PROGRESS®
ORBIX®

OrbixSSL Java Programmer's and Administrator's Guide

Version 3.3, SP11 March 2012

PROGRESS software
BUSINESS MAKING PROGRESS™

Progress Orbix v3.3.11

Third Party Acknowledgements: One or more products in the Progress Orbix v3.3.11 release includes third party components covered by licenses that require that the following documentation notices be provided:

Progress Orbix v3.3.11 incorporates OpenSSL/SSLeay v0.9.8.i technology from OpenSSL.org. Such Technology is subject to the following terms and conditions: LICENSE ISSUES

The OpenSSL toolkit stays under a dual license, i.e. both the conditions of the OpenSSL License and the original SSLeay license apply to the toolkit. See below for the actual license texts. Actually both licenses are BSD-style Open Source licenses. In case of any license issues related to OpenSSL please contact openssl-core@openssl.org.

Updated: 07-Mar-2012

Contents

Part I

Introduction

Part II

OrbixSSL Administration

Part III

OrbixSSL Programming

Part IV

OrbixSSL Java Reference

Part V

Appendices

Preface

OrbixSSL integrates Orbix, Progress Software implementation of the CORBA standard, with the Secure Sockets Layer Version 3.0 (SSL V3.0) protocol. This integration allows Orbix C++ and Java Edition applications to communicate using SSL security.

This guide presents details of the integration between Orbix Java Edition and SSL and explains how to add SSL security to Orbix Java Edition applications.

Audience

This guide is intended for programmers who wish to develop Orbix Java Edition applications that communicate using SSL security.

This guide does not assume that the reader has any knowledge of SSL security issues. This guide assumes that programmers have significant knowledge of Orbix Java Edition programming.

Orbix documentation is periodically updated. New versions between releases are available at this site:

http://communities.progress.com/pcom/docs/DOC-105220

If you need assistance with Orbix or any other Progress products, go to http://www.progress.com/orbix/orbix-support.html.

If you want to provide any comments on Progress documentation, go to http://www.progress.com/en/about/contact.html.

Organization of this Guide

This guide is divided into four parts:

Part I, "Introduction"

This part introduces SSL security, describes how OrbixSSL applications use SSL, and shows you how to add security to an existing Orbix Java Edition application. Read this part first.

Part II, "OrbixSSL Administration"

This part describes the system administration tasks required when running an OrbixSSL system.

Part III, "OrbixSSL Programming"

This part introduces the OrbixSSL Java application programming interface (API) and describes how you use it to control SSL security in your applications.

Part IV, "OrbixSSL Java Reference"

This part provides a complete reference for the Java classes defined in the OrbixSSL API.

Part V, "Appendices"

This part provides supplemental information about OrbixSSL security and the SSL administration tools supplied with OrbixSSL.

Document Conventions

This document uses the following typographical and keying conventions:

`Constant width` Constant width words or characters represent source code or system values you must use literally, such as commands, options, and path names.

Italic Italic words in normal text represent emphasis and new terms.

Italic words or characters in code and commands represent variable values you must supply, such as arguments or commands or path names for your particular system.

This guide uses the following keying conventions:

...
.
.
.
Horizontal or vertical ellipses in format and syntax descriptions indicate that material has been eliminated to simplify a discussion.

[ ] Brackets enclose optional items in format and syntax descriptions.

{ } Braces enclose a list from which you must choose an item in format and syntax descriptions.

| A vertical bar separates items in a list of choices enclosed in { } (braces) in format and syntax descriptions.

Part I

Introduction

1

An Introduction to OrbixSSL

OrbixSSL integrates Orbix with Secure Sockets Layer (SSL) security. Using OrbixSSL, distributed applications can transfer confidential data securely across a network.

An Overview of OrbixSSL

Secure Sockets Layer (SSL) provides data security for applications that communicate across networks. SSL is a transport layer security protocol layered between application protocols and TCP/IP.

Orbix applications communicate using the CORBA standard Internet Inter-ORB Protocol (IIOP) or Progress Software's proprietary Orbix protocol. These application-level protocols are layered above the transport-level protocol TCP/IP. OrbixSSL applications communicate using IIOP or the Orbix protocol layered above SSL. Figure 1.1 on page 18 illustrates how the SSL protocol layer integrates with Orbix communications.

All OrbixSSL components, including the Orbix daemon and Orbix utilities, and all OrbixSSL applications can communicate using SSL. OrbixSSL imposes few requirements on administrators and programmers who wish to support SSL communications in Orbix applications.

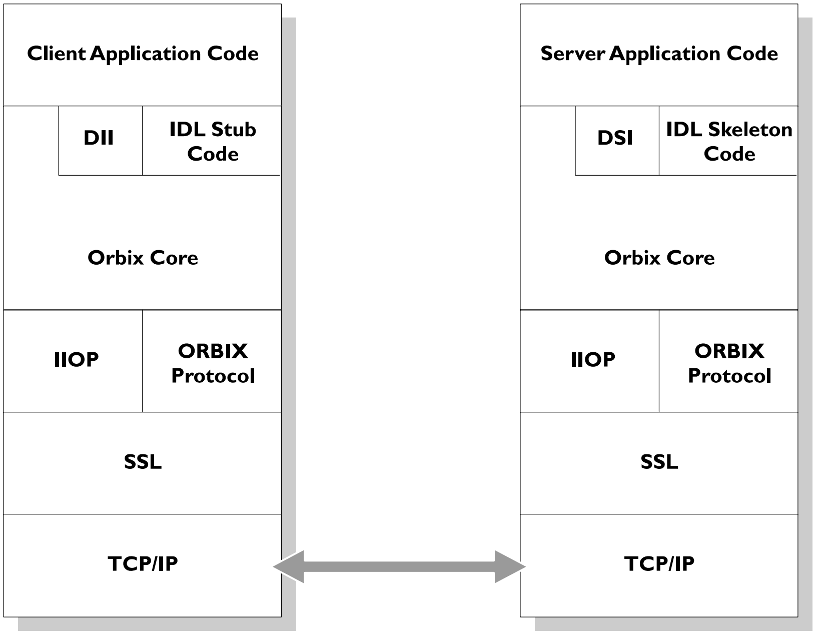

Figure 1.1: *The Role of SSL in Orbix Client/Server Communications*

OrbixSSL administrators use a single configuration file to configure a high-level security policy for a distributed system. OrbixSSL programmers develop standard Orbix applications that automatically communicate using SSL. The details of the SSL protocol are hidden, but programmers can use the OrbixSSL application programming interface (API) to customize SSL communications.

OrbixSSL applications can be configured to support any or all of the following options:

- IIOP
- IIOP over SSL

- Orbix Protocol
- Orbix Protocol over SSL

OrbixSSL acts as a dynamic upgrade to Orbix C++ and Orbix Java Edition. Existing applications continue to work as before.

An Overview of SSL Security

SSL provides authentication, privacy, and integrity for communications across TCP/IP connections. Authentication allows an application to verify the identity of another application with which it communicates. Privacy ensures that data transmitted between applications can not be eavesdropped on or understood by a third party. Integrity allows applications to detect if data was modified during transmission.

Authentication in SSL

SSL uses Rivest Shamir Adleman (RSA) public key cryptography for authentication. In public key cryptography, each application has an associated public key and private key. Data encrypted with the public key can be decrypted only with the private key. Data encrypted with the private key can be decrypted only with the public key.

Public key cryptography allows an application to prove its identity by encoding data with its private key. As no other application has access to this key, the encoded data must derive from the true application. Any application can check the content of the encoded data by decoding it with the application’s public key.

The SSL Handshake Protocol

Consider the example of two applications, a client and a server. The client connects to the server and wishes to send some confidential data. Before sending application data, the client must ensure that it is connected to the required server and not to an impostor.

When the client connects to the server, it confirms the server identity using the SSL handshake protocol. A simplified explanation of how the client executes this handshake in order to authenticate the server is as follows:

1. The client initiates the SSL handshake by sending the initial SSL handshake message to the server.
2. The server responds by sending its *certificate* to the client. This certificate verifies the server's identity and contains its public key.
3. The client extracts the public key from the certificate and encrypts a symmetric encryption algorithm session key with the extracted public key.
4. The server uses its private key to decrypt the encrypted session key which it will use to encrypt and decrypt application data passing to and from the client. The client will also use the shared session key to encrypt and decrypt messages passing to and from the server.

For a complete description of the SSL handshake, refer to the *Netscape Communications SSL V3.0* specification, available from `www.netscape.com`.

The SSL protocol permits a special optimized handshake in which a previously established session can be resumed. This has the advantage of not needing expensive public key computations. The SSL handshake also facilitates the negotiation of ciphers to be used in a connection.

The SSL protocol also allow the server to authenticate the client. Client authentication, which is supported by OrbixSSL, is optional in SSL communications.

As any application can have a public and private key pair, the transfer of the public key must be accompanied by additional information that proves the key is associated with the true server and not some other application. For this reason, the key is transmitted as part of a certificate.

Certificates in SSL Authentication

The public key is transmitted as part of a certificate. A certificate is used to ensure that the public key submitted is in fact the public key which belongs to the submitter. For the certificate to be acceptable to the client, it must have been digitally signed by a Certificate Authority (CA) that the client explicitly trusts.

The International Telecommunications Union (ITU) recommendation X.509 defines a standard format for certificates. SSL authentication uses X.509 certificates to transfer information about an application's public key.

An X.509 certificate includes the following data:

- The name of the entity identified by the certificate.
- The public key of the entity.
- The name of the certification authority that issued the certificate.

The role of a certificate is to match an entity name to a public key. A CA is a trusted authority that verifies the validity of the combination of entity name and public key in a certificate. You must specify trusted CAs in order to use OrbixSSL.

According to the SSL protocol, it is unnecessary for applications to have access to all certificates. Generally, each application only needs to access its own certificate and the corresponding issuing certificates. Clients and servers supply their certificates to applications that they want to contact during the SSL handshake. The nature of the SSL handshake is such that there is nothing insecure in receiving the certificate from an as yet untrusted peer. The certificate will be checked to make sure that it has been digitally signed by a trusted CA and the peer will have to prove its identity during the handshake.

Privacy of SSL Communications

When a client authenticates a server, confidential data sent by the client can be encoded by the server's public key. It is only the actual server application that will be able to decode this data, using the corresponding private key.

Immediately after authentication, an SSL client application sends an encoded data value to the server. This unique session encoded value is a key to a symmetric cryptographic algorithm.

A symmetric cryptographic algorithm is an algorithm in which a single key is used to encode and decode data. Once the server has received such a key from the client, all subsequent communications between the applications can be encoded using the agreed symmetric cryptographic algorithm. This feature strengthens SSL security.

Examples of symmetric cryptographic algorithms used to maintain privacy in SSL communications are the Data Encryption Standard (DES) and RC4.

Integrity of SSL Communications

The authentication and privacy features of SSL ensure that applications can exchange confidential data that cannot be understood by an intermediary. However, these features do not protect against the modification of encrypted messages transmitted between applications.

To detect if an application has received data modified by an intermediary, SSL adds a message authentication code (MAC) to each message. This code is computed by applying a function to the message content and the secret key used in the symmetric cryptographic algorithm.

An intermediary cannot compute the MAC for a message without knowing the secret key used to encrypt it. If the message is corrupted or modified during transmission, the message content will not match the MAC. SSL automatically detects this error and rejects corrupted messages.

2

Getting Started with OrbixSSL

OrbixSSL provides SSL security for communications between components of your CORBA applications. This chapter shows you how to introduce SSL security to an existing application.

Using OrbixSSL, your CORBA applications benefit from the authentication, privacy, and integrity of SSL communications. When you create an OrbixSSL application, you must supply the information necessary to complete the authentication process. OrbixSSL then ensures the privacy and integrity of your communications without any intervention from you.

The SSL handshake, described in Chapter 1, enables components of your OrbixSSL application to authenticate each other. To ensure every SSL handshake completes successfully, each authenticated component must be able to access its certificate and private key.

To provide this information to OrbixSSL applications, you use the OrbixSSL application programming interface (API). This chapter uses an OrbixSSL demonstration program to show how you can add SSL security to an existing Orbix Java Edition application.

Overview of the Application

The Orbix Java Edition grid demonstration implements a simple CORBA application. In this application, an Orbix server creates a single object that implements the IDL interface `grid`.

To begin communicating with the server, a client gets a reference to the `grid` object. The client uses the `grid` object to read and write numeric values stored in a two-dimensional grid.

The IDL definitions for this application are as follows:

```
// IDL
interface grid {
   readonly attribute short height;
   readonly attribute short width;

   void set(in short row, in short col,
      in long value);
   long get(in short row, in short col);
};
```

Running the Application without SSL

Without SSL, this application runs as follows:

1. The client gets a reference to the `grid` object. Implicitly, the client contacts the Orbix daemon, which launches the server.
2. The client calls an operation on the `grid` object. The server processes this call.
3. The client calls further operations on the `grid` object.

These steps are illustrated in Figure 2.1. When the application runs without SSL, all communications between parts of the application are insecure.

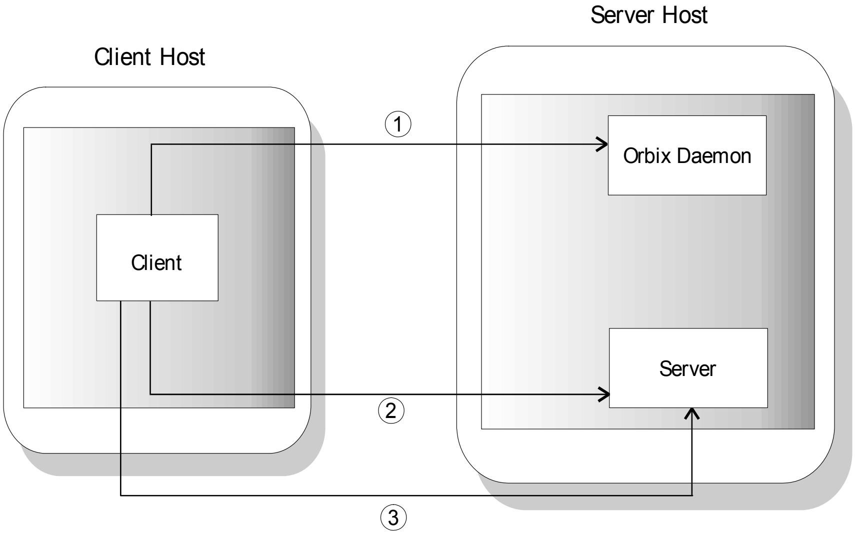

Figure 2.1: *Running the Grid Application*

Running the Application with SSL

When using SSL, each component of the application that acts as a server must be able to prove its identity. On first contact with another component, a server must be able to supply its certificate and encrypt messages with its private key. In this example, there are two servers: the bank server and the Orbix daemon.

