Avid® MediaCentral Platform Services

Creating Trusted Certificates for Single Servers

Important Information

The procedures in this document only apply to single-server installations. If you have a cluster configuration, refer to the instructions in the "*Creating Trusted Certificates for Clustered Systems*" document.

Secure Sockets Layer (SSL) certificates can be generated by the local system or obtained by a trusted Certificate Authority (CA) group. The latter has the advantage in that the certificate is automatically trusted by browsers. No warnings appear when a connection is made to a secure web page, nor do you need to add them to the Trusted Root Certification Authorities store (since the CA's root certificate is already there).

If you obtain a certificate/signature from a CA, or are yourself a CA, Avid recommends obtaining a Subject Alternative Name (SAN) certificate. SAN certificates can have a number of associated host names, domain names, IP addresses, etc., all within the same certificate.

As of November 2015, many Certification Authorities will only issue certificates that include the FQDN (Fully Qualified Domain Name). Short host names will no longer be accepted due to potential risks. If your Trusted CA-signed certificate is issued by an in-house source, short host names and IP addresses are acceptable through a Subject Alternative Name (SAN) certificate.

Generating a trusted certificate involves the following steps:

- Generating a Certificate Signing Request (CSR)

  This step creates the CSR file required by Trusted Certificate Authorities.
- Verifying and Submitting the CSR

  Ensures that the CSR file is valid before submitting it to a Certificate Authority.
- Installing the Trusted CA-Signed Certificate

  This process adds the certificate to the MediaCentral server.
- Updating the application.properties File

  In this step, the custom password information is added to the MCS configuration file.

Generating a Certificate Signing Request (CSR)

The process for obtaining a certificate varies with each CA, but it always begins with generating a Certificate Signing Request (CSR) from your local system.

This procedure requires the following information:

- Two-letter ISO country code (e.g. US, CA, DE)
- State, Province, Prefecture, etc. (spelled out – no abbreviations)
- City, locality or jurisdiction (e.g. Paris)
- Organization Name (e.g. Avid Technology)
- Organizational Unit (e.g. IT)
- Common Name: FQDN (e.g. mcs.mydomain.com) of the MCS server
- Email Address: Contact email address (optional)
- Challenge password (optional)
- Optional Company Name (optional)

To generate the Certificate Signing Request (CSR):

1. Log into Linux on the MCS server as the *root* user.
2. Change to a secure directory, for example, /root.

   ```
   cd /root
   ```

3. Generate a CSR and private key using the *openssl* command:

   ```
   openssl req -out jetty.csr -new -newkey rsa:2048 -nodes -keyout
   jettyPrivateKey.key
   ```

Since December 2010, most CAs require a key length of 2048 bits, as shown in the above example. Fewer bits are considered insecure, and may be rejected by the CA

4. Answer the questions as prompted.

   Your responses are inserted into the CSR, and will appear in the CA-signed SSL certificate. The details are visible to end-users when they view certificate details in a browser.

   The *openssl* command produces two files: *jetty.csr* and *jettyPrivateKey.key*

   - **jetty.csr**: This is the CSR you will submit to the CA. It contains the information you entered, and a public key. It does not contain your private key.
   - **jettyPrivateKey.key**: This is your enterprise's private key for the certificate just created.

     Keep the private key in a safe place (such as the root user home directory, /root). You will need it later to add the CA-signed key to the keystore.

Although your private key will be stored in the in the Jetty keystore, Jetty does not provide a means for extracting the key. Should you lose your private key, use a third-party tool, such as jksExportKey, to extract it from the Jetty keystore.

