep/home/security/certs** directory (version 16.0.0 or later) or the ***/EPInstallDir*/SASEPHome/security/certs** directory (version 15.0000 or earlier) should have `drwxr-xr-x` permissions.
  - The dcsecurity.properties file should have `-rwxr-xr-x` permissions.

4 Use the sasep-admin.sh script to copy the contents of the **security** directory to all nodes on the Hadoop cluster.

a Navigate to either the **/opt/sas/ep/home/bin/** directory (SAS Embedded Process version 16.0.0 or later) or the ***/EPInstallDir*/SASEPHome/bin/** directory (SAS Embedded Process version 15.0000 or earlier).

```
cd /opt/sas/ep/home/bin
....or....
cd /EPInstallDir/SASEPHome/bin
```

b Run the sasep-admin.sh script with the -security deploy argument.

```
./sasep-admin.sh -security deploy
```

This script deploys the SAS Data Connect Accelerator security settings to all nodes on the cluster. For more information, see "SASEP-ADMIN.SH Script" in *SAS Viya in Linux: Deployment Guide*.

Note: You can use `sasep-admin.sh –security deploy -force` to overwrite the current settings.

DCTCPMENCRYPT Option Setting Interaction

The DCTCPMENCRYPT option must be set for both the CAS server and the data provider. How the option is set on both sides determines whether the data being transferred is encrypted, the data is sent in plaintext, or the data transfer fails. The following table describes the interaction.

DCTCPMENCRYPT	CAS setting - YES	CAS setting - NO	CAS setting - OPT
Data provider setting - YES	Data transfer - encrypted	Data transfer - fails	Data transfer - encrypted
Data provider setting - NO	Data transfer - fails	Data transfer - plaintext	Data transfer - plaintext
Data provider setting - OPT	Data transfer - encrypted	Data transfer - plaintext	Data transfer - encrypted

You might want to use the OPT setting on the CAS server if you have more than one cluster set up as a client. If you want one cluster to use encrypted data transfer and one cluster to use plaintext, you would set the DCTCPMENCRYPT option of the first cluster to YES and the DCTCPMENCRYPT option of the second cluster to NO. You would then set the DCTCPMENCRYPT option of the CAS server to OPT.

Note: During deployment of SAS Viya, the DCTCPMENCRYPT option is set to OPT on the CAS server. You can change CAS server settings in the casconfig_usermods.lua file.

Updating the CAS Configuration File Options for Data Transfer

Note: This section applies only to a programming-only deployment.

You can check the current run-time data transfer encryption settings of the CAS server by using the CAS Server Monitor. The settings are on the Runtime Environment panel of the System State page. For more information about the CAS Server Monitor, see "Using CAS Server Monitor" in *SAS Viya Administration: SAS Cloud Analytic Services*.

CAS server options are stored in a configuration file. During deployment, the casconfig_deployment.lua is created in the **/opt/sas/viya/config/etc/cas/default** directory from content provided in the vars.yml file. When the sas-viya-casconrtroller-default service is started, the options in the Lua file are processed.

Changes to data transfer encryption options such as the DCTCPMENCRYPT option should be made in the casconfig_usermods.lua file.

For a complete list of options, see "CAS Configuration File Options for Data Transfer with the SAS Data Connect Accelerator" on page 135.

Updating the dcsecurity.properties File Options for Data Transfer

On Hadoop or Teradata, the file options for data transfer encryption are located in the dcsecurity.properties file. The dcsecurity.properties file is located in the following directory on your cluster:

- For Teradata, either the **/opt/SAS/ep/home/security/** directory (SAS Embedded Process version 16.0.0 or later) or the **/opt/SAS/SASTKInDatabaseServerForTeradata/*ep-version*/security** directory (SAS Embedded Process version 15.0000 or earlier).
- For Hadoop, either the **/opt/sas/ep/home/security/** directory (SAS Embedded Process version 16.0.0 or later) or the ***/EPInstallDir*/SASEPHome/security** directory (SAS Embedded Process version 15.0000 or earlier).

After you update the dcsecurity.properties file, copy the file to all nodes of the cluster.

- For Teradata, do a parallel file transfer to push the dcsecurity.properties file to all nodes.
- For Hadoop, use the sasep-admin.sh script to copy the contents of either the **/opt/sas/ep/home/security/** directory (SAS Embedded Process version 16.0.0 or later) or the ***/EPInstallDir*/SASEPHome/security** directory (SAS Embedded Process version 15.0000 or earlier) to all nodes on the Hadoop cluster. Run this command from either the **/opt/sas/ep/home/bin/** directory or the ***/EPInstallDir*/SASEPHome/bin** directory.

```
./sasep-admin.sh -security deploy
```

For a complete list of options, see "dcsecurity.properties File Options for Data Transfer with the SAS Data Connect Accelerator" on page 138.

Configure SAS 9.4 Clients to Work with SAS Viya

Configure a SAS 9.4 client to work with SAS Viya. From the SAS Viya deployment, you need the CA certificate that was used to sign the certificate that the CAS server is using. You also need the CA certificate that was used to sign the certificate that the Apache HTTP Server is using in order to interact with REST and Python.

Note: You must have SAS administrator privileges to import certificates from SAS Viya.

Perform the following tasks to locate the CA certificate that was used to sign the certificate that the CAS server is using and add that certificate to the certificate stores and truststores of the SAS 9.4 deployment.

1. On most SAS Viya deployments, the CA certificate files that CAS is using can be found as follows:
   - In a SAS Viya full deployment on Linux, you can use the vault-ca.crt file or the trustedcerts.pem file, located at **`/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts`**. The vault-ca.crt file contains two certificates. The first certificate is the SAS Viya root CA certificate issued by SAS Secrets Manager, and the second certificate is the SAS Viya intermediate CA certificate issued by SAS Secrets Manager. The trustedcerts files contain the trusted CA certificates and the intermediate certificates.
   - In a SAS Viya programming-only deployment on Linux, the trustedcerts files are located at **`/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts/trustedcerts`**.
   - In a SAS Viya Windows deployment, the trustedcerts files are located at **`C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\tls\cacerts\trustedcerts`**.
2. On a SAS 9.4 Windows client, import the SAS Viya root CA certificate and the intermediate certificate into the Window's certificate stores. These files have to be imported into the Window's certificate store one at a time. Because the vault-ca.crt file contains two files, the SAS Viya root CA certificate and the SAS Viya intermediate CA certificate, you need to create two unique files, one containing the root CA and the other containing the intermediate CA certificates. Use a text editor and cut and paste as appropriate.

   Each certificate in the file is denoted with a `-----BEGIN CERTIFICATE-----` and an `-----END CERTIFICATE-----` pair. Include the `-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----` header and footer in each of the two new files.

   Save these two files on your Windows machine and then add your certificates to the Windows CA store. You need to import the root certificate first and then the intermediate certificate. See "Import CA Certificates into the Windows Trusted Root Certificate Authorities Store" on page 72.
3. On a Linux 9.4m5 client, perform the following steps:
   a. Copy the SAS Viya CA certificates (vault-ca.crt or the trustedcerts.pem file) from the SAS Viya host to a location on your SAS 9.4 deployment where you can access the certificates. The directory structure where the SAS 9.4 trusted CA certificates (trustedcerts.pem or trustedcerts.jks) are found is at **`<SASHome>/SASSecurityCertificateFramework/1.1/cacerts`**.

      Note: Do not overwrite the existing trustedcerts files.

   b. Append the contents of the SAS Viya vault-ca.crt file (or the SAS Viya trustedcerts.pem file) to the end of the trustedcerts.pem file on the SAS 9.4 host. There are various ways to add your certificates to the trustedcerts.pem file:
      - Use the SAS Deployment Manager to add your certificates to the trusted CA bundle.
      - Use a text editor to add your certificates to the trustedcert.pem file.

   c. If you are using the December 2017 release of SAS 9.4M5 or later, you can skip this step.

Otherwise, on the Linux server, set environment variable CAS_CLIENT_SSL_CA_LIST= to the trust list that the client uses to connect to the server.

```
export CAS_CLIENT_SSL_CA_LIST=
'<SASHome>/SASSecurityCertificateFramework/1.1/cacerts/trustedcerts.pem'
```

If your SAS 9.4 client is SAS Studio, you can add the EXPORT statement to the sasenv_local file that is located at **/SASHome/SASFoundation/bin**.

Note:

In the December 2017 release of SAS 9.4M5, the CAS_CLIENT_SSL_CA_LIST= environment variable does not need to be set.

Perform the following tasks to locate the CA certificate that was used to sign the certificate that the Apache HTTP Server is using and add that certificate to the certificate stores and truststores of the SAS 9.4 deployment.

1. From the Apache HTTP Server deployed with SAS Viya, locate the CA certificate file that you need to copy to the SAS 9.4 deployment.

   Note: The certificate file might contain a root and intermediate certificates (chain file).

   a. If you have replaced the default self-signed CA certificates with your site-signed CA certificates, you can locate these certificates in the following directories.

      - On Red Hat Enterprise Linux and Equivalent Distributions, the certificate is located in **/etc/pki/tls/certs**.
      - On SUSE Linux Enterprise Server, the certificate is located in **/etc/apache2/ssl.crt**.

      For information about replacing the default certificates on the Apache HTTP Server, see "Update Apache HTTP Server TLS Certificates and Cryptography " on page 5.

   b. You can also use the CA certificate file that starts with the file name **httpproxy** located in **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts**. The certificate files complete name is dependent on the inventory name assigned to the host. For example, the complete name might be httpproxy-*inventoryname*-ca.crt.

2. On a SAS 9.4 Windows client, import the SAS Viya Apache HTTP root CA certificate and the intermediate certificates into the Window's certificate store. The certificate files have to be imported into the Window's certificate store one at a time. If you have copied a certificate chain file, you will need to create unique files for each certificate. Use a text editor and cut and paste as appropriate.

   Each certificate in the file is denoted with a `-----BEGIN CERTIFICATE-----` and an `-----END CERTIFICATE-----` pair. Include the `-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----` header and footer in each of the new files.

   Save these files on your Windows machine and then add all of the certificates to the Windows CA store.

3. On a SAS 9.4m5 Linux client, perform the following steps to add the SAS Viya Apache HTTP CA certificate to the SAS 9.4 trustedcerts.pem file.

   a. Copy the Apache HTTP CA certificate (site-signed CA certificate or httpproxy-*inventoryname*-ca.crt) from the SAS Viya host to a location on your SAS 9.4 deployment where you can access the certificates. The directory structure where the SAS 9.4 trusted CA certificates (trustedcerts.pem or trustedcerts.jks) are found is at **<SASHome>/SASSecurityCertificateFramework/1.1/cacerts**.

      Note: Do not overwrite the existing trustedcerts files.

   b. Append the contents of the Apache HTTP certificate file to the end of the trustedcerts.pem file on the SAS 9.4 host. There are various ways to add your certificates to the trustedcert.pem file:

      - Use the SAS Deployment Manager to add your certificates to the trusted CA bundle.
      - Use a text editor to add your certificates to the trustedcerts.pem file.

c If you are using the December 2017 release of SAS 9.4M5 or later, you can skip this step.

Otherwise, on the Linux server, set environment variable CAS_CLIENT_SSL_CA_LIST= to the trust list that the client uses to connect to the server.

```
export CAS_CLIENT_SSL_CA_LIST=
'<SASHome>/SASSecurityCertificateFramework/1.1/cacerts/trustedcerts.pem'
```

If your SAS 9.4 client is SAS Studio, you can add the EXPORT statement to the sasenv_local file that is located at **/SASHome/SASFoundation/bin**.

Note:

In the December 2017 release of SAS 9.4M5, the CAS_CLIENT_SSL_CA_LIST= environment variable does not need to be set.

Disable and Enable TLS (Linux Full Deployment)

Overview

To enable and disable TLS in a full deployment, you will at minimum need to update the vars.yml file to change the SECURE_CONSUL setting and disable ports using port families.

In SAS Viya, categories of ports known as port families were developed to allow a minimum number of settings to enable and disable broad categories of network traffic security using SAS Environment Manager. Port families are groupings of ports that share network usage characteristics. For example, for many customers, there is network and security infrastructure built around proxy servers and web applications. Machines running web applications frequently are managed as a group. This port family name is "web". Similarly, relational database servers are frequently grouped into similar network topologies and managed as a collective. This port family name is "databaseTraffic". SAS servers are frequently grouped together. That port family name is "sasData". Lastly, processes that monitor the health of the system, report usage statistics and control the configuration of the servers is port family "Server Control". See Table 6 on page 62.

The security of most ports is controlled by their corresponding port family configuration setting. However, a very small subset of ports do not use the port family configuration setting. TLS on these ports must be enabled and disabled differently.

- SAS Configuration Server ports are controlled by settings in the vars.yml file. The SECURE_CONSUL and the DISABLE_CONSUL_HTTP_PORT settings are TRUE by default. For more information, see “Enable or Disable TLS on the SAS Configuration Server Ports” on page 64.
- SAS Cloud Analytic Services inter-node communications are not secured by default, but can be enabled or disabled through configuration changes. See “Configure CAS Internode TLS (Linux Full Deployment)” on page 35.
- SAS Secrets Manager ports are always secured.
- For encryption in-motion with SAS Embedded Process, in a SAS Viya full-deployment, the configuration on the CAS side is complete by default. However, the SAS Embedded Process configuration, on either Hadoop or Teradata, must be updated to enable it. The DCTCPMENCRYPT option defines if encryption is used. See “Encrypt Data Transfer When Using the SAS Data Connect Accelerator (Linux Full Deployment)” on page 50.

For a list of ports in SAS Viya, see “Enable Required Ports” in *SAS Viya for Linux: Deployment Guide*.

Enable or Disable TLS Using Port Families

Information included in these sections describe what port families are configured by default and how those default port family configurations can be changed.

Port Families

The port families shown in Table 6 on page 62 are secured using TLS by default. You can enable or disable network security traffic for TLS using categories (families) of ports.

Table 6 *TLS Port Families - Enabled by Default*

Family Name	Description	Ports That Can Be Controlled
databaseTraffic	Port family that needs to control traffic to database servers that might be located on different network segments.	SAS Infrastructure Data Server (PostgreSQL), EP Data Connectors
sasData	Port family that controls traffic transporting data to SAS servers. The SAS Workspace Server and the SAS Object Spawner use this port family to enable and disable AES encryption at start-up.	CAS client, SAS Compute Server, SAS/CONNECT Server, SAS/CONNECT Spawner, SAS Event Stream Processing (ESP) Server, CAS Server Monitor SAS Workspace Server, SAS Object Spawner
serverControl	Port family that controls traffic sent between clustered servers to maintain the cluster.	SAS Launcher Server
web	Port family that enables and disables internal traffic routed to Apache where HTTP is used instead of HTTPS. External traffic to the HTTPS end-point is not affected.	Apache HTTP Server, all web apps and microservices, SAS Cache Locator (Apache Geode), SAS Message Broker (RabbitMQ), CAS Rest, ESP App, SAS Studio, CAS Server Monitor

Use Ansible to Enable or Disable TLS Port Families Pre-deployment

TLS is enabled by default. If you want to disable TLS across all ports prior to deployment, add the following to the sitedefault.yml file. For information about sitedefault.yml, see "Bulk Loading of Configuration Values (sitedefault.yml)" in *SAS Viya Administration: Configuration Properties*.

Note: Keep the indention.

```
config:
    application:
        sas.security:
            network.web.enabled: false
            network.sasData.enabled: false
            network.databaseTraffic.enabled: false
            network.serverControl.enabled: false
```

Customers can use SAS Environment Manager to override the default behavior by altering the sas.security/network settings.

Use SAS Environment Manager to Enable and Disable TLS Port Families Post-Deployment

To alter the port family settings using the SAS Environment Manager, edit the sas.security network settings.

Note: You need to turn off all four port families if you are disabling TLS across the deployment. This example turns off just one.

1 From the applications menu (☰), select **Manage Environment**.

2 From the side menu, click .

The Configuration page is an advanced interface. It is available only to SAS Administrators.

3 The default view is **Basic Services**. Select **Definitions** from the drop-down box.

4 In the **Definitions** list, filter on **sas.security**. Select **sas.security**.

5 The sas.security/network.web.enabled property is for the port family for which you want to turn off web-enabled TLS.

a Locate the **network.web.enabled** property. At the right corner of the Configuration window, click and set **network.web.enabled** to false.

Click **Save**.

Note: The system takes a few minutes to recognize the new key before starting to use the new key.

false Disables TLS for the port family.

true Enables TLS for the port family.

For a description of the properties, see "Configuration Properties: Reference (System)" in *SAS Viya Administration: Configuration Properties*.

6 Restart all services on all machines

Note: On a multiple-machine deployment, there is a sequence for starting and stopping SAS Viya servers and services. See "General Servers and Services: Operate (Linux)" in *SAS Viya Administration: General Servers and Services*.

```
sudo /etc/init.d/sas-viya-all-services stop
sudo /etc/init.d/sas-viya-all-services start
```

Programmatically Enable or Disable TLS Port Families

Using SAS Environment Manager to disable or enable TLS on port families is the preferred method. However, you can enable or disable the TLS ports programmatically by using the following sas-bootstrap-config commands. The SAS Bootstrap Config CLI must establish trust for the TLS handshake to proceed and allow secure communication. To establish trust, the truststore must be specified as an environment variable. Sourcing the consul.conf sets the SSL_CERT_FILE environment variable to the trusted certificates.

1 Log on to the SAS Configuration Server (Consul) as a user with root or sudo privileges.

2 To see whether the setting for a particular port family is enabled or disabled, you can enter the following commands. In this example, we want to see whether the TLS port family for PostgreSQL is enabled or disabled. PostgreSQL uses the network.databaseTraffic.enabled port family value and is set to true if TLS is enabled.

```
. /opt/sas/viya/config/consul.conf
/opt/sas/viya/home/bin/sas-bootstrap-config --token-file
/opt/sas/viya/config/etc/SASSecurityCertificateFramework/
tokens/consul/default/client.token kv read --recurse config|
grep network.databaseTraffic.enabled.
```

3 The following sas-bootstrap-config commands are used to disable TLS on the various port families. The port families are set to false to disable TLS on the port family. Log on to the SAS Configuration Server (Consul) as a user with root or sudo privileges.

```
. /opt/sas/viya/config/consul.conf
/opt/sas/viya/home/bin/sas-bootstrap-config
--token-file /opt/sas/viya/config/etc/SASSecurityCertificateFramework/
```

```
tokens/consul/default/client.token kv write --force --site-default
config/application/sas.security/network.databaseTraffic.enabled false

/opt/sas/viya/home/bin/sas-bootstrap-config
--token-file /opt/sas/viya/config/etc/SASSecurityCertificateFramework/
tokens/consul/default/client.token kv write --force --site-default
config/application/sas.security/network.sasData.enabled false

/opt/sas/viya/home/bin/sas-bootstrap-config  --token-file
/opt/sas/viya/config/etc/SASSecurityCertificateFramework/
tokens/consul/default/client.token kv write --force --site-default
config/application/sas.security/network.serverControl.enabled false

/opt/sas/viya/home/bin/sas-bootstrap-config --token-file
/opt/sas/viya/config/etc/SASSecurityCertificateFramework/
tokens/consul/default/client.token kv write --force --site-default
config/application/sas.security/network.web.enabled false
```

Note: TLS settings for the High Availability (HA) PostgreSQL cluster can be enabled and disabled at the global level for all clusters. Individual clusters can override the global setting by adding a cluster-specific key as shown in the following example:

```
. /opt/sas/viya/config/consul.conf
/opt/sas/viya/home/bin/sas-bootstrap-config
--token-file /opt/sas/viya/config/etc/SASSecurityCertificateFramework/
tokens/consul/default/client.token kv write --force --site-default
config/<CLUSTER_NAME>/sas.security/network.databaseTraffic.enabled false
```

4 Restart all services on all machines

Note: On a multiple-machine deployment, there is a sequence for starting and stopping SAS Viya servers and services. See "General Servers and Services: Operate (Linux)" in *SAS Viya Administration: General Servers and Services*.

```
sudo /etc/init.d/sas-viya-all-services stop
sudo /etc/init.d/sas-viya-all-services start
```

Enable or Disable TLS on the SAS Configuration Server Ports

To disable TLS across the deployment, in addition to turning off port families, you must also disable settings in SAS Configuration Server (Consul) . Consul ports are controlled by settings in the vars.yml file. In the file, the setting SECURE_CONSUL must be set to FALSE to completely disable TLS on the deployment. See "Use Ansible to Enable or Disable TLS Port Families Pre-deployment" on page 62.

Two settings control the HTTP and HTTPS ports for Consul. The two settings are configured as follows:

SECURE_CONSUL	is set to TRUE by default. This setting enables port 8501 with HTTPS. If you set SECURE_CONSUL to FALSE, only the unsecured HTTP port (8500) is available.
DISABLE_CONSUL_HTTP_PORT	is set to TRUE by default. This setting disables port 8500. If you set DISABLE_CONSUL_HTTP_PORT to FALSE, both the HTTP port (8500) and the HTTPS port (8501) are available.

The following instructions provide an example of how to disable TLS on Consul.

Note: To enable TLS on Consul, simply set SECURE_CONSUL=TRUE and continue the following steps.

1 Edit the vars.yml file and set SECURE_CONSUL=FALSE.

2 Run the Ansible playbook in its entirety. See "Deploy the Software" in *SAS Viya for Linux: Deployment Guide*.

```
sudo -u userid ansible-playbook -i inventory.ini site.yml
```

3 Stop and restart all services on all machines

Note: On a multiple-machine deployment, there is a sequence for starting and stopping SAS Viya servers and services. See "General Servers and Services: Operate (Linux)" in *SAS Viya Administration: General Servers and Services*.

```
sudo /etc/init.d/sas-viya-all-services stop
sudo /etc/init.d/sas-viya-all-services start
```

Note: When SAS Message Broker (RabbitMQ) is stopped, the epmd mapper process is terminated. However, the epmd process is restarted when you issue status checks. This status might mislead you to think that stopping RabbitMQ did not work when you stopped all of the processes. If you issue a status check to verify that all processes have been stopped and see that the epmd process is running, you can manually terminate it again if you think that it is necessary. To stop the epmd process, issue the following command:

```
${SASHOME}/bin/epmd -kill
```

Manage Truststores

Add Certificates to Truststores Using Ansible (Linux Full Deployment)

In a SAS Viya full deployment, you can use Ansible to add a new CA certificate to the truststores.

1 On the main Consul machine (the machine listed within the [consul] host group in the inventory.ini file), edit the sitedefault.yml file. That file is located at **/opt/sas/viya/config/etc/consul.d/default**.

Add the Base64 PEM encoded certificate to sitedefault.yml.

Note: Keep the indention of this file.

```
cacerts:
    sascaroot: |
        -----BEGIN CERTIFICATE-----
        certificate string
        -----END CERTIFICATE-----
    sassha2rootca: |
        -----BEGIN CERTIFICATE-----
         certificate string
        -----END CERTIFICATE-----
    MyRootCA: |
        -----BEGIN CERTIFICATE-----
        MIIFZTCCA02gAwIBAgIQIbEH2kWtB5NBMg9WdJc0jzANBgkqhkiG9w0BAQsFADBF
        MRMwEQYKCZImiZPyLGQBGRYDY29tMRMwEQYKCZImiZPyLGQBGRYDc2FzMRkwFwYD
        ...
        -----END CERTIFICATE-----
```

2 Restart the SAS Configuration Server to pick up the changes to the sitedefault.yml file. This act copies the changes to the CA bundle in the sitedefault.yml file to Consul's key-value (KV) store. Only new key-value pairs are added. Existing key-value pairs are not updated. How you run the following command depends on your operating system.

- Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

```
sudo systemctl restart sas-viya-consul-default
```

- Red Hat Enterprise Linux 6.x (or an equivalent distribution):

```
sudo service sas-viya-consul-default restart
```

3 Use the Ansible playbook to rebuild the truststores on each machine in the deployment. On an Ansible controller machine, run the rebuild-trust-stores.yml Ansible play. This act incorporates the updated CA bundle of trusted certificates (from Consul configuration) into the various truststores (trustedcerts.pem and trustedcerts.jks) on each machine in the SAS Viya deployment.

```
sudo ansible-playbook -i inventory.ini
./utility/rebuild-trust-stores.yml
```

4 Add the sitedefault.yml content from Step 1 on page 65 to the sitedefault.yml that forms part of the Ansible playbooks in **/roles/consul/files/sitedefault.yml**.

The new CA certificates are added to all truststores on all hosts. When the deployment playbook is rerun later, the correct content will remain in the truststores.

For more information, see "Deploy the Software" in *SAS Viya for Linux: Deployment Guide*.

Remove Certificates from the Truststores Using Ansible (Linux Full Deployment)

In a SAS Viya full deployment, you can use Ansible to remove CA certificates from the truststores. For example, you might want to remove an expired CA certificate.

1 On the main Consul machine (the machine listed within the [consul] host group in inventory.ini file), edit the sitedefault.yml file if you added that certificate to the file. The sitedefault.yml file is located at **/opt/sas/viya/config/etc/consul.d/default** and **/roles/consul/files/sitedefault.yml**. Remove the expired certificate from both files.

   Remove the Base64 PEM encoded certificate from the `cacerts` section of the file. Locate the certificate that you want to remove. For example, we are removing the certificate that we labeled MyRootCA. Remove the following highlighted rows.

