# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>3</td>
</tr>
<tr>
<td>Overview</td>
<td>3</td>
</tr>
<tr>
<td>Important Considerations</td>
<td>3</td>
</tr>
<tr>
<td>IBM Global Security Kit (GSKit)</td>
<td>4</td>
</tr>
<tr>
<td>Configuration</td>
<td>6</td>
</tr>
<tr>
<td>Setting up SSL with the Content Manager OnDemand Server</td>
<td>6</td>
</tr>
<tr>
<td>Create a key database</td>
<td>6</td>
</tr>
<tr>
<td>Using a CA-signed certificate</td>
<td>6</td>
</tr>
<tr>
<td>Creating a self-signed digital certificate</td>
<td>8</td>
</tr>
<tr>
<td>Configuring the Content Manager OnDemand Server</td>
<td>9</td>
</tr>
<tr>
<td>Setting up SSL with the Content Manager OnDemand Client</td>
<td>11</td>
</tr>
<tr>
<td>Create a key database</td>
<td>11</td>
</tr>
<tr>
<td>Configuring the Content Manager OnDemand Client</td>
<td>11</td>
</tr>
<tr>
<td>Using the Content Manager OnDemand Client</td>
<td>12</td>
</tr>
<tr>
<td>Appendix A</td>
<td>13</td>
</tr>
<tr>
<td>Overview of the SSL handshake</td>
<td>13</td>
</tr>
<tr>
<td>Digital certificates and certificate authorities</td>
<td>13</td>
</tr>
<tr>
<td>Public-key cryptography</td>
<td>14</td>
</tr>
<tr>
<td>Supported cipher suites</td>
<td>15</td>
</tr>
<tr>
<td>Appendix B</td>
<td>16</td>
</tr>
<tr>
<td>Default GSKit trusted root certificates</td>
<td>16</td>
</tr>
</tbody>
</table>

"Using Secure Sockets Layer (SSL) With IBM Content Manager OnDemand" Rev: September 10, 2010

©Copyright International Business Machines Corporation 2010. All rights reserved. US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
Introduction

Overview
Content Manager OnDemand Version 8.5 now supports Secure Sockets Layer (SSL) and its successor, Transport Layer Security (TLS), for all transmissions between the Content Manager OnDemand servers and clients. SSL is the standard technology for creating encrypted links between servers and clients. Without encryption, packets of information travel through networks in full view of anyone who has access to the network. The Content Manager OnDemand server can be configured to listen on either a non-SSL port or an SSL port, or it can listen on both types of ports.

Note: When this topic mentions SSL, the same information applies to TLS, unless otherwise noted.

Once a Content Manager OnDemand client (i.e. Win32 Client, arsdoc, OnDemand Web Enablement Kit (ODWEK) Java API, …) has been configured to logon to a Content Manager OnDemand library server with SSL – all communication that is made to support that client will be done using SSL.

- Between the client and the Library Server
- Between the client and the Object Server(s)
- Between the Object Server(s) and the Library Server

Important Considerations
Due to system performance concerns, use SSL only for sessions where it is needed. Consider adding additional processor resources on the Content Manager OnDemand server and/or client to manage the increased overhead.

With SSL, the identities of the parties are verified through the use of digital certificates. Digital certificates have expiration dates, once a digital certificate has expired, Content Manager OnDemand will not be able to establish connections thru SSL. Therefore, always be aware and plan ahead to avoid expired certificates.

The support of SSL and ODWEK refers specifically to the transfer of data between ODWEK and the Content Manager OnDemand server(s) and does not imply a level of support from the browser to ODWEK. Using SSL from the browser to ODWEK has always been allowed and does not require any support from ODWEK - it is the application/web developer’s responsibility to enable such support.
IBM Global Security Kit (GSKit)

Content Manager OnDemand Version 8.5 uses the IBM Global Security Kit in order to support SSL communication. You must ensure that GSKit is installed on your system. Regardless if you are installing a Content Manager OnDemand client or server, GSKit must be installed. If installing on AIX, HP-UX, Linux or Solaris – you must manually install GSKit. When installing on the Windows platform, the Content Manager OnDemand installer will install GSKit for you. It is also possible that you might need to install the 32bit and/or 64bit version of GSKit.

