As described in previously published BMC documentation, you can configure the User Interface Middleware (UIM) server to use certificates and private keys to enable Secure Sockets Layer/Transport Layer Security (SSL/TLS). For users who are not familiar with SSL/TLS certificates, this notice provides an overview of the certificates and general instructions for creating them.

**Note**
Best practices for creating and managing certificates vary depending on which System Authorization Facility (SAF) application you are using. Also, the approach that you use can require coordination between multiple organizations within your enterprise. The procedures in this notice represent examples that can help you get started.

This notice also offers:

- Additional references for more detailed information (see “Additional references” on page 21)
- Sample JCL for users of the IBM Resource Access Control Facility (RACF) or the CA Technologies CA ACF2 security package (see “JCL samples” on page 11)
  - For required RACF authorizations, see the RACF reference documentation (IRR.DIGTCERT.*). Other SAF systems offer similar functionality; refer to their reference documentation for specific commands.

You can use self-signed certificates or Certificate Authority-signed (CA–signed) certificates:

- Self-signed (see “Overview of self-signed certificates” on page 2)
- CA–signed (see “Overview of CA–signed certificates” on page 3)

**Note**
If you have questions, contact Customer Support at 1 800 537 1813 (United States or Canada) or call your local support center.
Overview of self-signed certificates

The simplest and easiest type of certificate to implement is a single self-signed certificate (although most production environments require CA-signed certificates). Many web browsers do not initially recognize self-signed certificates as "trusted" (signed by a known authority), prompting errors or warnings such as this example from Mozilla Firefox:

As the following example from the BMC Database Management console shows, you are usually offered the choice of accepting a self-signed certificate for one-time use or permanently:
Accept a self-signed certificate only if you are certain that you know and trust the certificate's origin.

**Note**
Most IT departments maintain a Windows repository called a truststore, which contains certificates and indicates whether they are trusted. Many web browsers and Oracle Java applications (such as the BMC Database Management console) also maintain their own truststores.

### Overview of CA–signed certificates

Most production environments require that you use Certificate Authority–signed (CA–signed) certificates instead of self-signed certificates.

Your enterprise might rely solely on external CAs or might have its own CA that uses a "signing certificate" to sign other certificates within the enterprise. The signing certificate might be signed by an external CA. Either way, the "trusted root" certificates are preapproved and then pushed to truststores (certificate repositories) throughout the enterprise.
The result is a "chain of trust": your application presents a certificate that was signed by another certificate, which might have been signed by another, and so on, down to a root certificate in your truststore.

For example, consider the certificate that the Google homepage presents via the Firefox Certificate Viewer. The certificate for www.google.com was signed by an intermediate Google CA named Google Internet Authority G2. This intermediate CA was signed by the GeoTrust Global CA (included in most truststores because GeoTrust is a major third-party CA provider).

Creating certificates

Use these procedures to create SSL/TLS certificates.

Creating a self-signed certificate

This example procedure creates and saves a self-signed certificate to a data set and directs UIM to use the certificate directly from that data set. Note that other alternatives are possible; for example, you might prefer to attach the certificate to a key ring instead of using it directly from a data set.

1 Create a self-signed certificate:

   a Create the GENCERT job by using one of the following JCL samples:
      ■ “Sample JCL to create a self-signed certificate for RACF” on page 12
      ■ “Sample JCL to create a self-signed certificate for CA ACF2” on page 14

   b Run the GENCERT job.

      After the job runs successfully, the certificate will exist within RACF or CA ACF2.

   c (optional) Verify the existence of your certificate by using one of the following JCL samples to list all of the certificates associated with your user ID:

      ■ For RACF

      ```
      [yourJobCardHere]
      //RACF   EXEC PGM=IKJEFT01
      //SYSTSPRT DD SYSOUT=*  //SYSTSNIN DD *
      RACDCERT ID(yourUserID) LIST
      /*
      //
      ```
For CA ACF2, the output also includes the suffix that was CA ACF2 generated for your user ID (for example, AUTO002). All subsequent commands for this certificate will need to include that suffix when specifying the user ID.