With SSL, the application runs as shown in Figure 2.2 on page 26:

1. The client gets a reference to the `grid` object. Implicitly, the client contacts the Orbix daemon, which launches the server.

   The Orbix daemon supplies its certificate to the client. The client uses this certificate to check the identity of the daemon.

2. The client calls an operation on the `grid` object. The server processes this call.

   The server supplies its certificate to the client. The client uses this certificate to check the identity of the server.

3. The client calls further operations on the `grid` object over a secure connection.

With SSL security, all the servers in the application can be identified and all communications between application components take place over secure connections.

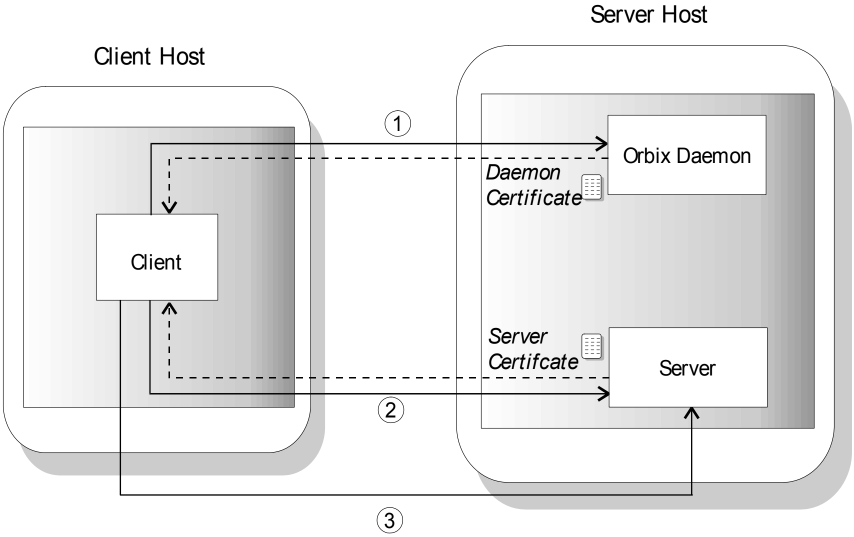

Figure 2.2: *Running the Grid Application with SSL Security*

To develop this example, you must modify the client and server programs. In the server, you must:

- Initialize OrbixSSL.
- Instruct OrbixSSL where to find the server certificate.
- Provide OrbixSSL with access to the server's private key.

In the client, you must:

- Initialize OrbixSSL.
- Provide OrbixSSL with information about which certificates to accept.

To run the example, you must use the SSL-enabled Orbix daemon, `orbixd`, on the server host instead of the Orbix Java Edition daemon, `orbixdj`. You must also provide the Orbix daemon with access to its certificate and private key.

Overview of the Certificates Used in the Example

In the grid application, the server and Orbix daemon use demonstration certificates installed with OrbixSSL. Each certificate has a corresponding file in the OrbixSSL `certificates` directory. The certificates for the grid application are shown in Table 2.1.

Server	Certificate File
Grid	`demos/demo_server_1`
Orbix daemon	`services/orbix`

Table 2.1: *Demonstration Certificates used by the Grid Application*

The `orbix` certificate is a general demonstration certificate for use with standard Orbix servers. The `demo_server_1` certificate is a demonstration certificate used with OrbixSSL server examples. Each of the demonstration certificates is signed by the OrbixSSL demonstration certificate authority (CA), called `demo_ca_1`.

WARNING: These certificates are completely insecure. Use them for OrbixSSL demonstration programs only. Do not use them in a deployed system. In a deployed system, you must create your own customized certificates for components of your application. The certificates for a deployed system should be signed by a CA that you can trust. Never trust the CA `demo_ca_1`. The process of creating and signing certificates is described in detail in Chapter 3 on page 39.

Adding SSL to the Example

The Orbix Java Edition grid application is located in the `demos/grid` directory of your Orbix Java Edition installation. OrbixSSL includes a secure version of this example in the OrbixSSL `demos/ssldemo` directory. This section describes the code changes introduced in this SSL-enabled version of the demonstration.

Adding SSL to the Server

As described in "Running the Application with SSL" on page 25, there are three steps required to add SSL security to the server program:

- Initialize OrbixSSL.
- Instruct OrbixSSL where to find the server certificate.
- Provide OrbixSSL with access to the server's private key.

This section describes each of these steps.

Initializing OrbixSSL

Every OrbixSSL program must initialize OrbixSSL using the OrbixSSL API. To import the API classes used by all servers, use the following statements:

```
import IE.Iona.OrbixWeb.SSL.IT_SSL;
import IE.Iona.OrbixWeb.SSL.IT_Format;
import IE.Iona.OrbixWeb.SSL.IT_X509Cert;
```

The OrbixSSL API contains a single initialization method that must be called in all your OrbixSSL programs. This method is called `IT_SSL.init()` and is defined as follows:

```
class IE.Iona.OrbixWeb.SSL.IT_SSL {
public:
  public static synchronized IT_SSL init()
    throws INITIALIZE;
  ...
};
```

The SSL-enabled grid server calls this method as follows:

```
...
import IE.Iona.OrbixWeb.SSL.IT_SSL;
import IE.Iona.OrbixWeb.SSL.IT_Format;
import IE.Iona.OrbixWeb.SSL.IT_X509Cert;

public class javaserver1 {
   ...

   public static void main(String args[]) {
      org.omg.CORBA.ORB orb =
         org.omg.CORBA.orb.init(args, null);
      IT_SSL ssl = IT_SSL.init();
      ...
   }
}
```

As shown here, you must call ORB.init() before calling IT_SSL.init(). In addition, for OrbixSSL initialization to succeed, you must call the method IT_SSL.init() before your OrbixSSL program attempts to make any remote operation calls.

Specifying the Location of a Server Certificate

In SSL, each application certificate is signed by a certificate authority (CA). The CA confirms that the identity of the application corresponds to the public key in the certificate. The CA can, in turn, be signed by another CA and this process continues until a self-signed CA certificate is reached. This process is known as *certificate chaining*.

Each OrbixSSL demonstration certificate has an associated certificate file in the OrbixSSL certificates directory. The grid server uses the demo_server_1 certificate, which is signed using the self-signed certificate demo_ca_1. The files associated with these certificates are demos/demo_server_1 and ca/demo_ca_1.

To specify the location of a server's certificate files, you must create an array that represents the server's certificate chain. In the case of the grid server, the demos/demo_server_1 certificate file is element zero in the array and the file ca/demo_ca_1 is element one.

For example, if the OrbixSSL certificates directory is located in /progress/OrbixSSL, create the certificate chain as follows:

```
...
import IE.Iona.OrbixWeb.SSL.IT_SSL;
import IE.Iona.OrbixWeb.SSL.IT_X509Cert;
import IE.Iona.OrbixWeb.SSL.IT_Format;

public class javaserver1 {

   public static void main(String args[]) {
      org.omg.CORBA.ORB orb =
         org.omg.CORBA.ORB.init(args,null);
      IT_SSL ssl = IT_SSL.init();
      IT_X509Cert certChain[] = new IT_X509Cert[2];

      try {
         certChain[0] = new IT_X509Cert
          ("/progress/OrbixSSL/certificates/demos/
          demo_server_1", IT_Format.IT_FMT_PEM);
         certChain[1] = new IT_X509Cert
          ("/progress/OrbixSSL/certificates/ca/
          demo_ca_1", IT_Format.IT_FMT_PEM );
         ssl.setApplicationCertChain(certChain);
         ...
      }
      ...
}
```

An object of type IT_X509Cert represents a single X.509 certificate. An array of these objects represents a certificate chain. The method IT_SSL.setApplicationCertChain() associates a certificate chain with the server program.

Providing Access to a Server Private Key

In this example, the private key associated with the certificate file demos/demo_server_1 is stored in the file demos/demo_server_1.jpk. This private key file is stored in encrypted Privacy Enhanced Mail (PEM) format. When a private key is encrypted in this way, you can access it only using a corresponding pass phrase.

When you launch an OrbixSSL server, it must specify where to locate its private key file and must supply the private key pass phrase to OrbixSSL. This allows OrbixSSL to read the private key and the server to encrypt data with this key, which is a critical part of SSL authentication.

The OrbixSSL API includes methods that allows you to specify the location of a private key file and the corresponding pass phrase. These methods are `IT_SSL.setPrivateKeyPassword()` and `IT_SSL.setRsaPrivateKeyFromFile()`. The demonstration server calls these methods as follows:

```
...
import IE.Iona.OrbixWeb.SSL.IT_SSL;
import IE.Iona.OrbixWeb.SSL.IT_X509Cert;
import IE.Iona.OrbixWeb.SSL.IT_Format;

public class javaserver1 {

   public static void main(String args[]) {
      org.omg.CORBA.ORB orb =
         org.omg.CORBA.ORB.init(args,null);
      IT_SSL ssl = IT_SSL.init();
      IT_X509Cert certChain[] = new IT_X509Cert[2];

      try {
         // Set certificate chain.
         ...

         // Set private key.
         ssl.setPrivateKeyPassword("demopassword");
         ssl.setRSAPrivateKeyFromFile(
          "/progress/OrbixSSL/certificates/demos/
          demo_server_1.jpk",IT_Format.IT_FMT_PEM);
      }
      ...
}
```

In this example, the pass phrase is hard coded in the server program. In fact, this is insecure and useful only for demonstration purposes. In a deployed system, you must provide a secure mechanism for retrieving the server pass phrase. For example, you could request the pass phrase from the user.

Adding SSL to the Client

As described in "Running the Application with SSL" on page 25, there are two steps required to add SSL security to the client program:

- Initialize OrbixSSL.
- Provide OrbixSSL with information about which certificates to accept.

This section describes each of these steps.

Initializing OrbixSSL

The steps required to initialize OrbixSSL in a client are the same as those described in "Initializing OrbixSSL" on page 28, with the exception that it is not necessary to use `IE.Iona.OrbixWeb.SSL.IT_X509Cert`. The following code initializes OrbixSSL for a client:

```
...
import IE.Iona.OrbixWeb.SSL.IT_SSL;
import IE.Iona.OrbixWeb.SSL.IT_Format;

public class javaclient1 {
   ...

   public static void main(String args[]) {
      org.omg.CORBA.ORB orb =
         org.omg.CORBA.orb.init(args, null);
      IT_SSL ssl = IT_SSL.init();
      ...
   }
}
```

Specifying which Certificates to Accept

Every certificate is signed by a CA. When a client receives a certificate from a server, the client checks that the certificate is signed by a trusted CA. If the client trusts the CA, it accepts the certificate and continues to authenticate the server, otherwise it rejects the certificate.

When running an OrbixSSL application, you must specify a list of CAs that the application should accept. To do this, call the method `IT_SSL.addTrustedCert()` for each trusted CA. This method takes the location of the CA certificate file as a parameter.

The grid example uses the insecure OrbixSSL demonstration CA, `demo_ca_1`. To specify that the client should accept certificates signed by `demo_ca_1`, call `IT_SSL.addTrustedCert()` as follows:

```
...
import IE.Iona.OrbixWeb.SSL.IT_SSL;
import IE.Iona.OrbixWeb.SSL.IT_Format;

public class javaclient1 {
   ...

   public static void main(String args[]) {
      org.omg.CORBA.ORB orb =
         org.omg.CORBA.orb.init(args, null);
      IT_SSL ssl = IT_SSL.init();

      try {
         ssl.addTrustedCert
          ("/progress/OrbixSSL/certificates/ca/
          demo_ca_1", IT_Format.IT_FMT_PEM);
      }
      ...
   }
}
```

This code assumes that the OrbixSSL `certificates` directory is located in `/progress/OrbixSSL/certificates`.

Running the Application

After you modify the client and server programs, run the application as follows:

1. On the client and server hosts, set the `CLASSPATH` to include both the Orbix Java Edition `classes` directory and the Orbix Java Edition `classes/SSL.zip` file.
2. On the server host, run the SSL-enabled Orbix daemon.
3. Register the server in the Implementation Repository with server name `SSLgrid1`.
4. Run the client.

There are special considerations that you must take into account when running the SSL-enabled Orbix daemon.

Running the Orbix Daemon

The SSL-enabled Orbix daemon, `orbixd`, is located in the `bin` directory of your Orbix installation. This daemon acts as an OrbixSSL C++ server and requires some configuration, as described in the ***OrbixSSL C++ Programmer's and Administrator's Guide***.

To run the daemon, do the following on the server host:

1. Edit the file `orbixssl.cfg`, located in the `cfg` directory of your OrbixSSL installation. Add the following text to this file:

```
OrbixSSL {
   IT_CERTIFICATE_PATH =
      OrbixSSL directory/certificates;
   IT_CA_LIST_FILE =
      OrbixSSL directory/ca_lists/demo_ca_list_1;
};

Orbix {
   orbixd {
      IT_CERTIFICATE_FILE =
            OrbixSSL.IT_CERTIFICATE_PATH +
            "services/orbix";
   };
};
```

In this text, replace *OrbixSSL directory* with the actual path of your OrbixSSL installation, for example `/progress/OrbixSSL`.

2. Set the environment variable `IT_CONFIG_PATH` to the location of the Orbix configuration file, `iona.cfg`.
3. On UNIX, run the OrbixSSL `update` command to specify the location of the OrbixSSL configuration file, `orbixssl.cfg`:

   ```
   update library OrbixSSL_directory 2
   ```

   Run this command for each of the OrbixSSL libraries, replacing *library* with the library file name and *OrbixSSL_directory* with the location of `orbixssl.cfg`.

   On Windows, set the environment variable `IT_SSL_CONFIG_PATH` to the location of `orbixssl.cfg`.
4. Set the environment variable that locates dynamic libraries, for example `PATH` on Windows, `LD_LIBRARY_PATH` on Solaris, or `SHLIB_PATH` on HP-UX, to include the Orbix `lib` directory.
5. Run the Orbix daemon:

   ```
   orbixd
   ```

For more information about securing the SSL-enabled Orbix daemon, refer to the *OrbixSSL C++ Programmer's and Administrator's Guide*.

Working with Secure Applets

Creating an applet version of an OrbixSSL Java application is similar to creating an applet version of an Orbix Java Edition application. The OrbixSSL `demos` directory includes an applet version of the grid demonstration, in a directory named `sslGridApplet`. The installed makefile for this example describes how to generate the signed applet and make it available through a web browser.