   Note: Keep the indention of this file.

```
cacerts:
    sascaroot: |
        -----BEGIN CERTIFICATE-----
        certificate string
        -----END CERTIFICATE-----
    sassha2rootca: |
        -----BEGIN CERTIFICATE-----
         certificate string
        -----END CERTIFICATE-----
    MyRootCA: |
        -----BEGIN CERTIFICATE-----
        MIIFZTCCA02gAwIBAgIQIbEH2kWtB5NBMg9WdJc0jzANBgkqhkiG9w0BAQsFADBF
        MRMwEQYKCZImiZPyLGQBGRYDY29tMRMwEQYKCZImiZPyLGQBGRYDc2FzMRkwFwYD
        ...
        -----END CERTIFICATE-----
```

2 To delete certificates, you must remove the certificates and keys from Consul using sas-bootstrap-config commands. The SAS Bootstrap Config CLI must establish trust for the TLS handshake to proceed and allow secure communication. To establish trust, the truststore must be specified as an environment variable. Sourcing the consul.conf sets the SSL_CERT_FILE environment variable to the trusted certificates.

   Note: The following commands should be run as a root or sudo user.

   The code is shown on more lines for display purposes only. This command should be on one line and should not contain line breaks.

a Set the Consul access token in the CONSUL_HTTP_TOKEN environment variable. This command needs to be performed before executing any utilities or services that might access Consul.

```
. /opt/sas/viya/config/consul.conf
export CONSUL_HTTP_TOKEN=$(sudo cat /opt/sas/viya/config/etc/SASSecurityCertificateFramework
/tokens/consul/default/client.token)
```

b To remove the internal CA certificate (`MyRootCA`) that you added previously, submit the following commands:

```
. /opt/sas/viya/config/consul.conf
/opt/sas/viya/home/bin/sas-bootstrap-config kv delete cacerts/MyRootCA
```

3 Restart the SAS Configuration Server to pick up the changes to the sitedefault.yml file. Only new key-value pairs are added. Existing key-value pairs are not updated. How you run the following command depends on your operating system.

- Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

```
sudo systemctl restart sas-viya-consul-default
```

- Red Hat Enterprise Linux 6.x (or an equivalent distribution):

```
sudo service sas-viya-consul-default restart
```

4 Use the Ansible playbook to rebuild the truststores on each machine in the deployment. On an Ansible controller machine, run the rebuild-trust-stores.yml Ansible play. This act incorporates the updated CA bundle of trusted certificates (from Consul configuration) into the various truststores (trustedcerts.pem and trustedcerts.jks) on each machine in the SAS Viya deployment.

```
sudo ansible-playbook -i inventory.ini ./utility/rebuild-trust-stores.yml
```

The new CA certificates are now removed from all of the truststores on all hosts. When the deployment playbook is rerun later, the correct content will remain in the truststores.

Note: If the deployment playbook is rerun later to update the license, the correct content remains in the truststores when you follow this process.

Replace Certificates in the Truststores Using Ansible (Linux Full Deployment)

In a SAS Viya full deployment, you can use Ansible to replace CA certificates in the truststores. For example, you might have a CA certificate that has been renewed.

1 On the main Consul machine (the machine listed within the [consul] host group in the inventory.ini file), edit the sitedefault.yml file. The sitedefault.yml file is located at **`/opt/sas/viya/config/etc/consul.d/default`** and **`/roles/consul/files/sitedefault.yml`**. Update the certificate in both files.

Update the base64 PEM encoded certificate in the `cacerts` section of the file. For our example, we updated the certificate that was labeled MyRootCA.

Note: Keep the indention of this file.

```
cacerts:
    sascaroot: |
        -----BEGIN CERTIFICATE-----
        certificate string
        -----END CERTIFICATE-----
    sassha2rootca: |
        -----BEGIN CERTIFICATE-----
         certificate string
        -----END CERTIFICATE-----
    MyRootCA: |
```

```
        -----BEGIN CERTIFICATE-----
        MIIFZTCCA02gAwIBAgIQIbEH2kWtB5NBMg9WdJc0jzANBgkqhkiG9w0BAQsFADBF
        MRMwEQYKCZImiZPyLGQBGRYDY29tMRMwEQYKCZImiZPyLGQBGRYDc2FzMRkwFwYD
        ...
        -----END CERTIFICATE-----
```

2 To update the certificates, you must first remove the certificates and keys from Consul that you want to replace or update. The SAS Bootstrap Config CLI must establish trust for the TLS handshake to proceed and allow secure communication. To establish trust, the truststore must be specified as an environment variable. Sourcing the consul.conf sets the SSL_CERT_FILE environment variable to the trusted certificates.

Note: The following commands should be run as a root or sudo user.

The code is shown on more lines for display purposes only. This command should be on one line and should not contain line breaks.

a Set the Consul access token in the CONSUL_HTTP_TOKEN environment variable. This command needs to be performed before executing any utilities or services that might access Consul.

```
. /opt/sas/viya/config/consul.conf
export CONSUL_HTTP_TOKEN=$(sudo cat /opt/sas/viya/config/etc/SASSecurityCertificateFramework
/tokens/consul/default/client.token)
```

b To remove the internal CA certificate (`MyRootCA`), submit the following command. You are removing the existing key-value pair from the `cacerts` tree in order for the new key-value pair to be added.

```
. /opt/sas/viya/config/consul.conf
/opt/sas/viya/home/bin/sas-bootstrap-config kv delete cacerts/MyRootCA
```

3 Restart the SAS Configuration Server to pick up the changes to the sitedefault.yml file. Only new key-value pairs are added. Existing key-value pairs are not updated. How you run the following command depends on your operating system.

- Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

```
sudo systemctl restart sas-viya-consul-default
```

- Red Hat Enterprise Linux 6.x (or an equivalent distribution):

```
sudo service sas-viya-consul-default restart
```

4 Use the Ansible playbook to rebuild the truststores on each machine in the deployment. On an Ansible controller machine, run the rebuild-trust-stores.yml Ansible play. This act incorporates the updated CA bundle of trusted certificates (from Consul configuration) into the various truststores (trustedcerts.pem and trustedcerts.jks) on each machine in the SAS Viya deployment.

```
sudo ansible-playbook -i inventory.ini ./utility/rebuild-trust-stores.yml
```

The new CA certificates are now updated on all of the truststores on all hosts. When the deployment playbook is rerun later, the correct content will remain in the truststores.

Note: If the deployment playbook is rerun later to update the license, the correct content remains in the truststores when you follow this process.

Add Certificates to or Remove Certificates from the Truststore or a Certificate Chain Manually

Add Certificates to the Trustedcerts Files

SAS provides a trusted list of root CA certificates at installation. This trusted list includes the Mozilla bundle of CA certificates, the default Apache httpd certificates, and the CA certificates issued by SAS Secrets Manager

(only in a SAS Viya full deployment). There are two files named trustedcerts that contain the trusted list of certificates, trustedcerts.pem and trustedcerts.jks.

- In a SAS Viya deployment on Linux, the trusted CA certificates are found at **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts**.
- In a SAS Viya deployment on Windows, the trusted CA certificates are found at **C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\cacerts**

You can add additional root CA certificates to the truststore or delete certificates from the truststore for the file format needed, JKS or PEM.

You can add your CA root certificates, self-signed certificates, or chained certificates to the trustedcerts.pem file. You can also delete the certificates that need to be deleted.

1 Use a text editor to add certificates to or remove certificates from the trustedcerts.pem file. Locate the certificates that you want to delete from or add to the trustedcerts.pem file. Here is an example template of certificates that a trustedcerts.pem file might contain.

```
<PEM encoded companycaroot>
-----BEGIN CERTIFICATE-----
certificate string
-----END CERTIFICATE-----
<PEM encoded companysha2rootca>
-----BEGIN CERTIFICATE-----
certificate string
-----END CERTIFICATE-----
<PEM encoded digicertrootca>
-----BEGIN CERTIFICATE-----
certificate string
-----END CERTIFICATE-----
<PEM encoded SASViyalocalhost>
-----BEGIN CERTIFICATE-----
certificate string
-----END CERTIFICATE-----
<PEM encoded customer-chain>
-----BEGIN CERTIFICATE-----
certificate string
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
certificate string
-----END CERTIFICATE-----
```

The content of the digital certificate in this example is represented as `<PEM encoded certificate>`. The content of each digital certificate is delimited with a `-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----` pair. All text outside the delimiters is ignored. Therefore, you might not want to use delimited lines for descriptive comments.

You can also concatenate the certificate authority files. For example, you can concatenate a root authority certificate file and a primary certificate file into a single PEM file. Here are two examples of concatenating certificates. The customer-chain.crt file has two certificates in the file. The vault-ca.crt file has two certificates in the file.

On Linux, use cat to concatenate.

```
cat customer-chain.crt >> trustedcerts.pem
cat vault-ca.crt >> trustedcerts.pem
```

On Windows, use type to concatenate.

```
type customer-chain.crt >> trustedcerts.pem
```

```
type vault-ca.crt >> trustedcerts.pem
```

Note: You can place these files in any order.

2 Because the digital certificate is encoded, it is unreadable. To view the file contents, you can use the following OpenSSL commands for your file type:

```
openssl x509 -in /opt/sas/viya/config/etc/
SASSecurityCertificateFramework/cacerts/trustedcerts.pem -text -noout
```

```
openssl x509 -in
C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\cacerts -text -noout
```

3 Place the new CA certificates that you added to the truststore in the **cacerts** directory. Also add all of your certificates, chain certificates, and private keys to the following directories:

- On a Linux programming-only deployment:
  - **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts**
  - **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tls/certs**
  - **opt/sas/viya/config/etc/SASSecurityCertificateFramework/private**
- On a Linux full deployment:
  - **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts**
  - **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tls/certs/cas/shared/default**
  - **• opt/sas/viya/config/etc/SASSecurityCertificateFramework/private/cas/shared/default**
- On Windows:
  - **C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\cacerts**
  - **C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\tls\certs**
  - **C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\private**

The following steps show how to add certificates to and remove certificates from trustedcerts.jks file (the Java truststore). Use the `keytool` command to add the certificates to the Java truststore.

Note: For more information about the keytool command, see keytool - Key and Certificate Management Tool.

1 Locate the default truststore for your Java applications.

- In a SAS Viya deployment on Linux, the trusted certificates are found at **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts/trustedcerts.jks**.
- In a SAS Viya deployment on Windows, the trusted CA certificates are found at **C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\cacerts\trustedcerts.jks**

2 When you import the CA certificates into the default trust store, add the root certificate first. Repeat these steps for the client certificates.

a On Linux, use the following commands to import a root CA certificate (customerCA.crt in our example) into the default truststore:

```
$ keytool -importcert -file
/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts/customerCA.crt
-alias customerCA -keystore
/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts/trustedcerts.jks
```

b On Windows, use the following keytool commands to import a root CA (customerCA.crt) certificate and a Client certificate into the default truststore (trustedcerts.jks):

Note: Keytool might be in a different directory.

```
C:\Program Files\Java\Java-version\bin\keytool -importcert -file
C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\cacerts\
customerCA.crt -alias customerCA -keystore
C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\cacerts\trustedcerts.jks
```

3 Delete the CA certificates from the default truststore. Repeat these steps for the client certificates.

a On Linux, use the following commands to delete a root CA certificate (customerCA.crt in our example) from the default truststore. Use the following commands to remove a root CA certificate (customerCA.crt in our example) from the default truststore:

```
keytool -delete -alias customerCA -keystore
/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts/trustedcerts.jks
```

b On Windows, use the following keytool commands to delete a root CA (customerCA.crt) certificate and a Client certificate from the default truststore (trustedcerts.jks):

Note: Keytool might be in a different directory.

```
C:\Program Files\Java\Java-version\bin\keytool -delete -alias customerCA -keystore
C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\cacerts\trustedcerts.jks
```

Note: Do not delete the trustedcerts files.

Note: Add your root certificate to the trustedcerts.pem and trustedcerts.jks files on every machine in the deployment. Also delete your root certificate from every machine in the deployment.

For more information about how to manage your certificates and protect your keys, see "Manage Certificates and Generate New Certificates" on page 74.

Add Certificates to the Certificate Chain

The list of TLS certificates, from the root certificate to the end-user certificate, represents the TLS certificate chain. We take the server identity certificates and chain the intermediate certificates with them. In a SAS Viya deployment, these chained certificate files are placed in the **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tls/certs** directory.

For detailed information about how to manage your certificates and protect your keys, see "Manage Certificates and Generate New Certificates" on page 74.

The following steps show how to add your intermediate certificates to the server identity certificates file to create a chain:

1 You can use a text editor to add your certificates to a file in PEM format called server.crt.

Chaining files contain the server certificate first, the intermediate certificate that validates the server certificate next, the intermediate certificate that validates the first intermediate certificate next, and so on down the chain all the way to the root certificate. The root certificate does not need to be in this file.

Here is a template of what a chained server.crt file might contain.

```
(Your Server Id Certificate)
 - - - - -BEGIN CERTIFICATE- - - - -
<PEM encoded certificate>
- - - - -END CERTIFICATE- - - - -
(Intermediate Certificate(s))
 - - - - -BEGIN CERTIFICATE- - - - -
<PEM encoded certificate>
  - - - - -END CERTIFICATE- - - - -
(Intermediate Certificate(s))
 - - - - -BEGIN CERTIFICATE- - - - -
<PEM encoded certificate>
```

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

The content of the digital certificate in this example is represented as `<PEM encoded certificate>`. The content of each digital certificate is delimited with a-----`BEGIN CERTIFICATE`----- and -----`END CERTIFICATE`----- pair. All text outside the delimiters is ignored. Therefore, you might not want to use delimited lines for descriptive comments.

You can also concatenate the certificate files. For example, you can concatenate an intermediate authority certificate file and a server certificate file into a single PEM file. Here is an example of concatenating certificates.

On Linux, use cat to concatenate.

```
cat int_ca.crt >> server_id.crt
```

On Windows, use type to concatenate.

```
type int_ca.crt >> server_id.crt
```

2 Because the digital certificate is encoded, it is unreadable. To view the file contents, you can use the following OpenSSL commands for your file type:

```
openssl x509 -in /config/etc/SASSecurityCertificateFramework/
tls/certs/server_id.crt -text -noout
```

Import CA Certificates into the Windows Trusted Root Certificate Authorities Store

Import your CA certificates into the Windows Trusted Root Certificate Authorities store using the Microsoft Management Console (MMC).

Note: The instructions shown here may vary slightly depending on which version of Windows you are using. See the official Microsoft Windows documentation.

1 Start the Microsoft Management Console (MMC). Right-click the Windows **Start** menu and select **Run**.

2 In the Run window, enter MMC, and press **OK**.

3 From the Microsoft Management Console window, click **File** and **Add/Remove Snap-in** from the drop-down menu.

   a In the Add or Remove snap-ins window, select **Certificates** from the **Available snap-ins** list.

   b Add **Certificates** to the **Selected snap-ins** list.

   c Select **Computer account** and click **Next**.

   d Select **Local computer** (the computer that this console is running on). Click **Finish**.

   e Click **OK**.

4 Expand the **Certificates (Local Computer)** list and click **Trusted Root Certification Authorities**.

5 Select the **Certificates** node to view a list of certificates.

6 From the left pane, right-click **Certificates**. If you are importing CA certificates, right-click the **Certificates** node within the **Certificates (Local Computer)**, **Trusted Root Certification Authorities**, **Certificates** hierarchy.

   Select **All Tasks**, and then **Import** from the drop-down menu.

7 From the Certificate Import Wizard, confirm that **Local Machine** is selected for the **Store Location**, and click **Next.**

a Select **Browse** to locate the file to import. Expand the list of file types and select **All Files** from the drop-down menu.

b From the list of files, select the certificate that you want to import into the trust store. Click **Next**.

c From the **Certificate Store** page of the Certificate Import Wizard, select **Place all certificates in the following store**. Place the certificates in the Certificate store for the Trusted Root Certification Authorities. Click **Next**.

d From the **Completing the Certificate Import Wizard** page of the Certificate Import Wizard, click **Finish**.

e Verify that you receive the message `The import was successful`, and click **OK**.

8 Observe that the imported certificate is now listed in the Microsoft Management Console. In the Console window, expand Trusted Root Certification Authorities to make sure that the certificate that you imported is listed.

9 Repeat the steps for any CA intermediate certificates.

Import the Client Certificate into the Windows Personal Machine Store

Import the client certificates into the Windows personal store on the local machine using the Microsoft Management Console (MMC).

1 Start the Microsoft Management Console (MMC). Right-click the Windows **Start** menu and select **Run**.

2 In the Run window, enter MMC, and press **OK**.

3 From the Microsoft Management Console window, click **File** and **Add/Remove Snap-in** from the drop-down menu.

a In the Add or Remove snap-ins window, select **Certificates** from the **Available snap-ins** list.

b Add **Certificates** to the **Selected snap-ins** list.

c Select **Computer account** and click **Next**.

d Select **Local** computer (the computer that this console is running on). Click **Finish**.

e Click **OK**.

4 Expand the **Certificates (Local Computer)** list and click **Personal**.

5 From the left pane, right-click and select the **Certificates** node within the **Certificates (Local Computer)**, **Personal**, **Certificates** hierarchy to view a list of certificates. Select **All Tasks**, and then **Import** from the drop-down menu.

6 From the Certificate Import Wizard, confirm that **Local Machine** is selected for the **Store Location**, and click **Next**.

a Select **Browse** to locate the file to import. Expand the list of file types and select **All Files** from the drop-down menu.

b From the list of files, select the certificates that you want to import. In our example, we are selecting the client certificate in PFX format that contains the certificate and private key file, customerCert.pfx. Click **Next**.

c From the **Private Key Protection** page of the Certificate Import Wizard, enter the password for customerCert.pfx. Select **Include all extended properties**. Do not select **Enable strong private key protection** and do not select **Mark this key as exportable**. Click **Next**.

d From the**Certificate Store** page of the Certificate Import Wizard, select **Place all certificates in the following store**. Place the certificates in the Personal Certificate store. Click **Next**.

e From the **Completing the Certificate Import Wizard** page, click **Finish**.

f Verify that you receive the message `The import was successful`, and click **OK**.

7 Observe that the imported certificates are now listed in the Microsoft Management Console.

Grant Read Permission to Authenticated Users for the Client Certificate's Private Key

In order to use the client's private key, the client certificate's private key must be readable. Perform the following tasks to grant Read permission to the authenticated users who will use the client certificate's private key.

1 Start the Microsoft Management Console (MMC). Right-click the Window's **Start** menu and select **Run**.

2 From the Run window, type MMC, and press **OK**.

3 Right-click the client certificate that you recently imported. See "Import the Client Certificate into the Windows Personal Machine Store" on page 73.

4 On the pop-up menu, select **All Tasks**, **Manage Private Keys**. A Permissions window appears.

5 From the Permissions window, within the **Security** tab, add **Authenticated Users**.

6 Ensure that authenticated users have Read permission. Select **Allow Read item**, deselect **Allow Full control**, and deselect **Allow Special permissions**. Click **OK**.

Manage Certificates and Generate New Certificates

Use Best Practices to Create and Manage Certificates

SAS recommends the following best practices when you are creating certificates and when managing certificates and securing your private keys.

- When generating new certificates, provide the following information for the certificate signing request:
  - Provide fixed host names (required by the SAS Viya environment). On Linux, you can use the `'hostname -f'` command to discover the hostname and provide that name when generating a certificate.
  - Provide fully qualified domain names (FQDN) that are 64 characters or fewer in length. One of the specifications for the certificate revocation list is a 64-character limit for the common name (CN) attribute. For more information, see RFC 5280.
  - Provide subject alternative names (SAN), including IP addresses. To use OpenSSL to generate new certificates, see "Create Certificates with SANs Using OpenSSL" on page 76.
  - For multi-tenancy on Linux, ensure that the certificates contain subject alternate names for each tenant or use a wildcard for the subdomain. For more information on multi-tenant DNS naming, see "Additional Requirements for Multi-tenancy" in *SAS Viya for Linux: Deployment Guide*. For an example where wildcards are specified for multi-tenancy, see the certificate signing request (CSR) conf file at "Create Certificates with SANs Using OpenSSL" on page 76.
- For Windows, if you are providing a certificate to enable TLS on the CAS binary port, provide a certificate in a PFX formatted file. The file contains the following information.
  - The PFX file has a private key embedded within it.

  - The private key within the PFX file is protected with a password.
  - The PFX file contains all certificates in the certification path (the PFX file contains the certificates that comprise the CA chain).
- Intermediate certificates need to be added to the server identity certificate in a certificate chain. The server identity certificate must be the first certificate in the chain. The intermediate certificate must be second. This order is important to allow validation with the private key to be successful.
- If your custom root certificate is site-signed or is not already included in the Mozilla bundle of trusted CA certificates, then you need to add the root certificate to the trustedcerts files.

  On Linux, you should also place a copy of the root certificate that you are adding to the trustedcerts files in the same directory. The root certificate should have a .crt file extension. This ensures that if the Ansible playbook needs to be rerun to update the installation, then this root certificate is automatically included in the regeneration of the trustedcerts files.

  For information, see "Manage Truststores" on page 65.

  Note: Do not delete the trustedcerts.jks and the trustedcerts.pem files.

  Note: Add your root certificate to the trustedcerts.pem and trustedcerts.jks files on every machine in the deployment.
- Encrypt your private key when possible.
- Password-protect your private key file.
- Place your password in your encrypted key file as the first line of the file. If you are using the SAS Viya provided certificate and key files, the name of that file is the encryption.key file. .
- When providing custom certificates, do not name your custom certificates and key files the same names as the SAS Viya provided or the Apache provided default certificate and key files (sas_encrypted.crt, sas_encrypted.key, encryption.key, localhost.crt, localhost.key). In a SAS Viya full-deployment on Linux, the default certificates and keys are renewed every time the primary controller is restarted. Therefore, the custom certificate and key files will be overwritten if stored using the same name as the default.

Manage Certificates Using Ansible Play Utilities (Linux Deployment)

When using the Ansible playbook, the following utilities can be used to manage certificates on a SAS Viya full deployment. These utilities are run from the **sas_viya_playbook** directory.

rebuild-trust-stores.yml

On an Ansible controller machine, from the **/viya/sas_viya_playbook/** directory, run the rebuild-trust-stores.yml play to incorporate the customer CA bundle of trusted certificates (from Consul configuration) into the various truststore files (trustedcerts.pem and trustedcerts.jks) on each machine in the SAS Viya deployment.

```
ansible-playbook -i inventory.ini ./utility/rebuild-trust-stores.yml
```

For an example of how this play is used, see "Add Certificates to Truststores Using Ansible (Linux Full Deployment)" on page 65.

distribute-httpd-certs.yml

This Ansible play adds your new custom certificate to **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts**. The play distributes copies of the certificate chain file to all machines with a name of httpproxy-**inventory name**-ca.crt. The play then rebuilds the trustedcerts.pem and trustedcerts.jks files and distributes them to every machine in the deployment.

On an Ansible controller machine, from the **sas_viya_playbook** directory, you can run the distribute-httpd-certs.yml play to distribute new certificates. On the Ansible controller machine, locate the utility file in the **/viya/sas_viya_playbook/utility** directory.

```
ansible-playbook -i inventory.ini ./utility/distribute-httpd-certs.yml
```

For an example of how this play is used, see "Replace Self-Signed Certificates with Custom Certificates (Post-deployment)" on page 10.

Create Certificates with SANs Using OpenSSL

To generate certificates with subject alternative names (SAN), you can use OpenSSL to create a new private key and a certificate that includes the SAN fields. Subject Alternative Names are an X509 Version 3 (RFC 2459) extension that allows a TLS certificate to specify multiple names that the certificate should match. A SubjectAltName can contain email addresses, IP addresses, regular DNS host names, and multi-tenant names.

In the following task, we are creating a self-signed CA certificate that includes subject alternative names .

1 Create a configuration file to be used by OpenSSL for the certificate signing request (CSR). Provide the following information in the file:

- CN = common name. This should be a fully qualified domain name (FQDN). An FQDN has two parts, a host-name and a domain name.
- subjectAltName = subject alternative names. Provide all of them, including fully qualified host names, short names, alternative names, IP addresses, multi-tenant names, and wildcards for multi-tenancy.
- basicConstraints = CA:true

Here is an example configuration file to generate a self-signed CA certificate using OpenSSL. Our configuration file is named req.conf. In this file, we are requesting that the new certificate contain fully qualified domain names, short names, subject alternative names, and multi-tenant names.

Note: For more information about creating an X509v3 configuration file, see x509v3_config

```
[req]
distinguished_name = req_distinguished_name
x509_extensions = v3_req
prompt = no
[req_distinguished_name]
C = US
O = Self-Signed Certificate
CN = <<enter your common name - use a fully qualified domain name>>
[v3_req]
keyUsage = keyEncipherment, dataEncipherment
extendedKeyUsage = serverAuth, clientAuth
subjectAltName = @alt_names
basicConstraints = CA:TRUE
[alt_names]
DNS.1 = <<fully qualified hostname>>
DNS.2 = <<short name>>
DNS.3 = <<alternative fully qualified hostname>>
DNS.4 = <<alternative short name>>
DNS.5 = *.<<fully qualified hostname>>
DNS.6 = *.<<alternative fully qualified hostname>>
DNS.7 = *.<<short name>>
DNS.8 = *.<<alternative short name>>
IP.2 = 0:0:0:0:0:0:0:1
IP.3 = <<IPv4 Address>>
IP.4 = <<IPv6 Address>>
```

Note: The DNS.5–DNS.9 alternative names are used for multi-tenancy. Ensure that the certificates contain subject alternate names for each tenant or use a wildcard for the subdomain.