The CSR (jetty.csr) file has contents similar to the following:

```
-----BEGIN NEW CERTIFICATE REQUEST-----
XXXCuTCCAaECAQAwdDELMAkGA1UEBhMCQ0ExDzANBgNVBAgTBlF1ZWJlYzERMA8GA1UEBxMITW9u
dHJlYWwxDTALBgNVBAoTBEF2aWQxCzAJBgNVBAsTAlFBMSUwIwYDVQQDExxoYS1pY3BzLTAyLmds
b2JhbC5hdmlkd3cuY29tMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAjoSSGeAskozI
G454aQ6RK3qOKix4gYapEAGgzJ+f+6LN/j0U5sIznW2F5RG047ChMiMdENVK2v4jlI1RtHozD4Yd
RO/xHVBejmP+SDAhtfNcXX2ThZYzGbHWm7mcxOnyXH5vU0KVWaWfz9K3Oh1FPwEgI6T4sb1TBY7h
F8r7uObeoUbXnaXObGHJnSzlX/PC3YcsHPPI4BFAmK/6oQpgUbnN+L7y7oBCIBwv5tY3Z16q9vgy
V9H8NpEIEkh9anJXWlo13aGeMYa0yM4g2cRsiBGAPRkmvx2lYUBf5TOcSk9mFZAZRudK1j5+mCtY
VljC9q5cZzL0IZ52U+kbW1IrRwIDAQAtT7bwDQYJKoZIhvcNAQEFBQADggEBAEObhmbSxAPwfyD7
jO6uR6R/1oN2fu29bxx9yTMBS8OeiLb1NcSaBAPxxcZaaHnmXeKIzh0ReHXt4GUNXGOL2HYVKjLa
OFqY9mhUgrEdIIEbbpXtOI41qzdQqP/CCv5j6fx8M5gdCVZghtk0+G+MME92e4YSib9Ghs+WVXCj
1uwzr4VwlpsaSAvGpNhLV6wTHgeYcGoKOi6gycdTNPIySc+KDGfZFVCAeg6nDqkanjvYUKInPByC
s3cD8B+ZxvHSIzRf4mbPSOm596XxpiaiJUEYo9jNWZjgdGJghlI2SKryjkWG8wJMWjDBGaHzcXXX
ET/c/veHF0c2XNjFSU/hHEA=
-----END NEW CERTIFICATE REQUEST-----
```

The private key (jettyPrivateKey.key) file is structured similarly.

Verifying and Submitting the CSR

Before submitting the CSR to your CA, you should validate its contents using one of the many online CSR validation tools. Generally, this involves cutting and pasting the contents of the CSR file into a web browser and viewing the parsed results. To ensure the CSR complies with your CA's requirements, use their CSR validation tool, if possible.

Alternatively, you can check the CSR on your local system through the same *openssl* command used to generate the keys. The verification command will use different options.

Once verified, you can submit the files to the Certificate Authority.

To verify the contents of the CSR:

1. Use the *openssl* command to check the contents of the key:

   ```
   openssl req -text -noout -verify -in jetty.csr
   ```

   The system should respond with a "`verify OK`" message and a structured dump of the CSR contents.

2. Double check the validity of the private key you created:

   ```
   openssl rsa -in jettyPrivateKey.key -check
   ```

   The system should respond with a "`RSA key ok`" message and a structured dump of the private key contents.

To submit the CSR to the Certificate Authority:

1. Submit the CSR to your CA, and take receipt of the CA-signed Certificate.

   The CA will provide you with a concatenated certificate container file. This file contains the CA-signed certificate. Its contents resemble the following (abbreviated for convenience):