Developing Secure Applets

It is important that applets shut down the ORB when they exit. This ensures that connections are closed and resources are freed. Failing to do this may result in the browser hanging on exit.

To ensure connections are properly closed, add the following line to your applet's `destroy()` method:

```
_CORBA.Orbix.shutdown(true);
```

Signed applets that are to run in Internet Explorer must assert their requirement for full permission in four applet methods: `init()`, `start()`, `stop()` and `destroy()`. When they are finished with their full permissions, they must revoke them.

To assert a requirement for full permissions, use the following code:

```
import com.ms.security.*;
...
PolicyEngine.assertPermission
            (PermissionID.SYSTEM);
```

To revoke permissions granted using this code, use:

```
PolicyEngine.revertPermission
            (PermissionID.SYSTEM);
```

OrbixSSL includes stub versions of the Microsoft classes used here. This means that the same applet code can be used irrespective of the target browser. The `com.ms.security` classes in Microsoft Internet Explorer take precedence over the OrbixSSL stub versions. In other browsers, the stub versions are used but they initiate no action.

Deploying Secure Applets

OrbixSSL can run in browsers only as a signed applet. This means that it requires privileges over and above what the browser sandbox permits. A number of applet signing techniques exist, each targeted at a particular browser. Techniques to transparently provide versions of an applet using different signing techniques also exist. For further information on the topic of applet signing and deployment, refer to the Progress Knowledge Base at `www.progress.com`.

Part II

OrbixSSL Administration

3

Managing Certificates

SSL authentication uses X.509 certificates. This chapter explains how you can create X.509 certificates that identify your OrbixSSL applications.

An X.509 certificate binds a name to a public key value. The role of a certificate is to guarantee that the public key can be used to verify the identity contained in the X.509 certificate.

Authentication of a secure application depends on the integrity of the public key value in the application's certificate. If an impostor replaced the public key with its own public key, it could impersonate the true application and gain access to secure data.

To prevent this form of attack, all certificates must be signed by a *certification authority* (CA). A CA is a trusted node that confirms the integrity of the public key value in a certificate.

A CA signs a certificate by adding its digital signature to the certificate. A digital signature is a message encoded with the CA's private key. The CA's public key is made available to applications by distributing a certificate for the CA. Applications verify that certificates are validly signed by decoding the CA's digital signature with the CA's public key.

Most of the demonstration certificates supplied with OrbixSSL are signed by the CA `demo_ca_1`. This CA is completely insecure because anyone can access its private key. To secure your system, you must create new certificates signed by a trusted CA. This chapter describes the certificates required by an OrbixSSL application and shows you how to create those certificates.

Creating Certificates for an Application

To set up a fully secure OrbixSSL system, you must generate a full set of certificates for the secure components of your system, such as server, authenticated clients, the Orbix daemon, Orbix services, and so on. There are three steps required to do this:

1. Set up a CA that you can trust.
2. Use the CA to create signed certificates.
3. Deploy the signed certificates.

If a component of your application must prove its identity during SSL authentication, that component requires a certificate signed by your chosen CA. In a secure system, this always includes the Orbix daemon, the Orbix utilities, the Orbix services, and your server programs. If you use client authentication, your clients also require certificates.

Overview of the OrbixSSL Demonstration Certificates

The OrbixSSL `certificates` directory contains a set of demonstration certificates that enable you to run the OrbixSSL example applications. The certificates contained in the `certificates` directory are described in Table 3.1.

Certificate	Description
`ca/demo_ca_1` `ca/demo_ca_2`	Contains the certificates for the example CAs `demo_ca_1` and `demo_ca_2`. The CA list file, `demo_ca_list_1`, in the OrbixSSL `ca_lists` directory, includes the certificate for `demo_ca_1`. Programs that set the value of `IT_CA_LIST_FILE` to this file accept only certificates signed by `demo_ca_1`.

Table 3.1: *Demonstration Certificates Supplied with OrbixSSL*

Certificate	Description
demos/bad_guy demos/bank_customer_1 demos/bank_customer_2 demos/secure_bank_server demos/demo_client demos/demo_client_ca2 demos/demo_server demos/demo_server_ca2 ...	Example certificates used in the OrbixSSL demonstration programs. These programs are contained in the OrbixSSL `demos` directory. These certificates are signed by `demo_ca_1`, with the exception of those with `_ca2` appended to the file name, which are signed by `demo_ca_2`.
services/orbix services/orbix_events services/orbix_manager services/orbix_names services/orbix_ots services/orbix_trader	Example certificates used by Orbix services and standard Orbix executable files, such as the Orbix daemon, the Orbix utilities, and the Interface Repository server.

Table 3.1: *Demonstration Certificates Supplied with OrbixSSL*

The remainder of this chapter describes the steps involved in setting up a CA and signing certificates. As an example, it then shows you how to replace the demonstration certificates in the OrbixSSL `certificates` directory with your own, secure certificates.

Choosing a Certification Authority

A CA must be trusted to keep its private key secure. When setting up an OrbixSSL system, it is important to choose a suitable CA, make the CA certificate available to all applications, and then use the CA to sign certificates for your applications.

There are two types of CA available. A *commercial CA* is a company that signs certificates for many systems. A *private CA* is a trusted node that you set up and use to sign certificates for your system only.

Commercial Certification Authorities

There are several commercial CAs available. The mechanism for signing a certificate using a commercial CA depends on which CA you choose.

An advantage of commercial CAs is that they are often trusted by a large number of people. If your applications are designed to be available to systems external to your organization, use a commercial CA to sign your certificates. If your applications are for use within an internal network, a private CA might be appropriate.

Before choosing a CA, examine the certificate signing policies of some commercial CAs and, if your applications are designed to be available on an internal network only, review the potential costs of setting up a private CA.

Private Certification Authorities

If you wish to take responsibility for signing certificates for your system, set up a private CA. To set up a private CA, you require access to a software package that provides utilities for creating and signing certificates. Several packages of this type are available.

One software package that allows you to set up a private CA is OpenSSL. OpenSSL is an implementation of SSL. The OpenSSL package includes basic command line utilities for generating and signing certificates and these utilities are available with every installation of OrbixSSL.

To set up a private CA using OrbixSSL, do the following:

1. Choose a suitable host to act as CA.
2. Install OrbixSSL on the CA host.
3. Use the OpenSSL utilities to create a certificate and private key for the CA.
4. Copy the CA certificate and private key to the required directories on the CA host.

When you complete these steps, you can use the OpenSSL utilities to sign application certificates for your system.

Choosing a Host for a Private Certification Authority

Choosing a host is an important step in setting up a private CA. The level of security associated with the CA host determines the level of trust associated with certificates signed by the CA.

If you are setting up a CA for use in the development and testing of OrbixSSL applications, use any host that the application developers can access. However, when you create the CA certificate and private key, do not make the CA private key available on hosts where security-critical applications run.

If you are setting up a CA to sign certificates for applications that you are going to deploy, make the CA host as secure as possible. For example, take the following precautions to secure your CA:

- Do not connect the CA to a network.
- Restrict all access to the CA to a limited set of trusted users.
- Protect the CA from radio-frequency surveillance using an RF-shield.

When you choose a suitable host to act as the CA host, install OrbixSSL and use the OpenSSL utilities to create the CA certificate and private key.

Creating a Self-Signed Certificate and Private Key

A self-signed certificate is a CA certificate in which the issuer and subject of the certificate are identical. It acts as the final authority in a certificate chain. To create a self-signed certificate and private key for your CA, use the OpenSSL utility `openssl` to run the command `req` as follows:

```
openssl req -config openssl_config_file -days 365
-out ca_cert_file.pem -new -x509
```

The utility `openssl` is located in the OrbixSSL `bin` directory. Replace ***openssl_config_file*** with the fully qualified name of the OpenSSL configuration file `openssl.cnf`. By default, OrbixSSL installs this file in the `config` directory of your Orbix installation.

The `req` command requests information that identifies the CA, including your organization name, organization address, and so on. This information comprises the CA's ***distinguished name***.

This command also asks you to specify a pass phrase with which `req` will encrypt the private key for the CA. Note the pass phrase and guard it carefully.

The req command outputs two files. The first output file is ca_cert_file.pem, which contains the CA certificate in Privacy Enhanced Mail (PEM) format. The second output file is named privkey.pem (this default filename can be overridden using the -keyout option) and contains the encrypted private key for your CA in PEM format.

Note: The integrity of your private CA depends on the security of the pass phrase used to encrypt the CA's private key and the integrity of the CA's private key file. These should be available only to trusted users of the CA.

An Example of Creating a Self-Signed Certificate and Private Key

Consider the example of creating a certificate and private key for a CA to be used in signing certificates within the finance department of Progress Software.

If the openssl.cnf file is installed in the default directory, run req as follows:

```
openssl req -config openssl config file -days 365
   -X509 -new -out demo_ca_1 -keyout demo_ca_1.pk
```

The req command begins by generating the private key for your CA. req prompts you to enter a pass phrase, which is used to encrypt the private key:

```
Generating a 512bit private key
.............+++++
...........+++++
writing new private key to 'privkey.pem'
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:
```

The default openssl.cnf file supplied with OrbixSSL configures the key length to 512 bits. This should be increased to 1024 bits for most live systems. When using 1024 bit keys, the initial SSL handshake is a number of times slower than for 512 bit keys, but the level of security obtained is very much greater.

The `req` command continues by requesting identification information for your CA:

```
Country Name (2 letter code) []: IE
State or Province Name (full name) []: Co. Dublin
Locality Name (eg, city) []: Dublin
Organization Name (eg, company) []: Progress Software
Organizational Unit Name (eg, section) []: Finance
Common Name (eg, YOUR name) []: Gordon Brown
Email Address []: gbrown@progress.com
```

The input for these identification fields should clearly identify the individual or group responsible for controlling the CA.

As a result of this operation, the `req` command outputs two files in the local directory. The CA certificate file is called `demo_ca_1`. The CA private key file is called `demo_ca_1.pk`.

Installing the Certificate and Private Key Files

To prepare the CA to sign certificates, do the following:

1. Ensure that the CA certificate file name matches the `certificate` value in the `openssl.cnf` file.
2. On the CA host, copy the CA certificate file to the *root certificate directory*. To locate this directory, consult the `dir` entry in `openssl.cnf`.
3. Ensure that the name of the CA private key file matches the `private_key` value in the `openssl.cnf` file.
4. On the CA host, copy the private key file to the directory specified by the `private_key` entry in `openssl.cnf`.

When you complete these steps, the CA is ready to sign application certificates.

Publishing a Certification Authority Certificate

To authenticate a certificate signed by a CA, an application requires access to the CA's own certificate. To install a CA certificate on an OrbixSSL application host, copy the CA certificate file to a directory on the local file system. Limit write access to this file to a single trusted user. Do no make the CA private key file available to hosts other than the CA itself.

The name of the CA certificate file must match the `certificate` value in the `openssl.cnf` file. To indicate that it trusts the CA, the application must call the method `IT_SSL.addTrustedCert()` specifying the name of the CA file as a parameter.

Certificates Signed by Multiple Certification Authorities

A CA certificate may be signed by another CA. For example, an application certificate may be signed by the CA for the finance department of Progress Software, which in turn is signed by a commercial CA.

This system of signing certificates is known as *certificate chaining*. An application can accept a signed certificate if the CA certificate for any CA in the signing chain is available in the certificate file in the local root certificate directory.

To limit the length of certificate chains that an application accepts, the application programmer calls the method `IT_SSL.setMaxChainDepth()`.

Signing Application Certificates

If using a commercial CA, you must follow the CA's procedures for obtaining signed certificates. If using a private CA, you can sign application certificates for use in your system. The process for generating a signed certificate is as follows:

1. An individual or group responsible for an application generates a *certificate signing request* (CSR).
2. The CSR is submitted to the CA for signing.
3. The CA signs and returns the new certificate.
4. The certificate file is copied to the OrbixSSL certificates directory on the host in which the application runs.

When this process is complete, the OrbixSSL application can use the signed certificate to prove its identity to other applications.

Generating a Certificate Signing Request

To generate a certificate signing request (CSR), run the OpenSSL command `req` as follows:

```
openssl req -nodes -config openssl config file -days 365
   -new -out csr_file.pem
```

The `req` command requests information that identifies your application. This information includes the components of the distinguished name for your organization. The `-nodes` argument ensures that the output private key is unencrypted. The ouput key must not be encrypted as a Java application cannot understand the OpenSSL encrypted private key format. Instead, you can encrypt the output file using the `keyenc` command. See “Creating a Private Key File for Java Programs” on page 48.

The `req` command outputs two files. The first output file is *csr_file*`.pem`, which contains the CSR for your application. The second output file is called `privkey.pem`. It is unencrypted and contains the application private key. This output file must immediately be encrypted using `keyinc`.

The file *csr_file*`.pem` should now be transferred to the CA for signing.

An Example of Generating a Certificate Signing Request

Consider the example of generating a CSR for an OrbixSSL server application with server name `Bank`. Run `req` as follows:

```
openssl req -nodes -config openssl config file -days 365
    -new -out Bank-csr.pem
```

The `req` command begins by generating a private key for your application:

```
Generating a 512 bit private key
....+++++
..........+++++
writing new private key to 'privkey.pem'
```

The req command continues by requesting identification information for your certificate:

```
Country Name (2 letter code) []:IE
State or Province Name (full name) []: Co. Dublin
Locality Name (eg, city) []: Dublin
Organization Name (eg, company) []: Progress Software
Organizational Unit Name (eg, section) []: Finance
Common Name (eg, YOUR name) []: CORBA Server:Bank
Email Address []: info@progress.com
```

Your organization should define a clear policy for the format and content of the identification fields added to your application certificates. Enter the requested fields according to this policy.

Signing a Certificate

To sign a certificate, run the ca command as follows:

```
openssl ca -config openssl config file -days 365
   -in csr_file.pem > certname.pem
```

The ca command displays the identification information contained in the CSR. It is critically important that you check that this information is correct with respect to the application for which the CSR was generated.

The ca command asks you if you wish to sign the application certificate. If you sign the certificate, the ca command outputs the certificate in PEM format to the file *certname*.pem. This *certname*.pem file is supplied to the originator of the certificate request.

To return the certificate to the person who issued the CSR, copy the file to disk and transfer this file from disk to a location accessible to that person. This certificate file can then be copied to the certificates directory on the application host. To locate this directory, consult the certs value in the local openssl.cnf file.