For this example, this certificate is being used by Apache httpd. The Apache httpd proxy is shared by all tenants, but each tenant gets a unique subdomain. If the proxy is named viya.abc.com, then users for the tenant1 tenant will access the system at tenant1.viya.abc.com. Customers are required to set up DNS

aliases (either explicit subdomain aliases or a wildcard alias) to deploy a multi-tenant system. The DNS subdomain must match (case insensitive) the tenant ID.

2 Generate the private key and self-signed certificate using the OpenSSL `req` command:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
path/customer.key -out path/customer.crt -config req.conf -extensions 'v3_req'
```

3 Three files are located in the current directory:

- req.conf is the configuration file specified in the OpenSSL `req` command.
- customer.key is the RSA 2048-bit private key file.
- customer.crt is the self-signed TLS certificate in PEM format.

Use the following OpenSSL command to verify that the new certificate file (customer.crt) includes the SAN information:

```
openssl x509 -in customer.crt -text -noout
```

Generate Site-Signed or Third-Party-Signed Certificates in PEM Format

You need to create two files, a private key file and a certificate file.

private key
: This private key is in RSA format and is saved in ASCII (Base64-encoded) PEM (Privacy Enhanced Mail) format.

third-party-signed certificate
: A certificate authority (CA) is a trusted third party. This certificate contains the CA's public key in X.509 certificate form and is saved in ASCII (Base64-encoded) PEM format.

SAS recommends the following best practices for managing certificates and securing your private keys for the CAS server.

- Place your server identity certificates in the **`/config/etc/SASSecurityCertificateFramework/tls/certs`** directory.

  Intermediate certificates need to be added to the server identity certificate in a certificate chain. The server identity certificate must be the first certificate in the chain. The intermediate certificate must be second. This order is important to allow validation with the private key to be successful.

- If your custom root certificate is site-signed or is not already included in the Mozilla bundle of trusted CA certificates, then you need to manually add the root certificate to the trustedcerts files under the **`/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts`** directory.

  You should also place a copy of the root certificate that you are adding to the trustedcerts files in the same directory. The root certificate should have a .crt file extension. This ensures that if the playbook needs to be rerun to update the installation, then this root certificate is automatically included in the regeneration of the trustedcerts files.

  For information, see "Add Certificates to or Remove Certificates from the Truststore or a Certificate Chain Manually" on page 68.

  Note: Do not delete the trustedcerts.jks and the trustedcerts.pem files.

  Note: Add your root certificate to the trustedcerts.pem and trustedcerts.jks files on every machine in the deployment.

- Place your private server keys in the **`/config/etc/SASSecurityCertificateFramework/private`** directory structure and reference this directory location in the environment variables that you are setting.
- Encrypt your private key when possible.

Note: This example is one way of possibly several to generate certificates for use with TLS. Consult your administrator for details about what is required for your site.

Generate site-signed or third-party-signed certificates in PEM format.

1 Decide which type of CA to use at your site.
   - site-signed
   - third-party-signed

2 Change the directory to where your OpenSSL commands reside. For example:

```
cd /usr/bin
```

3 Use the following OpenSSL command to generate a new private key in RSA format and a CA certificate signing request in PEM format. Store your private key in **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/private**.

```
openssl req -new -out /opt/sas/viya/config/etc/
SASSecurityCertificateFramework/tls/certs/certreq.csr
-newkey rsa:2048 -keyout /opt/sas/viya/config/etc/
SASSecurityCertificateFramework/private/private.key -nodes
```

It is recommended that you supply an encrypted password on the key file. To do so, submit the following request.

```
openssl rsa -aes128 -in /opt/sas/viya/config/etc/
SASSecurityCertificateFramework/private/private.key
-out /opt/sas/viya/config/etc/SASSecurityCertificateFramework/
private/tempprivate.key -passout pass:password
```

```
mv opt/sas/viya/config/etc/SASSecurityCertificateFramework/
private/tempprivate.key /opt/sas/viya/config/etc/
SASSecurityCertificateFramework/private/private.key
```

4 Verify your certificate signing request (CSR).

```
openssl req -noout -text -in /opt/sas/viya/config/etc/
SASSecurityCertificateFramework/tls/certs/certreq.csr
```

5 Submit your CSR file (certreq.csr) to your CA. This CA can be a CA at your site or a third party. You should receive the following certificates from your CA.
   - signed certificate (containing the CA's public key)
   - CA root certificate
   - One or more CA intermediate certificates

6 Store the signed certificates from your CA in **opt/sas/viya/config/etc/SASSecurityCertificateFramework/tls/certs**.

7 Add your site-signed root CA certificates to the truststore. See "Add Certificates to or Remove Certificates from the Truststore or a Certificate Chain Manually" on page 68.

See Also

For an example of using OpenSSL to generate site-signed or third-party-signed certificates in PEM format, see "Use OpenSSL to Create Site-Signed or Third-Party-Signed Certificates in PEM Format" on page 139.

Generate Site-Signed or Third-Party-Signed Certificates in Java Keystore Format

The following steps create site-signed or third-party-signed certificates in Java keystore (JKS) format. Details of each step are shown after this summary.

1 Create the machine's keystore.

2 Create a certificate signing request (CSR).

3 Submit a .csr file to a CA.

4 Receive a signed certificate, CA root certificate, and one or more CA intermediate certificates.

5 Add the server's identity certificate to the keystore.

6 Add the CA intermediate certificate to the keystore.

Note: This example is one way of possibly several to generate certificates for use with TLS. Consult your administrator for details about what is required for your site.

The keystore contains private keys and certificates used by TLS servers to authenticate themselves to TLS clients. By convention, such files are referred to as keystores.

SAS recommends the following best practices for managing certificates for Java.

- The signed certificate and private key are contained in one JKS format file. Add your certificates to the **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/java/jks** directory.
- Password-protect the private key.
- Password-protect the keystore. In the following example, the keystore file is named keystore.jks.
- Make the keystore file readable only by members of the appropriate group.
- Make the file where the keystore password is referenced readable only by members of the appropriate group. For example, you might make the init_usermods.properties file (where the password is referenced by a keystore password property)) readable only by members of the appropriate group.

You can obtain site-signed or third-party-signed certificates using the Java Keytool. In the following scenario, we are using a certificate authority (CA) as our third party.

1 Log on to your machine as a user with root or sudo privileges.

2 Change the directory to where your `keytool` command resides. For example:

```
cd $JAVA_HOME/bin
```

3 Use the `keytool` command to create a new private key and keystore and store the information in the keystore file named keystore.jks. In the following example, we are first generating a private key server.key. We are also using alias server.

```
keytool -genkey -alias server -keyalg RSA -keystore
/opt/sas/viya/config/etc/SASSecurityCertificateFramework/
java/jks/keystore.jks -storepass changeit -keypass password
-validity 360 -keysize 2048
```

The keystore password (which protects the keystore as a whole) and the key password (which protects the private key stored in the server entry) are set using the -storepass and -keypass options respectively.

Change the permissions on the keystore file (keystore.jks) to be readable only by members of the appropriate group. Use chmod or sudo to change the permissions.

```
chmod 600 keystore.jks
```

When you list the file, you see the permissions are Read/Write only (-rw-------) .

4 To query the contents of your Java keystore file, you can use the following command:

```
keytool -list -v -keystore /opt/sas/viya/config/etc
/SASSecurityCertificateFramework/java/jks/keystore.jks
-storepass changeit -keypass password
```

5 Use the `keytool` command to create a certificate signing request (CSR) for an existing keystore. Here is an example command:

```
keytool -certreq -alias server -keystore
/opt/sas/viya/config/etc/SASSecurityCertificateFramework
/java/jks/keystore.jks -file
/opt/sas/viya/config/etc/SASSecurityCertificateFramework
/java/jks/server.csr -storepass changeit -keypass password
```

This command generates the CSR and stores it in a file called server.csr.

6 Submit your CSR file to your CA. For our example, we have provided a name for each of the signed certificates that we might receive: server_ca.pem, root_ca.pem, and int_ca.pem. You should receive the following from your CA:

- signed identity certificate (server_ca.pem)
- CA root certificate (root_ca.pem)
- one or more CA intermediate certificates (int_ca.pem)

7 After you have submitted your CSR to the CA and received the CA's reply (containing the signed certificate), import the reply into your keystore, located at **`/opt/sas/viya/config/etc/SASSecurityCertificateFramework/java/jks`**, using the following `keytool` options.

This step imports the signed server identity certificate and one or more intermediate certificates in PEM format into the keystore.

a Add the server identity certificate to your keystore. In this example, server_cert.pem is the server identity certificate.

```
keytool -importcert -file /opt/sas/viya/config/etc/
SASSecurityCertificateFramework/java/jks/server_ca.pem
-keystore /opt/sas/viya/config/etc/SASSecurityCertificateFramework/
java/jks/keystore.jks -storepass changeit -keypass password
-trustcacerts -alias server_ca
```

b If your server certificate is signed by an intermediate CA, import the intermediate certificate into your keystore file. In this example, int_ca.pem is the CA intermediate certificate.

```
keytool -importcert -file /opt/sas/viya/config/etc/
SASSecurityCertificateFramework/java/jks/int_ca.pem -keystore
/opt/sas/viya/config/etc/SASSecurityCertificateFramework/java/
jks/keystore.jks -storepass changeit -keypass password
-trustcacerts -alias int_ca
```

c Verify that the certificates that you added to your keystore are present.

```
keytool -v -list -keystore /opt/sas/viya/config/etc/
SASSecurityCertificateFramework/java/jks/keystore.jks
-storepass changeit -keypass password
```

Generate Self-Signed Certificates

Self-signed certificates are signed by your own private key, rather than by an external CA. You can generate self-signed certificates or root certificates in PEM format using RSA or HMAC encryption or in Java keystore format.

A private key file and a self-signed certificate are needed.

private key
: This private key is in RSA format and is saved in ASCII (Base64-encoded) PEM format.

self-signed certificate
This certificate contains a public key in X.509 certificate form and is saved in ASCII (Base64-encoded) PEM format.

SAS recommends the following best practices for managing certificates and securing your private keys. See "Use Best Practices to Create and Manage Certificates" on page 74.

Generate self-signed certificates or root certificates in PEM format using RSA encryption.

Note: This example is one of several possible ways to generate certificates for use with TLS. Consult your administrator for details about what is required for your site.

1 Change the directory to the directory where your OpenSSL commands reside. For example:

```
cd /usr/bin
```

2 Use the following OpenSSL command to generate a self-signed certificate with new private key using RSA encryption. In this example, a self-signed CA certificate with subject alternative names is being requested. To see an example where an OpenSSL configuration file is used to specify extensions, see see "Create Certificates with SANs Using OpenSSL" on page 76.

Note: In this example, the certificate request is for use on Linux and the paths specified are Linux paths.

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/opt/sas/viya/config/etc/
SASSecurityCertificateFramework/private/customer.key -out
/opt/sas/viya/config/etc/SASSecurityCertificateFramework/
tls/certs/customer.csr
-addext "subjectAltName=DNS:fully qualified hostname"
-addext "subjectAltName=IP:ip-address"
-addext "basicConstraints=CA:true"
```

It is recommended that you supply an encrypted password on the key file. To do so, submit the following request:

```
openssl rsa -aes128 -in /opt/sas/viya/config/etc/
SASSecurityCertificateFramework/private/customer.key
-out /opt/sas/viya/config/etc/SASSecurityCertificateFramework/
private/tempcustomer.key -passout pass:password
mv opt/sas/viya/config/etc/SASSecurityCertificateFramework/
private/tempprivate.key /opt/sas/viya/config/etc/
SASSecurityCertificateFramework/private/customer.key
```

3 If you need to add your certificates to the trusted list of certificates, see "Add Certificates to or Remove Certificates from the Truststore or a Certificate Chain Manually" on page 68.

Generate self-signed certificates in Java keystore format.

Note: This example is one of several possible ways to generate certificates for use with TLS. Consult your administrator for details about what is required for your site.

1 For servers based on Java, generate a self-signed certificate using `keytool -genkeypair`. This command creates a public/private key pair and wraps the public key into a self-signed certificate. For example, the following command creates a self-signed test certificate for the host and stores it in a keystore. For this example, we are using alias `javahost`.

```
$ keytool -genkeypair -keystore
/opt/sas/viya/config/etc/SASSecurityCertificateFramework/
java/jks/javahost.jks -keyalg RSA -alias javahost
-dname "CN=javahost.example.com,O=Hadoop" -storepass changeit
-keypass password -validity 1000
```

Note: By default, self-signed certificates are valid for only 90 days. To increase this period, replace the previous command's `-validity <val_days>` parameter to specify the number of days for which the certificate should be considered valid.

2 If you need to add your certificates to the trusted list of certificates, see"Add Certificates to or Remove Certificates from the Truststore or a Certificate Chain Manually" on page 68.

Convert Digital Certificate File Formats Using OpenSSL

In OpenSSL, you can use many parameters to convert between the different digital certificate file formats. The following are some examples of a few ways to convert files from one format to another. See OpenSSL TLS Toolkit for more commands that can be used.

Convert DER to PEM File Format

Many certificate authorities provide certificates in DER format. If you have a DER formatted file, but need a PEM formatted file, you can convert the DER formatted file to PEM format using OpenSSL.

Note: You must convert a DER formatted file to PEM format before you can include it in a trust list on Linux.

Here is an example of how to convert a server digital certificate from DER input format to PEM output format:

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

Convert a PEM Encoded Certificate to DER File Format

If you have a PEM formatted file, but need a DER formatted file, you can convert the PEM formatted file to DER using OpenSSL.

Here is an example of how to convert a server digital certificate from PEM input format to DER output format:

```
OpenSSL x509 -outform der -in certificate.pem -out certificate.der
```

Convert PEM to PKCS#12 (.pfx .p12) File Format

If you are using a Java application that accepts only PKCS#12 format, you might need to convert your PEM formatted file that includes certificates and the separate key file to one file that includes both the certificate and the key file.

If you have a PEM formatted certificate file, but need a PKCS#12 formatted file, you can convert the PEM format certificate to a PKCS#12 format using OpenSSL. Here is one way of converting a PEM to a PKCS#12 formatted file for non-FIPS (Federal Information Processing Standard) libraries.

```
OpenSSL pkcs12 -export -out path/certificate.p12 -inkey path/privatekey.key
-in path/certificate.crt -certfile certs.pem
```

Use SAS Bootstrap Config CLI on Consul to Manage the KV Store and ACL Tokens

SAS Bootstrap Config CLI Commands

SAS Bootstrap Config CLI enables you to interact with the SAS Configuration Server (Consul) from the command line. This tool can be found at the following locations on Windows and Linux deployments:

- On Linux, **`/opt/sas/viya/home/bin`**
- On Windows, ***`Install-Directory`*\`Viya\bin\sas-bootstrap-config.exe`**

For general server information, see "SAS Configuration Server" in *SAS Viya Administration: Infrastructure Servers*.

On Linux, the general command syntax is as follows:

/opt/sas/viya/home/bin/sas-bootstrap-config *command*

On Windows, the general command syntax is as follows:

Install-Directory\Viya\bin\sas-bootstrap-config.exe *command*

The SAS Bootstrap Config CLI commands that can be used are shown in the following table. In this document, we use the `kv` (key-value) and `acl` (access control list) commands to update the key-value stores when managing certificates for TLS, and we use the `acl` command for managing ACL tokens.

Table 7 *SAS Bootstrap Config CLI Commands*

Commands	Description
`acl`	Manages access control lists (ACLs) in Consul.
`agent`	Manages the Consul agent.
`catalog`	Enables queries of endpoints that list known datacenters, nodes in a given datacenter, services in a given datacenter, nodes in a given service, and services provided by a node. The catalog endpoints register and deregister nodes, services, and checks in Consul.
`help` `h`	Shows a list of commands or help for one command.
`kv`	Manages key-value pairs in Consul.
`network`	Gets network information.
`node`	Gets the node ID.
`operator`	Provides cluster-level tools for Consul operators.
`status`	Gets information about the status of the Consul cluster.

Commands that enable you to interact with the key-value store are shown in Table 8 on page 83. The following syntax is for interacting with the key-value store on Linux:

/opt/sas/viya/home/bin/sas-bootstrap-config kv *command argument*

The following syntax is for interacting with the key-value store on Windows:

Install-Directory\Viya\bin\sas-bootstrap-config.exe kv *command argument*

Table 8 *SAS Bootstrap Config CLI Commands for Updating the Key-Value Store*

Commands	Description
`bulkload`	Loads key-value pairs into Consul.
`delete`	Deletes a given key in the Consul KV store.

Commands	Description
`exists`	Returns the exit code 64 if the key does not exist. Returns 0 if it exists.
`help`	Shows a list of commands or help for one command.
`read`	Reads a value for a key.
`write`	Writes a key-value pair to Consul.

Commands that enable you to create, update, destroy, and query ACL tokens are shown in the following table. The syntax for interacting with ACL tokens is as follows:

/opt/sas/viya/home/bin/sas-bootstrap-config acl *command*

Table 9 *SAS Bootstrap Config CLI Commands for Managing ACL Tokens*

Commands	Description
`clone`	Creates an ACL token by cloning an existing token.
`create`	Creates an ACL token with a given policy.
`destroy`	Destroys an ACL token.
`info`	Gets information about an ACL token.
`list`	Lists all active ACL tokens.
`update`	Updates an ACL token.

Establish a TLS Chain of Trust to Access SAS Bootstrap Config CLI

SAS Configuration Server in SAS Viya is secure by default and requires encryption in-motion using TLS. SAS Bootstrap Config CLI must establish trust for the TLS handshake to proceed. To establish trust, the truststore must be specified as an environment variable and point to a truststore that contains the CA certificates.

On a Linux deployment, if your environment is enabled for Transport Layer Security (TLS), you must set the SSL_CERT_FILE environment variable to the path location of the trustedcerts.pem file (if using the SAS default truststore) or the path location of your site-signed certificate (if using an internal truststore). For CLI users on Linux who are running the CLIs directly on the SAS machine, you can source the consul.conf file rather than setting the SSL_CERT_FILE environment variable manually.

Before invoking SAS Bootstrap Config CLI, source the consul.conf file.

```
. /opt/sas/viya/config/consul.conf
```

The consul.conf file contains the following environment variable settings:

```
# BEGIN Ansible managed Consul client connection options
export CONSUL_HTTP_ADDR=https://localhost:8501
export
SSL_CERT_FILE=/opt/sas/viya/config/etc/SASSecurityCertificateFramework/
cacerts/trustedcerts.pem
```

```
export CONSUL_CACERT=$SSL_CERT_FILE
export VAULT_CACERT=$SSL_CERT_FILE
# END Ansible managed Consul client connection options
```

If you are manually setting the SSL_CERT_FILE, before invoking the SAS Bootstrap Config CLI, set the environment variable to the path of the trustedcerts.pem file (if using the SAS default truststore) or the path of your site-signed certificate (if using an internal truststore). Here is an example pointing to the SAS default truststore:

```
export SSL_CERT_FILE=/opt/sas/viya/config/etc/
SASSecurityCertificateFramework/cacerts/trustedcerts.pem
```

Here is an example pointing to the customer's site-signed certificate.

```
export SSL_CERT_FILE=/path/customer.crt
```

For an example of removing the key-value from Consul using SAS Bootstrap Config CLI, see "Remove Certificates from the Truststores Using Ansible (Linux Full Deployment)" on page 66.

Authenticate to Access SAS Bootstrap Config CLI

SAS Configuration Server has access control in place. To use SAS Bootstrap Config CLI, an access token is required. There are two ways to set up your environment and access a token.

Note: The following code is shown on more than one line for display purposes only. The SAS Bootstrap Config commands need to be on one line and should not contain line breaks.

- Specify the following environment variable for the current shell session.

  ```
  export CONSUL_HTTP_TOKEN='sudo cat /opt/sas/viya/config/etc/
  SASSecurityCertificateFramework/tokens/consul/default/client.token'
  ```

- Specify the following as the first option to be read from a file.

  ```
  /opt/sas/viya/home/bin/sas-bootstrap-config --token-file
  /opt/sas/viya/config/etc/SASSecurityCertificateFramework/
  tokens/consul/default/client.token kv
  ```

Secure Credentials in the CAS Server with cas.servicesbaseurl (Linux Full Deployment)

Note: This section is applicable only if you have a full deployment. If you have a programming-only deployment, skip this section.

For credentials management, the cas.SERVICESBASEURL= option is set during deployment. The URL that enables a CAS server to use SAS Viya services is set using the cas.SERVICESBASEURL= option. For example, CAS client credentials are passed to the SASLogon service at the address specified in the cas.SERVICESBASEURL= option in order to obtain an OAuth token.

This option is set by default in the casconfig_deployment.lua file located at **/opt/sas/viya/config/etc/cas/default/**.

1. In the casconfig.lua file, ensure that the HTTPS URL is used to access the Apache HTTP Server machine.

   ```
   cas.servicesbaseurl='https://webserver-host-name'
   ```

   Note: The host name in the URL is the same as the Common Name used in the server identity certificate that Apache HTTP Server is using.

   Note: In a SAS Viya full deployment, the cas.SERVICESBASEURL= option defaults to port 443 for HTTPS access.

2 When you set the cas.SERVICESBASEURL= option to use HTTPS, the CAS_CALISTLOC= environment variable needs to be set in the casconfig_usermods.lua file to point to the CA certificates that the Apache HTTP Server is using.

```
env.CAS_CALISTLOC=
'/path-to-CA-chain-used-for-Apache-HTTP-Server-certificate'
```

Note: If the CA certificates are already imported in the OpenSSL truststore, setting the env.CAS_CALISTLOC= environment variable is not necessary.

3 If you are setting the CAS_CALISTLOC= environment variable, you should copy the change made to this environment variable to the vars.yml file. This change ensures that your settings are not changed when upgrades are made to the deployment.

Note: See "Modify the vars.yml File" in *SAS Viya for Linux: Deployment Guide*.

Add the following highlighted variables and their respective values:

```
CAS_CONFIGURATION:
   env:
     #CAS_DISK_CACHE: /tmp
     CAS_CLIENT_SSL_REQUIRED: 'true'
     CAS_CALISTLOC: path-to-CA-chain-used-for-Apache-HTTP-Server-certificate
   cfg:
     #gcport: 5580
     #httpport: 8777
     #port: 5570
     #colocation: 'none'
     servicesbaseurl: 'https://http-proxy-host-name'
```

Save and close the vars.yml file.

For information about using cas.SERVICESBASEURL=, see See "Configuration File Options Reference" in *SAS Viya Administration: SAS Cloud Analytic Services*.

Manage Tokens, Create JWT Signing Keys, and Update the Encryption Key

Generate Signing Keys for JSON Web Tokens

Overview

A JSON web token (JWT) is a JSON object that is defined in RFC 7519 as a safe way to pass a set of information between two parties. Access tokens issued by SAS Logon Manager are also OpenID Connect ID tokens, which are JWTs.

The token consists of three parts: a header, claims, and a signature. All of these parts are base64 encoded. Here is what an example token might look like. Each part is separated by a period to create header.claims.signature.


```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.
TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ
```


In a new SAS Viya deployment, SAS Logon Manager generates RSA keys and CAS gets the public key that it needs from SAS Logon Manager automatically if the cas.SERVICESBASEURL= property is set. You can configure your own signing keys, overriding the SAS Logon Manager behavior.

To configure your JWT keys, the sas.logon.jwt property needs to be set for CAS and SAS Logon Manager. The sas.logon.jwt property is used to secure JSON web tokens with RSA digital signatures. For a description of the

properties, see "Configuration Properties: Reference (Applications)" in *SAS Viya Administration: Configuration Properties*.

Generate a JWT Signing Key

The following example uses OpenSSL to generate an RSA signing key.

Note: This example is one way of many to generate RSA signing keys. Consult your administrator for details about what is required for your site.

1. Change the working directory to the directory where SAS stores keys. SAS stores keys in the **/opt/sas/viya/config/etc/SASSecurityCertificateFramework** directory structure. For example:

   ```
   cd /opt/sas/viya/config/etc/SASSecurityCertificateFramework/private
   ```

2. Use the following OpenSSL command to generate a new RSA private key. The following OpenSSL command generates the RSA private key in PKCS#1 format:

   ```
   openssl genrsa -out /opt/sas/viya/config/etc/
   SASSecurityCertificateFramework/private/jwt-privatekey.pem 2048
   ```

3. You can derive the public key from the private key using the following command:

   ```
   openssl rsa -in jwt-privatekey.pem -out public.pem -pubout
   ```

4. Copy the private key to the signingKey property using SAS Environment Manager. See "Configure the SAS Logon Manager with a New JWT Signing Key" on page 87.

Configure the SAS Logon Manager with a New JWT Signing Key

Use the SAS Environment Manager to set configuration properties that are used by the SAS Logon Manager. If you created a new JWT signing key, paste the key into the signingKey property.

1. From the side menu (☰), select **Manage Environment**.

2. In the navigation bar, click 🔧.

   The Configuration page is an advanced interface. It is available to only SAS Administrators.