**Example**

```
CERTDATA / RDADKS.AUTO002 LAST CHANGED BY RDADKS ON 09/13/16-14:11
```

2 Export the self-signed certificate to a data set:

   a Create the EXPORT job by using one of the following JCL samples:
      - “Sample JCL to export a self-signed certificate for RACF” on page 16
      - “Sample JCL to export a self-signed certificate for CA ACF2” on page 16

   b Run the job.

   After the job runs successfully, the certificate resides in the specified data set.

3 Update the UIM Startup configuration member to use a self-signed certificate:

   a Specify your certificate data set.

   Add or update the following parameter in the configuration member:

   ```xml
   <BMC_PARM ID="SSL_STORE" VALUE="DSN"/>
   <BMC_PARM ID="SSL_CERTIFICATE" VALUE="exportTargetDataSet" />
   </BMC_PARM>
   ```

   Specifying DSN for the SSL_STORE VALUE attribute indicates that the certificate is stored in a data set.

   The SSL_CERTIFICATE VALUE attribute specifies the data set that contains the certificate. Specify the name of the data set that you created with the EXPORT command.

   b Specify your private key.
Add or update the following parameter in the configuration member:

```xml
<BMC_PARM ID="SSL_STORE" VALUE="ICSF">
  <BMC_PARM ID="SSL_PRIVATE_KEY" VALUE="pkdsLabel" />
</BMC_PARM>
```

Specifying ICSF for the SSL_STORE VALUE attribute indicates that the private key associated with your certificate is stored in the ICSF Private Key Data Set (PKDS).

The SSL_PRIVATE_KEY VALUE attribute provides UIM with the unique identifier of the private key associated with your certificate. Specify the label that you used when you created the certificate with the GENCERT command.

c Specify the UIM encryption level.

Add or update the following parameter in the configuration member:

```xml
<BMC_PARM ID="ENCRYPTION_LEVEL" VALUE="SSL-IF" />
```

BMC recommends using the value SSL-IF. In this mode (SSL/TLS Conditional), UIM accepts both SSL-enabled and non-SSL-enabled connections.

Instead, you can specifying SSL-REQUIRED, which runs UIM in SSL/TLS Required mode. In this mode, UIM rejects all connection attempts that are not SSL enabled. Consequently, any UIM URLs that you have accessed via a web browser become unavailable; only their equivalent URLs beginning with https:// will be available.

4 Verify the results:

a If the UIM task is running, stop it.

b Start or restart the UIM task.

The MSGLOG output DD lists the UIM settings and includes your certificate and private key labels:

```
13:15:26.939 001 BMC340110I HTPMain **Initialized**
13:15:26.939 001 BMC340116I UIM Server started with these settings:
******************************************************************************
  13:15:26.940 001   Private Key Store:       ICSF
  13:15:26.940 001       SSL Private Key:      yourPkdsLabel
```

c Start a console session.
Until now, you have installed or launched BMC consoles by entering a URL in this format in a web browser:

http://host:port/

With SSL/TLS, you now use an equivalent secure URL in this format:

https://host:port/

As discussed in “Overview of self-signed certificates” on page 2, a warning message from your web browser indicates that the certificate is not trusted because it is self-signed. At the discretion of your IT department, accepting these warnings might be acceptable. Also, consider asking IT to add your self-signed certificate to the truststores for your enterprise.

Creating a CA–signed certificate

Use the following procedure to implement a CA–signed certificate. You will create a self-signed certificate (or start with an existing self-signed certificate), have it signed by a CA, and then import the CA-signed version into your SAF.

1 Create a self-signed certificate, or select one that already exists (see “Creating a self-signed certificate” on page 4).

2 Create a Certificate Signing Request (CSR):
   a Create the GENREQ job by using one of the following JCL samples:
      ■ “Sample JCL to create a CSR for RACF” on page 17
      ■ “Sample JCL to create a CSR for CA ACF2” on page 18
   b Run the job.

      A base-64 encoded version of the certificate is created, signed by its own private key.