- Content Manager OnDemand Server (64bit GSKit)
- Content Manager OnDemand Windows Client/Admin (32bit GSKit)
- ODWEK Java Api (32bit and/or 64bit GSKit)
- ODWEK CGI, ODWEK Servlet, and CICS Client do not support SSL

Content Manager OnDemand ships with GSKit Version 8.0.13.4 and uses the default GSKit installer (on UNIX, the GSKit shared libraries will have links in /usr/lib). Other products from IBM also support using GSKit, however the version of GSKit that ships with other products might be different than what ships with Content Manager OnDemand. You must ensure that Content Manager OnDemand utilizes GSKit 8.0.13.4 or later. Other IBM products might not use the default GSKit installation and therefore have their own version of GSKit in their product installation directory. On UNIX platforms, you might need to set the appropriate dynamic library path order environment variable (LIBPATH or LD_LIBRARY_PATH) in which to search for GSKit when running a Content Manager OnDemand command. Use the gsk8ver or gsk8ver_64 command to determine which version of GSKit is in the default path.

See the Content Manager OnDemand README file for further information on installing GSKit.

<table>
<thead>
<tr>
<th>Operating System</th>
<th>GSKit Install Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>/usr/opt/ibm/gsk8</td>
</tr>
<tr>
<td></td>
<td>/usr/opt/ibm/gsk8_64</td>
</tr>
<tr>
<td>HP-UX</td>
<td>/opt/ibm/gsk8_32</td>
</tr>
<tr>
<td></td>
<td>/opt/ibm/gsk8_64</td>
</tr>
<tr>
<td>Linux</td>
<td>/usr/local/ibm/gsk8</td>
</tr>
<tr>
<td></td>
<td>/usr/local/ibm/gsk8_64</td>
</tr>
<tr>
<td>Solaris</td>
<td>/opt/ibm/gsk8</td>
</tr>
<tr>
<td></td>
<td>/opt/ibm/gsk8_64</td>
</tr>
<tr>
<td>Windows</td>
<td>Standard install path:</td>
</tr>
<tr>
<td></td>
<td>C:\Program Files\IBM\GSK8</td>
</tr>
<tr>
<td></td>
<td>Windows 32 bit installations on x86_64 systems:</td>
</tr>
<tr>
<td></td>
<td>C:\Program Files (x86)\IBM\GSK8</td>
</tr>
</tbody>
</table>

"Using Secure Sockets Layer (SSL) With IBM Content Manager OnDemand" Rev: September 10, 2010

©Copyright International Business Machines Corporation 2010. All rights reserved. US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
To configure SSL support, you will need to create and manage your digital certificates with GSKit. Invoke GSKCapiCmd using the gsk8capicmd (or gsk8capicmd_64) command, as described in the GSKCapiCmd User’s Guide to manage these digital certificates. The server sends this certificate to the client during the SSL handshake to provide authentication for the server. To obtain a certificate, use GSKCapiCmd to create a new certificate request and submit it to a CA to be signed, or create a self-signed certificate for testing purposes.

Note: Windows Only - when invoking the GSKCapiCmd tool you must ensure that the GSKit bin and lib directories are in your PATH.

64bit:
PATH=C:\Program Files\IBM\GSK8\bin;C:\Program Files\IBM\GSK8\lib64;%PATH%

32bit:
PATH=C:\Program Files\IBM\GSK8\bin;C:\Program Files\IBM\GSK8\lib;%PATH%

32bit on a Windows 64bit system:
PATH=C:\Program Files (x86)\IBM\GSK8\bin;C:\Program Files (x86)\IBM\GSK8\lib;%PATH%

Configuration

Setting up SSL with the Content Manager OnDemand Server

Create a key database
Use the GSKCapiCmd tool to create your key database. It must be a Certificate Management System (CMS) type key database. The GSKCapiCmd is a non-Java-based command-line tool, Java™ does not need to be installed on your system to use this tool.

Although not required, it is recommended that you store the key database in the Content Manager OnDemand server installation config sub-directory.