3. The default view is **Basic Services**. Select **Definitions** from the drop-down box.

4. In the **Definitions** list, select **sas.logon.jwt**.

5. If no properties are configured for definition, complete the following:

   a. In the top right corner of the window, click **New Configuration**.

   b. In the New sas.logon.jwt Configuration dialog box, paste the PEM-encoded JWT private key into the value for the signingKey property.

      For a description of the properties, see "Configuration Properties: Reference (Applications)" in *SAS Viya Administration: Configuration Properties*.

   c. Click **Save**.

      Note: The system takes a few minutes to recognize the new key before starting to use the new key.

Obtain an OAuth Access Token to Register a New Client ID

You can use a curl command to obtain an OAuth access token. You can then use that token to register a new client ID.

A Consul token is required to obtain an OAuth access token for registration. A Consul token can be found at **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tokens/consul/default** and is named client.token.

An example curl command to request a registration token for a new client follows. In this example, the client is named APP.

```
curl -X POST "http://localhost/SASLogon/oauth/
clients/consul?callback=false&serviceId=app"
-H "X-Consul-Token: 29c4700f-ea89-41cd-8bc4-4198ccaa5bf9"
```

Note:

This request must pass a `callback=false` query string parameter and authenticate directly by passing a SAS Configuration Server (Consul) token. If the Consul token is valid, SASLogon returns the OAuth access token for registration in the response.

By default, tokens are valid for 12 hours. When you register the client ID, you can configure the amount of time that the token is valid. See how to register a client ID in "Authentication: How To" in *SAS Viya Administration: Authentication*.

For more information about using curl, see Curl Documentation.

Replace Tokens and Update the Encryption Key for SAS Configuration Server (Linux Full Deployment)

Overview

At installation, tokens are generated and placed in the **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tokens/consul/default** directory. Client, encryption, and management tokens are provided. The owner and group of these files is SAS.

client.token
: is the ACL client token that is used by all services to access values in the key-value store.

management.token
: is the ACL management token (acl_master_token) that is used to administer the ACLs.

encryption.token
: specifies the secret key that is used for encryption of Consul network traffic. It is used for Gossip communication.

You must use the value of an ACL token that is of type management to administer Consul ACLs. The value of this management token is created by the Ansible playbook and stored in the management.token file at **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tokens/consul/default**.

Replace ACL Tokens

You must use the value of an ACL token that is of type management to administer Consul ACLs. In the following example, the **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tokens/consul/default/management.token** file contains the ID of a management ACL that we want to change.

We are using the SAS Bootstrap Config CLI to replace ACL tokens.

Note: You can also use the Consul ACL HTTP CLI to manage ACL tokens. For more information, see ACL HTTP Endpoint.

1. Source the /etc/profile.d/lang.sh to set the LANG environment variable. It will be set to a value such as en_US.UTF-8

   ```
   source /etc/profile.d/lang.sh
   ```

2. The **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tokens/consul/default/management.token** file contains the ID of a management ACL that we want to change.

```
sudo cat /opt/sas/viya/config/etc/
SASSecurityCertificateFramework/tokens/consul/default/management.token
0329addc-bb72-489c-9f0a-5421890dd2fb
```

3 Create a backup copy of the original management.token.

```
sudo cp /opt/sas/viya/config/etc/
SASSecurityCertificateFramework/tokens/consul/default/
management.token /opt/sas/viya/config/etc/
SASSecurityCertificateFramework/tokens/consul/default/
management.token.OLD
```

4 List the ACLs using the SAS Bootstrap Config CLI. The SAS Bootstrap Config CLI must establish trust for the TLS handshake to proceed and allow secure communication. To establish trust, the truststore must be specified as an environment variable. Sourcing the consul.conf sets the SSL_CERT_FILE environment variable to the trusted certificates. After this trust is established, you can communicate using the SAS Bootstrap Config CLI and list the ACLs.

```
. /opt/sas/viya/config/consul.conf
sudo /opt/sas/viya/home/bin/sas-bootstrap-config
--token-file /opt/sas/viya/config/etc/
SASSecurityCertificateFramework/tokens/consul/default/
management.token acl list
```

These are the ACLs listed.


```
 {
        "CreateIndex": 4,
        "ModifyIndex": 4,
        "ID": "0329addc-bb72-489c-9f0a-5421890dd2fb",
        "Name": "Master Token",
        "Type": "management",
        "Rules": ""
    },
    {
        "CreateIndex": 3,
        "ModifyIndex": 64718,
        "ID": "anonymous",
        "Name": "Anonymous Token",
        "Type": "client",
        "Rules": "{\"service\":{\"\":{\"Policy\":\"read\"}}}"
    },
    {
        "CreateIndex": 19,
        "ModifyIndex": 64702,
        "ID": "eaa6de8a-3824-4c8f-a73a-dbd835c5cc97",
        "Name": "client",
        "Type": "client",
        "Rules": "{\"key\":{\"\":{\"Policy\":\"write\"}},\"service\":
        {\"\":{\"Policy\":\"write\"}},\"event\":{\"\":{\"Policy\":\"write\"}},
        \"query\":{\"\":{\"Policy\":\"write\"}}}"
    }
```


5 Clone the management token using the following command. The `c43b7d1a-ccee-3792-a1d8-9576a9dbe7d2` ID is returned by the execution of the following code. This ID is the new value that is inserted into the management.token file at **`/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tokens/consul/default`**.

```
sudo /opt/sas/viya/home/bin/sas-bootstrap-config --token-file
/opt/sas/viya/config/etc/SASSecurityCertificateFramework/
```

```
tokens/consul/default/management.token acl clone --acl-id
$(sudo cat /opt/sas/viya/config/etc/SASSecurityCertificateFramework/
tokens/consul/default/management.token)
{
    "ID": "c43b7d1a-ccee-3792-a1d8-9576a9dbe7d2"
}
```

6 List the ACLs again to verify that the new management ACL has been created.

```
sudo /opt/sas/viya/home/bin/sas-bootstrap-config --token-file
/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tokens/
consul/default/management.token acl list
```

Here are the ACLs listed now.

```
{
        "CreateIndex": 4,
        "ModifyIndex": 4,
        "ID": "0329addc-bb72-489c-9f0a-5421890dd2fb",
        "Name": "Master Token",
        "Type": "management",
        "Rules": ""
    },
    {
        "CreateIndex": 3,
        "ModifyIndex": 64899,
        "ID": "anonymous",
        "Name": "Anonymous Token",
        "Type": "client",
        "Rules": "{\"service\":{\"\":{\"Policy\":\"read\"}}}"
    },
    {
        "CreateIndex": 64927,
        "ModifyIndex": 64927,
        "ID": "c43b7d1a-ccee-3792-a1d8-9576a9dbe7d2",
        "Name": "Master Token",
        "Type": "management",
        "Rules": ""
    },
    {
        "CreateIndex": 19,
        "ModifyIndex": 64897,
        "ID": "eaa6de8a-3824-4c8f-a73a-dbd835c5cc97",
        "Name": "client",
        "Type": "client",
        "Rules": "{\"key\":{\"\":{\"Policy\":\"write\"}},
        \"service\":{\"\":{\"Policy\":\"write\"}},\"event\":
        {\"\":{\"Policy\":\"write\"}},\"query\":{\"\":{\"Policy\":\"write\"}}}"
    }
```

7 Replace the value in the management.token file with the value that was returned from the clone command.

```
sudo bash -c 'echo c43b7d1a-ccee-3792-a1d8-9576a9dbe7d2 >
/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tokens/
consul/default/management.token'
```

8 Destroy the old management ACL.

```
sudo /opt/sas/viya/home/bin/sas-bootstrap-config
--token-file /opt/sas/viya/config/etc/SASecurityCertificateFramework/
tokens/consul/default/management.token acl destroy --acl-id $(sudo cat
```

```
/opt/sas/viya/config/etc/SASSecurityCertificateFramework/
tokens/consul/default/management.token.OLD)
```

9 After the new ACLs have been created in Consul and the management.token and client.token files have been updated with the new values, copies of the original .token files can be deleted.

Replace ACL Tokens Using the sas-crypto-management Tool

You can use the sas-crypto-management application located at **/opt/sas/viya/home/SASSecurityCertificateFramework/bin/** to generate a value that can be used as the ID for an ACL. The sas-crypto-management tool must establish trust for the TLS handshake to proceed and allow secure communication. To establish trust, the truststore must be specified as an environment variable. Sourcing the consul.conf sets the SSL_CERT_FILE environment variable to the trusted certificates. After this trust is established, you can communicate using the sas-crypto-management tool.

```
. /opt/sas/viya/config/consul.conf
sudo /opt/sas/viya/home/bin/SASSecurityCertificateFramework/
bin/sas-crypto-management uuid --out-file /opt/sas/viya/config/etc/
SASSecurityCertificateFramework/tokens/consul/default/client.token
```

You can then use the value of the ACL ID that was generated using the sas-crypto-management tool (instead of the value that Consul generates using its clone command as shown in "Replace ACL Tokens" on page 88). Then, use the `create` command to specify the ID that should be used.

Replace an Encryption Key on Consul

SAS Configuration Server (Consul) uses two network communication protocols:

- Gossip protocol is used for communication between servers and agents. Encryption is enabled for Gossip communication by default in a SAS Viya deployment.
- RPC protocol is used for communication between agents and servers.

SAS Viya services interact with the Consul server agents (for example, communication of REST calls over HTTPS).

Note: In a SAS Viya full deployment, HTTP end-point is disabled by default.

All Consul agents that are running as servers or clients need to have an encryption key. The Consul agent supports encrypting all of its network traffic. The SASSecurityCertificateFramework provides the encryption key that is used for Gossip communication. Enabling Gossip encryption requires only that you set an encryption key when starting the Consul agent.

The Consul RPM start script generates a file named config-gossip.json in **/opt/sas/viya/config/etc/consul.d**. The consul RPM uses the value obtained from the gossip.token file in **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tokens/consul/default**. You can see the type of information contained in the file by submitting the following command:

```
sudo cat /opt/sas/viya/config/etc/consul.d/config-gossip.json
```

The generated file contains encryption information that looks like the following. The encryption key is 16 bytes and Base64 encoded.

```
{     "encrypt": "y/k+KRpeZZVmzHCVrvbR6A==" }
```

The encrypt option specifies the secret key to use for encryption of Consul network traffic. This key must be 16 bytes that are Base64 encoded. All nodes within a cluster must share the same encryption key to communicate. The provided key is automatically persisted to the data directory and loaded automatically whenever the agent is restarted. More information about this option can be found at Consul Configuration Command-line Options.

There are situations when the encryption key might need to be replaced.

1 Sign on to the machine that runs the SAS Configuration Server (Consul) as the SAS install user (sas) or with sudo privileges.

2 On the host running the SAS Configuration Server, use the `keygen` command to generate a new key on all hosts.

```
/opt/sas/viya/home/bin/consul keygen
```

3 Copy the value that is generated (for example, `X4SYOinf2pTAcAHRhpj7dA==`) into the config-gossip.json on all hosts. This file is located at **`/opt/sas/viya/config/etc/consul.d/`**.

   Use the copied string as the value for the `encrypt` parameter:

```
"encrypt": "X4SYOinf2pTAcAHRhpj7dA=="
```

4 Stop Consul. How you run the following command depends on your operating system.

   - Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

     ```
     sudo systemctl restart sas-viya-consul-default
     ```

   - Red Hat Enterprise Linux 6.x (or an equivalent distribution):

     ```
     sudo service sas-viya-consul-default restart
     ```

5 Delete the local.keyring and remote.keyring files in **`/opt/sas/viya/config/data/consul/serf`**.

   All nodes within a cluster must share the same encryption key to communicate. The provided key is automatically persisted to the data directory and loaded automatically whenever the agent is restarted. This option is provided on each agent's initial start-up sequence. The value of this secret key is persisted to the **`/opt/sas/viya/config/data/consul/serf`** directory to files local.keyring and remote.keyring.

   Note: If a key is provided after Consul has been initialized with an encryption key, then the provided key is ignored and a warning is displayed.

6 Restart SAS Configuration Server (Consul). How you run the following command depends on your operating system.

   - Red Hat Enterprise Linux 7.x (or an equivalent distribution) and SUSE Linux Enterprise Server 12.x:

     ```
     sudo systemctl restart sas-viya-consul-default
     ```

   - Red Hat Enterprise Linux 6.x (or an equivalent distribution):

     ```
     sudo service sas-viya-consul-default restart
     ```

   When Consul is restarted, the Consul RPM start script regenerates the config-gossip.json file and Consul reads this value and re-creates the local.keyring and remote.keyring files.

You can read about how this is done for Consul at Encryption for Consul.

Concepts

SAS/SECURE

SAS/SECURE Overview

SAS/SECURE software provides industry-standard encryption capabilities for data in motion. This affects communications among SAS servers and between SAS servers, SAS desktop clients, and SAS web applications.

Refer to “NETENCRYPT System Option” on page 111 and “NETENCRYPTALGORITHM= System Option” on page 112 for details. You can specify various encryption algorithms as well as TLS to secure data in motion.

Linux supports the following encryption algorithms:

- RC2
- RC4
- DES
- TripleDES
- AES

Refer to "Encryption Algorithms" on page 109 for more information about encryption algorithms supported for use with SAS/SECURE.

SAS/SECURE Software Availability

For software delivery purposes, SAS/SECURE is a product within SAS. SAS/SECURE is included with the SAS Viya software (except where prohibited by import restrictions). This change makes strong encryption available in all deployments.

SAS/SECURE Export Restrictions

For U.S. export purposes, SAS designates export status for each product based on the encryption algorithms and the product's functional capability. SAS/SECURE is available to most commercial and government users inside and outside the U.S. However, some countries (for example, Russia, China, and France) have import restrictions on products that contain encryption, and the U.S. prohibits the export of encryption software to specific embargoed or restricted destinations.

SAS/SECURE for Linux includes the following encryption algorithms.

- RC2 using up to 128-bit keys
- RC4 using up to 128-bit keys
- DES using up to 56-bit keys
- TripleDES using up to 168-bit keys
- AES using 256-bit keys

SAS/SECURE Installation and Configuration

SAS/SECURE is installed and delivered on every installation except in countries (for example, Russia, China, and France) that have import restrictions on products that contain encryption. The U.S. prohibits the export of encryption software to specific embargoed or restricted destinations. Whether SAS/SECURE is used depends on the options that are set.

To use encryption provided by SAS/SECURE for communications and networking, specify the NETENCRYPT system option and set the NETENCRALG= system option to a value of RC2, RC4, DES, TRIPLEDES, AES, or SSL. Refer to "NETENCRYPT System Option" on page 111 and "NETENCRYPTALGORITHM= System Option" on page 112.

Transport Layer Security (TLS)

Transport Layer Security (TLS) Overview

Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are cryptographic protocols that are designed to provide communication security. TLS and SSL are protocols that provide network data privacy, data integrity, and authentication.

Note: All discussion of TLS is also applicable to the predecessor protocol, Secure Sockets Layer (SSL).

TLS uses X.509 certificates and hence asymmetric cryptography to assure the party with whom they are communicating and to exchange a symmetric key. As a consequence of choosing X.509 certificates, certificate authorities and a public key infrastructure are necessary to verify the relation between a certificate and its owner, as well as to generate, sign, and administer the validity of certificates. For information about certificates, see "Certificates Used by TLS and HTTPS" on page 95.

In addition to providing encryption services, TLS performs client and server authentication, and it uses message authentication codes to ensure data integrity. The client requests a certificate from the server, which it validates against the public certificate of the certificate authority used to sign the server certificate. The client then verifies the identity of the server and negotiates with the server to select a cipher (encryption method). The cipher that is selected is the first match between the ciphers that are supported on both the client and the server. All subsequent data transfers for the current request are then encrypted with the selected encryption method.

TLS System Requirements

SAS supports TLS on the Linux and Windows operating environments.

Supported TLS Software and Libraries Used

SAS Viya uses the OpenSSL libraries provided and installed on your operating system.

- For Windows, SAS uses the Microsoft Schannel libraries that come with the Windows operating system to provide the TLS protocols.
- For Linux, SAS Viya uses the OpenSSL implementation for TLS protocols. SAS Viya supports the version provided for your operating system and the OpenSSL libraries installed. Those protocols are configurable and can use various ciphers depending on their version.

  Note: Refer to your operating system vendor documentation when using the vendor's OpenSSL libraries. There might be additional procedures that need to be followed to make the libraries work properly in your environment.

On Linux, SAS Viya supports TLS version 1.2. The default minimum protocol for OpenSSL is TLS 1.2.

On Windows, the Schannel SSP implementation of the TLS protocols use algorithms from a cipher suite to create keys and encrypt information. SAS Viya supports the Cipher Suites provided by Windows. A cipher suite is a set of cryptographic algorithms. Cipher suites can be negotiated only for TLS versions that support them. The highest supported TLS version is always preferred in the TLS handshake. Refer to Cipher Suites in TLS/SSL (Schannel SSP) for details about supported protocols of your Windows Operating System.

TLS Configuration

- On Linux, SAS Viya supports TLS using the operating system's OpenSSL libraries. In a full deployment of SAS Viya, almost all external network connections are secured by default. SAS Viya is deployed with Transport Layer Security (TLS) to secure network connections and is fully compliant with SAS security standards.
- On Windows, SAS Viya uses Schannel libraries that are provided by Windows.

See "How To" on page 3 for additional configuration to provide a higher level of security to a SAS Viya deployment.

TLS Terminology

The following concepts are fundamental to understanding TLS:

certificate authorities (CAs)
: Cryptography products provide security services by using digital certificates, public-key cryptography, private-key cryptography, and digital signatures. Certificate authorities (CAs) create and maintain digital certificates, which also help preserve confidentiality.

Various commercial CAs, such as VeriSign and Thawte, provide competitive services for the e-commerce market. You can also develop your own CA by using products from companies such as RSA Security and Microsoft or from the Open-Source Toolkit OpenSSL.

digital signatures
: A digital signature affixed to an electronic document or to a network data packet is like a personal signature that concludes a hand-written letter or that validates a credit card transaction. Digital signatures are a safeguard against fraud. A unique digital signature results from using a private key to encrypt a message digest. A document that contains a digital signature enables the receiver of the document to verify the source of the document. Electronic documents are said to be verified if the receiver knows where the document came from, who sent it, and when it was sent.

  Another form of verification comes from message authentication codes (MAC), which ensure that a signed document has not been changed. A MAC is attached to a document to indicate the document's authenticity. A document that contains a MAC enables the receiver of the document (who also has the secret key) to know that the document is authentic.

digital certificates
: Digital certificates are electronic documents that ensure the binding of a public key to an individual or an organization. Digital certificates provide protection from fraud.

  Usually, a digital certificate contains a public key, a user's name, and an expiration date. It also contains the name of the certificate authority (CA) that issued the digital certificate and a digital signature that is generated by the CA. The CA's validation of an individual or an organization allows that individual or organization to be accepted at sites that trust the CA.

public and private keys
: Public-key cryptography uses a public and a private key pair. The public key can be known by anyone, so anyone can send a confidential message. The private key is confidential and known only to the owner of the key pair, so only the owner can read the encrypted message. The public key is used primarily for encryption, but it can also be used to verify digital signatures. The private key is used primarily for decryption, but it can also be used to generate a digital signature.

symmetric key
: In symmetric key encryption, the same key is used to encrypt and decrypt the message. If two parties want to exchange encrypted messages securely, they must both have a copy of the same symmetric key. Symmetric key cryptography is often used for encrypting large amounts of data because it is computationally faster than asymmetric cryptography. Typical algorithms include DES, TripleDES, RC2, RC4, and AES.

asymmetric key
: Asymmetric or public key encryption uses a pair of keys that have been derived together through a complex mathematical process. One of the keys is made public, typically by asking a CA to publish the public key in a certificate for the certificate-holder (also called the subject). The private key is kept secret by the subject and never revealed to anyone. The keys work together where one is used to perform the inverse operation of the other: If the public key is used to encrypt data, only the private key of the pair can decrypt it. If the private key is used to encrypt, the public key must be used to decrypt. This relationship allows a public key encryption scheme where anyone can obtain the public key for a subject and use it to encrypt data that only the user with the private key can decrypt. This scheme also specifies that when a subject encrypts data using its private key, anyone can decrypt the data by using the corresponding public key. This scheme is the foundation for digital signatures.

Certificates Used by TLS and HTTPS

Overview of Certificates

Certificates are required for configuring TLS and HTTPS. TLS is used as the mechanism to provide encryption in-motion. Digital certificates are used in a network security system to guarantee that the two parties exchanging

information are really who they claim to be. Certificates are used to authenticate a server process or a human user. Digital certificates are issued and signed by a certificate authority (CA).

Configuring a server process to use TLS and HTTPS requires both a private key and a signed X.509 certificate. The signed server certificate also contains the public key.

A TLS server that is configured correctly should present to the client both the server certificate and any intermediate certificates. This means that the TLS client requires access only to the Root Certificate Authority certificate in order to establish trust of the server certificate.

In a SAS Viya deployment, certificates and security artifacts (certificate and key files, tokens) are provided at deployment. In a full deployment of SAS Viya, these certificate artifacts are used to provide security by default when the software is deployed. In a programming-only deployment of SAS Viya, security artifacts are provided to be used to configure and secure the software post-deployment.

When customers order a certificate through a commercial third party, they are submitting a certificate signing request (CSR). This CSR is a request to the certificate authority to sign the certificate and return that signed certificate to the customer for use.

A CA is an organization that verifies the information or the identity of computers on a network and issues digital certificates of authenticity and public keys. As part of a public key infrastructure (PKI), a CA checks with a registration authority to verify information provided by the requestor of a digital certificate. If the registration authority verifies the requestor's information, the CA can then issue a certificate.

There are three types of certificates that can be used to authenticate entities:

- third-party-signed

  You go to a commercial third-party certificate authority (VeriSign, GeoTrust, Thawte, DigiCert, Comodo, and so on), or a company can create its own CA and then use it to generate server and client certificates.

- site-signed

  You go to the IT department at your site to obtain a certificate.

- self-signed

  You serve as your own certificate authority.

Figure 2 *Types of Certificates and Who Acts as the CA for Each*

The CA signs the certificate that has been requested and does the following:

1. Ensures that the signing process is compatible with the private key type (either RSA or ECC). If you intend to use ECC certificates, the signing process also ensures that the certificates are generated correctly.
2. Provides the set of public X.509 certificates to establish the trust chain. The chain of trust must be established so that end-users will not see warnings in their browsers.

Multiple X.509 certificates are required for the chain of trust when there are site-signed or third-party-signed certificates. There should be the Root Certificate Authority X.509 certificate and a number of Intermediate Certificate Authority X.509 certificates.

A certificate chain is a sequence of certificates, where each certificate in the chain is signed by the subsequent certificate. The purpose of a certificate chain is to establish a chain of trust from a peer certificate to a trusted CA certificate. The CA vouches for the identity in the peer certificate when it signs it. If the CA is one that you trust (a copy of the CA certificate is in your root certificate directory), you can trust the signed peer certificate as well.

Server certificates are used to secure servers (most common are web servers) when you connect to sites that support HTTPS. A CA-signed server certificate is the type of certificate you would need to deploy if you do not want web browsers to display a warning when users attempt to connect to your secure server.

CA certificates are the certificates in your browser. Before any major web browser such as Chrome or Firefox connects to your server using HTTPS, it already has in its possession a set of certificates that can be used to verify the digital signature that will be found on your server certificate. These certificates are called CA (Certificate Authority) certificates. On these certificates is a copy of the public key of the CA that might issue (sign) your server certificate.

SAS Truststores

A SAS Viya deployment provides the SAS Security Certificate Framework that includes two truststores (two files), one in Base64 PEM encoded format (trustedcerts.pem) and one in a Java keystore format (trustedcerts.jks). The trustedcerts files are located on every machine in a SAS Viya deployment. These two files have the same content and are located in the following directories:

- On Linux and equivalent distributions, **`/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts`**
- On Windows, **`C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\cacerts`**

These two trustedcerts files contain the Mozilla bundle of trusted certificate authority (CA) certificates and other certificates added as part of the deployment process. Other certificates are shown in Table 10 on page 98.

Table 10 *Certificates That Can Be Included in the Truststore*

Type of Certificate	Description
Mozilla trusted certificate authority (CA) certificate bundle	Certificates from Mozilla that are provided by SAS Viya. These certificates are included in the trustedcerts.pem and the trustedcerts.jks files.
CA certificates issued by SAS Secrets Manager	Certificates generated using the SAS Secrets Manager. These are the Vault Root CA and intermediate certificates that are generated on the machines that have the SAS Secrets Manager deployed. These CA certificates are included in the file named vault-ca.crt. **Note:** The SAS Secrets Manager is deployed only in a full Linux Deployment.
Certificates generated by SAS	Certificates generated by SAS Viya. These certificates are usually named sas_encrypted.crt or sas.crt.
Any certificate chain. On a Linux deployment, on the Apache Server, this chain certificate is pointed to by HTTPD_CERT_PATH in the vars.yml file.	Certificates that are provided by SAS Viya or customer provided certificates added to the Apache server pre-deployment or post-deployment.
Custom certificates	Certificates that are provided by the customer.

Your web browser inherently trusts all certificates that have been signed by any root that has been embedded in the browser itself or in an operating system on which it relies.

In a Linux full deployment, during the Ansible deployment process, SAS Viya automatically obtains the HTTPS certificate from the Apache HTTP Server and adds this to the SAS Configuration Server under the key-value store named **`cacerts`**. Ansible uses the value of HTTPD_CERT_PATH to create an additional file in the SAS Security Certificate Framework under the **`cacerts`** directory. The deployment process then builds the trustedcerts files using the Mozilla bundle and these items. The trustedcerts files are distributed across all the hosts in the Ansible inventory.