3 Send the CSR to a Certificate Authority (CA), and provide the required information (which varies, depending on which CA you use).

You might need to send the certificate via FTP to a local machine in order to email it to the CA for signing. If so, use ascii mode for the FTP transfer. You should be able to open the request with a text editor and confirm that it resembles this sample request:

-----BEGIN NEW CERTIFICATE REQUEST-----
MIIDICAgIBAgIKBgkqhkiG9w0BAQEFADBzMQswCQYDVQQGEwJBSUVJ灌yBhعشyQ0Ew
-----END NEW CERTIFICATE REQUEST-----
4 Retrieve and upload the signed certificate from the CA:

a When the CA returns the certificate chain (typically a PFX file type, which conforms to the PKCS#12 standard), browse it by using the Microsoft Windows certmgr utility or a similar application to verify that it looks correct:

b Upload the signed certificate to your IBM z/OS environment by using FTP in binary mode to prevent data corruption.

5 Import the CA–signed certificate:

a Import the CA–signed certificate into your SAF by using one of the following JCL samples:

   - “Sample JCL to import CA–signed certificates for RACF” on page 18
   - “Sample JCL to import CA–signed certificates for CA ACF2” on page 19

b Run the job.

6 (required for RACF; optional for CA ACF2) Determine the labels of intermediate and root CA–signed certificates.
The label values are needed to create a key ring for RACF. These label values might be in the output of your ADD (for RACF) or INSERT (for CA ACF2) job. If not, use the following procedure:

a Use one of the following JCL samples to determine the labels:

- Sample JCL if you use RACF

```jcl
[yourJobCardHere]
//RACF    EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=* //SYSTSIN DD *
RACDCERT CERTAUTH LIST /*
/* //
```

- Sample JCL if you use CA ACF2

```jcl
[yourJobCardHere]
//ACFJOB EXEC PGM=ACFBATCH //SYSPRINT DD SYSOUT=* //SYSLBC DD DISP=SHR,DSN=dataSetName //SYSHELP DD DISP=SHR,DSN=dataSetName //SYSIN DD * SET PROFILE(USER) DIV(CERTDATA)
LIST (-) /* //
```

b Run the job.

The output includes all CA certificates, which should include any certificates in your certificate chain. You can identify the certificates by matching their attributes (such as subject name and serial number) to the certificate chain in the certmgr utility.

7 Create a key ring:

a Create a key ring for your SAF by using one of the following JCL samples:

- “Sample JCL to create a key ring for RACF” on page 19
- “Sample JCL to create a key ring for CA ACF2” on page 21

b Run the job.

8 Update the UIM Startup configuration member to use a CA–signed certificate:

a Specify your key ring.

Add or update the following parameter in the configuration member:

```xml
<BMC_PARM ID="SSL_STORE" VALUE="R_DATALIB" />
<BMC_PARM ID="SSL_CERTIFICATE"
    VALUE="userID/yourKeyRingLabel" />
</BMC_PARM>
```
Specifying R_DATALIB for the SSL_STORE VALUE attribute indicates that the certificate is attached to a key ring.

The SSL_CERTIFICATE VALUE attribute specifies the label for the key ring that you created and its associated user ID (both of which are required for UIM to access the key ring). This value must be specified as a user ID, followed by a forward slash (/), followed by the key ring label. No additional white spaces can be present.

b Specify your private key.

Add or update the following parameter in the configuration member:

```xml
<BMC_PARM ID="SSL_STORE" VALUE="ICSF">
  <BMC_PARM ID="SSL_PRIVATE_KEY" VALUE="yourPkdsLabel" />
</BMC_PARM>
```

Specifying ICSF for the SSL_STORE VALUE attribute indicates that the private key associated with your certificate is stored in the ICSF Private Key Data Set (PKDS).

The SSL_PRIVATE_KEY VALUE attribute provides UIM with the unique identifier of the private key associated with your certificate. Specify the same PKDS label value that you used when creating the initial, self-signed version of your certificate. This value should be the same value you used during your ADD (for RACF) or INSERT (for ACF2) command.

c Specify the UIM encryption level.