   ```
   -----BEGIN CERTIFICATE-----
   XXXXfDCCBGSgAwIBAgIQfju3hLvGVKvSuNZ37MOUqDANBgkqhkiG9w0BAQUFADCB
   jDELMAkGA1UEBhMCVVMxFzAVBgNVBAoTDlZlcmlTaWduLCBJbmMuMTAwLgYDVQQL
   EydGb3IgVGVzdCBQdXJwb3NlcyBPbmx5LiAgTm8gYXNzdXJhbmNlcy4xMjAwBgNV
   BAMTKVZlcmlTaWduIFRyaWFsIFNlY3VyZSBTZXJ2ZXIgUm9vdCBDQSAtIEcyMB4X
   DTA5MDQwMTAwMDAwMFoXDTE5MDMzMTIzNTk1OVowgcsxCzAJBgNVBAYTAlVTMRcw
   FQYDVQQKEw5WZXJpU2lnbiwgSW5jLjEwMC4GA1UECxMnRm9yIFRlc3QgUHVycG9z
   ZXMgT25seS4gIE5vIGFzc3VyYW5jZXMuMUIwQAYDVQQLEzlUZXJtcyBvZiB1c2Ug
   CCsGAQUFBwEMsdfEwX6FdoFswWTBXMFUWCWltYWdL2dpZjAhMB8wBwYFKw4DAhoE
   FI/l0xqGrI2Oa8PPgGrUSBgsexkuMCUWI2h0dHA6Ly9sb2dvLnZlcmlzaWduLmNv
   bS92c2xvZ28uZ2lmMB0GA1UdDfjdsBQoFxOKvdaitdwGLLe2jtoQZmBu5TAfBgNV
   HSMEGDAWgBRIGeeSb5KdNGOZwPCZyNaljIx/ZTANBgkqhkiG9w0BAQUFAAOCAQEA
   NgvA9cj2h5yFC2SJMmE8a9trUmjnorZWO/Ifmdf5ADuQuf+k8arodHpdSeq/f2Gj
   wDIo3oYL2bT/66tw46Kx3Q/Z02pp7YW+BRvKejBYXN9FJxsXEkPKpz4SRvSQLl5Y
   Bst7q03nK0lZQ4/LE+hufgvx08JdqGd4o4cOvZ6o4MQaMgX/0lwNjC+4PWuKfmrK
   mr+RhpSkc72cEEO9//XsYesTMetY3nOuqXAQqiH4lz3KAzqNgohXXcF8W1F3ytTT
   q1yWbMkJonRpXbE3U0rWI33ywiS7LAk8n4dR8tNoqqIrP6sDtvolx9/+qTPWGnYy
   tV0P/hcJt5CbqE7008EnPQ==
   -----END CERTIFICATE-----
   ```

The CA can supply you the file as a PEM (.pem), a PXF (.pxf) or using another file extension but the certificate content is what truly matters for you to create the Trusted CA-signed certificate.

   The concatenated certificate container file and CA-signed certificate it contains is a public document. It does not contain any private information and does not need to be stored securely. However, it is recommended you keep it in a safe place such as the root user home directory (/root).

2. Verify the contents of the file in human-readable form:

   ```
   openssl x509 -in <filename> -noout -text
   ```

   Where `<filename>` is the name of the file provided to you by the CA.

Technical information for the certificate includes the issuer (e.g. VeriSign, Thawte, Digicert), the certificate's validity, public key, and the information you supplied in the CSR.

Installing the Trusted CA-Signed Certificate

Once you have obtained an SSL certificate that has been signed by a recognized Certificate Authority (CA) you can add it to the MCS server's Jetty keystore.

This procedure requires that you have the following files:

- Certificate file (.pem, .pxf, or other): The file containing the CA-signed certificate.
- Key file (.key): The file containing the private used to generate the CSR.

The following procedures require the creation of the following passwords:

- **Export Password** (used in step 6 below)

  This is required when you use *openssl* to combine the two files into a PKCS file. This password protects the PKCS file and it also protects the certificate once it is inside the keystore.

  In another procedure, you will add this password to the MediaCentral application.properties file, so it can be used by MediaCentral to serve (i.e. export) the certificate from the keystore.

- **Destination Keystore Password** (used in step 9 and 10 below)

  This is required when you use *keytool* to create the Jetty keystore.

  Similarly, in another procedure you will add this password to the MediaCentral application.properties file, so it can be used by MediaCentral to gain access to the keystore.

For simplicity, it is recommended you use the same value for both the Export and Destination Keystore passwords.

This procedure requires that you know the following passwords:

- **Import Password** (used in step 7 below)

  This is the same password as the Export Password. It is required when you use openssl to verify the PKCS file is created correctly.

- **Certificate Password** (used in step 8 below)

  This is the password used to create the certificate file in the previous procedure.

- **Source Keystore Password** (used in step 9 below)

  This is the same password as the Export Password. It is required when you use keytool to create the Jetty keystore.

To install the signed certificate:

1. Log into Linux on the MCS server as the *root* user.
2. Change to the directory containing the private key you generated to obtain the CSR:

   ```
   cd /root
   ```

3. In case you need to revert to the original configuration, Avid recommends backing up the current Jetty keystore file:

   ```
   cp /opt/avid/etc/avid/avid-interplay-central/ssl/jetty.keystore /tmp/
   jetty.keystore.bak
   ```

Do not back up the keystore to the /ssl directory. Jetty reads all files in the keystore directory and unnecessary files can result in errors.