Creating a Private Key File for Java Programs

A Java program that needs to use certificates must be able to access its private key in a special encrypted format. To create the private key file for an authenticated Java program, run the OrbixSSL utility keyenc on the unencrypted private key file output by OpenSSL, for example:

```
keyenc privkey.pem privkey.jpk password
```

Replace pass phrase with a pass phrase that encodes the private key. The program should call `IT_SSL.setRSAPrivateKeyFromFile()` to use the private key stored in the output file `privkey.jpk`.

An Example of Signing a Certificate

Consider the example CSR described in "An Example of Generating a Certificate Signing Request" on page 47. Sign this certificate by running `ca` (on the CA host) as follows:

```
openssl ca -config openssl config file
-days 365 -in Bank-csr.pem -out Bank-cert.pem
```

The output from this command begins by requesting the pass phrase used to encode the CA private key:

```
Enter PEM pass phrase:
```

If you enter the correct pass phrase, `ca` displays the identification information contained in the CSR:

```
Check that the request matches the signature
Signature ok

The Subjects Distinguished Name is as follows
countryName              :PRINTABLE:'IE'
stateOrProvinceName      :PRINTABLE:'Co. Dublin'
localityName             :PRINTABLE:'Dublin'
organizationName         :PRINTABLE:'Progress Software'
organizationalUnitName   :PRINTABLE:'Finance'
commonName               :PRINTABLE:'CORBA Server:Bank'
emailAddress             :IA5STRING:'info@progress.com'

Certificate is to be certified until Dec 12 14:11:12 1998
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
```

Check that the identification information contained in the CSR is correct in accordance with the security policy of your organization. If the information is correct, sign the certificate and commit the operation when prompted.

This command produces a signed application certificate in the file `Bank-cert.pem`. Use the `keyenc` command to encrypt the certificate immediately after it is created. See "Creating a Private Key File for Java Programs" on page 48.

Part III

OrbixSSL Programming

4

Defining a Security Policy

Each installation of OrbixSSL includes a set of Java classes that allow you to specify how your applications use SSL security. This chapter describes how you can use these classes to configure SSL security for each of your applications.

Defining a security policy means configuring your OrbixSSL applications to achieve the level of security required by your system. The OrbixSSL API includes methods that enable you to specify the location of certificates, which certificates applications should use, which certificates they should accept, and so on.

This chapter begins with an overview of the OrbixSSL API. It then describes each of the method calls required to define a comprehensive security policy. This guide provides a complete reference for all the Java classes in the OrbixSSL API. Refer to this part for more information about classes and methods introduced in this chapter. More specifically, the classes described here allow you to:

- Configure server authentication.
- Configure client authentication.
- Cofigure OrbixSSL types.
- Configure ciphers.
- Configure session caching.
- Provide IORs with security information. Non-Orbix clients can sometimes require this information.

Overview of the OrbixSSL API

The OrbixSSL Java classes are located in the OrbixSSL `classes` directory. These classes are defined in the package `IE.Iona.OrbixWeb.SSL`.

In this package, the class `IT_SSL` provides the core features of the OrbixSSL API. To access this class, first create an `IT_SSL` object using the static method `IT_SSL.init()`, for example:

```
...
import IE.Iona.OrbixWeb.SSL.IT_SSL;

public class OrbixSSLExample {
   public static void main(String args[]) {
      org.omg.CORBA.ORB orb =
         org.omg.CORBA.orb.init(args, null);
      IT_SSL ssl = IT_SSL.init();
      ...
   }
}
```

You must call the method `ORB.init()`, from package `org.omg.CORBA`, before calling `IT_SSL.init()`.

In addition to class `IT_SSL`, most SSL programs use the OrbixSSL class `IT_Format`. All authenticated applications also use the class `IT_X509Cert`.

Configuring Server Authentication

When developing an OrbixSSL application, you must do the following to ensure that server authentication succeeds:

- Specify which certificate each server should use.
- Specify the private key file and pass phrase for each server.
- Specify which certificates each client should accept.

This section describes how to implement each of these tasks using the OrbixSSL API. For the purposes of SSL communications, a server is any Orbix Java Edition program that can accept operation calls. This includes Orbix Java Edition servers and clients that accept callbacks from servers.

Specifying the Location of Certificates

To specify the location of the certificate files associated with a server, you must create an array of `IT_X509Cert` objects that represents the server's certificate chain. You then pass this array to the method `IT_SSL.setApplicationCertChain()`.

For example, if your server uses the OrbixSSL demonstration certificate file `demos/demo_server_1`, signed by the CA certificate in file `ca/demo_ca_1`, use the following code in your server program:

```
...
import IE.Iona.OrbixWeb.SSL.IT_SSL;
import IE.Iona.OrbixWeb.SSL.IT_X509Cert;
import IE.Iona.OrbixWeb.SSL.IT_Format;

public class OrbixSSLExample {
   public static void main(String args[]) {
      org.omg.CORBA.ORB orb =
         org.omg.CORBA.ORB.init(args,null);
      IT_SSL ssl = IT_SSL.init();
      IT_X509Cert certChain[] = new IT_X509Cert[2];
```

```
     try {
        certChain[0] = new IT_X509Cert
         ("OrbixSSL directory/certs/demos/
         demo_server_1", IT_Format.IT_FMT_PEM);
        certChain[1] = new IT_X509Cert
         ("OrbixSSL directory/certs/ca/demo_ca_1",
         IT_Format.IT_FMT_PEM );
        ssl.setApplicationCertChain(certChain);
        ...
     }
     ...
}
```

The method `IT_SSL.setApplicationCertChain()` accepts certificate chains in which the certificate files are coded using one of the following formats: ASN.1 Distinguished Encoding Rules (DER), Privacy Enhanced Mail (PEM), or RSA Laboratories' Public Key Cryptography Standards #12 (PKCS#12). PKCS is a set of informal standard protocols developed by RSA Laboratories for exchanging security information on the Internet. Web browsers commonly support certificate files in PKCS format.

Specifying the Private Key File and Pass Phrase

Each authenticated application has an associated certificate and private key. In OrbixSSL for Java, the private key is stored in a file separate from the certificate file. Consequently, you must use the method `IT_SSL.setRsaPrivateKeyFromFile()` to specify the location of the private key file.

For example, if your program uses the private key in the OrbixSSL `demos/demo_server_1.jpk` file, call this method as follows:

```
IT_SSL ssl = IT_SSL.init();
...

try {
   ssl.setRSAPrivateKeyFromFile(
   "/progress/OrbixSSL/certs/demos/
demo_server_1.jpk",
   IT_Format.IT_FMT_PEM);
}
...
```

In this example, the private key is stored in encrypted format, so you must also provide the pass phrase used to encrypt the private key. To do this, call the method `IT_SSL.setPrivateKeyPassword()`. For example:

```
...
import IE.Iona.OrbixWeb.SSL.IT_SSL;
import IE.Iona.OrbixWeb.SSL.IT_X509Cert;
import IE.Iona.OrbixWeb.SSL.IT_Format;

public class OrbixSSLExample {
   public static void main(String args[]) {
      ...
      IT_SSL ssl = IT_SSL.init();
      ...

      try {
         // Set private key.
         ssl.setPrivateKeyPassword("demopassword");
         ssl.setRSAPrivateKeyFromFile(
         "/progress/OrbixSSL/certs/
demo_server_1.jpk",
         IT_Format.IT_FMT_PEM);
      }
      ...
}
```

Specifying Certificates to Accept

The program that receives a certificate must validate it to ensure the identity of the server. OrbixSSL does some basic validation, and you can add more. To enable OrbixSSL to do this basic validation, you provide some information about which certificates your programs should accept.

The method `IT_SSL.addTrustedCert()` allows you to add a CA to the list of CAs that a program trusts. When you call this method, you pass the location of the CA certificate file as a parameter. This certificate file must be available to the program.

For example, if CA `newCA` is identified by the certificate file `/local/certs/newCA`, call `IT_SSL.addTrustedCert()` as follows:

```
...
import IE.Iona.OrbixWeb.SSL.IT_SSL;
import IE.Iona.OrbixWeb.SSL.IT_X509Cert;
import IE.Iona.OrbixWeb.SSL.IT_Format;

public class OrbixSSLExample {
   public static void main(String args[]) {
      ...
      IT_SSL ssl = IT_SSL.init();
      ...

      try {
         // Add trusted CA to list.
         ssl.addTrustedCA("/local/certs/newCA",
            IT_Format.IT_FMT_PEM);
      }
      ...
}
```

Using certificate chaining, a CA certificate can be signed by another CA. To ensure security for an application, it is often necessary to limit the maximum number of certificates in a chain in addition to specifying the list of trusted CAs.

To limit the default maximum chain depth that your program will accept, call the method `IT_SSL.setMaxChainDepth()`. During authentication, any certificate chain that exceeds the specified depth will cause the SSL handshake to fail.

For example, to set the maximum chain depth to five, call `IT_SSL.setMaxChainDepth()` as follows:

```
IT_SSL ssl = IT_SSL.init();
...

try {
   ssl.setMaxChainDepth(5);
}
...
```

A chain depth of one indicates that a certificate can be signed by one trusted CA only. A chain depth of two indicates that the CA certificate can in turn be signed by a trusted CA. If any CA in the chain is trusted, the application certificate is considered valid by OrbixSSL.

Configuring Client Authentication

Some secure applications, for example Internet banking systems, require that clients can identify themselves to servers. These applications use an extended SSL handshake, in which the server validates the client certificate. Client authentication is optional in SSL security.

To specify that a server should authenticate clients, call the method `IT_SSL.setClientAuthentication()` with a `true` parameter value:

```
IT_SSL ssl = IT_SSL.init();
...

try {
   ssl.setClientAuthentication(true);
}
...
```

When you call this method, the server requires each secure client to supply a certificate during the SSL handshake. If the server cannot authenticate the client, the handshake fails.

If the server uses client authentication, it must call `IT_SSL.addTrustedCert()` to establish a list of trusted CAs. A client that communicates with the server must have an associated certificate and private key.

In addition to servers, clients can use this method to authenticate the suppliers of any callbacks they receive.

Configuring OrbixSSL Application Types

Orbix Java Edition defines two general application types: clients, which call IDL operations on CORBA objects, and servers, which contain those objects. However, these roles are sometimes reversed. For example, in many applications, servers make callbacks to objects located in clients.

In OrbixSSL, it is important to be aware that all programs can potentially act as clients and servers. For each program, OrbixSSL allows you to specify an *invocation policy*. This policy determines whether the program uses SSL when connecting to a server and whether it uses SSL when it accepts connection attempts from clients. An invocation policy is a combination of these two independent settings.

Possible settings for making connections are:

- Only make connections to servers using SSL.
- Only make connections to servers without using SSL.
- Make connections using SSL, but allow insecure connections to specified interfaces or servers.
- Make connections to servers using SSL or without using SSL, as required.

Possible setting for accepting connection attempts are:

- Accept only connection attempts that use SSL.
- Accept only connection attempts that do not use SSL.
- Accept either connection attempts that use SSL or attempts that do not. In this case, the client determines whether to use SSL.

This chapter describes how you set the invocation policy for an OrbixSSL program and how programs interact based on their policy settings.

Choosing Invocation Policies

The most secure OrbixSSL system architecture is one in which all applications connect using SSL. If SSL security is available to all applications in your system, you should ensure that each application has a fully secure policy for making and accepting connections. This is the default setting for an OrbixSSL application.

The least secure system architecture is one in which no applications use SSL security. It is unlikely that your OrbixSSL system will consist of only insecure applications, but it may be acceptable for some of your applications to interact without using SSL.

For example, in a secure system it is sometimes necessary to accommodate existing applications that cannot communicate over SSL. In this case, your system could consist of a combination of fully secure applications, fully insecure applications, and applications that combine secure communications with insecure communications.

Setting an Invocation Policy

To specify the invocation policy for a program, call the method `IT_SSL.setInvocationPolicy()`. This method is defined as follows:

```
class IE.Iona.OrbixWeb.SSL.IT_SSL {
public:
   public void setInvocationPolicy(int pol)
      throws IT_SSLException;
   ...
};
```

The parameter `pol` specifies which invocation policy the application should use. This integer is a bitwise `OR` combination of the values defined in the class `IT_SSLInvocationOptions`. These values are:

```
IT_SECURE_ACCEPT
IT_INSECURE_ACCEPT
IT_INSECURE_CONNECT
IT_SECURE_CONNECT
IT_SPECIFIED_INSECURE_CONNECT
IT_SPECIFIED_SECURE_CONNECT
```

The values IT_SECURE_ACCEPT and IT_INSECURE_ACCEPT determine how the program behaves when receiving operation calls. The other values determine how the program behaves when making operation calls.

For example, to specify that a program should be able to receive both secure and insecure operation calls, but should make only secure operation calls, do the following:

```
IT_SSL ssl = IT_SSL.init();
...

try {
   ssl.setInvocationPolicy(
   IT_SSLInvocationOptions.IT_SECURE_ACCEPT |
   IT_SSLInvocationOptions.IT_INSECURE_ACCEPT |
   IT_SSLInvocationOptions.IT_SECURE_CONNECT);
}
...
```

You can specify only one connect option when calling this method.

How Invocation Policies Affect OrbixSSL Communications

Table 4.1 describes the set of client and target invocation policies that communicate successfully and indicates the type of communications associated with each case. The first column of this table indicates the client policy of the application that attempts to establish a connection. The second column indicates the target policy of the application that receives this connection attempt.

Client Policy	Target Policy	Resulting Communications
`IT_SECURE_CONNECT`	`IT_SECURE_ACCEPT`	Secure.
`IT_SECURE_CONNECT`	`IT_SECURE_ACCEPT` `IT_INSECURE_ACCEPT`	Secure.
`IT_SECURE_CONNECT`	`IT_INSECURE_ACCEPT`	N/A.
`IT_SPECIFIED_INSECURE_CONNECT`	`IT_SECURE_ACCEPT`	Secure.
`IT_SPECIFIED_INSECURE_CONNECT`	`IT_SECURE_ACCEPT` `IT_INSECURE_ACCEPT`	Secure unless explicitly specified by client.
`IT_SPECIFIED_INSECURE_CONNECT`	`IT_INSECURE_ACCEPT`	Insecure only if explicitly specified by client; otherwise N/A.
`IT_SPECIFIED_SECURE_CONNECT`	`IT_SECURE_ACCEPT`	Secure only if explicitly specified by client; otherwise N/A.
`IT_SPECIFIED_SECURE_CONNECT`	`IT_SECURE_ACCEPT` `IT_INSECURE_ACCEPT`	Insecure unless explicitly specified by client; otherwise secure.
`IT_SPECIFIED_SECURE_CONNECT`	`IT_INSECURE_ACCEPT`	Insecure unless explicitly specified by client; otherwise N/A.
`IT_INSECURE_CONNECT`	`IT_SECURE_ACCEPT`	N/A.
`IT_INSECURE_CONNECT`	`IT_SECURE_ACCEPT` `IT_INSECURE_ACCEPT`	Insecure.
`IT_INSECURE_CONNECT`	`IT_INSECURE_ACCEPT`	Insecure.