Add or update the following parameter in the configuration member:

```xml
<BMC_PARM ID="ENCRYPTION_LEVEL" VALUE="SSL-IF" />
```

BMC recommends using the value SSL-IF. In this mode (SSL/TLS Conditional), UIM accepts both SSL-enabled and non-SSL-enabled connections.

Instead, you can specifying SSL-REQUIRED, which runs UIM in SSL/TLS Required mode. In this mode, UIM rejects all connection attempts that are not SSL enabled. Consequently, any UIM URLs that you have accessed via a web browser become unavailable; only their equivalent URLs beginning with `https://` will be available.

9 Verify the results:

a If the UIM task is running, stop it.

b Start or restart the UIM task.
The MSGLOG output DD lists the UIM settings and includes your certificate and private key labels:

<table>
<thead>
<tr>
<th>Time</th>
<th>Message ID</th>
<th>Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>13:15:26.939</td>
<td>001 BMC340110I</td>
<td>HTPMain <strong>Initialized</strong></td>
</tr>
<tr>
<td>13:15:26.939</td>
<td>001 BMC340116I</td>
<td>UIM Server started with these settings:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>13:15:26.940</td>
<td>001</td>
<td>Private Key Store: ICSF</td>
</tr>
<tr>
<td>13:15:26.940</td>
<td>001</td>
<td>SSL Private Key: yourPkdsLabel</td>
</tr>
</tbody>
</table>

c Start a console session.

Until now, you have installed or launched BMC consoles by entering a URL in this format in a web browser:

http://host:port/

When using SSL/TLS, you now use an equivalent secure URL in this format:

https://host:port/

You should not receive any warning messages from your web browser indicating that the certificate is not trusted. If you receive any messages, your key ring probably is not defined correctly, or your root certificate is not present in your truststore.

JCL samples

This topic provides JCL samples for creating SSL/TLS certificates:

- “Sample JCL to create a self-signed certificate for RACF” on page 12
- “Sample JCL to create a self-signed certificate for CA ACF2” on page 14
- “Sample JCL to export a self-signed certificate for RACF” on page 16
- “Sample JCL to export a self-signed certificate for CA ACF2” on page 16
- “Sample JCL to create a CSR for RACF” on page 17
- “Sample JCL to create a CSR for CA ACF2” on page 18
- “Sample JCL to import CA–signed certificates for RACF” on page 18
- “Sample JCL to import CA–signed certificates for CA ACF2” on page 19
- “Sample JCL to create a key ring for RACF” on page 19
- “Sample JCL to create a key ring for CA ACF2” on page 21
Sample JCL to create a self-signed certificate for RACF

Use the following sample JCL to create a self-signed certificate for use with RACF:

```
[yourJobCardHere]
//RACF    EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
RACDCERT GENCERT  +
   SUBJECTSDN( CN('yourServerCom') +
               OU('organizationalUnit') +
               O('organization') +
               L('locality') +
               S('state') +
               C('country') ) +
   SIZE(2048)  +
   WITHLABEL('yourLabelHere') +
   PCICC('yourPkdsLabelHere') +
   KEYUSAGE( HANDSHAKE )
/*
*/
```

The following information explains this JCL:

- **SUBJECTSDN (Distinguished Name)**
  
  A combination of subparameters that describes the entity for which the certificate is being created. In this example, the entity is the system on which you will be running UIM.
  
  — **CN (Common Name)**
  
  The value that applications such as web browsers will use to determine whether the certificate was created for the server that is presenting it. If this value does not match the result of DNS lookups, most applications will issue a warning or prevent the connection completely.

  **Note**
  
  This value can use an asterisk wildcard. For example, '*.bmc.com' is valid.