AIX: /usr/lpp/ars/config
HP-UX, Linux, Solaris: /opt/ondemand/config
Windows: C:\Program Files\IBM\OnDemand for Windows\config

For example, the following command creates a key database called ondemand.kdb and a stash file called ondemand.sth:

gsk8capicmd_64 -keydb -create -db "ondemand.kdb" -pw "myKeyDBpasswd" -stash -populate

The -stash option creates a stash file at the same path as the key database, with a file extension of .sth.
At Content Manager OnDemand start-up, GSKit uses the stash file to obtain the password to the key database.

Note: You should use strong file system protection on the stash file.

When you create a key database with the -populate option, it is automatically populated with a number of predefined trusted certificate authority (CA) certificates. A trusted CA is one whose root certificate is noted as trusted in the key database. See Appendix B for the list of default trusted root certificates.

Using a CA-signed certificate
To obtain a CA-signed certificate for your server (instead of a self-signed certificate), you need to generate a certificate signing request and pay a well known CA, such as VeriSign, to sign the certificate. After receiving the signed certificate, you need to store it in the server key database.

The following example demonstrates how to request and receive a certificate. It uses a trial version of a certificate.

1) Create a Certificate Signing Request (CSR) for ondemand.kdb. The following command creates a new RSA private-public key pair and a PKCS10 certificate request in the specified key database. For CMS key
databases, the certificate request information is stored in the file with the ".rdb" extension. The file specified by the -file option is the one that needs to be sent to your CA.

```
gsk8capicmd_64 -certreq create -db "ondemand.kdb" -pw "myKeyDBpasswd" -label "mycert" \
-dn "CN=myhost.mycompany.com,O=myOrganization,OU=myOrganizationUnit,L=Boulder,ST=CO,C=US" \
-file "mycertRequestNew"
```

The following command lists the detailed information of the certificate request for my server:

```
gsk8capicmd_64 -certreq details -db "ondemand.kdb" -pw "myKeyDBpasswd" -label "mycert"
```

In case you need to delete the certificate request, use a command similar to the following example:

```
gsk8capicmd_64 -certreq delete -db "ondemand.kdb" -pw "myKeyDBpasswd" -label "mycert"
```

2) Go to the CA web site and register. You will be asked to cut and paste the request file to submit the request. For trial version, you would receive an email that contains the signed certificate. The email also contains links for downloading the trial root CA certificate and the trial intermediate CA certificate. Use notepad or vi to save all three certificates into files (these three are in a chain of trust):

- RootCert.arm
- IntermediateCert.arm
- MyCertificate.arm

3) Add the trial Root CA Certificate into ondemand.kdb with the following command:

```
gsk8capicmd_64 -cert add -db "ondemand.kdb" -pw "myKeyDBpasswd" -label "trialRootCACert" \
-file RootCert.arm -format ascii
```

4) Add the trial Intermediate CA Certificate into ondemand.kdb with the following command:

```
gsk8capicmd_64 -cert add -db "ondemand.kdb" -pw "myKeyDBpasswd" -label "trialIntermediateCACert" \
-file IntermediateCert.arm -format ascii
```

5) Receive the trial Certificate into ondemand.kdb with the following command:

```
gsk8capicmd_64 -cert receive -file MyCertificate.arm -db "ondemand.kdb" -pw "myKeyDBpasswd" \
-format ascii
```
List all the certificates in mydbserver.kdb with the following command:

```bash
gsk8capicmd_64 -cert -list all -db "ondemand.kdb" -pw "myKeyDBpasswd"
```

Certificates found

* default, - personal, ! trusted
  `-! mycert
  `! trialIntermediateCACert
  `! trialRootCACert
  `-! myselfsigned

6) If the certificate is signed by a trusted CA, obtain the certificate, install it in the key database, and restart the server. Because the certificate is provided by a trusted authority, the certificate is accepted by Content Manager OnDemand and communication between server and client can commence without having to update the Content Manager OnDemand client key database.