Table 4.1: *How Programs with Different Invocation Policies Communicate*

Limitations Imposed by Incompatible Invocation Policies

Because of incompatible security capabilities, limitations exist on the interaction between some programs. For example, an insecure client cannot communicate with a fully secure server. Such instances have the value `N/A` in the communications column of Table 4.1.

If a secure client attempts to communicate securely with an insecure target, for example by resolving a reference to an object in the target program, the client application receives an `SSL_FAILURE` exception.

If an insecure client attempts to communicate with a fully secure target, the client receives a `NO_PERMISSION` exception, or a communication failure.

Specifying Exceptions to an Invocation Policy

If your program has a client policy of `IT_SPECIFIED_INSECURE_CONNECT`, it can make insecure calls to specified interfaces or servers only. To specify the list of interfaces, the client must call the function `IT_SSL.specifySecurityForInterfaces()`. To specify the list of servers, the client must call `IT_SSL.specifySecurityForServers()`.

Similarly, if your program has a client policy of `IT_SPECIFIED_SECURE_CONNECT`, it can make secure calls to specified interfaces or servers only. The functions `IT_SSL.specifySecurityForInterfaces()` and `IT_SSL.specifySecurityForServers()` also allow a client to specify these interfaces and servers.

It is important to limit use of `IT_SPECIFIED_INSECURE_CONNECT` or `IT_SPECIFIED_SECURE_CONNECT`, because it is not difficult for a program to change the server name or interface that it uses. If a client passes sensitive data to a server, it should always use `IT_SECURE_CONNECT`. If a client does not pass sensitive data to a server, but the server passes sensitive data to the client, the server should force the client to connect using SSL.

Configuring Ciphers

OrbixSSL allows you to specify which ciphers should be used for SSL encryption. A *cipher suite* is a combination of the following SSL settings:

- Specification of the key exchange algorithm.

  RSA certificates are useful for key exchanges as RSA is a widely used public-key algorithm that can be used for either encryption or digital signing.
- Specification of the cipher to be used.

  Permitted ciphers are taken from the following list: `RC2`, `RC4`, `DES`, `3DES_EDE`, `CBC`.
- Specification of the hash algorithm to be used.

  Permitted hashes include `MD5` and `SHA`.

The OrbixSSL class `IT_SSLCipherSuite` defines each of the cipher suites permitted by OrbixSSL. These are:

```
IT_SSLV3_RSA_WITH_RC4_128_SHA
IT_SSLV3_RSA_WITH_RC4_128_MD5
IT_SSLV3_RSA_WITH_3DES_EDE_CBC_SHA
IT_SSLV3_RSA_WITH_DES_CBC_SHA

IT_SSLV3_RSA_EXPORT_WITH_DES40_CBC_SHA
IT_SSLV3_RSA_EXPORT_WITH_RC2_CBC_40_MD5
IT_SSLV3_RSA_EXPORT_WITH_RC4_40_MD5
```

To specify which cipher suites your application can use, first create an array of `IT_SSLCipherSuite` objects, then set each element of the array to a required cipher suite and pass the array to the method `IT_SSL.specifyCipherSuites()`. For example:

```
IT_SSL ssl = IT_SSL.init();
...

IT_SSLCipherSuite ciphers[] =
   new IT_SSLCipherSuite[2];
ciphers[0] = IT_SSLCipherSuite.
   IT_SSLV3_RSA_WITH_DES_CBC_SHA;

ciphers[0] = IT_SSLCipherSuite.
```

```
   IT_SSLV3_RSA_WITH_RC4_128_MD5;

try {
   ssl.specifyCipherSuites(ciphers);
}
...
```

You cannot use any combination of ciphers other than those defined in class `IT_SSLCipherSuite`.

OrbixSSL Session Caching Configuration

SSL session caching allows the reuse of information previously agreed between a client and server thus enabling faster subsequent reconnection. This can significantly increase server throughput if clients repeatedly reconnect to the server.

The method `IT_SSL.setCacheOptions()` allows you to configure session caching in your application. This method takes an integer parameter that contains a bitwise `OR` combination of the values defined in class `IE.Iona.OrbixWeb.SSL.IT_SSLCacheOptions`. This class is defined as follows:

```
class IT.Iona.OrbixWeb.SSL.IT_SSL {
   public:
      public static final in IT_SSL_CACHE_NONE;
      public static final in IT_SSL_CACHE_CLIENT;
      public static final in IT_SSL_CACHE_SERVER;
}
```

These options control the use of SSL session caching as follows:

`IT_SSL_CACHE_NONE`

This value means that OribxSSL clients and servers will not use SSL session caching. That is, they cannot accept re-used SSL session IDs offered by SSL clients, and will not offer to resume previously established SSL sessions when contacting servers for a second or subsequent time.

`IT_SSL_CACHE_CLIENT` This value means that OrbixSSL client programs will cache any sessions that are successfully established with servers. However, if subsequent attempts are made to reconnect to the server, then the initial session will be offered for reuse to the server. Whether the session is actually reused or not depends on the server’s policy with respect to session caching. This applies to servers when they are acting as clients as well as pure clients.

`IT_SSL_CACHE_SERVER` This value means that servers of OrbixSSL will cache any sessions that are successfully established with clients. If subsequent attempts are made to reconnect by clients, then the previously established session that is being offered by the client will be accepted provided that it has not been flushed from the OrbixSSL session cache.

It is important to note that for an OrbixSSL cache to be reused, SSL session caching has to be enabled for clients and servers. This applies to clients when they are receiving callbacks as well as to pure clients.

Providing IORs with SSL Information

When a non-Orbix client wants to obtain a server IOR from the Orbix daemon by means of IIOP, it is necessary to provide that IOR with SSL information. You can do this by means of the `putit` utility:

This is the full `putit` command syntax:

```
putit [-v] [-h <host>] [-per-client | -per-client-pid]
[ [-shared | -unshared] [-marker <marker>] ]
[ -j | -java [-classpath <classpath> | -addpath <path> ] ]
[ -oc <ORBclass> -os <ORBSingletonClass>] [ -jdk2]
| [-per-method [-method <method>] ]
[-port <iiop portnumber>]
[ -n <number of servers> ] [ -l ]
[ -ssl_secure | -ssl_semi_secure  [-ssl_client_auth]  [-
ssl_support_null_enc | -ssl_support_null_enc_only]  [-
ssl_support_null_auth | -ssl_support_null_auth_only] ]
```

```
<serverName> [ <commandLine> | -persistent ]
```

The ssl parameters are described in Table 4.2. To use them, you must specify either `-ssl_secure` or `-ssl_semi_secure` first.

putit Flag	Description
`-ssl_client_auth`	Indicates that the server authenticates clients.
`-ssl_support_null_enc`	This indicates that the NULL encryption SSL ciphersuites (which do not support confidentiality) are supported by the server.
`-ssl_support_null_enc_only`	This indicates that only the server supports the NULL encryption SSL ciphersuites..
`-ssl_secure`	This is the minimal flag needed to indicate that the server is SSL enabled. If this flag or –ssl_semi_secure are not supplied then the server is insecure and no SSL related data should be written to the IR. One of these two flags must be supplied before any other SSL flag is acceptable. An error should be presented to the user if they are not.
`-ssl_semi_secure`	This indicates a `SEMI_SECURE` server policy. If this flag or `–ssl_secure` are not supplied to `putit` then the policy is `INSECURE` and no SSL related stuff should be written to the IR. One of these two flags must be supplied before any other SSL flag is acceptable. An error should be presented to the user if they are not.
`-ssl_support_null_auth`	This flag indicates that the server supports null authentication. OrbixSSL servers do not currently support this.
`-ssl_support_null_auth_only`	This flag indicates that the server supports null authentication. OrbixSSL servers do not currently support this.

Table 4.2: *putit SSL Parameters*

Using the putit SSL Parameters

There are four groups of SSL parameters. If you want to use them, you must use one from Group 1, followed by one or none from each of the other three groups:

Group 1

```
-ssl_secure
-ssl_semi_secure
```

Group 2

```
-ssl_support_null_enc
-ssl_support_null_enc_only
```

Group 3

```
-ssl_support_null_auth
-ssl_support_null_auth_only
```

Group 4

```
-ssl_client_auth
```

As OrbixSSL supports per server process security policy settings, those settings specified by putit apply to all objects created by the server.

The most common use cases are:

```
Putit –ssl_secure demo/grid grid.exe
Putit –ssl_secure –ssl_client_auth demo/grid grid.exe
Putit –ssl_semi_secure demo/grid grid.exe
```

The following might be less common:

```
Putit –ssl_semi_secure –ssl_client_auth demo/grid grid.exe
```

5

Validating Certificates

During SSL authentication, OrbixSSL checks the validity of an application's certificate. This chapter describes how OrbixSSL validates a certificate and how you can use the OrbixSSL API to introduce additional validation to your applications.

The OrbixSSL API allows you to define functions that implement custom validation of certificates. During SSL authentication, OrbixSSL validates a certificate and then passes it to your custom validation function for examination. This functionality is very important in systems that log information about certificates or have application-specific requirements for the contents of each certificate.

An X.509 certificate contains information about the supplier and the CA that issued the certificate. The structure of a certificate is specified in Abstract Syntax Notation One (ASN.1), a standard syntax for describing messages that can be sent or received on a network.

OrbixSSL provides a set of Java classes that enable you to extract the information from a certificate without a detailed understanding of the corresponding ASN.1 definitions. When writing your certificate validation functions, use these classes to examine the certificate contents.

Overview of Certificate Validation

Figure 5.1 shows a server sending its certificate to a client during an SSL handshake. OrbixSSL code at the server reads the certificate from file and transmits it as part of the handshake. OrbixSSL code at the client reads the certificate from the network, checks the validity of its contents, and either accepts or rejects the certificate.

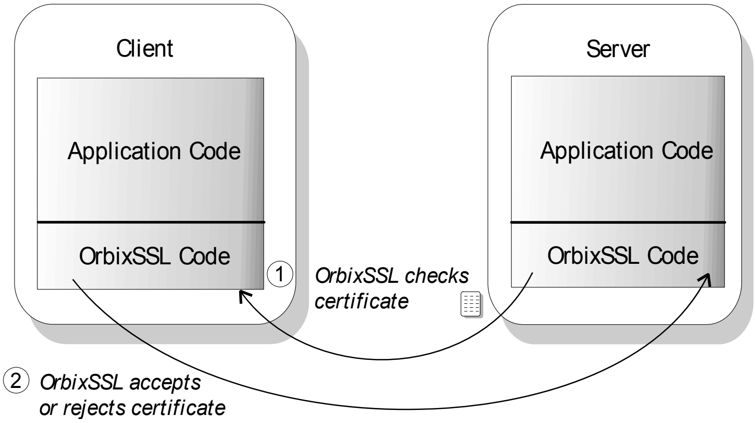
Client
Application Code
OrbixSSL Code
Server
Application Code
OrbixSSL Code
1 OrbixSSL checks certificate
2 OrbixSSL accepts or rejects certificate

Figure 5.1: *OrbixSSL Validating a Certificate*

The default certificate validation in OrbixSSL provides full security for the application. OrbixSSL checks:

- That the certificate is a validly constructed X.509 certificate.
- That the signature is correct for the certificate.
- That the certificate chain is validly constructed, consisting of the peer certificate plus valid issuer certificates up to the maximum allowed chain depth.

For some applications, it is necessary to introduce additional validation. For example, an application might require validation based on detailed information stored in the certificate common name, or some other certificate data.

Using OrbixSSL, you can register a method that carries out extra validation on certificates. When OrbixSSL receives a certificate, it validates it in the usual way and then passes it to your custom validation method, with an error code indicating whether the default validation succeeded or failed. You can then use the OrbixSSL API to examine the full contents of the certificate and instruct OrbixSSL whether to accept or reject it.

Figure 5.2 illustrates how a custom validation method interacts with OrbixSSL code during an SSL handshake.

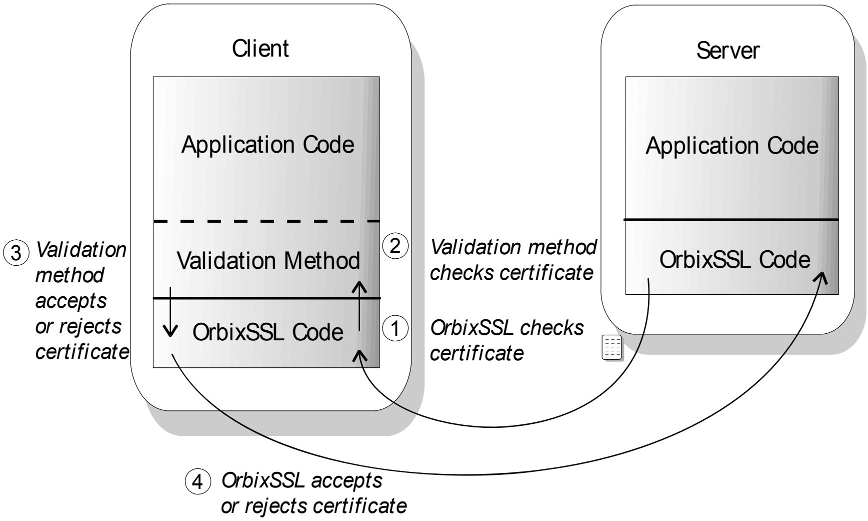
Client
Application Code
Validation Method
OrbixSSL Code
Server
Application Code
OrbixSSL Code
3 Validation method accepts or rejects certificate
2 Validation method checks certificate
1 OrbixSSL checks certificate
4 OrbixSSL accepts or rejects certificate

Figure 5.2: *Using a Custom Validation Method*

Introducing Additional Validation

OrbixSSL allows you to register two objects for additional certificate validation: one for validating certificates received from servers, and another for validating certificates received from clients. These two types of certificate often require different validation at the application level.