  — **OU (Organizational Unit)**
  
  The name of your organization, within your enterprise

  — **O (Organization)**
  
  The name of your enterprise

  — **L (Locality)**
  
  Generally, the city in which your enterprise is headquartered

  — **S or ST (State)**
  
  Generally, the state or province in which your enterprise is headquartered
— C (Country)

Generally, the two-letter country code in which your enterprise is headquartered

■ SIZE (Key Size)

The size (in bits) of the private key that is generated and associated with this certificate. The certificate contains a corresponding paired public key that is generated at the same time. Larger keys are more secure. This number is not arbitrary, although there are a few predefined values. If you want to change the value, refer to the RACF documentation for other valid values.

■ WITHLABEL (Certificate Label)

A unique identifier that RACF uses for your certificate. The identifier can be up to 32 characters, can contain blanks, and can use mixed case. For more information, see http://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.1.0/com.ibm.zos.v2r1.icha400/le-add.htm#le-add.

■ PCICC (Private Key)

A private key that is generated for a certificate, when the certificate is generated. The PCICC parameter specifies the type of key to be created. The label you provide is a unique identifier that RACF uses to identify your private key within the Private Key Data Set (PKDS). The PKDS label must be unique, can be up to 64 character, is not case-sensitive, and must consist of alphanumeric characters, a @, a #, a $, or a period (.). The first character must be alphanumeric, @, #, or $, and you cannot use blanks. For more information, see https://www.ibm.com/support/knowledgecenter/en/SSLTBW_1.13.0/com.ibm.zos.r13.icha400/radcrtg.htm#pkdsic.

■ KEYUSAGE (Key Usage/Purpose)

Declaration of the purpose of this key. (You can use keys for a number of purposes, but you must declare them when creating the key.) Because this key will be used only for the security handshake (a step in the SSL/TLS protocol), HANDSHAKE is the appropriate value.

**Tip**

You might find it useful to define multiple URLs for which your certificate is valid. Certificates can contain Subject Alternate Name (SAN) values. The RACDCERT GENCERT command allows you to specify only one SAN by including the ALTNAME parameter:

```
ALTNAME(DOMAIN( 'yourDomainHere' ) ) +
```

You might be able to work around the RACF single-SAN limitation by using wildcards in your SAN value (for example, '*.bmc.com'). For more information about this subparameter, see the RACF documentation.
Sample JCL to create a self-signed certificate for CA ACF2

Use the following sample JCL to create a self-signed certificate for use with CA ACF2:

```
[yourJobCardHere]
//ACFJOB EXEC PGM=ACFBATCH
//SYSPRINT DD SYSOUT=*  
//SYSLBC DD DISP=SHR,DSN=dataSetName  
//SYSHELP DD DISP=SHR,DSN=dataSetName  
//SYSIN DD *  
   SET PROFILE(USER) DIV(CERTDATA)
   GENCERT userID SUBJSDN(CN='yourServerCom' -  
       OU='organizationalUnit' O='organization' -  
       L='locality' SP='state' C='country' ) -  
       SIZE(2048) LABEL('yourLabelHere') -  
       KEYUSAGE(HANDSHAKE) -  
       ALTNAME(DOMAIN=sampleDomainCom) -  
       PCICC PKDSLBL('yourPkdsLabelHere')  
```

The following information explains this JCL:

- **userID (User ID)**
  The user ID with which you want to associate the certificate
  The following guidelines apply to the user ID:
  
  — If you use the same user ID for multiple certificates, CA ACF2 appends suffixes to your ID as described in the *CA ACF for z/OS Administrator Guide*.
  
  — If the sample CA ACF2 JCL requires specifying a user ID as the first parameter of a command, include a suffix if one is associated with that certificate.
  
  — Instead of specifying a user ID, you can consider using the CERTAUTH or SITECERT parameter. For more information, see the *CA ACF for z/OS Administrator Guide*.

- **SUBJSDN (Distinguished Name)**
  A combination of subparameters that describes the entity for which the certificate is being created. In this example, the system on which you will be running UIM.
  
  — **CN (Common Name)**
    The value that applications such as web browsers will use to determine whether the certificate was created for the server that is presenting it. If this value does not match the result of DNS lookups, most applications will issue a warning, or prevent the connection completely.
Note

This value can use an asterisk wildcard. For example, '*.bmc.com' would be valid.