Ensuring that the truststores are updated with any additional certificates is critical to a correctly operating environment. For information about managing the truststores, see "Manage Truststores" on page 65.

Apache HTTPD Certificates

Using Default Self-Signed Certificates Provided with SAS Viya Deployment (Linux)

An Apache HTTP Server is used as a reverse proxy server to secure your environment. The deployment process automatically installs Apache httpd on the machines that you designate as targets for the HTTP proxy installation unless it has already been installed. Apache httpd with the mod_ssl module is required in order to create the Apache HTTP Server, which provides security to the SAS Viya components.

The default Apache HTTP security settings use self-signed certificates provided by Apache. These settings are reasonably secure, but they are not compliant with SAS security standards. In a Linux full deployment of SAS Viya, the playbook inspects existing certificates and the CA chain to determine whether they comply with SAS security requirements. See "How SAS Viya Determines If Certificates Meet the SAS Security Standards on an Installed Linux HTTP Server" on page 100.

If certificates compliant with the SAS security standards are found, they are used without changes. If only the default mod_ssl certificates are found, the playbook generates SAS Viya self-signed certificates and configures mod_ssl to use them.

If you keep the self-signed certificates provided by SAS, end users see a standard web browser warning message.

CAUTION! SAS Viya self-signed certificates prior to the July 2019 release of SAS Viya are valid for only one year. In the July 2019 release of SAS Viya, the self-signed certificates provided by SAS Viya have a seven-year expiration time. Prior to the July 2019 release of SAS Viya, the self-signed certificates expire in only one year. Contact SAS Technical Support to renew the self-signed certificates before they expire. expiration time. Prior to the July 2019 release of SAS Viya, the self-signed certificates expired in only one year. Perform the task toupdate SAS Viya default self-signed certificate to extend the expiration date, or contact SAS Technical Support for help.

SAS recommends replacing the certificates before giving end users access to the software. SAS recommends that you install Apache httpd and configure the Apache HTTP Server to use certificates that comply with the security policies at your enterprise before you start the deployment process on a Linux deployment. In a SAS Viya full deployment on Linux, when you replace certificates before deployment of SAS Viya, the playbook automatically configures the certificates to secure the servers. See "Replace Self-Signed Certificates with Custom Certificates (Linux Pre-deployment)" on page 7.

Replacing the certificates can also be performed post-deployment on Linux. On Windows deployments, the Apache httpd certificates can be replaced only at post-deployment. See "Replace Self-Signed Certificates with Custom Certificates (Post-deployment)" on page 10.

The certificates and key files that the Apache HTTP Server uses are specified in the ssl.conf file or in the ssl-global-conf file on Linux or the httpd-ssl.conf file on Windows. The locations of these files are as follows:

- on RHEL and equivalent distributions, the ssl.conf file in **`/etc/httpd/conf.d/`**
- on SUSE Linux Enterprise Server 12.x, the ssl-global.conf file in **`/etc/apache2/`** and the vhost-ssl.conf file in **`/etc/apache2/vhosts.d/`**
- on Windows, the httpd-ssl.conf in **`C:\ProgramData\SAS\Viya\etc\httpd\conf\extra`**

The certificate and key files are specified using the directives in these configuration files. On Linux, the default certificate and key files provided by Apache are named localhost.crt and locahost.key. The certificate and key files provided by SAS Viya for Linux and Windows are named sas_encrypted.crt and sas_encrypted.key.

- The certificate filename is specified using directive SSLCertificateFile or SSLCertificateChainFile, or both.

  Note: In Apache HTTP Server version 2.4.8, the SSLCertificateFile directive was extended to load intermediate CA certificates from the server certificate file. This change enables you to use the

SSLCertificateFile directive instead of the SSLCertificateChainFile directive. See an explanation at SSLCertificateChainFile Directive.

- On a Red Hat Enterprise Linux and equivalent distributions, specify `SSLCertificateFile /etc/pki/tls/certs/`.
- On SUSE Linux Enterprise Server, specify `SSLCertificateFile /etc/apache2/ssl.crt/`.
- On Windows, specify `SSLCertificateFile C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\tls\certs\`.

- The certificate key file and path are specified using directive SSLCertificateKey.
  - The default RSA private key associated with certificates is named localhost.key.
    - On Red Hat Enterprise Linux and equivalent distributions, the certificate filename is set as `SSLCertificateKeyFile /etc/pki/tls/private/localhost.key`.
    - On SUSE Linux Enterprise Server, the certificate filename is set as `SSLCertificateKeyFile /etc/apache2/ssl.crt/localhost.key`.
    - On Windows, the certificate filename is set as `SSLCertificateKeyFile C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\private\sas.key`

How SAS Viya Determines If Certificates Meet the SAS Security Standards on an Installed Linux HTTP Server

Note: This information applies to a Linux Apache HTTP Server that is not initially set up by SAS. This section does not apply to a Windows deployment of SAS Viya.

During the deployment of SAS Viya on an existing Apache HTTP Server, the server identity certificates and the CA chain of certificates are investigated to determine whether they comply with SAS security requirements.

- Certificates that meet SAS security standards are those provided by the customer and signed by a trusted commercial CA or your own internal CA. In a certificate, the Basic Constraints extension cannot have the CA field set to false (CA:FALSE) to meet the SAS security standards.
- If certificates that do not meet SAS security standards are found, a self-signed certificate signed by SAS Viya is used, and mod_ssl is configured to use it. By default, mod_ssl issues a self-signed certificate where the Basic Constraints extension has the CA field set to false (CA:FALSE). This type of certificate does not meet the SAS security standards. Therefore, SAS Viya provides a self-signed certificate where the CA extension is excluded.

If you do not add compliant certificates, end users see a standard web browser warning message. SAS recommends replacing the non-compliant certificates before giving end users access to SAS Viya. You can add your own certificates pre-deployment on Linux or post-deployment on Linux and Windows. See "Replace Self-Signed Certificates with Custom Certificates (Linux Pre-deployment)" on page 7and "Replace Self-Signed Certificates with Custom Certificates (Post-deployment)" on page 10.

Note: Replacing your certificates post-deployment requires a brief outage.

Who Needs to Know about the Apache HTTPD Certificates?

The Apache HTTPD certificates are used by the deployment infrastructure. The following components must know about the Apache httpd certificates.

- SAS Configuration Server (Consul)
  - Each time that Consul is started, the sas-viya-httpproxy start script harvests all certificates found in the configuration files. In the Apache ssl.conf configuration file, directives SSLCertificateFile and SSLCertificateChainFile point to the files that contain any combination of the server, the root CA, and the intermediate CA certificates.

  - All of the certificates harvested when the sas-viya-httpproxy start script runs are stored in the Consul key value store. Ansible adds these certificates in the **/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts/** path. Ansible adds this certificate chain file as the file httpproxy-*host definition*-ca.crt. The *host definition* is taken from the inventory.ini file.
  - Any restart of the sas-viya-httpproxy service updates the certificates stored in Consul. If a certificate has been removed from the certificate chain, it is removed from Consul automatically when the sas-viya-httpproxy service is restarted.
- the HTTPD_CERT_PATH environment variable in the vars.yml file
  - The value of the environment variable is maintained by the user running the playbook.
  - When creating the certificate file that the HTTPD_CERT_PATH environment variable points to, the certificate file should be created as described in "Create a Certificate Chain of Trust for Apache httpd" on page 101.
  - On Red Hat Enterprise Linux and equivalent distributions, the path to the default location for the certificate chain files is as follows:

    ```
    HTTPD_CERT_PATH: '/etc/pki/tls/certs/customer.crt'
    ```

    On SUSE Linux Enterprise Server, the default location for certificates is as follows:

    ```
    HTTPD_CERT_PATH: '/etc/apache2/ssl.crt/customer.crt'
    ```

    Note: When you add custom certificates (either pre-deployment or post-deployment), set the environment variable to the path where the default certificates are located. Provide a file name other than the default name "localhost". The customer-provided certificate file that contains the certificate chain is named *customer.crt* for this example.
- the SAS truststores
  - The certificates in Consul are added, deleted, or updated in the SAS truststore each time the sas-viya-httpdproxy service is restarted.
  - In the vars.yml file, the certificates found at the location pointed to by HTTPD_CERT_PATH are distributed to all hosts and then added to each of the host truststores.
  - When the Ansible rebuild-trust-stores.yml and the distribute-httpd-cert.yml plays are run, the truststores are updated.

Create a Certificate Chain of Trust for Apache httpd

You need to create certificate chain of trust certificate files to update the certificates that Apache httpd uses. Then you set the configuration file directives SSLCertificateFile or SSLCertificateChainFile, or both, to point to those certificate files. These directives are set in configuration files ssl.conf file and ssl-global.conf on Linux, and httpd-ssl.conf on Windows.

Note: In Apache HTTP Server version 2.4.8, the SSLCertificateFile directive was extended to load intermediate CA certificates from the server certificate file. This change enables you to use the SSLCertificateFile directive for all of your certificates, including the chained certificate if you have an Apache HTTP Server version 2.4.8 or later. The SSLCertificateChainFile directive is then not needed. See an explanation at SSLCertificateChainFile Directive.

Here are the types of certificates that the certificate files should contain to ensure that there are no gaps in the certificate chain of trust.

1. Configure the SSLCertificateFile (or SSLCertificateChainFile directive if your Apache HTTP Server version is earlier than 2.4.8) to point to a certificate file that contains a full chain of certificates. This chain of certificates includes the server certificate, root CA certificate, and all intermediate CA certificates. This configuration provides the following results:

a Apache httpd is aware of the entire chain of certificates. Because Apache httpd knows the complete chain of certificates, the sas-viya-httpproxy service start script knows about the complete chain of certificates. Because the sas-viya-httpproxy service knows about the entire chain of certificates, all CA certificates in the chain are stored in Consul and also in the host truststores.

b Because the entire CA chain of trust is in Consul, and therefore also in each of the host truststores, the value for HTTPD_CERT_FILE in the vars.yml file can be left blank.

2 If your Apache HTTP Server version is earlier than 2.4.8, you can configure both the SSLCertificateFile directive and the SSLCertificateChainFile directive to point to two different certificate files that create a complete chain of trust. The SSLCertificateFile directive can point to a certificate file that contains only the server certificate. The SSLCertificateChainFile directive then points to a certificate file that contains the full chain of CA certificates (all intermediate CA certificates and the root CA certificate).

This configuration is functionally equivalent to the configuration described in Step 1 on page 101.

The following certificate configurations are problematic:

1 If you configure the SSLCertificateFile directive to point to a file that contains only the server certificate, and the SSLCertificateChainFile directive points to a file that contains a subset of the chain of CA certificates (for example, the file might contain only the signing intermediate CA certificate), the full chain of trust is not known. This configuration is not recommended and is problematic for the following reasons:

a In this configuration, the SSLCertificateChainFile directive is pointing to a certificate file that does not contain the root CA certificate and possibly does not contain other necessary intermediate CA certificates. The SSLCertificateChainFile might also not contain other necessary certificates.

b Because there are certificates missing, Apache httpd is not aware of the complete chain of trust.

c Because Apache httpd is not aware of the complete chain of trusted certificates, the sas-viya-httpproxy start script also does not know about the complete chain of trusted certificates. This gap in the chain of trusted certificates creates a problem in Consul and in the host truststores.

d Because a gap exists in the chain of trusted certificates, the vars.yml HTTPD_CERT_FILE value must be set. The certificate file that HTTPD_CERT_FILE points to must contain the remainder of the chain of trust.

e The missing certificates are needed to complete the chain of trust when building the truststores. The HTTPD_CERT_FILE file that contains the missing certificates restores continuity in the chain of trusted certificates.

2 If you configure the SSLCertificateFile directive to point to a file that contains only the server certificate, and the SSLCertificateChainFile directive is not set, there is a gap in the chain of trusted certificates. This configuration is problematic for the following reasons:

a This configuration is functionally equivalent to the configuration described in Step 1a on page 102.

b The file referenced by HTTPD_CERT_PATH contains only the server certificate. This certificate chain must contain the entirety of the chain of CA certificates, including the root CA certificate if only the SSLCertificateFile directive is used.

Certificates Issued by SAS Secrets Manager (Linux Full Deployment)

Note: SAS Secrets Manager is not used in a Linux programming-only deployment nor in a Windows deployment.

In a full deployment of SAS Viya, SAS Configuration Server (Consul) and SAS Secrets Manager are designed to be accessible only from within the SAS Viya environment. Consul and SAS Secrets Manager are not end-points that end users connect to. However, services and web applications connect to these end-points.

SAS Secrets Manager, which is based on HashiCorp Vault, is the trusted party that generates and signs root and intermediate TLS certificates. SAS Secret Manager generates both a Root Certificate Authority certificate and private key, as well as an Intermediate Certificate Authority certificate and private key. The Intermediate Certificate Authority is used to sign the individual server certificates.

These TLS certificates are used to secure communication between various SAS Viya processes. SAS Secrets Manager provides a point of contact for services that require certificates needed to maintain secured communication. These certificates are signed by a CA root and CA intermediate certificate created by SAS Secrets Manager. See Table 11 on page 103.

Note: SAS recommends installing a full deployment, which includes the product visual interfaces and microservices.

When SAS Secrets Manager is running, a SAS Viya service is deployed with an authentication token that allows it to contact SAS Secrets Manager as the service needs to. When this service is started, it uses its token to request a new certificate from SAS Secrets Manager and then secures its own internal port with that certificate. This process happens every time the service is started, as well as at the time of the deployment. Services can generate new certificates on any given start. The service handles all communication and interaction with SAS Secrets Manager programmatically. The service secures its end-points with the certificate provided by SAS Secrets Manager.

Certificates issued and key files are placed on the CAS controller, SAS/CONNECT server, SAS Configuration Server (Consul), SAS launcher server, SAS Message Broker (RabbitMQ), and SAS Infrastructure Data Server (PostgreSQL). For more information about servers in a deployment, see "Infrastructure Servers: Overview " in *SAS Viya Administration: Infrastructure Servers*.

Note: All the microservices have signed certificates that are stored in SAS Secrets Manager.

Table 11 *Certificate and Key Artifacts Provided by SAS Secrets Manager for a SAS Viya Full Deployment*

Security Artifact	Deployment File Name	Location	Description
Certificate truststore	trustedcerts.pem trustedcerts.jks	`/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts`	Contains the trusted list of CA certificates. These include the Mozilla bundle of trusted CA certificates, the SAS Secrets Manager site-signed certificates, the Apache httpd certificates, any custom certificates, and the chain of trust certificates.
Certificate file	vault-ca.crt	`/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts`	Contains the CA's certificates. It contains two certificates: the CA's root certificate and the CA's intermediate certificate.

Security Artifact	Deployment File Name	Location	Description
Certificate file	sas_encrypted.crt	`/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tls/certs`	Contains the root CA certificate issued by SAS Secrets Manager. It is sometimes referred to as the machine certificate. Each machine in a deployment has its own root CA certificate. This file is placed on all other machines in the deployment to allow those machines to trust this machine when they connect to it. This certificate file has a plaintext private key contained in file sas_encrypted.key.
Certificate private key file	sas_encrypted.key	`/opt/sas/viya/config/etc/SASSecurityCertificateFramework/private`	Contains the RSA private key associated with the public key. This RSA private key is encrypted. Its decryption key is the contents of the file encryption.key.
Certificate private key passphrase file	encryption.key (optional)	`/opt/sas/viya/config/etc/SASSecurityCertificateFramework/private`	This is the passphrase (or key) used to encrypt and decrypt the RSA private key in the file sas_encrypted.key.

In SAS Environment Manager, the interface for managing certificates is SAS Secrets Manager. SAS Secrets Manager is based on HashiCorp Vault 0.7.5. SAS Secrets Manager uses, stores, and generates secrets such as Transport Layer Security (TLS) certificates. For more information, see "SAS Secrets Manager (Linux)" in *SAS Viya Administration: Infrastructure Servers*.

Note: SAS Secrets Manager is installed on the same machines where SAS Configuration Server resides. SAS Configuration Server must be running for SAS Secrets Manager to be operational.

For information about the configuration properties that SAS Secrets Manager uses, see "Configuration Properties: Reference (Services)" in *SAS Viya Administration: Configuration Properties*.

Self-Signed Certificates Issued by SAS Viya

In a Linux programming-only or a Windows deployment of SAS Viya, self-signed certificates are provided for configuring TLS for the deployment.

Note: These self-signed certificate files are named the same as the SAS Secrets Manager certificate and key files provided in a full deployment of Linux. In a SAS Viya full deployment on Linux, these certificates are site-signed certificates that are signed by SAS Secrets Manager.

The following SAS self-signed certificates and key files are provided for the various machines in the deployment.

Table 12 Certificate and Key Artifacts Provided at Installation for SAS Viya Deployments

Security Artifact	Deployment File Name	Location	Description
Certificate truststore	trustedcerts.pem trustedcerts.jks	Linux: `/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts` Windows: `C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\cacerts\`	Contains the trusted list of CA certificates. These include the Mozilla bundle of trusted CA certificates, the SAS Viya self-signed certificates, the Apache httpd certificates, and the chain of trust certificates.
Certificate file	sas_encrypted.crt	`/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tls/certs` `C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\tls\certs`	Contains the server identity certificate generated by SAS Viya. These are self-signed certificates that are encrypted.
Certificate private key file	sas_encrypted.key	`/opt/sas/viya/config/etc/SASSecurityCertificateFramework/private` `C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\private`	Contains the private key generated by SAS Viya.
Certificate private key passphrase file	encryption.key (optional)	`/opt/sas/viya/config/etc/SASSecurityCertificateFramework/private` `C:\ProgramData\SAS\Viya\etc\SASSecurityCertificateFramework\private`	Contains the encrypted passphrase file provided by SAS Viya.

On Apache, SAS determines which certificates and keys to use during the deployment process. SAS determines whether to use the default Apache certificates, your own custom certificates, or the self-signed certificates provided by SAS. See "Apache HTTPD Certificates" on page 99.

Certificate File Formats

There are many file formats used to structure certificates. Here are some of them:

- encodings (also used as extensions)

  PEM

  Privacy Enhanced Email (.pem) is a container format (Base64-encoded x.509). The .pem extension is used for different types of X.509v3 files, which contain ASCII (Base64) armored data prefixed with a `-----BEGIN-----` line.

  Examples are CA certificate files or an entire certificate chain. This file can contain an issued public certificate, a public key, a private key, and intermediate and root certificates.

  The PEM file format is preferred by open-source software. It can have a variety of extensions (.pem, .key, .cer, .cert, and so on). For information about converting between file formats, see "Convert Digital Certificate File Formats Using OpenSSL" on page 82.

  DER

  Distinguished Encoding Rules (.der) is used for binary DER encoded certificates. A PEM file is just a Base64-encoded DER file. OpenSSL can convert these to PEM. DER supports storage of a single certificate. These files can also bear the .cer extension or the .crt extension. For information about converting between file formats, see "Convert Digital Certificate File Formats Using OpenSSL" on page 82.

  JKS

  JKS is a file format that is specific to Java. It is the Java keystore implementation. A keystore is a storage facility for cryptographic keys and certificates. Keytool is a key and certificate management utility that uses JKS as the file format of the key and certificate databases (KeyStore and TrustStores).

  PKCS#12 .P12

  Public-Key Cryptography Standards (.pkcs12) is a file format that has both public and private keys in the file and all certificates in a certification path. This container file is fully encrypted with a password-based symmetric key. PFX is a predecessor to PKCS#12.

  Note: The PKCS#12 format is the only file format that can be used to export a certificate and its private key.

  For information about converting between file formats, see "Convert Digital Certificate File Formats Using OpenSSL" on page 82.

- common extensions

  CRT

  The CRT extension is used for certificates. It supports storage of a single certificate. The certificates can be encoded as binary DER or as ASCII PEM. The CER and CRT extensions are nearly synonymous.

  Note: The only time CRT and CER can safely be interchanged is when the encoding type can be identical. For example, PEM-encoded CRT is the same as PEM-encoded CER.

  CSR

  This is a certificate signing request. Some applications can generate these for submission to certificate authorities. It includes some of the key details of the requested certificate, such as subject, organization, and state, as well as the public key of the certificate that will be signed. These are signed by the CA and a certificate is returned. The returned certificate is the public certificate. Note that this public certificate can be in a couple of formats.

  KEY

  The KEY extension is used both for public and private keys. The keys can be encoded as binary DER or as ASCII PEM.

SSH (Secure Shell)

SSH (Secure Shell) Overview

SSH is an abbreviation for Secure Shell. SSH is a protocol that enables users to access a remote computer via a secure connection. SSH is available through various commercial products and as freeware. OpenSSH is a free version of the SSH protocol suite of network connectivity tools.

Although SAS software does not directly support SSH functionality, you can use the tunneling feature of SSH to enable data to flow between a SAS client and a SAS server. Port forwarding is another term for tunneling. The SSH client and SSH server act as agents between the SAS client and the SAS server, tunneling information via the SAS client's port to the SAS server's port.

Linux operating systems can access an OpenSSH server on another Linux system. To access an OpenSSH server, Linux systems require OpenSSH software.

Windows systems require PuTTY software.

Currently, SAS supports the OpenSSH client and server that supports protocol level SSH-2 in Linux environments. Other third-party applications that support the SSH-2 protocol currently are untested. Therefore, SAS does not support these applications.

To understand the configuration options that are required for the OpenSSH and PuTTY clients and the OpenSSH server, it is recommended that you have a copy of the book *SSH, the Secure Shell: The Definitive Guide* by Daniel J. Barrett, Richard E. Silverman, and Robert G. Byrnes. This book is an invaluable resource when you are configuring the SSH applications, and it describes in detail topics that include public key authentication, SSH agents, and SSHD host keys.

SSH System Requirements

SAS supports SSH in these operating environments:

- Linux
- UNIX
- Windows
- z/OS

SSH Software Availability

OpenSSH supports SSH protocol versions 1.3, 1.5, and 2.0.

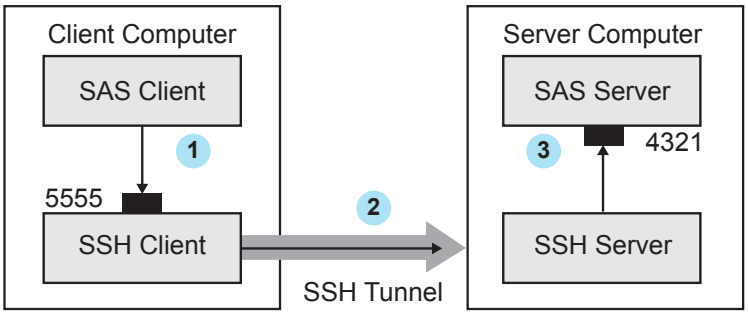
To build the OpenSSL software, refer to the following resources:

- www.openssh.com
- www.ssh.com
- PuTTY Download Page
- Barrett, Daniel J., Richard E. Silverman, and Robert G. Byrnes. 2005. *SSH, the Secure Shell: The Definitive Guide.* Sebastopol, CA: O'Reilly.

SSH Tunneling Process

An inbound request from a SAS client to a SAS server is shown as follows:

Figure 3 *SSH Tunneling Process*



1 The SAS client passes its request to the SSH client's port 5555.

2 The SSH client forwards the SAS client's request to the SSH server via an encrypted tunnel.

3 The SSH server forwards the SAS client's request to the SAS server via port 4321.

Outbound, the SAS server's reply to the SAS client's request flows from the SAS server to the SSH server. The SSH server forwards the reply to the SSH client, which passes it to the SAS client.

SSH Tunneling: Process for Installation and Setup

SSH software must be installed on the client and server computers. Exact details about installing SSH software at the client and the server depend on the particular brand and version of the software that is used. See the installation instructions for your SSH software.

The process for setting up an SSH tunnel consists of the following steps:

1 SSH tunneling software is installed on the client and server computers. Details about tunnel configuration depend on the specific SSH product that is used. On Linux, you use OpenSSH software to access your Linux OpenSSH server.

2 The SSH client is started as an agent between the SAS client and the SAS server.

3 The components of the tunnel are set up. The components are a listen port, a destination computer, and a destination port. The SAS client accesses the listen port, which is forwarded to the destination port on the destination computer. SSH establishes an encrypted tunnel that indirectly connects the SAS client to the SAS server.

Encrypting PDF Files Generated by ODS

You can use ODS to generate PDF output. When these PDF files are not password protected, any user can use Acrobat to view and edit the PDF files. You can encrypt and password-protect your PDF output files by specifying the PDFSECURITY= system option. Valid security levels for the PDFSECURITY= option are NONE or HIGH. SAS encrypts PDF documents using a 128-bit encryption algorithm. With PDFSECURITY=HIGH, at least one password must be set using the PDFPASSWORD= system option. A password is required to open a PDF file that has been generated with ODS.