To register an object for server certificate validation, call the method IT_SSL.setValidateServerCallback(). This method is defined as:

```
class IE.Iona.OrbixWeb.SSL.IT_SSL {
public:
   public synchronized void
      setValidateServerCallback(
      IT_ValidateX509CertCB cb);
   ...
};
```

To register an object for server certificate validation, call the method IT_SSL.setValidateClientCallback(). This method is defined as:

```
class IE.Iona.OrbixWeb.SSL.IT_SSL {
public:
   public synchronized void
      setValidateClientCallback(
      IT_ValidateX509CertCB cb);
   ...
};
```

The single parameter to each of these methods is an object that implements interface IT_ValidateX509CertCB. To create a callback object, you define a class that implements this interface. The interface contains a single method, called validateCert() that OrbixSSL calls when it validates a certificate.

For example, you could use the following class to create a callback object:

```
import IE.Iona.OrbixWeb.SSL.*;

public class CertCallBack implements
IT_ValidateX509CertCB {
   public IT_CertValidity validateCert
      (IT_CertValidity systemOpinion,
      IT_X509CertChain peerCertChain) {
      ...
   }
}
```

To register an object of this type as a server certificate callback object, do the following:

```
IT_SSL ssl = IT_SSL.init();
...

try {
   CertCallBack serverValidCB =
      new CertCallBack();
   ssl.setValidateServerCallback(serverValidCB);
}
...
```

When OrbixSSL calls your validation method, `validateCert()`, it supplies two parameters. The first parameter is of type `IT_CertValidity`. This parameter indicates whether the default certificate validation succeeded or failed. The class `IT_CertValidity` defines the following values:

`IT_SSL_VALID_YES`	Indicates that the default certificate validation succeeded.
`IT_SSL_VALID_NO_APP_DECISION`	Indicates the default certificate validation failed, but the application can chose whether to accept or reject the certificate.
`IT_SSL_VALID_NO`	Indicates the default certificate validation failed, and the application must reject the certificate.

The second parameter is of type `IT_X509CertChain`. This parameter provides access to the full certificate chain. "Examining the Contents of a Certificate" on page 76 describes how you use this parameter to examine the contents of the peer certificate.

Your custom validation method must return an `IT_CertValidity` value. If this return value is `IT_SSL_VALID_NO_APP_DECISION`, OrbixSSL rejects the certificate. If the return value is `IT_SSL_VALID_YES`, OrbixSSL accepts the certificate. The return value has no effect if the first parameter passed to the method is `IT_SSL_VALID_NO`.

The OrbixSSL demonstration applications, located in the OrbixSSL `demos` directory, provide basic examples of creating certificate validation methods.

Examining the Contents of a Certificate

The role of a certificate is to associate an identity with a public key value. In more detail, a certificate includes:

- X.509 version information.
- A *serial number* that uniquely identifies the certificate.
- A *common name* that identifies the supplier.
- The *public key* associated with the common name.
- The name of the user who created the certificate, which is known as the *subject name*.
- Information about the certificate issuer.
- The signature of the issuer.
- Information about the algorithm used to sign the certificate.
- Some optional X.509 version three extensions. For example, an extension exists that distinguishes between CA certificates and end-entity certificates.

The second parameter to your custom validation method, of type IT_X509CertChain, provides access to the certificate chain received by OrbixSSL. Class IT_X509CertChain is defined as follows:

```
class IE.Iona.OrbixWeb.SSL.IT_X509CertChain {
public:
    public IT_X509CertChain();
    public void add(IT_X509Cert cert);
    public IT_X509Cert getCert(int pos);
    public IT_X509Cert getCurrentCert();
    public int getCurrentDepth();
    public IT_CertError getErrorInfo();
    public int numCerts();
    public String toString();
};
```

The method numCerts() indicates the number of certificates in the certificate chain. For example, if the peer certificate is signed by a single, self-signed CA, this method returns a value of two. The method getCert() returns a certificate from a particular position in the chain, starting at one. Repeated calls to getCurrentCert() iterate through the certificate chain.

When you call getCert() or getCurrentCert(), you receive an object of type IT_X509Cert that represents the required certificate. Class IT_X509Cert is defined as follows:

```
class IE.Iona.OrbixWeb.SSL.IT_X509Cert {
public:
  public IT_X509Cert(byte certData[])
    throws IT_X509BadCertException;
  public IT_X509Cert(String file,
    IE_Format filetype)
    throws IT_X509BadCertException,
    java.io.FileNotFoundException,
    java.io.IOException;
  public byte[] convert(IT_Format f);
  public IT_ExtensionList getExtensions();
  public IT_AVAList getIssuer();
  public IT_UTCTime getNotAfter();
  public IT_UTCTime getNotBefore();
  public java.math.BigInteger getSerialNumber();
  public IT_Signature getSignature();
  public IT_AVAList getSubject();
  public IT_PublicKeyInfo getSubjectPublicKey();
```

```
    public int getVersion();
    public int length(IT_Format f);
    public String toString();
};
```

This guide provides detailed information about the methods of this class. These methods return Java types corresponding to the ASN.1 types of the certificate contents. For example, `IT_X509Cert.getVersion()` returns an integer value that indicates the X.509 version number in use. In accordance with the X.509 standard, a value of `0` corresponds to version one, `1` corresponds to version two, and `2` corresponds to version three.

Working with Distinguished Names

An X.509 certificate uses ASN.1 ***distinguished name*** structures to store information about the certificate issuer and subject. A distinguished name consists of a series of attribute value assertions (AVAs). Each AVA associates a value with a field from the distinguished name.

For example, the distinguished name for a certificate issuer could be represented in string format as follows:

```
/C=IE/ST=Co. Dublin/L=Dublin/O=Progress/OU=PD/
CN=Progress
```

In this example, AVAs are separated by the `/` character. The first field in the distinguished name is `C`, representing the country of the issuer, and the corresponding value is the country code `IE`. This example distinguished name contains six AVAs.

When you call the methods `IT_X509Cert.getIssuer()` or `IT_X509Cert.getSubject()`, OrbixSSL returns the corresponding distinguished name as an object of type `IT_AVAList`. Class `IT_AVAList` is defined as follows:

```
class IE.Iona.OrbixWeb.SSL.IT_AVAList {
public:
     public IT_AVAList();
     public void add(IT_OID_Tag oid, IT_AVA ava);
     public byte[] convert(IT_Format f);
     public IT_AVA getAVA(int pos);
     public IT_AVA getAVAByOID(int seq[]);
     public IT_AVA getAVAByOIDTag(IT_OID_Tag t);
```

```
        public int getNumAVAs();
        public int length(IT_Format f);
    };
```

To retrieve a particular AVA from a distinguished name, use the `IT_AVAList` object that represents the name. Each AVA in a distinguished name has an associated ASN.1 object identifier (OID).

You can retrieve a particular field using any one of the following three methods:

`getAVA()`	Returns an AVA from a particular position in the distinguished name. To use this, you must understand the contents of the distinguished name that you receive.
`getAVAByOID()`	Returns an AVA associated with a particular OID. To use this, you must know the OID of the field you require.
`getAVAByOIDTag()`	Returns an AVA associated with a particular OID, but uses the tags defined in type `IT_OIDTag` instead of the actual OID. Using this method, you can access some of the commonly required distinguished name fields without knowing the corresponding OIDs or positions in the distinguished name.

Each of these functions returns an object of type `IT_AVA`. You can then use the methods of class `IT_AVA` to convert the AVA to a number of different formats, such as string format or DER format, and retrieve the associated OID. Refer to class `IT_AVA` on page 93 for more details.

Working with X.509 Extensions

Some X.509 version three certificates include extensions. These extensions can contain several different types of information. If you wish to extract information from the extensions included in a certificate, call `IT_X509Cert.getExtensions()` on the certificate object.

This method returns an object of type of type IT_ExtensionList. This class is defined as follows:

```
class IT_ExtensionList {
public:
   virtual int convert(char* buf, IT_Format f);
   virtual unsigned int getNumExtensions();
   virtual int getExtension(int pos,
      IT_Extension& retExt);
   virtual int getExtensionByOID(IT_OID oid);
   virtual int getExtensionByOIDTag(
      IT_OID_Tag oid);
   virtual int length(IT_Format f);
};
```

Like AVAs, each possible extension is associated with an ASN.1 OID. Given a list of extensions, you can retrieve the extension you require using any one of the following three methods:

`getExtension()` Returns an extension from a particular position in the extension list. To use this, you must understand the list of extensions included in the certificate.

`getExtensionByOID()` Returns an extension associated with a particular OID. To use this, you must know the OID of the extension you require. Use this method only when the extension you require is not available from `getExtensionByOIDTag()`.

`getExtensionByOIDTag()` Returns an extension associated with a particular OID, but uses the tags defined in type `IT_OIDTag` instead of the actual OID. Using this method, you can access some extensions without knowing the corresponding OIDs or positions in the extension list.

Each of these functions returns an object of type `IT_Extension`. You can then use the methods in class `IT_Extension` to convert the extension information to a number of different formats, such as string format or DER format, and retrieve the associated OID.

Example of a Certificate Validation Function

This section describes a simple validation function, registered in an OrbixSSL client, that prints the common name (CN) of a server to which the client connects. The code for this function is as follows:

```
...
import IE.Iona.OrbixWeb.SSL.*;

public class CertCallBack implements
IT_ValidateX509CertCB {
   public IT_CertValidity validateCert
      (IT_CertValidity systemOpinion,
      IT_X509CertChain peerCertChain) {

      IT_CertValidity certValidity =
         systemOpinion;
      if (systemOpinion.equals
         (IT_CertValidity.IT_SSL_VALID_YES) ) {
         if (peerCertChain.getCurrentDepth() == 0){
            IT_X509Cert peerCert =
               peerCertChain.getCurrentCert();
1           IT_AVAList subject =
               peerCert.getSubject();
2           IT_AVA commonName =
               subject.getAVAByOIDTag
               (IT_OID_Tag.IT_OIDT_commonName);
            System.out.println
               ("Common Name is" + commonName);
            String acceptableServerCN =
               "OrbixSSL for Java Demo " +
               "Certificate(no warranty!)";
            String daemonCN = "Orbix";
            String commonNameStr =
               commonName.toString();
            if (commonNameStr.equals
               (acceptableServerCN) ||
               commonNameStr.equals (daemonCN) ) {
               certValidity =
                  IT_CertValidity.IT_SSL_VALID_YES;
            }
```

```
                else {
                   certValidity =
                      IT_CertValidity.IT_SSL_VALID_NO;
                }
             }
          }

          return certValidity;
       }
    }
```

The code is explained as follows:

1. The `getSubject()` method returns the subject's distinguished name field from an X.509 certificate.
2. The common name field is extracted from the subject name. The common name field is the name of the entity to whom the certificate was issued.

To specify that this class validates incoming server certificates, include the following code in the client:

```
CertCallBack certCallback = new CertCallBack();
_CORBA.Orbix.SSL.setValidateServerCertCallback(certCallback);
```

6

Managing Pass Phrases

Every server secured with OrbixSSL has an associated certificate and private key. To access its private key, and use it to encrypt messages, a server must retrieve the associated pass phrase. This chapter shows you how to use OrbixSSL administration to supply pass phrases to servers.

As described in Chapter 2, “Getting Started with OrbixSSL”, a programmer can use the OrbixSSL API to specify the pass phrase associated with the private key of any OrbixSSL program. For example, the programmer might request the pass phrase from the user and then supply this to OrbixSSL.

One problem with this approach is that many OrbixSSL servers are launched automatically by the Orbix daemon. Ideally, such servers would not require user intervention to obtain a pass phrase.

For this reason, OrbixSSL provides an administrative solution to the problem of providing private key pass phrases to servers. The OrbixSSL server *key distribution mechanism* (KDM) is a utility that enables you to supply pass phrases to servers at runtime.

Using a Central Repository for Servers

The OrbixSSL server key distribution mechanism (KDM) allows an administrator to maintain a database of servers and their associated private key pass phrases. When the Orbix daemon launches an OrbixSSL server, OrbixSSL uses the KDM to retrieve the pass phrase.

This section describes the KDM in detail. It explains how the KDM works, how you can maintain the database of server pass phrases, and how you can replace the KDM with other key distribution systems.

Overview of the Key Distribution Mechanism

The KDM is a single process that runs on each server host in your secure system. The KDM stores an encrypted repository of server names and their associated pass phrases. When a client connects to an OrbixSSL server, the Orbix daemon uses the KDM to provide the correct pass phrase to the server.

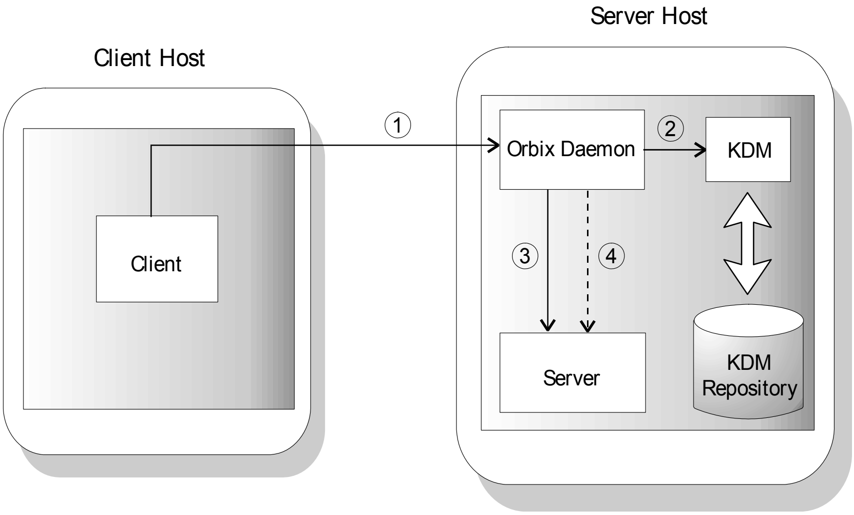

Figure 6.1: *Role of the Key Distribution Mechanism*

As shown in Figure 6.1, the following events happen when a client connects to a server that uses the KDM:

1. The client contacts the Orbix daemon on the server host.
2. The Orbix daemon requests security details for the server from the KDM.
3. The Orbix daemon launches the server, and simultaneously sends the pass phrase to the server.

All communications between the Orbix daemon and the KDM use SSL security. To ensure that only the Orbix daemon has access to server pass phrases, the KDM always uses client authentication. If another process requests a pass phrase from the KDM, this authentication fails. The configuration variable `IT_KDM_CLIENT COMMON _NAMES` described on page 88 specifies which clients can talk to the KDM.

Communications between the Orbix daemon and the server is secure. This ensures that an external process cannot read the server pass phrase when the daemon transfers it to the server process.