— OU (Organizational Unit)
   The name of your organization, within your enterprise

— O (Organization)
   The name of your enterprise

— L (Locality)
   Generally, the city in which your enterprise is headquartered

— SP (State)
   Generally, the state or province in which your enterprise is headquartered

— C (Country)
   Generally, the two-letter country code in which your enterprise is headquartered

■ SIZE (Key Size)
   The size (in bits) of the private key that is generated and associated with this certificate. The certificate contains a corresponding paired public key that is generated at the same time. Larger keys are more secure. This number is not arbitrary, though there are a few predefined values. If you want to change the value, you might also need to change the method used to generate and store the key (PCICC in this example). For more information, see the CA ACF2 documentation.

■ LABEL (Certificate Label)
   A unique identifier that CA ACF2 uses for your certificate.

■ PCICC PKDSLBL (Private Key)
   The type of key to create (specified by the PCICC parameter), and whether this key should be stored in the ICSF Private Key Data Store or PKDS (specified by the PKDSLBL subparameter). The label you provide is a unique identifier that CA ACF2 uses to identify your private key.

■ KEYUSAGE (Key Usage/Purpose)
   Declaration of the purpose of this key. (You can use keys for a number of purposes, but you must declare them when creating the key.) Because this key will be used only for the security handshake (a step in the SSL/TLS protocol), HANDSHAKE is the appropriate value.
Tip
You might find it useful to define multiple URLs for which your certificate is valid. Certificates can contain Subject Alternate Name (SAN) values. The GENCERT command allows you to specify any number of these by including one or more ALTNAME parameters:

\[ \text{ALTNAME(DOMAIN=yourDomainHere)} \]

Do not add quotes to yourDomainHere. The value is included in the generated SAN value and quotes would make the SAN value incorrect.

Sample JCL to export a self-signed certificate for RACF

Use the following sample JCL to export a self-signed certificate for use with RACF:

```plaintext
[yourJobCardHere]
//RACF EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=* 
//SYSTSIN DD * 
RACDCERT EXPORT ( LABEL('yourLabelHere') ) +
    DSN('yourDataSetNameHere') FORMAT(CERTDER)
/* 
// 
```

The following information explains this JCL:

- **LABEL (Certificate Label)**
  The value you used for the WITHLABEL parameter of the GENCERT command.

- **DSN (Data Set Name)**
  The name of the data set where the certificate will be exported. You will subsequently specify this data set name in your UIM configuration.

- **FORMAT (Certificate Format)**
  The certificate standard that you want to use. Several common standards are available. Specifying CERTDER exports the certificate in x.509 format, which UIM accepts.

Sample JCL to export a self-signed certificate for CA ACF2

Use the following sample JCL to export a self-signed certificate for use with CA ACF2:

```plaintext
[yourJobCardHere]
//ACFJOB EXEC PGM=ACFBATCH
//SYSPRINT DD SYSOUT=* 
//SYSLBC DD DISP=SHR, DSN=shopDependentValue
//SYSHELP DD DISP=SHR, DSN=shopDependentValue
//SYSSIN DD * 
```
SET PROFILE(USER) DIV(CERTDATA)  
EXPORT userID DSNAME('yourDataSetNameHere') -  
LABEL('yourLabelHere') -  
FORMAT(CERTDER)

The following information explains this JCL:

■ userID (User ID)  
The user ID with which the certificate is associated, suffixed if necessary.

■ LABEL (Certificate Label)  
The value you used for the WITHLABEL parameter of the GENCERT command.

■ DSN (Data Set Name)  
The name of the data set where the certificate will be exported. You will subsequently specify this data set name in your UIM configuration.

■ FORMAT (Certificate Format)  
The certificate standard that you want to use. Several common standards are available. Specifying CERTDER exports the certificate in x.509 format, which UIM accepts.