Table 13 *PDF System Options*

Task	System Option
Specifies whether text and graphics from PDF documents can be edited.	PDFACCESS | NOPDFACCESS

Task	System Option
Controls whether PDF documents can be assembled.	PDFASSEMBLY | NOPDFASSEMBLY
Controls whether PDF document comments can be modified.	PDFCOMMENT | NOPDFCOMMENT
Controls whether the contents of a PDF document can be changed.	PDFCONTENT | NOPDFCONTENT
Controls whether text and graphics from a PDF document can be copied.	PDFCOPY | NOPDFCOPY
Controls whether PDF forms can be filled in.	PDFFILLIN | NOPDFFILLIN
Specifies the page layout for PDF documents.	PDFPAGELAYOUT=
Specifies the page viewing mode for PDF documents.	PDFPAGEVIEW=
Specifies the password to use to open a PDF document and the password used by a PDF document owner.	PDFPASSWORD=
Controls the resolution used to print the PDF document.	PDFPRINT=
Controls the printing permissions for PDF documents.	PDFSECURITY=

Encryption Algorithms

The following encryption algorithms are provided with SAS Viya:

SAS Proprietary for SAS data set encryption with passwords
: is a cipher that uses parts of the passwords that are stored in the SAS data set as part of the 32-bit rolling key encoding of the data. This encryption provides a medium level of security. With the speed of today's computers, it could be subjected to a brute force attack on the 2,563,160,682,591 possible combinations of valid password values, many of which must produce the same 32-bit key.

SAS Proprietary Encryption for communications
: is a cipher that provides basic fixed encoding services under all operating environments that are supported by SAS. The algorithm expands a single message to approximately one-third by using 32-bit fixed encoding. This encoding is used for passwords in configuration files, login passwords, internal account passwords, and so on.

RC2
: is a block cipher that encrypts data in blocks of 64 bits. A *block cipher* is an encryption algorithm that divides a message into blocks and encrypts each block. The RC2 key size ranges from 8 to 256 bits. SAS/SECURE uses a configurable key size of 40 or 128 bits. (The NETENCRYPTKEYLEN system option is used to configure the key length.) The RC2 algorithm expands a single message by a maximum of 8 bytes. RC2 is an algorithm developed by RSA Data Security, Inc.

RC4
: is a stream cipher. A *stream cipher* is an encryption algorithm that encrypts data one byte at a time. The RC4 key size ranges from 8 to 2048 bits. SAS/SECURE uses a configurable key size of 40 or 128 bits. (The

NETENCRYPTKEYLEN system option is used to configure the key length.) RC4 is an algorithm developed by RSA Data Security, Inc.

DES (Data Encryption Standard)
is a block cipher that encrypts data in blocks of 64 bits by using a 56-bit key. The algorithm expands a single message by a maximum of 8 bytes. DES was originally developed by IBM but is now published as a U.S. Government Federal Information Processing Standard (FIPS 46-3).

TripleDES
is a block cipher that encrypts data in blocks of 64 bits. TripleDES executes the DES algorithm on a data block three times in succession by using a single 56-bit key. This has the effect of encrypting the data by using a 168-bit key. TripleDES expands a single message by a maximum of 8 bytes. TripleDES is defined in the American National Standards Institute (ANSI) X9.52 specification.

AES (Advanced Encryption Standard)
is a block cipher that encrypts data in blocks of 128 bits by using a 256-bit key. AES expands a single message by a maximum of 16 bytes. Based on its DES predecessor, AES has been adopted as the encryption standard by the U.S. Government. AES is one of the most popular algorithms used in symmetric key cryptography. AES is published as a U.S. Government Federal Information Processing Standard (FIPS 197).

DSA (Digital Signature Algorithm)
The Digital Signature Algorithm (DSA) is a public-key (or assymetric-key) cryptography algorithm. A digital signature is a mathematical scheme for demonstrating the authenticity of a digital message or document. A DSA is used to compute and verify digital signatures. Essentially, the DSA helps verify that data has not been changed after it is signed, thus providing message integrity.

In 1994, the National Institute of Standards and Technology (NIST) issued a Federal Information Processing Standard for digital signatures, known as the DSA or DSS. This was adopted as FIPS 186 in 1993.

Elliptic Curve (ECC)
is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. ECC requires smaller keys compared to non-ECC cryptography (based on plain Galois fields) to provide equivalent security. Elliptic curves are applicable for encryption, digital signatures, pseudo-random generators, and other tasks.

MD5 (Message Digest)
is a series of byte-oriented algorithms that produce a 128-bit hash value from an arbitrary-length message. It is an algorithm used for hashing. It was developed by Rivest.

Note: This algorithm is not FIPS 140-2 compliant.

RSA (Rivest-Shamir-Adleman)
RSA is a public-key (or assymetric-key) cryptography algorithm and is widely used for secure data transmission. It is used for both encryption and authentication. Encryption and decryption are carried out using two different keys, the public key and the private key. A public-key system means that the algorithm for encrypting a message is publicly known, but the algorithm to decrypt the message is only privately known. In RSA, the public key is a large number that is a product of two primes, plus a smaller number. The private key is a related number.

SHA-1 (Secure Hash Algorithm)
produces a 160-bit (20-byte) hash value. An SHA-1 hash value is typically rendered as a hexadecimal number 40 digits long. This algorithm was developed by the U.S. National Security Agency (NSA) and published in 2001 by the NIST as a U.S. Federal Information Processing Standard (FIPS) PUB 180-1.

SHA-256 (Secure Hash Algorithm)
is essentially a 256-bit block cipher algorithm that encrypts the intermediate hash value using the message block as key. This algorithm was developed by the U.S. National Security Agency (NSA) and published in 2001 by the NIST as a U.S. Federal Information Processing Standard (FIPS) PUB 180-4.

SHA-384 (Secure Hash Algorithm)
: SHA384 is a truncated version of SHA512. Ir is essentially a 384-bit block cipher algorithm that encrypts the intermediate hash value using the message block as key. SHA-512 uses 64-bit words.

SHA-512 (Secure Hash Algorithm)
: is essentially a 512-bit block cipher algorithm that encrypts the intermediate hash value using the message block as key. SHA-512 uses 64-bit words.

Reference

SAS System Options for Encryption

This section contains the SAS System options that can be used to configure encryption. These options can be specified in a number of different ways: in configuration files (connect_usermods.sh file, connectserver_usermods.sh file, connect_usermods.bat file, connectserver_usermods.bat), in properties files, in SAS programs in the OPTIONS statement, on the SAS/CONNECT spawner command line, and in the SAS System Options window in SAS 9.

These system options are used for SAS/CONNECT and workspace servers. These system options are used in a SAS Viya programming-only deployment on Linux and on Windows deployments. In a SAS Viya full deployment, TLS is configured and enabled by default. Disable TLS on Port Families and Across the Deployment.

NETENCRYPT System Option

Specifies whether encryption is required for the connection.

NETENCRYPT | NONETENCRYPT

NETENCRYPT
: specifies that encryption is required.

NONETENCRYPT
: specifies that encryption is not required, but is optional.

Client	Optional
Server	Optional
Valid in	Configuration file, OPTIONS statement, SAS System Options window in SAS 9, SAS invocation, SAS/CONNECT spawner command line, connectserver_usermods.sh script
Category	Communications: Networking and Encryption
PROC OPTIONS GROUP=	Communications
Default	NONETENCRYPT
Operating environment	Linux
See	“NETENCRYPTALGORITHM= System Option” on page 112

The default for this option specifies that encryption is used if the NETENCRYPTALGORITHM= option is set and if both the client and the server are capable of encryption. If encryption algorithms are specified, but either the client or the server is incapable of encryption, then encryption is not performed.

Encryption might not be supported at the client or at the server in these situations:

- You are using a release of SAS (prior to SAS 8) that does not support encryption.
- Your site (the client or the server) does not have a security software product installed.
- You specified encryption algorithms that are incompatible in SAS sessions on the client and the server.

NETENCRYPTALGORITHM= System Option

Specifies the algorithm or algorithms to be used for encrypted client/server data transfers.

NETENCRYPTALGORITHM= *algorithm | ("algorithm-1"... "algorithm-n")*

algorithm | ("algorithm-1"... "algorithm-n")
specifies the algorithm or algorithms that can be used for encrypting data that is transferred between a client and a server across a network. These algorithms are specified on the server.

When you specify two or more encryption algorithms, use a space or a comma to separate them, and enclose the algorithms in parentheses.

The following algorithms can be used.

Note: The value specified for the TLS algorithim for this system option is SSL.

- AES
- DES
- RC2
- RC4
- TripleDES
- SASProprietary
- SSL

Client	Optional
Server	Optional
Valid in	Configuration file, OPTIONS statement, SAS System Options window in SAS 9, SAS invocation, SAS/CONNECT spawner command line, connectserver_usermods.sh script
Category	Communications: Networking and Encryption
PROC OPTIONS GROUP=	Communications
Alias	NETENCRALG=
Default	No algorithim is defined
Operating environment	Linux
Tip	The NETENCRYPTALGORITHM= option must be specified in the server session.
See	"NETENCRYPT System Option" on page 111
Example	`options netencryptalgorithm=(ssl);`

Use the NETENCRYPTALGORITHM= option to specify one or more encryption algorithms that you want to use to protect the data that is transferred across the network. If more than one algorithm is specified, the client

session negotiates the first specified algorithm with the server session. If the client session does not support that algorithm, the second algorithm is negotiated, and so on.

If either the client session or the server session specifies the NETENCRYPT option (which makes encryption mandatory) but a common encryption algorithm cannot be negotiated, the client cannot connect to the server.

If the NETENCRYPTALGORITHM= option is specified in the server session only, then the server's values are used to negotiate the algorithm selection. If the client session supports only one of multiple algorithms that are specified in the server session, the client can connect to the server.

There is an interaction between either NETENCRYPT or NONETENCRYPT and the NETENCRYPTALGORITHM= option.

Table 14 *Client/Server Connection Outcomes*

Server Settings	Client Settings	Connection Outcome
NONETENCRYPT NETENCRALG=*alg*	No settings	If the client is capable of encryption, the client/server connection is encrypted. Otherwise, the connection is not encrypted.
NETENCRYPT NETENCRALG=*alg*	No settings	If the client is capable of encryption, the client/server connection is encrypted. Otherwise, the client/server connection fails.
No settings	NONETENCRYPT NETENCRALG=*alg*	A client/server connection is not encrypted.
No settings	NETENCRYPT NETENCRALG=*alg*	A client/server connection fails.
NETENCRYPT or NONETENCRYPT NETENCRALG=*alg-1*	NETENCRALG=*alg-2*	Regardless of whether NETENCRYPT or NONETENCRYPT is specified, a client/server connection fails.

NETENCRYPTKEYLEN= System Option

Specifies the key length that is used by the encryption algorithm for encrypted client/server data transfers.

NETENCRYPTKEYLEN= 0 | 40 | 128

0
specifies that the maximum key length that is supported at both the client and the server is used.

40
specifies a key length of 40 bits for the RC2 and RC4 algorithms.

128
specifies a key length of 128 bits for the RC2 and RC4 algorithms. If either the client or the server does not support 128-bit encryption, the client cannot connect to the server.

Client	Optional
Server	Optional

Valid in	Configuration file, OPTIONS statement, SAS System Options window in SAS 9, SAS invocation, SAS/CONNECT spawner command line, connectserver_usermods.sh script
Category	Communications: Networking and Encryption
PROC OPTIONS GROUP=	Communications
Alias	NETENCRKEY=
Default	0
Operating environment	Linux

The NETENCRYPTKEYLEN= option supports only the RC2 and RC4 algorithms. The SAS Proprietary, DES, TripleDES, TLS, and AES algorithms are not supported.

By default, if you try to connect a computer that is capable of only a 40-bit key length to a computer that is capable of both a 40-bit and a 128-bit key length, the connection is made using the lesser key length. If both computers are capable of 128-bit key lengths, a 128-bit key length is used.

Using longer keys consumes more CPU cycles. If you do not need a high level of encryption, set NETENCRYPTKEYLEN=40 to decrease CPU usage.

SSLCACERTDIR= System Option

Specifies the location of the trusted certificate authorities (CA) found in OpenSSL format.

SSLCACERTDIR="*file-path*"

"*file-path*"
specifies the location where the public certificates for all of the trusted certificate authorities (CA) in the trust chain are filed. There is one file for each CA. Each CA certificate file must be PEM-encoded (base64). For more information, see "Certificate File Formats" on page 105.

The names of the files are the value of a hash that OpenSSL generates.

Note: OpenSSL generates different hash values for each OpenSSL version. For example, OpenSSL 0.9.8 generates different hash values than does OpenSSL 1.0.2.

OpenSSL looks up the CA certificate based on the x509 hash value of the certificate. SSLCACERTDIR= requires that the certificates are located in the specified directory where the certificate names are the value of a hash that OpenSSL generates.

If you are upgrading from a version of OpenSSL that is older than 1.0.0, you need to update your certificate directory links. Starting with code base 1.0.0, SHA hashing is used instead of MD5. You can use the OpenSSL C_REHASH utility to re-create symbolic links to files named by the hash values.

You can discover the hash value for a CA and then create a link to the file named after the certificate's hash value. Note that you must add ".0" to the hash value.

```
ln -s cacert1.pem 'openssl x509 -noout -hash -in
/u/myuser/sslcerts/cacert1.pem'.0
```

If you list the CA file, you see the link between the file named after the certificate's hash value and the CA file.

```
lrwxrwxrwx 1 myuser rnd 10 Apr 7 14:42 6730c6a9.0 -> cacert1.pem
```

To verify the path of the server certificate file (cacert1.pem for our example), use the following OpenSSL command:

```
openssl verify -CApath /u/myuser/sslcerts cacert1.pem
```

Client	Optional
Server	Optional
Valid in	Configuration file, SAS invocation, SAS/CONNECT spawner start-up, connectserver_usermods.sh script
Categories	Communications: Networking and Encryption
	System Administration: Security
Default	The default file and location for certificates is **`/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts/trustedcerts.pem`**. You can point to a different file and location using the SSLCALISTLOC= system option or the SSLCACERTDIR= system option. There is one trusted certificate file pointed to by the SSLCALISTLOC= system option. By contrast, the SSLCACERTDIR= system option allows the customer to specify a location where multiple certificate files reside. See "SSLCALISTLOC= System Option" on page 115.
Operating environment	Linux
Examples	The SSLCACERTDIR= system option points to the directory where the CA certificate is located. Export the environment variable on Linux hosts for the Bourne Shell: `export SSLCACERTDIR=/u/myuser/sslcerts/`
	Set the environment variable on Linux hosts for the C Shell directory where the CA certificates are located: `SETENV SSLCACERTDIR /u/myuser/sslcerts/`
	Set the environment variable at SAS invocation for Linux hosts: `-set "SSLCACERTDIR=/u/myuser/sslcerts/"`

For Foundation servers such as workspace servers and stored process servers, if certificates are used, SAS searches for certificates in the following order:

1. SAS looks for SAS system option SSLCALISTLOC= to find the file trustedcerts.pem.
2. SAS looks for the SSLCALISTLOC= environment variable to find the file trustedcerts.pem.
3. If the SSLCALISTLOC= system option or environment variable is not used, the trustedcerts.pem file located in */opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts* is used as the default.
4. If trustedcerts.pem exists, and the SSL_CERT_DIR and SSLCACERTDIR environment variables are set, SAS checks trustedcerts.pem first before it searches the directory.
5. If trustedcerts.pem does not exist, but the certificates are in the directory defined by SSL_CERT_DIR or SSLCACERTDIR, then SAS ignores SSLCALISTLOC=.
6. If trustedcerts.pem does not exist, and the SSL_CERT_DIR and SSLCACERTDIR environment variables are not set, SAS reports an error.

Note: A trusted CA certificate is required at the client in order to validate a server's digital certificate. The trusted CA certificate must be from the CA that signed the server certificate.

SSLCALISTLOC= System Option

Specifies the location of the public certificate(s) for trusted certificate authorities (CA).

SSLCALISTLOC="*file–path*"

"*file-path*"
specifies the location of a single file that contains the public certificate(s) for all of the trusted certificate authorities (CA) in the trust chain.

Note: Specify this option on the client. Optionally, specify this option on the server.

Client	Optional
Server	Optional
Valid in	Configuration file, OPTIONS statement, SAS System Options window in SAS 9, SAS invocation, SAS/CONNECT spawner command line, connectserver_usermods.sh
Category	Communications: Networking and Encryption
PROC OPTIONS GROUP=	Communications
Operating environment	Linux
Notes	If the SSLCALISTLOC= system option is not specified, SAS defaults to a file named trustedcerts.pem located in `/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts` The trustedcerts.pem file contains the list of trusted CA certificates provided by SAS at installation. If you use this option, it must be specified on the client, but does not have to also be specified on the server.

The SSLCALISTLOC= system option specifies the location of a single file that contains the public certificate(s) for all of the trusted certificate authorities (CA) in the trust list. The CA file must be PEM-encoded (base64).

The location of the trusted certificate file specified by the SSLCALISTLOC= system option or SSLCACERTDIR= system option or the trustedcerts.pem file is needed on the spawner to verify the certificate from the SAS/CONNECT server.

The default path set for the SSLCALISTLOC= system option on the workspace server is **`/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts/trustedcerts.pem`**. By default, the trustedcerts.pem file contains a managed set of trusted root certificates (Mozilla bundle of certificates and others) provided at SAS installation.

Note: The SSLCACERTDIR= system option can be used instead of using the SSLCALISTLOC= system option. SSLCACERTDIR= points to a directory that contains all of the public certificate file(s) of all CA(s) in the trust list. One file exists for each CA in the trust list. For more information, see "SSLCACERTDIR= System Option" on page 114.

For Foundation servers such as workspace servers and stored process servers, if certificates are used, SAS searches for certificates in the following order:

1. SAS looks for SAS system option SSLCALISTLOC= to find the file trustedcerts.pem.
2. SAS looks for the SSLCALISTLOC= environment variable to find the file trustedcerts.pem.
3. If the SSLCALISTLOC= system option or environment variable is not used, the trustedcerts.pem file located in */opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts* is used as the default.

4 If trustedcerts.pem exists, and the SSL_CERT_DIR and SSLCACERTDIR environment variables are set, SAS checks trustedcerts.pem first before it searches the directory.

5 If trustedcerts.pem does not exist, but the certificates are in the directory defined by SSL_CERT_DIR or SSLCACERTDIR, then SAS ignores SSLCALISTLOC=.

6 If trustedcerts.pem does not exist, and the SSL_CERT_DIR and SSLCACERTDIR environment variables are not set, SAS reports an error.

Note: A trusted CA certificate is required at the client in order to validate a server's digital certificate. The trusted CA certificate must be from the CA that signed the server certificate. The SSLCALISTLOC= option is required at the server only if the SSLCLIENTAUTH option is also specified at the server.

Note: Unless the SSLCACERTDIR= system option is set or the default trustedcerts.pem file is used, the SSLCALISTLOC= system option is needed on the spawner to verify the certificate from the SAS/CONNECT server.

SSLCACERTDATA= System Option

Specifies the name of the issuer of the digital certificate that TLS should use.

SSLCACERTDATA="*encoded-string*"

"*encoded-string*"
specifies the base64-encoded x509 text that represents a single certificate authority (CA) certificate. This string is in PEM format. The text string starts with the line "-----BEGIN CERTIFICATE-----" and ends with the line "-----END CERTIFICATE-----".

This option provides a way to programmatically specify a CA certificate rather than having to point to a file that contains the certificate information. The certificate must be PEM-encoded (base64) format.

Here is an example of how you might use the SSLCACERTDATA= system option to specify a certificate.

```
data _null_;
  length certInfo $3200.;
  input txt $67.;
  retain certInfo;

  if _N_ = 1 then
    certInfo=txt;
  else
    certInfo=catx('0a'x,certInfo,txt);
  call symput('certInfo',trim(left(certInfo)));
  datalines;
-----BEGIN CERTIFICATE-----
MIICbzCCAfagAwIBAgIJAP7q5/tk7+laMAoGCCqGSM49BAMCMHYxCzAJBgNVBAYT
AlVTMQswCQYDVQQIDAJOQzENMAsGA1UEBwwEQ2FyeTEWMBQGA1UECgwNU0FTIElu
c3RpdHV0ZTEMMAoGA1UECwwDSURCMSUwIwYDVQQDDBxkZW1vUm9vdENBLUVDRFNB
LVAzODQtU0hBMjU2MB4XDTE2MTEwNDE4MDMzMVoXDTI2MTEwMjE4MDMzMVowdjEL
MAkGA1UEBhMCVVMxCzAJBgNVBAgMAk5DMQ0wCwYDVQQHDARDYXJ5MRYwFAYDVQQK
DA1TQVMgSW5zdGl0dXRlMQwwCgYDVQQLDANJREIxJTAjBgNVBAMMHGRlbW9Sb290
Q0EtRUNEU0EtUDM4NC1TSEEyNTYwdjAQBgcqhkjOPQIBBgUrgQQAIgNiAAQghfjE
5iiiPQtb/Ors/GeNuLRXWnUhqnPWw4X0veIQT5rXFWZmiwReIjaYt9KChhmFkPno
cQ1m3HpdVnP86cPLPpLSvcAG/dO6o2W2SakiOWa1cA1UKsRhy/kUMnTSGJSjUDBO
MB0GA1UdDgQWBBSXhRRVQTNHpe1A9NsdUa+Y/IxhTTAfBgNVHSMEGDAWgBSXhRRV
QTNHpe1A9NsdUa+Y/IxhTTAMBgNVHRMEBTADAQH/MAoGCCqGSM49BAMCA2cAMGQC
MFJf5/2+eRSwCxrOyVjgyI4Teiofggrji5StKyQzHhDnXPljdYRss0WxxhbdBcxo
8wIwDjX8Yx61lY52U/h0q8ZkuNJWu0gJ8ZmrOVttkUBYUU0D1Cer6pdl4gQd6mUz
oXrB
-----END CERTIFICATE-----
```

```
    ;
run;

options SSLCACERTDATA="&certInfo";
```

SSLCERTISS= System Option

Specifies the name of the issuer of the digital certificate that TLS should use.

SSLCERTISS="*issuer-of-digital-certificate*"

"*issuer-of-digital-certificate*"
specifies the name of the issuer of the digital certificate that should be used by TLS.

Client	Optional
Server	Optional
Valid in	Configuration file, OPTIONS statement, SAS System Options window in SAS 9, SAS invocation, SAS/CONNECT spawner command line, connect_usermods.sh file, connectserver_usermods.sh file, connect_usermods.bat file, connectserver_usermods.bat file
Category	Communications: Networking and Encryption
PROC OPTIONS GROUP=	Communications
Windows specifics	Windows

SSLCERTLOC= System Option

Specifies the location of the digital certificate for the machine's public key. This is used for authentication.

SSLCERTLOC="*file-path*"

"*file-path*"
specifies the location of a file that contains a digital certificate for the machine's public key. The certificate must be PEM-encoded (base64). This is used by servers to send to clients for authentication.

Client	Optional
Server	Optional
Valid in	Configuration file, OPTIONS statement, SAS System Options window in SAS 9, SAS invocation, SAS/CONNECT spawner command line, connectserver_usermods.sh script
Category	Communications: Networking and Encryption
PROC OPTIONS GROUP=	Communications
Operating environment	Linux
Note	If you use this option, it must be specified on the server, but does not have to also be specified on the client.

The SSLCERTLOC= option is required for a server. It is required at the client only if the SSLCLIENTAUTH option is specified at the server. In order for a TLS connection to succeed, the SAS/CONNECT server needs to

be started with the SSLCERTLOC= and SSLPVTKEYLOC= system options set in the SAS/CONNECT spawner. Alternatively, the SSLPKCS12LOC= system option can be used.

In SAS Viya, set the TLS options on the spawner and the server in the connectserver_usermods.sh file (`/opt/sas/viya/config/etc/connectserver/default`) and in the connect_usermods.sh file (`/opt/sas/viya/config/etc/connect/default`). For configuration information, see "Use SAS/CONNECT with TLS Enabled to Import Data" on page 44.

SSLCERTSERIAL= System Option

Specifies the serial number of the digital certificate that TLS should use.

SSLCERTSERIAL="*serial-number*"

"serial-number"
specifies the serial number of the digital certificate that should be used by TLS.

The SSLCERTSERIAL= option is used with the SSLCERTISS= option to uniquely identify a digital certificate from the Microsoft Certificate Store. You can also use the SSLCERTSUBJ= option to identify a digital certificate instead of using the SSLCERTISS= and SSLCERTSERIAL= options.

Client	Optional
Server	Optional
Valid in	Configuration file, OPTIONS statement, SAS System Options window in SAS 9, SAS invocation, SAS/CONNECT spawner command line, connect_usermods.sh file, connectserver_usermods.sh file, connect_usermods.bat file, connectserver_usermods.bat file
Category	Communications: Networking and Encryption
PROC OPTIONS GROUP=	Communications
Windows specifics	Windows

SSLCERTSUBJ= System Option

Specifies the subject name of the digital certificate that TLS should use.

SSLCERTSUBJ="*subject-name*"

"subject-name"
specifies the subject name of the digital certificate that TLS should use.

Client	Optional
Server	Optional
Valid in	Configuration file, OPTIONS statement, SAS System Options window in SAS 9, SAS invocation, SAS/CONNECT spawner command line, connect_usermods.sh file, connectserver_usermods.sh file, connect_usermods.bat file, connectserver_usermods.bat file
Category	Communications: Networking and Encryption
PROC OPTIONS GROUP=	Communications
Windows specifics	Windows

SSLCIPHERLIST= System Option

Specifies the ciphers that can be used on Linux for OpenSSL.