4. Copy the certificate file (e.g. jetty.pem) received from the CA to your current directory.
5. Combine the certificate and private key files into a single Public-Key Cryptography Standards (PKCS or .pkcs12) file:

   ```
   openssl pkcs12 -inkey <privatekeyfile> -in <certificatefile> -export
   -out jetty.pkcs12
   ```

The key file you specify in the above command must be the same one that was used to generate the CSR.

6. When prompted, enter and verify your Export Password.
7. Verify that the PKCS file was created correctly:

   ```
   openssl pkcs12 -info -in jetty.pkcs12
   ```

   Enter your Import Password as prompted.

   A structured dump of the PKCS file is displayed. Verify its contents, including the “issuer” values.

8. At the end of the dump, you are prompted to enter the Certificate Password (pass phrase). Enter the password used to create the private key, to obtain an encrypted output of the private key.
9. Create the Jetty keystore, and import the file you created into it:

   ```
   keytool -importkeystore -srckeystore jetty.pkcs12 -srcstoretype PKCS12 -
   destkeystore jetty.keystore
   ```

   At the prompts, provide your Destination Keystore Password for the Jetty keystore and enter the Source Keystore Password.

   For simplicity, it is suggested you use the same password for both.

10. Before proceeding, verify the contents of the keystore:

    ```
    keytool -list -keystore jetty.keystore
    ```

At the prompt enter the Destination Keystore Password.

The contents of the keystore are displayed, in a structured form, similar to the following:

```
Keystore type: JKS

Keystore provider: SUN

Your keystore contains 1 entry

1, Apr 5, 2013, PrivateKeyEntry,

Certificate fingerprint (MD5):
3C:35:2F:D5:40:8F:CF:18:4C:9A:BE:F1:9C:15:2C:D3
```

Take note of the MD5 fingerprint. You can use it later when browsing to MediaCentral to verify the correct certificate is being served.

11. Stop the Avid Interplay Central service:

    ```
    service avid-interplay-central stop
    ```

    You will restart the service after updating the application.properties file in the next section.

Be aware that this step will disconnect any users currently working on the system.

12. Copy the new Jetty keystore containing the CA certificate to its final location:

    ```
    cp jetty.keystore /opt/avid/etc/avid/avid-interplay-central/ssl/
    jetty.keystore
    ```

Updating the application.properties File

Once the certificate is created, you must update the MediaCentral configuration to make use of the new certificate. The *application.properties* file is used to add custom configuration changes to the MediaCentral system.

In this procedure you obtain obfuscated (disguised) passwords from Jetty and add them to the MediaCentral *application.properties* file. This allows MediaCentral to make use of the SSL certificate.

Be sure to add both the “password” and the “keypassword” to the file (similar to the following):

```
system.org.ops4j.pax.web.ssl.password=OBF\:1c3x1mf71jnb1sov1jk71mbf1c35
system.org.ops4j.pax.web.ssl.keypassword=OBF\:1c3x1mf71jnb1sov1jk71mbf1c35
```

Plain-text passwords can also be used. For reasons of security it is recommended you use obfuscated passwords instead.

This procedure requires that you have the following passwords from the previous section:

- Export Password
- Destination Keystore Password

To update the application.properties File:

1. Obtain an obfuscated string for the password used to create the *jetty.pkcs12* file in the previous procedure:

   ```
   java -cp /opt/avid/avid-interplay-central/lib/org.eclipse.jetty.jetty-util.jar org.eclipse.jetty.util.security.Password <Export_Password>
   ```

   Where `<Export_Password>` is the password you specified when creating the jetty.pks12 file.

Do not copy / paste this command from this document as some characters may be lost in the process.

   The system responds by outputting the password, the obfuscated password string (OBF:) and its MD5 hash value (MD5:).