Sample JCL to create a CSR for RACF

Use the following sample JCL to create a Certificate Signing Request (CSR) for use with RACF:

[yourJobCardHere]
//RACF EXEC PGM=IKJEFT01  
//SYSTSPRT DD SYSOUT=*  
//SYSTSIN DD *  
RACDCERT GENREQ ( LABEL('yourCertificateLabel') ) +  
   DSN( 'yourRequestDSN' )
/*  
*/

The following information explains this JCL:

■ LABEL (Certificate Label)  
The value you used for the WITHLABEL parameter with the GENCERT command.

■ DSN (Data Set Name)  
The name of the data set where the CSR should be created.
Sample JCL to create a CSR for CA ACF2

Use the following sample JCL to create a Certificate Signing Request (CSR) for use with CA ACF2:

```
[yourJobCardHere]
//ACFBATCH EXEC PGM=ACFBATCH
//SYSPRINT DD SYSOUT=* 
//SYSLBC DD DISP=SHR,DSN=dataSetName
//SYSHELP DD DISP=SHR,DSN=dataSetName
//SYSIN DD *
  SET PROFILE(USER) DIV(CERTDATA)
  GENREQ userID DSN('yourRequestDSN') - 
    LABEL('yourCertificateLabel')
```

The following information explains this JCL:

- **userID (User ID)**
  The user ID with which the certificate is associated, suffixed if necessary.

- **DSN (Data Set Name)**
  The name of the data set where the CSR should be created.

- **LABEL (Certificate Label)**
  The value you used for the LABEL parameter with the GENCERT command.

Sample JCL to import CA–signed certificates for RACF

Use the following sample JCL to import the CA–signed certificates into RACF:

```
[yourJobCardHere]
//IKJEFT01 EXEC PGM=IKJEFT01 
//SYSTSPRT DD SYSOUT=* 
//SYSTSIN DD *
  RACDCERT ADD('yourCertificateChainDSN')      + 
    WITHLABEL('yourCertificateLabel')
/*  */
```

The following information explains this JCL:

- **DSN (Data Set Name)**
  The name of the data set where you uploaded the CA–signed certificate chain.

- **WITHLABEL (Certificate Label)**
  The label you used when creating the certificate for the CSR. The self-signed version of this certificate will be replaced by the CA–signed version.
Sample JCL to import CA–signed certificates for CA ACF2

Use the following sample JCL to import the CA–signed certificates into CA ACF2:

```java
[yourJobCardHere]
//ACFJOB EXEC PGM=ACFBATCH
//SYSPRINT DD SYSOUT=* 
//SYSLBC DD DISP=SHR, DSN=dataSetName 
//SYSHELP DD DISP=SHR, DSN=dataSetName 
//SYSIN DD *
SET PROFILE(USER) DIV(CERTDATA)
INSERT userID DSN('yourCertificateChainDSN') -
   LABEL(yourCertificateLabel) -
   TRUST -
   PKDSLBL(yourPkdsLabel)
```

The following information explains this JCL:

- **userID** (User ID)
  The user ID with which the CA–signed certificate will be associated, suffixed if necessary.

- **DSN** (Data Set Name)
  The name of the data set where you uploaded the CA–signed certificate chain.

- **LABEL** (Certificate Label)
  A new certificate label for the CA–signed version of the certificate.

- **PKDSLBL** (Private Key Label)
  The same private-key label that you used in the job that created the certificate on which you based the CSR. Although this will exist as a separate certificate, it must use the same private key as its self-signed predecessor.

Sample JCL to create a key ring for RACF

Use the following sample JCL to create a key ring for use with RACF:

```sql
[yourJobCardHere]
//RACF EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=* 
//SYSTSIN DD *
RACDCERT ADDRING(yourKeyRingName) ID(yourUserID)
RACDCERT CONNECT ID(yourUserID) LABEL('yourCertificateLabel') +
   RING(yourKeyRingName) USAGE(PERSONAL) ) ID(yourUserID)
RACDCERT CONNECT CERTAUTH LABEL('CAcertificateLabel1') +
   RING(yourKeyRingName) )
RACDCERT CONNECT CERTAUTH LABEL('CAcertificateLabel2') +
   RING(yourKeyRingName) )
/*
```

JCL samples
**WARNING**

The order in which you add certificates to your key ring is critical. You *must* begin with the certificate you created, followed by the certificate that signed it, and on up the chain so that the last certificate you add is the root certificate.