SSLCIPHERLIST=*openssl_cipher_list*

openssl-cipher-list

The SSLCIPHERLIST= system option specifies the ciphers that can be used on Linux for OpenSSL. Refer to the OpenSSL Ciphers document to see how to format the *openssl-cipher-list* and for a complete list of the ciphers that work with your TLS version. The OpenSSL Ciphers information can be found at https://www.openssl.org/docs/man1.1.1/man1/ciphers.html.

Note: SAS does not support CAMELLIA, IDEA, MD2, and RC5 ciphers.

Note: The protocol and cipher information for the actual connection can be seen by setting `dumpCurrentCipherInfo` at the SAS DEBUG level.

Note: If you set a minimum protocol that does not allow some ciphers, you might get an error.

Client	Optional
Server	Optional
Valid in	Configuration file, command line
Categories	Communications: Networking and Encryption
	System Administration: Security
Restriction	If the SSLMODE= option is set, this option is ignored.
Operating environment	Linux
Notes	This option can also be specified as an environment variable.
	This system option must be set before TLS is loaded. It cannot be changed after TLS is loaded. You must set the environment variable before the SAS/CONNECT spawner is started and before SAS is started on the client.
Example	Specify the system option: `-SSLCIPHERLISTS= HIGH`

SSLCLIENTAUTH System Option

Specifies whether a server should perform client authentication.

SSLCLIENTAUTH | NOSSLCLIENTAUTH

SSLCLIENTAUTH

specifies that the server should perform client authentication. Server authentication is always performed, but the SSLCLIENTAUTH option enables a user to control client authentication. This option is valid only when used on a server.

> **TIP** If you enable client authentication, a certificate for each client is needed.

NOSSLCLIENTAUTH

specifies that the server should not perform client authentication.

Default	NOSSLCLIENTAUTH is the default.
Server	Optional
Valid in	Configuration file, OPTIONS statement, SAS System Options window in SAS 9, SAS invocation, SAS/CONNECT spawner command line, connectserver_usermods.sh script
Category	Communications: Networking and Encryption
PROC OPTIONS GROUP=	Communications
Operating environment	Linux
Note	If you use this option, it is specified on the server.

SSLCRLCHECK System Option

Specifies whether a Certificate Revocation List (CRL) is checked when a digital certificate is validated.

SSLCRLCHECK | NOSSLCRLCHECK

SSLCRLCHECK
specifies that CRLs are checked when digital certificates are validated.

NOSSLCRLCHECK
specifies that CRLs are not checked when digital certificates are validated.

Client	Optional
Server	Optional
Valid in	Configuration file, OPTIONS statement, SAS System Options window in SAS 9, SAS invocation, SAS/CONNECT spawner command line, connectserver_usermods.sh script
Category	Communications: Networking and Encryption
PROC OPTIONS GROUP=	Communications
Operating environment	Linux
Note	If you use this option, it can be specified on the client and server.
See	“SSLCRLLOC= System Option” on page 121

A certificate revocation list (CRL) is published by a certificate authority (CA) and contains a list of revoked digital certificates. The list contains only the revoked digital certificates that were issued by a specific CA.

The SSLCRLCHECK option is required at the server only if the SSLCLIENTAUTH option is also specified at the server. Because clients check server digital certificates, this option is relevant for the client.

SSLCRLLOC= System Option

Specifies the location of a certificate revocation list (CRL).

SSLCRLLOC=“*file-path*”

“file-path”
specifies the location of a file that contains a certificate revocation list (CRL).

Client	Optional
Server	Optional
Valid in	Configuration file, OPTIONS statement, SAS System Options window in SAS 9, SAS invocation, SAS/CONNECT spawner command line, connectserver_usermods.sh script
Category	Communications: Networking and Encryption
PROC OPTIONS GROUP=	Communications
Operating environment	Linux
Notes	If you use this option, it can be specified on the client and server.
	The SSLCRLLOC= option is required only when the SSLCRLCHECK option is specified.
See	“SSLCRLCHECK System Option” on page 121

SSLMODE= System Option

Sets the allowed TLS version and cipher suites to be used for TLS.

SSLMODE=*ssl-mode*

ssl-mode
sets the allowed TLS version and cipher suites to be used for TLS. Valid *ssl-mode*s are as follows:

SSLMODESUITEB128
is the mode of operation that uses the cipher-suites specified in the NIST Suite B Cryptography using 128 AES encryption.

SSLMODESUITEB192
is the mode of operation that uses the cipher-suites specified in the NIST Suite B Cryptography using 192 AES encryption.

SSLMODESP800131A
is the DEFAULT configuration mode for TLS communication. The SSLMODESP800131A mode is the mode of operation that uses the cipher-suites specified in the NIST Special Publication 800-131A.

SSLMODEDEPRECATED
allows for older cipher suites. These are cipher suites that do not meet NIST standards.

CAUTION! Cipher suites that do not meet NIST standards should not be used.

Client	Optional
Server	Optional
Valid in	Configuration file, command line, SAS/CONNECT spawner start-up if this option is used as an environment variable, connectserver_usermods.sh script
Categories	Communications: Networking and Encryption

	System Administration: Security
Default	SSLMODESP800131A
Restriction	If the SSLMODE= option is set, the SSLCIPHERLIST= system option is ignored.
Operating environment	Linux
See	For a list of ciphers that are supported for each of the modes that can be specified for the SSLMODE= system option, see SSLMODE= System Option Supported Ciphers.
Example	Specify the system option as follows: `-sslmode SSLMODESP800131A`

SAS uses the National Institute of Standards and Technology (NIST) Special Publication 800-131A (SP800-131A) as the minimum compliance standard for TLS and to extend the FIPS standards. TLS version 1.2 is the default version of TLS that SAS supports. For details of SP800-131A, see NIST Special Publication 800-131A, Revision 1.

Suite B cryptography allows TLS client and server applications to specify a profile compliant with Suite B cryptography as defined in RFC 5430: Suite B Profile for Transport Layer Security (TLS). Suite B cryptography specifies the cryptographic algorithms that can be used in a "Suite B Compliant" TLS V1.2 session. Suite B requires the key establishment and authentication algorithms that are used in TLS V1.2 sessions to be based on Elliptic Curve Cryptography, and the encryption algorithm to be AES.

For a list of ciphers that are supported for each of the modes that can be specified for the SSLMODE= system option, see SSLMODE= System Option Supported Ciphers.

SSLPKCS12LOC= System Option

Specifies the location of the PKCS#12 encoding package file.

SSLPKCS12LOC="*file-path*"

"file-path"
specifies the location of the PKCS#12 DER encoding package file that contains the certificate and the private key.

Client	Optional
Server	Optional
Valid in	Configuration file, OPTIONS statement, SAS System Options window in SAS 9, SAS invocation, SAS/CONNECT spawner command line, connectserver_usermods.sh script
Category	Communications: Networking and Encryption
PROC OPTIONS GROUP=	Communications
Operating environment	Linux
Notes	If you use this option, it can be specified on the client and server.
	You must specify both the SSLPKCS12LOC= option and the SSLPKCS12PASS= option together.

See	“SSLPKCS12PASS= System Option” on page 124

If the SSLPKCS12LOC= option is specified, the PKCS#12 DER encoding package must contain both the certificate and private key. The SSLCERTLOC= and SSLPVTKEYLOC= options are ignored.

You must specify both the SSLPKCS12LOC= option and the SSLPKCS12PASS= option in order for the SAS/CONNECT server to access the appropriate server scripts. In SAS Viya, set the TLS options on the spawner and the server in the connectserver_usermods.sh file (**`/opt/sas/viya/config/etc/connectserver/default`**) and in the connect_usermods.sh file (**`/opt/sas/viya/config/etc/connect/default`**). For configuration information, see “Use SAS/CONNECT with TLS Enabled to Import Data” on page 44.

SSLPKCS12PASS= System Option

Specifies the password that TLS requires to decrypt the PKCS12 file.

SSLPKCS12PASS=*password*

password
specifies the password that TLS requires in order to decrypt the PKCS#12 DER encoding package file. The PKCS#12 DER encoding package is stored in the file that is specified by using the SSLPKCS12LOC= option.

Client	Optional
Server	Optional
Valid in	Configuration file, OPTIONS statement, SAS System Options window in SAS 9, SAS invocation, SAS/CONNECT spawner command line
Category	Communications: Networking and Encryption
PROC OPTIONS GROUP=	Communications
Operating environment	Linux
Notes	If you use this option, it can be specified on the client and server.
	You must specify both the SSLPKCS12LOC= option and the SSLPKCS12PASS= option together.
See	“SSLPKCS12LOC= System Option” on page 123

The SSLPKCS12PASS= option is required only when the PKCS#12 DER encoding package is encrypted.

You must specify both the SSLPKCS12LOC= option and the SSLPKCS12PASS= option in order for the SAS/CONNECT server to access the appropriate server scripts. In SAS Viya, set the TLS options on the spawner and the server in the connectserver_usermods.sh file (**`/opt/sas/viya/config/etc/connectserver/default`**) and in the connect_usermods.sh file (**`/opt/sas/viya/config/etc/connect/default`**). For configuration information, see “Use SAS/CONNECT with TLS Enabled to Import Data” on page 44.

SSLPVTKEYLOC= System Option

Specifies the location of the private key that corresponds to the digital certificate.

SSLPVTKEYLOC="*file-path*"

"file-path"
specifies the location of the file that contains the private key that corresponds to the digital certificate that was specified by using the SSLCERTLOC= option.

Client	Optional
Server	Optional
Valid in	Configuration file, OPTIONS statement, SAS System Options window in SAS 9, SAS invocation, SAS/CONNECT spawner command line, connectserver_usermods.sh script
Category	Communications: Networking and Encryption
PROC OPTIONS GROUP=	Communications
Operating environment	Linux
Notes	If you use this option, it can be specified on the client and server.
	You must specify the SSLCERTLOC= option if you specify the SSLPVTKEYLOC= option. SSLPVTKEYPASS= is required only when the private key is encrypted.
See	"SSLCERTLOC= System Option" on page 118 and "SSLPVTKEYPASS= System Option" on page 125.

The SSLPVTKEYLOC= option is required at the server only if the SSLCERTLOC= option is also specified at the server.

The key must be PEM-encoded (base64). For more information, see "Certificate File Formats" on page 105.

You must specify both the SSLCERTLOC= option and the SSLPVTKEYLOC= option in order for the SAS/CONNECT server to access the appropriate server scripts. In SAS Viya, set the TLS options on the spawner and the server in the connectserver_usermods.sh file (`/opt/sas/viya/config/etc/connectserver/default`) and in the connect_usermods.sh file (`/opt/sas/viya/config/etc/connect/default`). For configuration information, see "Use SAS/CONNECT with TLS Enabled to Import Data" on page 44.

SSLPVTKEYPASS= System Option

Specifies the password that TLS requires for decrypting the private key.

SSLPVTKEYPASS="*password*"

"password"
specifies the password that TLS requires in order to decrypt the private key. The private key is stored in the file that is specified by using the SSLPVTKEYLOC= option.

Client	Optional
Server	Optional
Valid in	Configuration file, OPTIONS statement, SAS System Options window in SAS 9, SAS invocation, SAS/CONNECT spawner command line, connectserver_usermods.sh script

Category	Communications: Networking and Encryption
PROC OPTIONS GROUP=	Communications
Operating environment	Linux
Notes	If you use this option, it can be specified on the client and server.
	You must specify the SSLCERTLOC= option if you specify the SSLPVTKEYLOC= option. SSLPVTKEYPASS= is required only when the private key is encrypted.
See	"SSLCERTLOC= System Option" on page 118and "SSLPVTKEYPASS= System Option" on page 125.

The SSLPVTKEYPASS= option is required only when the private key is encrypted. OpenSSL performs key encryption.

Note: No SAS system option is available to encrypt private keys.

SSLSNIHOSTNAME= System Option

Enables the client to specify the Server Name Indication (SNI) in the TLS handshake that identifies the server name that it is trying to connect to.

SSLSNIHOSTNAME= "*hostname*"

"*hostname*"
specifies the host name that is used for the Server Name Indication (SNI) TLS extension. If it is not specified, the target host name is used. The client uses SNI in the first message of the TLS handshake (connection setup) to identify the server name that it is trying to connect to.

When making a TLS connection, the client requests a digital certificate from the web server. After the server sends the certificate, the client examines it and compares the name that it was trying to connect to with the name or names included in the certificate. If a match is found, the connection proceeds as normal.

Client	Optional
Server	Optional
Valid in	Configuration file, SAS invocation, SAS/CONNECT spawner start-up if this option is used as an environment variable, connectserver_usermods.sh script
Category	Communications: Networking and Encryption
PROC OPTIONS GROUP=	Communications
Default	The default is the name of the host being contacted.
Operating environment	Linux
Notes	This option can also be specified as an environment variable.
	The TLS SNI extension is always sent to the web server.
Example	Specify the system option as follows: `-SSLSNIHOSTNAME="www.example.org"`

SAS Environment Variables for Encryption

Overview of Environment Variables

Linux environment variables are variables that apply to both the current shell and to any subshells that it creates. The way in which you define an environment variable depends on the shell that you are running. For more information, see Defining Environment Variables in UNIX Environments.

SSL_USE_SNI Environment Variable

Disables the use of Server Name Indication (SNI) in the TLS handshake for the client.

SSL_USE_SNI

Linux clients and servers support TLS Server Name Indication (SNI). The client uses SNI in the first message of the TLS handshake (connection setup) to identify the server name that it is trying to connect to.

When making a TLS connection, the client requests a digital certificate from the web server. After the server sends the certificate, the client examines it and compares the name that it was trying to connect to with the name or names included in the certificate. If a match is found, the connection proceeds as normal.

Client	Optional
Server	Optional
Valid in	SAS invocation, configuration file
Categories	Communications: Networking and Encryption
	System Administration: Security
Default	By default, the TLS SNI extension is sent as part of the TLS handshake.
Restriction	System option SSLSNIHOSTNAME= is used to specify the Server Name Indication (SNI) that identifies the server name that it is trying to connect to. This environment variable is now used to turn off SNI. SNI is sent by default.
Operating environment	Linux
Examples	Export the environment variable on Linux hosts for the Bourne Shell: `export SSL_USE_SNI=1`
	Set the environment variable at SAS invocation for Linux hosts: `SETENV SSL_USE_SNI`

CAS TLS Environment Variables

CAS server options are stored in configuration files. For information about the CAS configuration files and when they are used, see “SAS Cloud Analytic Services: Reference” in *SAS Viya Administration: SAS Cloud Analytic Services*.

These are the configuration options that can be used for configuring TLS on CAS servers and clients.

env.CAS_CALISTLOC=<'*path*/*CA-list-file*'>

Specifies the location of a single file that contains the public certificate(s) for all of the trusted certificate authorities (CA) in the trust list. This is the CA list location when CAS is acting as a client.

Client	Optional
Valid in	Server configuration file, cas.settings file, cas configuration files, and operating system command line
Used by	CAS Server
Category	Security
Requirement	The certificate files and the key files being referenced by these environment variables must be PEM-encoded (Base64 ASCII).
Example	`env.CAS_CALISTLOC='/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts/trustedcerts.pem'`

env.CAS_CERTLOC=<'*path/certificate-file*'>

Specifies the path and filename of the file that contains the PEM-formatted certificate to be used for TLS communications.

This environment variable can also point to a certificate chain that starts with the server identity certificate and includes one or more intermediate CA certificates in the order in which they were signed.

Valid in	Server configuration file, cas.settings file, cas configuration files, and operating system command line
Used by	CAS REST API
Category	Security
Requirement	Use with env.CAS_PVTKEYLOC and env.CAS_PVTKEYPASS.
Example	`env.CAS_CERTLOC='/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tls/certs/cas/shared/default/sas_encrypted.crt'`

env.CAS_CLIENT_SSL_CA_LIST=<'*path/certificates-file*'>

Specifies the path and filename of the file that contains the list of trusted certificate authorities (CAs). This environment variable can be used by the CAS server or by the client connecting to the CAS server. For the server, this environment variable points to the trust list used to accept connections to the server. For the client, this environment variable points to the trust list that the client uses to connect to the server.

This environment variable might also need to be specified if you have a Linux 9.4m5 client connecting to a SAS Viya CAS server that is TLS enabled. If you have the December 2017 release of SAS 9.4M5, the CAS_CLIENT_SSL_CA_LIST= environment variable no longer needs to be set. For more information, see "Configure SAS 9.4 Clients to Work with SAS Viya" on page 59.

Client	Optional
Server	Optional
Valid in	CAS Lua configuration files, and operating system command line
Used by	CAS client, Lua client, Python client, CAS server, SAS 9.4 client
Category	Security
Example	`export CAS_CLIENT_SSL_CA_LIST='/opt/sas/viya/config/etc/SASSecurityCertificateFramework/cacerts/trustedcerts.pem'` `export CAS_CLIENT_SSL_CA_LIST='/opt/sas/viya/`

```
config/etc/SASSecurityCertificateFramework/cacerts/vault-ca.crt
export <SASHome>/SASSecurityCertificateFramework/1.1/
cacerts/trustedcerts.pem
```

env.CAS_CLIENT_SSL_CERT=<'*path*/*certificate-file*'>

Specifies the path and filename of the file that contains the certificate that the client uses to connect to the server for TLS communications. This environment variable is used when accepting connections to the CAS server.

This environment variable can also point to a certificate chain that starts with the server identity certificate and includes one or more intermediate CA certificates in the order in which they are signed.

Server	Optional
Valid in	Operating system command line
Used by	CAS server
Category	Security
Notes	Environment variables CAS_CLIENT_SSL_KEY=, CAS_CLIENT_SSL_KEYPWLOC=, and CAS_CLIENT_SSL_CERT= are specified together. The contents of this file are not confidential.
Example	`env.CAS_CLIENT_SSL_CERT='/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tls/certs/sas_encyrpted.crt'` `env.CAS_CLIENT_SSL_CERT='/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tls/certs/cas/default/sas_encrypted.crt'`

env.CAS_CLIENT_SSL_CLIENT_CERT=<'*path*/*certificate-file*'>

Specifies the path and filename of the file that contains the certificate that the client uses to connect to the server for TLS communications.

This environment variable is specified when the client presents a certificate to the server. In most configurations, only the server presents a certificate to the client.

This environment variable can also point to a certificate chain that starts with the server identity certificate and includes one or more intermediate CA certificates in the order in which they are signed.

Client	Optional
Valid in	Operating system command line
Used by	CAS client
Category	Security
Notes	The contents of this file are not confidential. Environment variables CAS_CLIENT_SSL_CLIENT_KEY=, CAS_CLIENT_SSL_CLIENT_KEYPW=, and CAS_CLIENT_SSL_CLIENT_CERT= are specified together.
Example	`env.CAS_CLIENT_SSL_CLIENT_CERT='/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tls/certs/CASClient.crt'` `env.CAS_CLIENT_SSL_CERT='/opt/sas/viya/config/etc/SASSecurityCertificateFramework/tls/certs/cas/default/CASClient.crt'`

env.CAS_CLIENT_SSL_CLIENT_KEY=<'*path*/*key-file*'>

Specifies the path and filename of the file that contains the private key for the client to use to connect to the server for TLS communications.

This environment variable is specified when the client presents a certificate to the server. In most configurations, only the server presents a certificate to the client.

Client	Optional
Valid in	Operating system command line
Used by	CAS client
Category	Security
Note	Environment variables CAS_CLIENT_SSL_CLIENT_KEY=, CAS_CLIENT_SSL_CLIENT_KEYPW=, and CAS_CLIENT_SSL_CLIENT_CERT= are specified together.
Tip	The contents of this file should be kept confidential.
Example	`env.CAS_CLIENT_SSL_CLIENT_KEY='/opt/sas/viya/config/etc/SASSecurityCertificateFramework/private/CASClient.key'` `env.CAS_CLIENT_SSL_CLIENT_KEY='/opt/sas/viya/config/etc/SASSecurityCertificateFramework/private/cas/default/CASClient.key'`

env.CAS_CLIENT_SSL_CLIENT_KEYPW=<'*password*'>

Specifies the password for the client's private key. The password in this variable should match the password used to generate the private key file specified by the CAS_CLIENT_SSL_CLIENT_KEY= environment variable.

Note: This password is not encoded.

This password should be set in a Lua configuration file that is readable only by the CAS service account.

Client	Optional
Valid in	Operating system command line
Used by	CAS client
Category	Security
Note	Environment variables CAS_CLIENT_SSL_CLIENT_KEY=, CAS_CLIENT_SSL_CLIENT_KEYPW=, and CAS_CLIENT_SSL_CLIENT_CERT= are specified together.
Example	`env.CAS_CLIENT_SSL_CLIENT_KEYPW='encryptedpassword'`

env.CAS_CLIENT_SSL_KEY=<*path*/*key-file*>

Specifies the path and filename of the file that contains the private key for the client to be used for TLS communications. This key is used when accepting connections to the server.

Server	Optional
Valid in	Operating system command line
Used by	CAS server
Category	Security

Note Environment variables CAS_CLIENT_SSL_KEY=, CAS_CLIENT_SSL_KEYPWLOC=, and CAS_CLIENT_SSL_CERT= are specified together.

Tip The contents of this file should be kept confidential.

Example

```
env.CAS_CLIENT_SSL_KEY='/opt/sas/viya/config/etc/
SASSecurityCertificateFramework/private/sas_encrypted.key'
```

```
env.CAS_CLIENT_SSL_KEY='/opt/sas/viya/config/etc/
SASSecurityCertificateFramework/private/cas/default/sas_encrypted.key'
```

env.CAS_CLIENT_SSL_KEYPWLOC==<'*path*/*certificate-file*'>

Specifies the location of the password file for the server's private key. The password in this variable should match the password used to generate the private key file specified by the CAS_CLIENT_SSL_KEY= environment variable.

Server Optional

Valid in Operating system command line

Used by CAS server

Category Security

Note Environment variables CAS_CLIENT_SSL_KEY= and CAS_CLIENT_SSL_CERT= are specified together.

Example

```
env.CAS_CLIENT_SSL_KEYPWLOC='/opt/sas/viya/
SASSecurityCertificateFramework/private/cas/default/encryption.key'
```

```
env.CAS_CLIENT_SSL_KEYPWLOC='/opt/sas/viya/
SASSecurityCertificateFramework/private/encryption.key'
```

env.CAS_CLIENT_SSL_REQUIRED=<'true' | 'false' >

Determines whether encryption is used between the client and the server.

Valid in Operating system command line

Used by CAS server

Category Security

Example

```
env.CAS_CLIENT_SSL_REQUIRED='true'
```

env.CAS_CLIENT_SSL_CERTISS=<"*issuer-name*">

Specifies the certificate issuer. *issuer-name* is the common name of the issuer of the server certificate to be used.

You can print the issuer of the client certificate using the following OpenSSL command.

```
C:\>openssl.exe x509 -in customerClient.pem -issuer -noout
```

Note the issuer of the certificate. For example, if `issuer= /DC=com/DC=Company/CN=Company SHA2 Issuing CA02`, you will need the value of `CN=`, which is `Company SHA2 Issuing CA02`. Save that value.

Valid in Server configuration file, cas.settings file, cas configuration files files and operating system command line

Category Security

Example

```
set env.CAS_CLIENT_SSL_CERTISS="Company SHA2 Issuing CA02"
```

env.CAS_CLIENT_SSL_CERTSERIAL=<"*serial-number*">

Specifies the certificate serial number. *serial-number* is a hexadecimal number.

You can use the following OpenSSL command to print the serial number of the client certificate.

```
C:\>openssl.exe x509 -in customerClient.pem -serial -noout
```

Note the value of serial= in the output. Save that numerical value. For example, if the output of the OpenSSL command is serial="190000AB8122B4DEC1D0AD1A7800000000AB57". Save the numeric string to set the environment variable.

Valid in	Server configuration file, cas.settings file, cas configuration files, and operating system command line
Category	Security
Example	

env.CAS_INTERNODE_DATA_SSL=<true | false>

Enables encryption for the analytics cluster when set to **`true`**. This value must be the same on every node in the cluster.

Valid in	Server configuration file, cas.settings file, and operating system command line
Category	Security
Example	`env.CAS_INTERNODE_DATA_SSL=true`

env.CAS_INTERNODE_SSL_CA_LIST=<'*path*/*keystore*'>

Specifies the path and filename of the file that contains the list of trusted certificate authorities (CAs). This setting is likely to be the same for all nodes in the grid.

Valid in	Server configuration file, cas.settings file, and operating system command line
Category	Security
Example	`env.CAS_INTERNODE_SSL_CA_LIST='/opt/sas/viya/config /etc/SASSecurityCertificateFramework/cacerts/trustedcerts.pem'`

env.CAS_INTERNODE_SSL_CERT=<'*path*/*certificate-file*'>

Specifies the path and filename of the file that contains the certificate to be used for TLS communications for the certificate specific to the node being configured.

This environment variable can point to a certificate chain that starts with the server identity certificate and includes the intermediate CA certificates in the order in which they are signed.

Valid in	Server configuration file, cas.settings file, and operating system command line
Category	Security
Example	`env.CAS_INTERNODE_SSL_CERT='/opt/sas/viya/config/etc/ SASSecurityCertificateFramework/tls/certs/sas_encrypted.crt'` `env.CAS_INTERNODE_SSL_CERT='/opt/sas/viya/config/etc/ SASSecurityCertificateFramework/tls/certs/cas/default/sas_encrypted.crt'`

env.CAS_INTERNODE_SSL_KEY=<'*path*/*key-file*'>

Specifies the path and filename of the file that contains the private key used to sign the certificate specific to the CAS node being configured. This setting is likely to be different on every machine.