**Creating a self-signed digital certificate**

To create a self-signed certificate with a label of myselfsigned, use the GSKCapiCmd command as shown in the following example:

```bash
gsk8capicmd_64 -cert -create -db "ondemand.kdb" -pw "myKeyDBpasswd" -label "myselfsigned" 
  -dn "CN=myhost.mycompany.com,O=myOrganization,OU=myOrganizationUnit,L=Boulder,ST=CO,C=US"
```

Extract the certificate you just created to a file, so that you can distribute it to computers running clients that will be establishing SSL connections to your Content Manager OnDemand server. For example, the following GSKCapiCmd command extracts the certificate to a file called ondemand.arm:

```bash
gsk8capicmd_64 -cert -extract -db "ondemand.kdb" -pw "myKeyDBpasswd" -label "myselfsigned" 
  -target "ondemand.arm" -format ascii
```

To display the certificate, issue the following command:

```bash
gsk8capicmd_64 -cert -details -db "ondemand.kdb" -pw "myKeyDBpasswd" -label "myselfsigned"
```
**Configuring the Content Manager OnDemand Server**

On AIX: Modify the ars.ini file and add the following lines:

```ini
[@SRV@_ARCHIVE]
HOST=ondemand.boulder.ibm.com
PROTOCOL=2
PORT=1445
SSL_PORT=1456
SRVR_INSTANCE=ARCHIVE
SRVR_INSTANCE_OWNER=root
SRVR_OD_CFG=/usr/lpp/ars/config/ars.cfg
SRVR_DB_CFG=/usr/lpp/ars/config/ars.dbfs
SRVR_SM_CFG=/usr/lpp/ars/config/ars.cache
SSL_KEYRING_FILE=/usr/lpp/ars/config/ondemand.kdb
SSL_KEYRING_STASH=/usr/lpp/ars/config/ondemand.sth
SSL_KEYRING_LABEL=IBM Content Manager OnDemand
SSL_CLNT_USE_SSL=0
```

On HP-UX, Linux, and Solaris: Modify the ars.ini file and add the following lines:

```ini
[@SRV@_ARCHIVE]
HOST=ondemand.boulder.ibm.com
PROTOCOL=2
PORT=1445
SSL_PORT=1456
SRVR_INSTANCE=ARCHIVE
SRVR_INSTANCE_OWNER=root
SRVR_OD_CFG=/usr/lpp/ars/config/ars.cfg
SRVR_DB_CFG=/usr/lpp/ars/config/ars.dbfs
SRVR_SM_CFG=/usr/lpp/ars/config/ars.cache
SSL_KEYRING_FILE=/opt/ondemand/config/ondemand.kdb
SSL_KEYRING_STASH=/opt/ondemand/config/ondemand.sth
SSL_KEYRING_LABEL=IBM Content Manager OnDemand
SSL_CLNT_USE_SSL=0
```

In the example above, the Content Manager OnDemand server will listen on both the non-SSL port 1445 and the SSL port 1456. If you do not want an SSL port, set SSL_PORT=0. If you do not want a non-SSL port, then set PORT=-1 (minus 1).

The SSL_PORT defines the port number in which to listen on for SSL. The SSL_KEYRING_FILE is the absolute file name for the key database which contains the digital certificates. The SSL_KEYRING_STASH is the absolute file name which contains the stash file for the key database. The SSL_KEYRING_LABEL is the name of the certificate in the key database. The SSL_CLNT_USE_SSL variable is used by Content Manager OnDemand server based-clients (ie. arsdoc, arsmaint, arsload, ...) to determine whether or not to use SSL (by default they will not use SSL). To have these commands use SSL set SSL_CLNT_USE_SSL=1

"Using Secure Sockets Layer (SSL) With IBM Content Manager OnDemand" Rev: September 10, 2010

©Copyright International Business Machines Corporation 2010. All rights reserved. US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
On Windows: Use the Configurator

The SSL_KEYRING_FILE and SSL_KEYRING_STASH default to the Content Manager OnDemand for Windows installation directory location. For example:

C:\Program Files\IBM\OnDemand for Windows\config\ondemand.kdb
C:\Program Files\IBM\Ondemand for Windows\config\ondemand.sth
Setting up SSL with the Content Manager OnDemand Client

Create a key database

Use the GSKCapiCmd tool to create your key database. It must be a Certificate Management System (CMS) type key database. The GSKCapiCmd is a non-Java-based command-line tool, and Java™ does not need to be installed on your system to use this tool.

We recommend you store the key database in the Content Manager OnDemand client installation config sub-directory.