   The following represents sample output. It is the string following OBF that is needed ("XXXXXX" represents the clear-text password):

   ```
   XXXXXX

   OBF:1c3x1mf71jnb1sov1jk71mbf1c35

   MD5:4c88dafcf38a9b90b1e32efe798f95f0
   ```

2. If you used a different password to create the Jetty keystore ("Destination Keystore Password"), repeat the step for the second password.
3. Edit (or create) the MediaCentral *application.properties* file using a text editor such as vi:

   ```
   vi /opt/avid/etc/avid/avid-interplay-central/config/application.properties
   ```

   In most cases, this file will not exist. Create the file using vi and add the lines indicated in the steps below.

   You can examine the contents of the default file in the following directory: /opt/avid/avid-interplay-central/config. However, do not make your changes in that file since it will be overwritten any time you upgrade MCS. Make your changes in the file you create in the /opt/avid/**etc**/avid/avid-interplay-central/config, as indicated in this step.

If you use the default file as a model, the one you create should only contain the values you wish to override.

4. Locate (or add) the entry for the Export Password:

   ```
   system.org.ops4j.pax.web.ssl.password=OBF\:1c3x1mf71jnb1sov1jk71mbf1c35
   ```

   Replace the obfuscated string (displayed as "`1c3x1mf71jnb1sov1jk71mbf1c35`" above) with the one you generated.

Those upgrading from ICS 1.2 or earlier (i.e. from a Windows server to a Linux server) please note the following difference in Linux syntax. If you are re-using the obfuscated string from the Windows server, be sure to add the Linux "escape" character ("\") in front of the colon in the entry for the password.

   A plain text entry would look like the following:

   ```
   system.org.ops4j.pax.web.ssl.password=<visible password>
   ```

For reasons of security it is recommended you use obfuscated passwords.

5. Locate (or add) the entry for the Destination Keystore Password:

   ```
   system.org.ops4j.pax.web.ssl.keypassword=OBF\:1c3x1mf71jnb1sov1jk71mbf1c35
   ```

   Replace the obfuscated string displayed as "`1c3x1mf71jnb1sov1jk71mbf1c35`" above) with the one you generated to create the Jetty keystore.

6. Save and exit the file:

   ```
   <Esc>:wq
   ```

7. Start the avid-interplay-central service (which also starts the Jetty web server):

   ```
   service avid-interplay-central start
   ```

   The system responds with the following:

   ```
   Avid Interplay Central process has been started.
   Wait for Interplay Central web interface...
   Avid Interplay Central has been started successfully (32 seconds)
   ```

Once the service has restarted, it serves the trusted certificate for its HTTPS connections.

Additional Information

The procedures and information in this section provide additional detail about the certificate configuration process.

Restoring the Original System Files

In the event that you are dissatisfied with the results of the CA-signed certificate, the original jetty.keystore file can be restored using the following procedure.

To restore the original Jetty keystore and MediaCentral configuration file:

1. Stop the Avid MediaCentral service:

   ```
   service avid-interplay-central stop
   ```

2. Overwrite the Jetty keystore you created with the original:

   ```
   cp /tmp/jetty.keystore.bak /opt/avid/etc/avid/avid-interplay-central/
   ssl/jetty.keystore
   ```

3. Delete the modified MediaCentral application.properties file:

   ```
   rm –f /opt/avid/etc/avid/avid-interplay-central/config/
   application.properties
   ```

4. Restart the Avid Interplay Central service:

   ```
   service avid-interplay-central restart
   ```

Be aware that this step will disconnect any users currently working on the system.

Examining the MediaCentral application.properties File

The following table summarizes the entries in the MediaCentral application properties file that are related to SSL certificates. MediaCentral makes use of these values to send SSL certificates to browsers.

Keys	Descriptions
system.org.osgi.service.ssl.password	The password protecting the certificate within the keystore. Must match the value given for the -storepass parameter when you generate the new certificate. Clear text or Jetty obfuscated (recommended).

Keys	Descriptions
system.org.osgi.service.ssl.keypassword	The password protecting the keystore itself. Must match the value given for -keypass when you generated the new keystore. Clear text or Jetty obfuscated (recommended).
system.org.osgi.service.ssl.algorithm	The encryption algorithm. Must match the value given for -keyalg when you generated the new keystore. Default is RSA.
system.org.osgi.service.https.keystore.path	The path and name of the key store file. Must match the value given for -keystore when you generated the new keystore. The path is relative to the standard SSL folder used for storing the key store file.