Configuring the Key Distribution Mechanism

Before running the KDM, add the following settings to the OrbixSSL configuration file on your server host:

```
OrbixSSL {
   IT_KDM_ENABLED = "TRUE";
   IT_KDM_REPOSITORY = "repository directory";
   IT_KDM_SERVER_PORT = "server port";
};

KDM {
   server {
      IT_CERTIFICATE_FILE =
   OrbixSSL.IT_CERTIFICATE_PATH +
         "KDM server cert file";
   };
   putkdm {
      IT_CERTIFICATE_FILE =
         OrbixSSL.IT_CERTIFICATE_PATH +
         "KDM client cert file";
   };
};
```

These configuration settings do the following:

`OrbixSSL.IT_KDM_ENABLED`
Enables the KDM. If the value of this variable is `TRUE`, all automatically launched servers on the host use the KDM. Otherwise, no servers use the KDM.

`OrbixSSL.IT_KDM_REPOSITORY`
Specifies the absolute path of the directory in which the KDM stores its database of pass phrases. The user that runs the KDM should have full read and write access to this directory.

`OrbixSSL.IT_KDM_SERVER_PORT`
Specifies the port number on which the KDM listens for incoming communications. You can use any available port for this value.

`KDM.server.IT_CERTIFICATE_FILE`
Specifies the certificate file that the KDM server should use to prove its identity. If you are using the OrbixSSL demonstration certificates, set this variable to the file `services/kdm_server` in the OrbixSSL `certificates` directory.

`KDM.putkdm.IT_CERTIFICATE_FILE`
Specifies the certificate file that the KDM utility `putkdm` should use to prove its identity to the KDM server. If you are using the OrbixSSL demonstration certificates, set this variable to the file `services/kdm_client` in the OrbixSSL `certificates` directory.

Configuring Client Authentication

To ensure that the KDM supplies accepts pass phrases from the `putkdm` utility only and supplies pass phrases to the Orbix daemon only, the KDM server always uses client authentication. To configure client authentication, add the following setting to the OrbixSSL configuration file:

```
OrbixSSL {
    IT_KDM_CLIENT_COMMON_NAMES =
        "Orbix daemon CN, putkdm CN";
};
```

Replace ***Orbix daemon CN*** with the common name from the Orbix daemon certificate. Replace ***putkdm CN*** with the common name from the certificate used by `putkdm`. For example, if you are using the OrbixSSL demonstration certificates, the required values are as follows:

```
OrbixSSL {
    IT_KDM_CLIENT_COMMON_NAMES =
        "Orbix, KDM Client";
};
```

If you have replaced the demonstration certificates, these common names must be the same as those you entered when creating your Orbix daemon and `putkdm` certificates.

Running the Key Distribution Mechanism

The KDM is an OrbixSSL server that the Orbix daemon contacts using an IDL interface. The KDM server executable is called `kdm` and is located in the `bin` directory of your installation.

Although the KDM is an OrbixSSL server, it is unlike a normal server in one respect: you can run the KDM before running the Orbix daemon. The KDM must be started before any automatically launched secure servers. To run the KDM:

1. Add the OrbixSSL `bin` directory to your path.
2. Run the following command:

   ```
   kdm
   ```

3. The KDM requests the pass phrase associated with its certificate.

If the KDM server uses the demonstration certificate `services/kdm_server`, enter `demopassword` as the pass phrase. If the KDM uses another certificate, enter the pass phrase for the associated private key.

Maintaining the Database

Before the Orbix daemon launches a server that uses the KDM, you must ensure that the server has a corresponding entry in the KDM database. To add an entry to the database, use the `putkdm` command:

```
putkdm server_name pass_phrase
```

The server name must match the name used to register the server in the Implementation Repository. The private key pass phrase must be at least six characters in length.

Verifying the Integrity of Server Executables

As an optional feature, the KDM allows you to ensure that the Orbix daemon only supplies pass phrases to the correct server executables. This prevents a malicious user from replacing a server executable with another program.

To support this feature, OrbixSSL provides a command-line utility, called `ccsit`, that takes a server executable file as input and outputs a ***cryptographic checksum*** based on the contents of the file. If the file is changed, the checksum becomes invalid.

Before running the `ccsit` utility, add the following settings to the OrbixSSL configuration file:

```
OrbixSSL {
   IT_CHECKSUMS_ENABLED = "TRUE";
   IT_CHECKSUMS_REPOSITORY = "checksums directory";
};
```

Replace ***checksums directory*** with a directory that can contain the checksums created by `ccsit`. In a production system, limit write access to your checksums directory to a single trusted user.

To register a checksum for a server, run the `ccsit` utility as follows:

```
ccsit server_file server_name
```

Replace *server_file* with the fully qualified name of the server executable. Replace *server_name* with the name used to register the server in the Implementation Repository.

Using the Key Distribution Mechanism

When the Orbix daemon launches a server and supplies its pass phrase using the KDM, it is not necessary for the server to call the API function `IT_SSL::setPrivateKeyPassword()`. If the server calls this function, it overrides the value supplied by the KDM. For information about how to write server code that uses the KDM when available, but supplies a password explicitly when the KDM is not available, refer to "Specifying the Private Key File and Pass Phrase" on page 56.

An IT_SSL object can use the hasPassword() method to determine whether a KDM password is available. For example, if an IT_SSL object `sslObj` has been initialized, the following code sample can make use of KDM:

```
if(!sslObj.hasPassword())
{
     read_password(): //user supplied
}
else
{
     //do nothing unless you want to override the
     //KDM-supplied password
}
```

Part IV

OrbixSSL Java Reference

Class IE.Iona.OrbixWeb.SSL.IT_AVA

Synopsis

As described in Chapter 5, “Validating Certificates”, an IT_AVAList is an abstraction of a distinguished name from a certificate. An IT_AVAList consists of a number of IT_AVA objects.

Individual IT_AVA objects represent an element of the distinguished name such as the common name field (CN) or organization unit (OU). You can retrieve a desired IT_AVA object can using the IT_AVAList class.

IT_AVA objects can be converted to a number of different forms such as string format or DER format. For more information on these formats, refer to convert() on page 93 and length() on page 94.

Java

```
class IE.Iona.OrbixWeb.SSL.IT_AVA {
public:
   public byte[] convert(IT_Format f);
   public int length(IT_Format f);
   public String toString();
};
```

IT_AVA.convert()

Synopsis

```
public byte[] convert(IT_Format f);
```

Description

This method converts data (Attribute Value Assertions or AVAs) to an another data format.

Parameters

f	The format of the required conversion. Currently, the only format supported is IT_Format.IT_FMT_DER.

Return Value

Returns an array of bytes that store the result of the conversion. Returns null if the required conversion is not supported.

IT_AVA.length()

Synopsis `public int length(IT_Format f);`

Description This method obtains the number of bytes required to store the result of converting to the format specified.

Parameters

`f`	The format of the required conversion. Currently, the only format supported is `IT_Format.IT_FMT_DER`.

Return Value Returns the number of bytes required to store the result of the conversion. Returns `-1` if the required conversion is not supported.

IT_AVA.toString()

Synopsis `public String toString();`

Description This method returns a string representation of the object. It overrides `toString()` in class `Object`.

Class IE.Iona.OrbixWeb.SSL.IT_AVAList

Synopsis

An IT_AVA_List consists of a number of IT_AVA objects and is an abstraction of the distinguished name fields in a certificate. This class provides a number of methods for obtaining individual IT_AVA objects.

A distinguished name is composed of a number of Attribute Value Assertions (AVAs). Each IT_AVA instance represents one component of the distinguished name. IT_AVA instances may be selected from an IT_AVAList using IT_OID_Tag values as keys, or by using an integer array that represents the ASN.1 object identifier. It is also possible to iterate over the list.

Java

```
class IE.Iona.OrbixWeb.SSL.IT_AVAList {
public:

      public IT_AVAList();
      public void add(IT_OID_Tag oid, IT_AVA ava);
      public byte[] convert(IT_Format f);
      public IT_AVA getAVA(int pos);
      public IT_AVA getAVABYOID(int seq[]);
      public IT_AVA getAVAByOIDTag(IT_OID_Tag t);
      public int getNumAVAs();
      public int length(IT_Format f);
};
```

See Also

```
IE.Iona.OrbixWeb.SSL.IT_AVA
IE.Iona.OrbixWeb.SSL.IT_Format
IE.Iona.OrbixWeb.SSL.IT_OID_Tag
```

IT_AVAList.IT_AVAList()

Synopsis

```
public IT_AVAList();
```

Description

This method constructs an empty AVA list.

IT_AVAList.add()

Synopsis `public void add(IT_OID_Tag oid, IT_AVA ava);`

Description This method adds an AVA to the list using the supplied ASN.1 object identifier as a key.

Parameters

`oid`	The object identifier associated with the AVA.
`ava`	The AVA instance to be added.

IT_AVAList.convert()

Synopsis `public byte[] convert(IT_Format f);`

Description This method converts to an alternate data format.

Parameters

`f`	The format of the required conversion. Currently, the only format supported is `IT_Format.IT_FMT_DER`.

Return Value Returns an array of bytes that store the result of the conversion. Returns `null` if the required conversion is not supported.

See Also class `IE.Iona.OrbixWeb.SSL.IT_Format`

IT_AVAList.getAVA()

Synopsis `public IT_AVA getAVA(int pos);`

Description This method obtains the AVA at the specified index.

Parameters

`pos`	The index position of the required AVA.

Return Value Returns the AVA at the index `pos`, if `pos` is a valid index. Returns `null` otherwise.

IT_AVAList.getAVAByOID()

Synopsis `public IT_AVA getAVAByOID(int seq[]);`

Description This method obtains the `IT_AVA` element of the `IT_AVAList` that has the requested object identifier. An ASN.1 object identifier is a sequence of numbers that identify a component in a hierarchical structure.

Parameters

`seq`	An ASN.1 OID.

Return Value Returns the AVA associated with the OID `seq`. Returns `null` if there is no AVA associated with the supplied OID.

See Also

```
IE.Iona.OrbixWeb.SSL.IT_OID
IE.Iona.OrbixWeb.SSL.IT_OID_Tag
IE.Iona.OrbixWeb.SSL.IT_AVAList.getAVAByOIDTag()
IE.Iona.OrbixWeb.SSL.IT_AVA.OID()
```

IT_AVAList.getAVAByOIDTag()

Synopsis `public IT_AVA getAVAByOIDTag(IT_OID_Tag t);`

Description This method obtains the `IT_AVA` that corresponds to the requested `IT_OID_Tag` value.

Parameters

`t`	A tag corresponding to an ASN.1 OID.

Return Value Returns the AVA associated with the OID `t`. Returns `null` if there is no AVA associated with `t`.

IT_AVAList.getNumAVAs()

Synopsis `public int getNumAVAs();`

Description This method obtains the number of AVA instances contained in this `IT_AVAlist`.

Return Value Returns the number of AVA elements.

IT_AVAList.length()

Synopsis

```
public int length(IT_Format f);
```

Description

This method returns the number of bytes required to store the result of converting to a specified format.

Parameters

`f`	The format of the required conversion. Currently, the only format supported is `IT_Format.IT_FMT_DER`.

Return Value

Returns the number of bytes required to store the result of the conversion. Returns `-1` if the required conversion is not supported.

See Also

class `IE.Iona.OrbixWeb.SSL.IT_Format`

Class IE.Iona.OrbixWeb.SSL.IT_CertError

Synopsis This class is used to obtain error information gathered during certificate chain processing.

Java

```
class IE.Iona.OrbixWeb.SSL.IT_CertError {
public:
      public IT_CertError(int errorCode, int depth);
      public int depth;
      public int errorCode;
};
```

IT_CertError.IT_CertError()

Synopsis

```
public IT_CertError(int errorCode, int depth);
```

Description Constructs an `IT_CertError` instance that holds error information gathered during certificate chain processing.

Parameters

`depth`	This field refers to the depth in the certificate chain at which point the error was encountered.
`errorCode`	This field contains the error code that OrbixSSL has associated with the certificate chain during validation of the certificate. Refer to class `IE.Iona.OrbixWeb.SSL.IT_SSLException` on page 105 for a complete list of error codes.

See Also

```
IE.Iona.OrbixWeb.SSL.IT_ValidateX509CertCB
IE.Iona.OrbixWeb.SSL.IT_X509CertChain.getErrorInfo()
IE.Iona.OrbixWeb.SSL.IT_SSL.setServerCertValidationCB()
IE.Iona.OrbixWeb.SSL.IT_SSL.setClientCertValidationCB()
```

Class IE.Iona.OrbixWeb.SSL.IT_CertValidity

Synopsis This class maintains constants used to indicate acceptance or rejection of a peer certificate during certificate validation. Specifically, it is used in certificate validation callbacks where OrbixSSL passes an instance of this class to the callback. If this instance indicates that OrbixSSL did not accept the certificate, you can get more information by calling `IT_X509CertChain.getErrorInfo()`.

Java

```
class IE.Iona.OrbixWeb.SSL.IT_CertValidity {
public:
     public static final IT_CertValidity IT_SSL_VALID_NO;
     public static final IT_CertValidity
             IT_SSL_VALID_NO_APP_DECESION;
     public static final IT_CertValidity IT_SSL_VALID_YES;
};
```

See Also `IE.Iona.OrbixWeb.SSL.IT_ValidateX509CertCB`

IT_CertValidity.IT_SSL_VALID_NO

Synopsis `public static final IT_CertValidity IT_SSL_VALID_NO;`

Description `IT_SSL_VALID_NO` indicates that OrbixSSL has rejected the certificate.

IT_CertValidity.IT_SSL_VALID_NO_APP_DECESION

Synopsis `public static final IT_CertValidity IT_SSL_VALID_NO_APP_DECESION;`

Description `IT_SSL_VALID_NO_APP_DECESION` indicates that OrbixSSL has rejected the certificate, but the application can choose to accept it.

IT_CertValidity.IT_SSL_VALID_YES

Synopsis `public static final IT_CertValidity IT_SSL_VALID_YES;`

Description `IT_SSL_VALID_YES` indicates that OrbixSSL has accepted the certificate, but the application can choose to reject it.

Class IE.Iona.OrbixWeb.SSL.IT_CommsSecuritySpec

Synopsis This class represents the name of an IDL interface or server and whether it is designated secure or insecure.

Java

```
class IE.Iona.OrbixWeb.SSL.IT_CommsSecuritySpec {
public:
      IT_CommsSecuritySpec(String, IT_Sec_CommsCategory);
      public string id;
      public IT_SecCommsCategory commsCat;
};
```

See Also `IE.Iona.OrbixWeb.SSL.IT_SecCommsCategory`

IT_CommsSecuritySpec.IT_CommsSecuritySpec()

Synopsis

```
public IT_CommsSecuritySpec
         (String id, IT_Sec_CommsCategory commsCat);
```

Description This method constructs `IT_CommsSecuritySpec` with the specified interface and security.