On Windows: `C:\Program Files\IBM\OnDemand32\config`

For example, the following command creates a key database called `ondemand.kdb` and a stash file called `ondemand.sth`:

```
gsk8capicmd_64 -keydb -create -db "ondemand.kdb" -pw "myKeyDBpasswd" -stash -populate
```

The -stash option creates a stash file in the same path as the key database, with a file extension of .sth. At Content Manager OnDemand start-up, GSKit uses the stash file to obtain the password to the key database.

**Note:** You should use strong file system protection on the stash file.

When you create a key database with the -populate option, it is automatically populated with a number of predefined trusted certificate authority (CA) certificates. A trusted CA is one whose root certificate is noted as trusted in the key database. See appendix A for the list of default trusted root certificates.

Configuring the Content Manager OnDemand Client

If your server certificate is signed by a well known CA, your client key database might already contain the CA certificate that signed your server certificate. If it does not, you must obtain the CA certificate, which is usually done by visiting the web site of the CA.

Obtain the signer certificate of the server digital certificate on the client. The server certificate can either be a self-signed certificate or a certificate signed by a certificate authority (CA).

If your server certificate is a self-signed certificate, you must extract its signer certificate to a file on the server computer and then distribute it to all computers running clients that will be establishing SSL connections to that server.

Add the signer certificate into the client key database. For example, the following gsk8capicmd command adds the certificate from the file `ondemand.arm` into the key database called `ondemand.kdb`:

```
gsk8capicmd_64 -cert -add -db "ondemand.kdb" -pw "myKeyDBpasswd" -label "dbselfsigned" -file "ondemand.arm" -format ascii
```

"Using Secure Sockets Layer (SSL) With IBM Content Manager OnDemand" Rev: September 10, 2010

©Copyright International Business Machines Corporation 2010. All rights reserved. US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
Using the Content Manager OnDemand Client

Now, when defining a Content Manager OnDemand server to the client, you can select the Use Secure Sockets Layer check box. The Content Manager OnDemand client will then use the following key database to manager and validate any received certificate from the Content Manager OnDemand server:

C:\Program Files\IBM\OnDemand32\config\ondemand.kdb
C:\Program Files\IBM\OnDemand32\config\ondemand.sth
Appendix A

Overview of the SSL handshake

During an SSL handshake, a public-key algorithm, usually RSA, is used to securely exchange digital signatures and encryption keys between a client and a server. This identity and key information is used to establish a secure connection for the session between the client and the server. After the secure session is established, data transmission between the client and server is encrypted using a symmetric algorithm, such as AES.

The client and server perform the following steps during the SSL handshake:

1. The client requests an SSL connection and lists its supported cipher suites.
2. The server responds with a selected cipher suite.
3. The server sends its digital certificate to the client.
4. The client verifies the validity of the server certificate, for authentication purposes. It can do this by checking with the trusted certificate authority that issued the server certificate or by checking in its own key database.
5. The client and server securely negotiate a session key and a message authentication code (MAC).
6. The client and server securely exchange information using the key and MAC selected.

Note: The Content Manager OnDemand system does not support the (optional) authentication of the client during the SSL handshake.

Digital certificates and certificate authorities

Digital certificates are issued by trusted parties, called certificate authorities, to verify the identity of an entity, such as a client or server.

The digital certificate serves two purposes: It verifies the owner’s identity and it makes the owner’s public key available. It is issued with an expiration date, after which it is no longer guaranteed by the certificate authority (CA).

To obtain a digital certificate, you send a request to the CA of your choice, such as Verisign, or RSA. The request includes your distinguished name, your public key, and your signature. A distinguished name (DN) is a unique identifier for each user or host for which you are applying for a certificate. The CA checks your signature using your public key and performs some level of verification of your identity (this varies with different CAs). After verification, the CA sends you a signed digital certificate that contains
your distinguished name, your public key, the CA’s distinguished name, and the signature of the certificate authority. You store this signed certificate in your key database.