Valid in	Server configuration file, cas.settings file, and operating system command line

Category Security

Tip The contents of this file should be kept confidential.

Example
```
env.CAS_INTERNODE_SSL_CERT='/opt/sas/viya/config/etc/
SASSecurityCertificateFramework/private/cas/default/sas_encrypted.key'
```
```
env.CAS_INTERNODE_SSL_CERT='/opt/sas/viya/config/etc/
SASSecurityCertificateFramework/private/sas_encrypted.key'
```

env.CAS_INTERNODE_SSL_KEYPW=<'*password*'>

Specifies the password for the private key.

The setting is the password for the encrypted private key used to sign the certificate specific to the node being configured.

Note: This password is not encoded.

Valid in Server configuration file, cas.settings file, and operating system command line

Category Security

Example
```
env.CAS_INTERNODE_SSL_KEYPW='encryptedpassword'
```

env.CAS_INTERNODE_SSL_KEYPWLOC =<'*path*/*certificate-file*'>

Specifies the location of the password/key file for the encrypted private key used to sign the certificate specific to the node being configured.

Valid in Server configuration file, cas.settings file, and operating system command line

Category Security

Example
```
env.CAS_INTERNODE_SSL_KEYPWLOC='opt/sas/viya/config/etc/
SASSecurityCertificateFramework/private/cas/default/encryption.key'
```
```
env.CAS_INTERNODE_SSL_KEYPWLOC='opt/sas/viya/config/
SASSecurityCertificateFramework/private/encryption.key'
```

env.CAS_PKCS12LOC=<'*path*/*certificate-file*'>

Specifies the path and filename of the PKCS#12 (DER formatted binary) file that contains the certificate and private key.

Valid in Server configuration file, cas.settings file, and operating system command line

Category Security

Example
```
env.CAS_PKCS12LOC='/opt/sas/viya/config/etc/
SASSecurityCertificateFramework/tls/certs/sas_encrypted.p12'
```
```
env.CAS_PKCS12LOC='/opt/sas/viya/config/etc/
SASSecurityCertificateFramework/tls/certs/cas/default/sas_encrypted.p12'
```

env.CAS_PKCS12PASS=<'*path*/*password-file*'>

Specifies the password for the private key specified by env.CAS_PKCS12LOC=.

Note: This password is not encoded.

Valid in Server configuration file, cas.settings file, and operating system command line

Category Security

Example `env.CAS_PKCS12PASS='password'`

env.CAS_PVTKEYLOC=<'*path*/*key-file*'>

Specifies the path and filename of the file that contains the private key that corresponds to the digital certificate.

Valid in Server configuration file, cas.settings file, and operating system command line

Used by CAS REST API

Category Security

Tip The contents of this file should be kept confidential.

Example
```
env.CAS_PVTKEYLOC='/opt/sas/viya/config/etc/
SASSecurityCertificateFramework/private/sas_encrypted.key'
```
```
env.CAS_PVTKEYLOC='/opt/sas/viya/config/etc/
SASSecurityCertificateFramework/private/cas/default/sas_encrypted.key'
```

env.CAS_PVTKEYPASS=<'*password*'>

Specifies the password for the private key specified by env.CAS_PVTKEYLOC=.

Note: This password is not encoded.

Valid in Server configuration file, cas.settings file, and operating system command line

Used by CAS REST API

Category Security

Example `env.CAS_PVTKEYPASS='password'`

env.CAS_PVTKEYPASSLOC=<'*path*/*key-file*'>

Specifies the path and filename of the file that contains the private key that corresponds to the digital certificate.

Valid in Server configuration file, cas.settings file, and operating system command line

Used by CAS REST API

Category Security

Tip The contents of this file should be kept confidential.

Example
```
env.CAS_PVTKEYLOC='/opt/sas/viya/config/etc/
SASSecurityCertificateFramework/private/encryption.key'
```
```
env.CAS_PVTKEYLOC='/opt/sas/viya/config/etc/
SASSecurityCertificateFramework/private/cas/default/encryption.key'
```

env.CAS_SSLCLIENTAUTH=<true>

When set to any value, causes client certificates to be validated when TLS connections are initiated.

Valid in Server configuration file, cas.settings file, and operating system command line

Category Security

Example `env.CAS_SSLCLIENTAUTH=true`

env.CAS_SSLCRLCHECK=<false>

When set to any value, causes the certificate revocation list (CRL) to be checked when TLS connections are initiated.

Valid in	Server configuration file, cas.settings file, and operating system command line
Used by	CAS REST
Category	Security
Example	`env.CAS_SSLCRLCHECK='false'`

env.CAS_SSLNAMECHECK=<true>

When set to any value, causes the name of the server to be checked against the host name specified in the server identity certificate pointed to by env.CAS_CERTLOC to validate the server's identity.

Valid in	Server configuration file, cas.settings file, and operating system command line
Used by	CAS REST API
Category	Security
Example	`env.CAS_SSLNAMECHECK=true`

env.CAS_USE_HTTPS_ALL=<'TRUE' | 'FALSE'>

When set to TRUE, causes connections using the CAS REST API to use HTTPS.

Valid in	Server configuration file, cas.settings file, CAS Lua config files, and operating system command line
Used by	CAS REST API
Category	Security
Default	FALSE
Example	`env.CAS_USE_HTTPS='FALSE'`

Configuration File Options for Data Transfer

CAS Configuration File Options for Data Transfer with the SAS Data Connect Accelerator

Note: The following configuration file options are supported only on Linux.

CAS server options are stored in a configuration file. During deployment, this configuration file, casconfig_deployment.lua, is created in the **`/opt/sas/viya/config/etc/cas/default`** directory. When you start the server with the `sas-viya-cascontroller-default start` command, the options are read.

For more information about the CAS server configuration files, see "Understanding Configuration Files and Start-up Files" in *SAS Viya Administration: SAS Cloud Analytic Services*.

These are the configuration options that can be used for data transfer encryption with the SAS Data Connect Accelerator. For a complete list of CAS configuration file options, see *SAS Viya Administration: SAS Cloud Analytic Services*.

cas.DCSSLCERTLOC='*pathname*'

Specifies the location of a file that contains the digital certificate for the machine's public key also known as the identity certificate. This is used by servers to send to clients for authentication.

Requirement	The certificate file must be PEM-encoded (base64).
Operating environment	Linux
See	"Certificate File Formats" on page 105

cas.DCSSLPKCS12LOC='*pathname*'

Specifies the location of the PKCS#12 DER encoding package file that contains the identity certificate and the private key.

Requirement	If the cas.DCSSLPKCS12LOC= option is specified, the PKCS#12 DER encoding package must contain both the certificate and private key. The cas.DCSSLCERTLOC= and cas.DCSSLPVTKEYLOC= options are ignored.
Operating environment	Linux
See	"Certificate File Formats" on page 105
	cas.DCSSLPKCS12PASS on page 136

cas.DCSSLPKCS12PASS=*password*

Specifies the password that TLS requires in order to decrypt the PKCS#12 DER encoding package file.

Interaction	The PKCS#12 DER encoding package is stored in the file that is specified by using the cas.DCSSLPKCS12LOC= option.
Operating environment	Linux
Note	The cas.DCSSLPKCS12PASS= option is required only when the PKCS#12 DER encoding package is encrypted.
See	cas.DCSSLPKCS12LOC on page 136

cas.DCSSLPVTKEYLOC='*pathname*'

Specifies the location of the file that contains the private key that corresponds to the digital certificate that was specified by using the cas.DCSSLCERTLOC= option.

Requirement	The key must be PEM-encoded (base64).
Operating environment	Linux
Note	The cas.DCSSLPVTKEYLOC= option is required at the server only if the cas.DCSSLCERTLOC= option is also specified at the server.
See	"Certificate File Formats" on page 105
	cas.DCSSLCERTLOC on page 136
	cas.DCSSLPVTKEYPASS on page 136

cas.DCSSLPVTKEYPASS=*password*

Specifies the password that TLS requires in order to decrypt the private key.

Interaction	The private key is stored in the file that is specified by using the cas.DCSSLPVTKEYLOC= option.
Operating environment	Linux
Note	The cas.DCSSLPVTKEYPASS= option is required only when the private key is encrypted. OpenSSL performs key encryption.
See	cas.DCSSLPVTKEYLOC on page 136

cas.DCSSLPVTKEYPASSLOC='*pathname*'

Specifies the location of the file that contains the password that TLS requires in order to decrypt the private key.

Interactions	The private key is stored in the file that is specified by using the cas.DCSSLPVTKEYLOC= option.
	If the cas.DCSSLPVTKEYPASS= option is specified, it is used. Otherwise, the cas.DCSSLPVTKEYPASSLOC= option is used.
Operating environment	Linux
See	cas.DCSSLPVTKEYPASS on page 136

cas.DCTCPMENCRYPT='YES' | 'NO' | 'OPT'

Specifies whether encryption is required for the connection.

'YES' means that data encryption is required.

'NO' means that data will be sent as plaintext.

'OPT' means that data encryption is preferred but not required.

Aliases	'REQ' or 'REQUIRED' for 'YES'
	'OPTIONAL' for 'OPT'
Default	No value. However, if you specify cas.DCTCPMENCRYPTALGORITHM='SSL' and cas.DCTCPMENCRYPT= is not specified, cas.DCTCPMENCRYPT= defaults to 'YES'.
Requirement	The option values must be uppercase.
Interactions	Encryption is determined by the setting of this option on both the client (data provider) and server (CAS) side. For more information, see “DCTCPMENCRYPT Option Setting Interaction” on page 57.
	If you specify cas.DCTCPMENCRYPTALGORITHM='SSL' and cas.DCTCPMENCRYPT is not specified, cas.DCTCPMENCRYPT defaults to 'YES'.
Operating environment	Linux
Note	If you have multiple clusters and you set the DCTCPMENCRYPT= option on the client (data provider) side to YES for one cluster and NO for another cluster, you might want to set the server (CAS) side cas.DCTCPMENCRYPT= option to 'OPT'.

cas.DCTCPMENCRYPTALGORITHM='SSL'

Specifies the algorithm to be used for encrypted data transfers using the SAS Data Connect Accelerator.

Default	SSL
Requirement	The option value, SSL, must be uppercase.
Interaction	If you specify cas.DCTCPMENCRYPTALGORITHM='SSL' and cas.DCTCPMENCRYPT= is not specified, cas.DCTCPMENCRYPT= defaults to 'YES'.
Operating environment	Linux
Note	TLS (option value is SSL) is the only algorithm available at this time for encrypted data transfers using the SAS Data Connect Accelerator.

dcsecurity.properties File Options for Data Transfer with the SAS Data Connect Accelerator

You can set the following options in the dcsecurity.properties file. The dcsecurity.properties file is located in the following directory on your cluster.

- For Teradata, **`/opt/SAS/SASTKInDatabaseServerForTeradata/`*`ep-version`*`/security`**
- For Hadoop, ***`EPInstallDir`*`/sasexe/SASEPHOME/security`**

The syntax for setting the properties is as follows:

```
-option-name
option-setting
```

Here is an example.

```
-DCTCPMENCRYPTALGORITHM SSL
```

These are the options that you can set in the dcsecurity.properties file options for data transfer encryption with the SAS Data Connect Accelerator.

DCSSLCACERTDIR '*pathname*'

Specifies the directory where the public certificates for all of the trusted certificate authorities (CA) in the trust list are filed.

Requirement	Each CA certificate file must be PEM-encoded (base64).
Interaction	The DCSSLCALISTLOC= option can be used instead of or in conjunction with the DCSSLCACERTDIR= option.
Note	Different versions of OpenSSL generate different hash values. For example, OpenSSL 0.9.8 generates different hash values from those generated by OpenSSL 1.x.
See	“Certificate File Formats” on page 105

DCSSLCALISTLOC '*pathname*'

Specifies the location of a single file that contains the public certificate(s) for all of the trusted certificate authorities (CA) in the trust list.

Requirement	The CA file must be PEM-encoded (base64).
Interaction	The DCSSLCACERTDIR= option can be used instead of or in conjunction with the DCSSLCALISTLOC= option.
See	“Certificate File Formats” on page 105

DCTCPMENCRYPT YES | NO | OPT

Specifies whether encryption is required for the connection.

YES means that data encryption is required.

NO means that data will be sent as plaintext.

OPT means that data encryption is preferred but not required.

Aliases	REQ or REQUIRED for YES
	OPTIONAL for OPT
Default	NO. However, if you specify the DCTCPMENCRYPTALGORITHM option and DCTCPMENCRYPT is not specified, DCTCPMENCRYPT defaults to YES.
Requirement	The option values must be uppercase.
Interaction	Encryption is determined by the setting of this option on both the client (data provider) and server (CAS). For more information, see "DCTCPMENCRYPT Option Setting Interaction" on page 57.

DCTCPMENCRYPTALGORITHM SSL

Specifies the algorithm to be used for encrypted data transfers using the SAS Data Connect Accelerator.

Default	SSL
Requirement	The option value, SSL, must be uppercase.
Note	TLS is the only algorithm available at this time for encrypted data transfers using the SAS Data Connect Accelerator.

Examples

Use OpenSSL to Create Site-Signed or Third-Party-Signed Certificates in PEM Format

Generate a Private Key in RSA Format and a Certificate Signing Request

The tasks that you perform to request a digital certificate for the CA, the server, and the client are similar. However, the values that you specify are different.

In this example, Proton, Inc. is the organization that is applying to become a CA. A certificate request is sent to a certificate authority to get it signed, thereby becoming a CA. After Proton, Inc. becomes a CA, it can serve as a CA for issuing other digital certificates to clients and servers on its network. The certificates generated by the Proton, Inc. CA are considered site-signed certificates.

Note: You can also sign the certificate yourself if you have your own certificate authority or create a self-signed certificate.

To create a site-signed certificate using OpenSSL, first you need to generate a private key in RSA format. This file is not protected with a passphrase and is saved in the ASCII (Base64-encoded) PEM format.

1. Edit your existing openssl.cnf file or create an openssl.cnf file. OpenSSL by default looks for a configuration file in **`/usr/lib/ssl/openssl.cnf`**. It is good practice to add `-config ./openssl.cnf` to the commands OpenSSL CA or OpenSSL REQ to ensure that OpenSSL is reading the correct file.

Note: You can find where the openssl.cnf file is located by submitting the following OpenSSL command:

```
openssl version -d
```

.

Here is an example of some of the information that can be specified in the openssl.cnf file. You need to specify where OpenSSL should look for information. Here is a partial file example. Much more information about certificates can be specified.

Figure 4 *Example of an OpenSSL.cnf File*

```
File  Edit  Format  View  Help
#
# OpenSSL example configuration file.
# This is being used for generation of certificate requests.
#

##################################################################
[ ca ]
default_ca      = CA_default            # The default ca section

##################################################################
[ CA_default ]

dir             = ./demoCA              # where everything is kept
certs           = $dir/certs            # where the issued certs are kept
crl_dir         = $dir/crl              # where the issued crl are kept
database        = $dir/index.txt        # database index file.
new_certs_dir   = $dir/newcerts         # default place for new certs.

certificate     = $dir/cacert.pem       # The CA certificate
serial          = $dir/serial           # The current serial number
crl             = $dir/crl.pem          # The current CRL
private_key     = $dir/private/cakey.pem# The private key
RANDFILE        = $dir/private/.rand    # private random number file

x509_extensions = usr_cert              # The extentions to add to the cert

default_days    = 365                   # how long to certify for
default_crl_days= 30                    # how long before next CRL
default_md      = sha1                  # which sha to use.
preserve        = no                    # keep passed DN ordering
policy          = policy_match

# For the CA policy
[ policy_match ]
```

2 Select the `apps` subdirectory of the directory where OpenSSL was built.

3 Initialize OpenSSL.

```
$ openssl
```

4 Issue the appropriate command to request a digital certificate. In the following example, we are creating an RSA private key and generating a certificate signing request all at once.

Table 15 *OpenSSL Commands for Requesting a Private Key*

Recipient of Certificate Request	OpenSSL Command
CA	`req -config ./openssl.cnf -new -out ca.csr -newkey rsa:2048 -keyout cakey.pem -sha256`

Recipient of Certificate Request	OpenSSL Command
Server	`req -config ./openssl.cnf -new -out server.csr -newkey rsa:2048 -keyout serverkey.pem -sha256`
Client	`req -config ./openssl.cnf -new -out client.csr -newkey rsa:2048 -keyout clientkey.pem -sha256`

Table 16 Arguments and Values Used in OpenSSL Commands

OpenSSL Arguments and Values	Functions
`req`	Requests a certificate.
`-config ./openssl.cnf`	Specifies the storage location for the configuration details for the OpenSSL program.
`-new`	Identifies the request as new.
`-out ca.csr`	Specifies the storage location for the certificate request.
`-newkey rsa:2048`	Generates a new private key along with the certificate request that is 2048 bits in length using the RSA algorithm.
`-keyout cakey.pem`	Specifies the storage location for the private key.
`-nodes`	Prevents the private key from being encrypted. This option is not recommended. For best practice, encrypt the private key.
`-sha256`	Specifies that the SHA-256 hash algorithm be used. Without this option, the default is SHA-1.

5 Informational messages are displayed and prompts for additional information appear according to the specific request.

To accept a default value, press the Enter key. To change a default value, type the appropriate information and press the Enter key.

Note: Unless the -NODES option is used in the OpenSSL command when creating a digital certificate request, OpenSSL prompts you for a password before allowing access to the private key.

Here is an example of a request for a digital certificate:

```
OpenSSL> req -config ./openssl.cnf -new -out ca.req -newkey rsa:2048
-keyout privkey.pem -nodes
Using configuration from ./openssl.cnf
Generating a 2048 bit RSA private key
.........................++++++
.........................................++++++
writing new private key to 'cakey.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
```

```
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:
State or Province Name (full name) [North Carolina]:
Locality Name (city) [Cary]:
Organization Name (company) [Proton Inc.]:
Organizational Unit Name (department) [IDB]:
Common Name (YOUR name) []: proton.com
Email Address []:Joe.Bass@proton.com

Please enter the following 'extra' attributes to be sent with
your certificate request
A challenge password []:
An optional company name []:
OpenSSL>
```

The request for a digital certificate is complete.

Note: For the server, the Common Name must be the name of the computer that the server runs on. In our examples, we are using proton.com.

Generate a Public Certificate

1 Issue the appropriate command to generate a public certificate from the certificate signing request.

Table 17 OpenSSL Commands for Generating Digital Certificates

Recipient of Generated Certificate	OpenSSL Command
CA	`x509 -req -in ca.csr -signkey cakey.pem -out cacert.pem -sha256` **Note:** This command generates a self-signed certificate.
Server	`ca -config ./openssl.cnf -in server.csr -out server.pem -md sha256` **Note:** This command creates certificates signed by the CA. These are defined in the openssl.cnf file.
Client	`ca -config ./openssl.cnf -in client.csr -out client.pem -md sha256` **Note:** This command creates certificates signed by the CA. These are defined in the openssl.cnf file.

Table 18 Arguments and Values Used in OpenSSL Commands to Generate a Certificate

OpenSSL Arguments and Values	Functions
`x509`	Identifies the certificate display and signing utility. Typically used to generate a self-signed certificate.
`-req`	Specifies that a certificate be generated from the request.

OpenSSL Arguments and Values	Functions
`ca`	Identifies the Certificate Authority utility.
`-config ./openssl.cnf`	Specifies the storage location for the configuration details for the OpenSSL utility.
`-in filename.csr`	Specifies the storage location for the input for the certificate request.
`-out filename.pem`	Specifies the storage location for the certificate.
`-signkey cakey.pem`	Specifies the private key that is used to sign the certificate that is generated by the certificate request.
`-md sha256`	Specifies that the SHA-256 hash algorithm be used. Without this option, the default is SHA-1.

2 Informational messages are displayed and prompts for additional information appear according to the specific request.

To accept a default value, press the Enter key. To change a default value, type the appropriate information, and press the Enter key.

Here is a sample of the messaging from a CSR for a server digital certificate:

Note: The password is for the CA's private key.

```
Using configuration from ./openssl.cnf
Enter PEM pass phrase: password
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
countryName           :PRINTABLE:'US'
stateOrProvinceName   :PRINTABLE:'NC'
localityName          :PRINTABLE:'Cary'
organizationName      :PRINTABLE:'Proton, Inc.'
organizationalUnitName:PRINTABLE:'IDB'
commonName            :PRINTABLE:'proton.com'
Certificate is to be certified until April 16 17:48:27 2016 GMT (365 days)
Sign the certificate? [y/n]:y
1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries Data Base Updated
```

The subject's Distinguished Name is obtained from the digital certificate request.

The generation of a digital certificate is complete.

Check Your Digital Certificate Using OpenSSL

To check a digital certificate, issue the following command:

```
openssl> x509  -text -in filename.pem
```

A digital certificate contains data that was collected to generate the digital certificate timestamps, a digital signature, and other information. However, because the generated digital certificate is encoded (usually in PEM format), it is unreadable.

Create a Certificate Chain Using OpenSSL

After generating a digital certificate for the CA, the server, and the client (optional), you must identify for the client application one or more CAs that are to be trusted. This list is called a *chain of trust*. This chain includes a set of certificates in which each one has been signed by the one that comes after it.

On the client, if there is only one CA to trust, specify in the client application the name of the file that contains the CA digital certificate. If multiple CAs are to be trusted, you can copy and paste into a new file the contents of all the digital certificates of CAs to be trusted by the client application. These CAs can be primary, intermediate, or root certificates. Add the root CAs to the client's truststore. For the server, do not include the root CA in the server's certificate chain.

Generally, .pem files are returned by default when you use OpenSSL to generate certificates. Most CAs return .crt files (Microsoft returns .cer files). SAS Viya provides Base64 encoded certificate files with the .crt extension. You can add these files together (regardless of your file extension) by manually cutting and pasting the files together or you can concatenate the certificate files together. For example, you can take an intermediate authority certificate file, a root authority certificate file, and a primary certificate file and concatenate them into a single file.

Here is an example of concatenating certificates into one file. To manually create a certificate chain of trust, use the following template.

```
(Your Server Certificate - ssl.crt)

-----BEGIN CERTIFICATE-----

<PEM encoded certificate>

-----END CERTIFICATE-----

(Your Intermediate CA Certificate(s))

-----BEGIN CERTIFICATE-----

<PEM encoded certificate>

-----END CERTIFICATE-----

(Your Root CA Certificate)

-----BEGIN CERTIFICATE-----

<PEM encoded certificate>

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

The content of the digital certificate in this example is represented as `<PEM encoded certificate>`. The content of each digital certificate is delimited with a `-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----` pair. All text outside the delimiters is ignored. Therefore, you might not want to use delimited lines for descriptive comments.

```
cat server.crt > certchain.pem
cat intermediateCA.crt >> certchain.pem
cat rootCA.pem >> certchain.pem
```

Because the digital certificate is encoded, it is unreadable. You will see a string of hexadecimal characters. To view the file contents, you can use the following OpenSSL commands to view the contents of your file by type:

```
openssl x509 -in cert.pem -text -noout
```

```
openssl x509 -in cert.cer -text -noout
openssl x509 -in cert.crt -text -noout
```

Use the following OpenSSL command to view a DER-encoded certificate:

```
openssl x509 -in certificate.der -inform der -text -noout
```

Note: If you are including a digital certificate that is stored in DER format into your certificate chain, you must first convert it to PEM format. For more information, see "Convert DER to PEM File Format" on page 82.

Verify Certificates in the Trust Chain Using OpenSSL

Clients and servers exchange and validate each other's digital certificates. All of the CA certificates that are needed to validate a server certificate compose a trust chain. All CA certificates in a trust chain have to be available for server certificate validation.

You can use the following OpenSSL command to verify that certificates are signed by a recognized certificate authority (CA):

```
openssl verify -verbose -CAfile <your-CA_file>.pem <your-server-cert>.pem
```

If your local OpenSSL installation recognizes the certificate or its signing authority and everything checks out (dates, signing chain, and so on), you get a simple `OK` message.

Troubleshooting TLS

SSL Error: Invalid subject name in partner's certificate.

This message is generated when trying to sign-on to a SAS/CONNECT spawner. SAS 9.4M4 was unable to start a SAS/CONNECT session to the Viya 3.3 CAS server. The server certificate needs server alternative name (SAN) extension entries in the server certificate for each name the host can be known by. The certificate needs the physical host name and DNS alias listed in the SAN.

```
NOTE: Remote signon to MYSERV commencing (SAS Release 9.04.01M4P110916).
ERROR: A communication subsystem partner link setup request failure
has occurred.
ERROR: Network request failed (rc 0x00000001105AB1B0) -
SSL Error: Invalid subject name in partner's certificate.
Subject name must match machine name.
ERROR: Remote signon to MYSERV canceled.
```

In this case, the sas-crypto-management tool needs to be used to generate a new default certificate that includes the SAN information. Contact SAS Technical Support for assistance.

service [SSL_CERTIFICATE_REISSUE] The SSL certificate is scheduled to be reissued at <date and time>

The TLS certificate generated for the Java microservice is not automatically being reissued when the certificate nears its expiration of one year. Hotfix 63529 is available to fix this issue.

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. Copyright © 2019, SAS Institute Inc., Cary, NC, USA. All Rights Reserved. July 2022 3.4-P1:calencryptmotion