You can ascertain the correct order by viewing the certificate chain using the Windows certmgr utility or a similar application. The certmgr utility lists certificates starting with the root and proceeding to the site-specific; this is the *opposite* order in which the certificate labels should be added to your JCL.

The following information explains this JCL:

- **ADDRING (Key Ring Name)**
  A unique name for your key ring.

- **(optional) ID (User ID)**
  The user ID to associate with the key ring. If you do not include this subparameter, the key ring is created under your user ID. Because your key ring and CA-signed certificate are both associated with your ID, the ADDRING and CONNECT commands for your certificate both use your ID.

- **(first instance) LABEL (Certificate Label)**
  Your personal certificate label.

- **(subsequent instances) LABEL (Certificate Label)**
  The labels of the CA–signed certificates. Use as many of these statements as needed.
Sample JCL to create a key ring for CA ACF2

Use the following sample JCL to create a key ring for use with CA ACF2:

```jcl
[yourJobCardHere]
//ACFJOB EXEC PGM=ACFBATCH
//SYSPRINT DD SYSOUT=* 
//SYSLBC  DD DISP=SHR,DSN=dataSetName 
//SYSHELP  DD DISP=SHR,DSN=dataSetName 
//SYSIN   DD *
SETS PROFILE(USER) DIV(KEYRING)
INSERT userID RINGNAME(yourKeyRingName)
CONNECT CERTDATA(userID) LABEL(yourCertificateLabel) -
       KEYRING(userID) RINGNAME(yourKeyRingName) -
       DEFAULT USAGE (PERSONAL)
```

The following information explains this JCL:

- **userID** (User ID for INSERT)
  The user ID with which to associate the certificate. CA ACF2 generates suffixes for each key ring and associates each one to this same user ID.

- **RINGNAME** (Key Ring Name for INSERT)
  A unique name for your key ring.

- **userID** (User ID for CONNECT)
  The user ID, suffixed if appropriate, that is associated with your CA–signed certificate.

- **LABEL** (Certificate Label)
  The same label you used for your CA–signed certificate when importing the certificate.

- **userid** (User ID for KEYRING)
  The same user ID value, suffixed if appropriate, that you specified for the INSERT command in this job.

- **RINGNAME** (Key Ring Name for KEYRING)
  The same name you specified for the INSERT command in this job.

Additional references

See the following resources for additional information about RACF and CA ACF2:

- For RACF:


■ For CA ACF2:


— CA ACF2 for z/OS Cookbook, "Digital Certificate Support”


Where to get the latest product information

To view the latest BMC documents, see the Support Central website at http://www.bmc.com/support.

Notices such as flashes, technical bulletins, and release notes are available on the website. You can subscribe to proactive alerts to receive e-mail messages when notices are issued or updated. For more information about proactive alerts, see the Support Central website.

© Copyright 2016 BMC Software, Inc.
BMC, BMC Software, the BMC logo, the BMC Software logo, and other BMC marks are the exclusive properties of BMC Software, Inc. and are registered or may be registered in the U.S. and in other countries. BladeLogic and other BladeLogic marks are the exclusive properties of BladeLogic, Inc. and are registered or may be registered in the U.S. and in other countries. All other trademarks or registered trademarks are the property of their respective owners. IBM, IBM z, RACF, ibm.com, and z/OS are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

The information included in this documentation is the proprietary and confidential information of BMC Software, Inc., its affiliates, or licensors. Your use of this information is subject to the terms and conditions of the applicable End User License agreement for the product and to the proprietary and restricted rights notices included in the product documentation.

BMC SOFTWARE INC
2103 CITYWEST BLVD, HOUSTON TX 77042-2827, USA
Telephone 1 713 918 8800 (or 1 800 841 2031 United States and Canada)
Customer Support: 1 800 537 1813 (United States and Canada) or contact your local support center