When you send this certificate to a receiver, the receiver performs two steps to verify your identity:

1. Uses your public key that comes with the certificate to check your digital signature.
2. Verifies that the CA that issued your certificate is legitimate and trustworthy. To do this, the receiver needs the public key of the CA. The receiver might already hold an assured copy of the public key of the CA in their key database, but if not, the receiver must acquire an additional digital certificate to obtain the public key of the CA. This certificate might in turn depend on the digital certificate of another CA; there might be a hierarchy of certificates issued by multiple CAs, each depending on the validity of the next. Eventually, however, the receiver needs the public key of the root CA. The root CA is the CA at the top of the hierarchy. To trust the validity of the digital certificate of the root CA, the public-key user must receive that digital certificate in a secure manner, such as through a download from an authenticated server, or with preloaded software received from a reliable source, or on a securely delivered diskette.

Many applications that send a digital certificate to a receiver send not just their own certificate, but also all of the CA digital certificates necessary to verify the hierarchy of certificates up to the root CA certificate.

For a digital certificate to be entirely trustworthy, the owner of the digital certificate must have carefully protected their private key, for example, by encrypting it on their computer's hard drive. If their private key has been compromised, an imposter could misuse their digital certificate.

You can use self-signed digital certificates for testing purposes. A self-signed digital certificate contains your distinguished name, your public key, and your signature.

Public-key cryptography

SSL uses public-key algorithms to exchange encryption key information and digital certificate information for authentication. Public-key cryptography (also known as asymmetric cryptography) uses two different encryption keys: a public key to encrypt data and an associated private key to decrypt it.

Conversely, symmetric key cryptography uses just one key, which is shared by all parties involved in the secure communication. This secret key is used both to encrypt and decrypt information. The key must be safely distributed to, and stored by, all parties, which is difficult to guarantee. With public-key cryptography, the public key is not secret, but the messages it encrypts can only be decrypted by using it's associated private key. The private key must be securely stored, for example, in your key database, or encrypted on your computer's hard drive.
Public-key algorithms alone do not guarantee secure communication; you also need to verify the identity of whoever is communicating with you. To perform this authentication, SSL uses digital certificates. When you send your digital certificate to someone, the certificate provides them with your public key. You have used your private key to digitally sign your certificate and so the receiver of the communication can use your public key to verify your signature. The validity of the digital certificate itself is guaranteed by the certificate authority (CA) that issued it.

**Supported cipher suites**

During an SSL handshake, the client and server negotiate which cipher suite to use to exchange data. A cipher suite is a set of algorithms that are used to provide authentication, encryption, and data integrity.

The Content Manager OnDemand system uses GSKit running in FIPS mode to provide SSL support. GSKit supports the following cipher suites:

- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_3DES_EDE_CBC_SHA

The name of each cipher suite specifies the algorithms that it uses for authentication, encryption, and data integrity checking. For example, the cipher suite TLS_RSA_WITH_AES_256_CBC_SHA uses RSA for authentication; AES 256-bit and CBC for encryption algorithms; and SHA-1 for the hash function for data integrity.

You cannot prioritize which cipher suite is selected.
Appendix B

Default GSKit trusted root certificates

- Thawte Personal Premium CA
- Thawte Personal Freemail CA
- Thawte Personal Basic CA
- Thawte Premium Server CA
- Thawte Server CA
- VeriSign Class 3 Secure Server CA
- VeriSign International Server CA - Class 3
- VeriSign Class 4 Public Primary Certification Authority - G3
- VeriSign Class 3 Public Primary Certification Authority - G5
- VeriSign Class 3 Public Primary Certification Authority - G3
- VeriSign Class 2 Public Primary Certification Authority - G3
- VeriSign Class 1 Public Primary Certification Authority - G3
- VeriSign Class 4 Public Primary Certification Authority - G2
- VeriSign Class 3 Public Primary Certification Authority - G2
- VeriSign Class 2 Public Primary Certification Authority - G2
- VeriSign Class 1 Public Primary Certification Authority - G2
- VeriSign Class 3 Public Primary Certification Authority
- VeriSign Class 2 Public Primary Certification Authority
- VeriSign Class 1 Public Primary Certification Authority
- Entrust.net Global Secure Server Certification Authority
- Entrust.net Global Client Certification Authority
- Entrust.net Client Certification Authority
- Entrust.net Certification Authority (2048)
- Entrust.net Secure Server Certification Authority