Parameters

`id`	This parameter specifies the name of the target IDL interface or server.
`commsCat`	This parameter specifies whether the interface or server is secure or insecure.

See Also `IE.Iona.OrbixWeb.SSL.IT_SecCommsCategory`

Class IE.Iona.OrbixWeb.SSL.IT_Extension

Synopsis

This class and class IE.Iona.OrbixWeb.SSL.IT_ExtensionList provide the OrbixSSL developer with an interface to any X.509 version three extensions that an X.509 certificate can include. The extension may be critical or it may be optional. The method getExtensions() in class IE.Iona.OrbixWeb.SSL.IT_X509Cert is used to obtain an IT_ExtensionList object. This class has a number of methods for retrieving individual extensions.

Class IE.Iona.OrbixWeb.SSL.IT_Extension allows you to access the data for an extension. Using the convert() and length() methods in class IE.Iona.OrbixWeb.SSL.IT_Extension, you can convert the extension data into a number of representations.

Java

```
class IE.Iona.OrbixWeb.SSL.IT_Extension {
public:
        public IT_Extension(byte data[],
                boolean critical, int asnOid[]);
        public byte[] convert(IT_Format f);
        public boolean critical();
        public IT_OID oid();
        public int length(IT_Format f);
};
```

IT_Extension.IT_Extension()

Synopsis

```
IT_Extension(byte data[], boolean critical, int asnOid[]);
```

Description

This method constructs an extension containing the specified data, whether the extension is critical, and the specified ASN.1 object identifier. The data value should be DER encoded.

Parameters

`data`	The data to be added to the X.509 certificate.
`critical`	This parameter should be `true` if the extension is required; `false` if the extension is optional.
`asnOID`	The ASN.1 object identifier.

IT_Extension.convert()

Synopsis `public byte[] convert(IT_Format f);`

Description This method converts extension data to an alternate data format.

Parameters

`f`	The format of the required conversion.

Return Value Returns an array of bytes that store the result of the conversion. Returns `null` if the required conversion is not supported.

IT_Extension.critical()

Synopsis `public boolean critical();`

Description This method determines whether or not this extension has been designated as critical. A critical extension must be present in the certificate.

IT_Extension.length()

Synopsis `public int length(IT_Format f);`

Description This method obtains the number of bytes required to store the result of converting to the specified format.

Parameters

`f`	The format of the required conversion.

Return Value Returns the number of bytes required to store the result of the conversion. Returns `-1` if the required conversion is not supported.

IT_Extension.oid()

Synopsis

```
public IT_OID oid();
```

Description This method obtains the ASN.1 object identifier associated with this extension.

Return Value Returns an instance of `IT_OID` that makes the ASN.1 object identifier available as an array of `int`.

Class IE.Iona.OrbixWeb.SSL.IT_ExtensionList

Synopsis This class and class IE.Iona.OrbixWeb.SSL.IT_Extension provide the OrbixSSL developer with an interface to any X.509 version three extensions that an X.509 certificate can include.

The method getExtensions() in class IE.Iona.OrbixWeb.SSL.IT_X509Cert is used to obtain an IT_ExtensionList object. This class has a number of methods for retrieving individual IT_Extension extensions.

Class IE.Iona.OrbixWeb.SSL.IT_Extension provides an interface to accessing the data for one extension. Class IE.Iona.OrbixWeb.SSL.IT_ExtensionList provides methods to retrieve IT_Extension instances using object identifiers, and using integer indices.

Java

```
class IE.Iona.OrbixWeb.SSL.IT_ExtensionList {
public:
      public IT_ExtensionList();
      public void add(IT_OID_Tag tag, IT_Extension extension);
      public byte[] convert(IT_Format f);
      public IT_Extension getExtension(int pos);
      public IT_Extension getExtensionByOID(int seq[]);
      public IT_Extension getExtensionByOIDTag(IT_OID_Tag t);
      public int getNumExtensions();
      public int length(IT_Format f);
};
```

IT_ExtensionList.IT_ExtensionList()

Synopsis `public IT_ExtensionList();`

Description Constructs an empty extension list.

IT_ExtensionList.add()

Synopsis `public void add(IT_OID_Tag tag, IT_Extension extension);`

Description This method adds an extension associated with the supplied object identifier to the list.

Parameters

`tag`	The object identifier to be associated with `extension`. This object identifier may be used later to retrieve the extension instance.
`extension`	The extension to store in the list.

IT_ExtensionList.convert()

Synopsis `public byte[] convert(IT_Format f);`

Description This method converts to an alternate data format.

Parameters

`f`	The format of the required conversion. Currently, no formats are supported. If conversion is required, individual extensions should be retrieved and converted instead.

Return Value Returns an array of bytes that store the result of the conversion. Returns `null` if the required conversion is not supported.

See Also class `IE.Iona.OrbixWeb.SSL.IT_Format`

IT_ExtensionList.getExtension()

Synopsis `public IT_Extension getExtension(int pos);`

Description This method obtains the extension at the specified index in the list.

Parameters

`pos`	The index position of the required extension in this list.

Return Value Returns the extension at index `pos`, if `pos` is a valid index. Returns `null` otherwise.

IT_ExtensionList.getExtensionByOID()

Synopsis

```
public IT_Extension getExtensionByOID(int seq[]);
```

Description This method obtains the extension associated with the specified object identifier. This differs from `getExtensionsByOIDTag()` in that the object identifier is specified as an `int` array.

Parameters

`seq`	The object identifier of the extension required.

Return Value Returns the extension associated with `seq`. Returns `null` if there is no extension associated with the supplied object identifier.

See Also

```
IE.Iona.OrbixWeb.SSL.IT_OID
IE.Iona.OrbixWeb.SSL.IT_OID_Tag
IE.Iona.OrbixWeb.SSL.IT_ExtensionList.getExtension()
IE.Iona.OrbixWeb.SSL.IT_Extension.OID()
```

IT_ExtensionList.getExtensionByOIDTag()

Synopsis

```
public IT_Extension getExtensionByOIDTag(IT_OID_Tag t);
```

Description This method obtains the extension associated with the specified object identifier.

Parameters

`t`	The object identifier of the extension required.

Return Value Returns the extension associated with `t`. Returns `null` if there is no extension associated with the supplied object identifier.

See Also

```
IE.Iona.OrbixWeb.SSL.IT_OID
IE.Iona.OrbixWeb.SSL.IT_OID_Tag
IE.Iona.OrbixWeb.SSL.IT_ExtensionList.getExtension()
IE.Iona.OrbixWeb.SSL.IT_Extension.OID()
```

IT_ExtensionList.getNumExtensions()

Synopsis `public int getNumExtensions();`

Description This method obtains the number of extension instances in the list.

Return Value Returns the number of extension instances in the list.

IT_ExtensionList.length()

Synopsis `public int length(IT_Format f);`

Description This method obtains the number of bytes required to store the result of converting to a specified format.

Parameters

`f` The format of the required conversion.

Currently, no formats are supported. Individual extensions should be retrieved and converted instead.

Return Value Returns the number of bytes required to store the result of the conversion. Returns `-1` if the required conversion is not supported.

See Also

```
IE.Iona.OrbixWeb.SSL.IT_Format
IE.Iona.OrbixWeb.SSL.IT_ExtensionList.convert()
```

Class IE.Iona.OrbixWeb.SSL.IT_Format

Synopsis This class maintains a list of options for the `convert()` and `length()` methods found in several OrbixSSL classes. Each option signifies a different type of conversion that `convert()` can implement. When passed to the `length()` method, the number of bytes required to store the result of the required conversion is returned.

Java

```
class IE.Iona.OrbixWeb.SSL.IT_Format {
public:
      public static final IT_Format IT_FMT_DER;
      public static final IT_format IT_FMT_PEM;
      public String toString();
};
```

IT_Format.IT_FMT_DER

Synopsis `public static final IT_Format IT_FMT_DER;`

Description This option represents DER encoding; that is, bytes of raw data in ASN.1 (DER) format.

IT_Format.IT_FMT_PEM

Synopsis `public static final IT_format IT_FMT_PEM;`

Description This option represents PEM format. In this format, the certificate description precedes the certificate PEM data. PEM format is an ASCII encoding that is suitable for transmission in e-mail.

IT_Format.toString()

Synopsis

```
public String toString();
```

Description This method returns a string representation of the object. It overrides `toString()` in class `Object`.

Class IE.Iona.OrbixWeb.SSL.IT_OID

Synopsis

This class is used by OrbixSSL to hold information identifying an ASN.1 object. An ASN.1 object identifier is a sequence of integer values used to identify certificate components. ASN.1 is the low-level format in which X.509 certificates are stored. This class maintains an array of integers corresponding to the ASN.1 sequence of integers in an object identifier (OID).

OrbixSSL handles object identifiers as follows:

1. It provides an `IE.Iona.OrbixWeb.SSL.IT_OID_Tag` class which has values for a number of common objects. For example, `IT_OIDT_commonName` identifies the common name (CN) component of a subject field in a certificate. Use of this class is sufficient for most OrbixSSL developer requirements.
2. If class `IE.Iona.OribxWeb.SSL.IT_OID_Tag` does not list the desired OIDs, developers can directly supply the sequence of integers that corresponds to an OID.

For simplicity, the data members of this class are made public.

Java

```
class IE.Iona.OrbixWeb.SSL.IT_OID {
public:
        public int OID[];
        public IT_OID_Tag tag;
};
```

IT_OID.IT_OID()

Synopsis

```
public IT_OID(int oid[]);
```

Description

This method constructs an OID with the specified ASN.1 object identifier sequence.

See Also

`IE.Iona.OrbixWeb.SSL.IT_OID_Tag`

Class IE.Iona.OrbixWeb.SSL.IT_OID_Tag

Synopsis

`IT_OID_Tag` is a value that is used to identify an OID. Accessing a certificate component using an `IT_OID_Tag` is more convenient than using a raw sequence of integers.

Java

```
class IE.Iona.OrbixWeb.SSL.IT_OID_Tag {
public:
   public static final IT_OID_Tag ASNOidToITOid(int ANSOid[]);
   public static final IT_Oid_Tag ASNOidToITOid(String ASNOid);
   public String toString();

   public static final IT_OID_Tag
            IT_OIDT_authority_key_identifier;

   public static final IT_OID_Tag IT_OIDT_basic_constraints;
   public static final IT_OID_Tag IT_OIDT_bf_cbc;
   public static final IT_OID_Tag IT_OIDT_bf_cfb64;
   public static final IT_OID_Tag IT_OIDT_bf_ecb;
   public static final IT_OID_Tag IT_OIDT_bf_ofb64;

   public static final IT_OID_Tag IT_OIDT_certificate_policies;
   public static final IT_OID_Tag IT_OIDT_commonName;
   public static final IT_OID_Tag IT_OIDT_countryName;
   public static final IT_OID_Tag IT_OIDT_crl_number;

   public static final IT_OID_Tag IT_OIDT_des_cbc;
   public static final IT_OID_Tag IT_OIDT_des_cfb64;
   public static final IT_OID_Tag IT_OIDT_des_ecb;
   public static final IT_OID_Tag IT_OIDT_des_ede;

   public static final IT_OID_Tag IT_OIDT_des_ede3;
   public static final IT_OID_Tag IT_OIDT_des_ede3_cbc;
   public static final IT_OID_Tag IT_OIDT_des_ede3_cfb64;
   public static final IT_OID_Tag IT_OIDT_des_ede3_ofb64;

   public static final IT_OID_TAG IT_OIDT_des_ede_cbc;
   public static final IT_OID_Tag IT_OIDT_des_ede_cfb64;
   public static final IT_OID_Tag IT_OIDT_des_ede_ofb64;
   public static final IT_OID_Tag IT_OIDT_des_ofb64;
public static final IT_OID_Tag IT_OIDT_desx_cbc;
public static final IT_OID_Tag IT_OIDT_dhKeyAgreement;
public static final IT_OID_Tag IT_OIDT_dsa;
public static final IT_OID_Tag IT_OIDT_dsaWithSHA;
public static final IT_OID_Tag IT_OIDT_dsaWithSHA1;

public static final IT_OID_Tag IT_OIDT_idea_cbc;
public static final IT_OID_Tag IT_OIDT_idea_cfb64;
public static final IT_OID_Tag IT_OIDT_idea_ecb;
public static final IT_OID_Tag IT_OIDT_idea_ofb64;

public static final IT_OID_Tag IT_OIDT_issuer_alt_name;
public static final IT_OID_Tag IT_OIDT_key_usage;
public static final IT_OID_Tag IT_OIDT_ld_ce;
public static final IT_OID_Tag IT_OIDT_localityName;

public static final IT_OID_Tag IT_OIDT_md2;
public static final IT_OID_Tag IT_OIDT_md2WithRSAEncryption;
public static final IT_OID_Tag IT_OIDT_md5;
public static final IT_OID_Tag IT_OIDT_md5WithRSAEncryption;
public static final IT_OID_Tag IT_OIDT_mdc2;
public static final IT_OID_Tag IT_OIDT_mdc2WithRSA;

public static final IT_OID_Tag IT_OIDT_netscape;
public static final IT_OID_Tag IT_OIDT_netscape_base_url;
public static final IT_OID_Tag IT_OIDT_netscape_ca_policy_url;
public static final IT_OID_Tag
        IT_OIDT_netscape_ca_revocation_url;

public static final IT_OID_Tag IT_OIDT_netscape_cert_extension;
public static final IT_OID_Tag IT_OIDT_netscape_cert_sequence;
public static final IT_OID_Tag IT_OIDT_netscape_cert_type;
public static final IT_OID_Tag IT_OIDT_netscape_comment;
public static final IT_OID_Tag IT_OIDT_netscape_data_type;
public static final IT_OID_Tag IT_OIDT_netscape_renewal_url;
public static final IT_OID_Tag IT_OIDT_netscape_revocation_url;
public static final IT_OID_Tag
        IT_OIDT_netscape_ssl_server_name;

public static final IT_OID_Tag IT_OIDT_organisationalUnitName;
public static final IT_OID_Tag IT_OIDT_organisationName;
public static final IT_OID_Tag IT_OIDT_pbeWithMD2AndDES_CBC;
public static final IT_OID_Tag IT_OIDT_pbeWithMD5AndDES_CBC;
public static final IT_OID_Tag IT_OIDT_pbeWithSHA1AndRC4;
public static final IT_OID_Tag IT_OIDT_pbeWithSHA1AndRC2_CBC;
public static final IT_OID_Tag IT_OIDT_pkcs;
public static final IT_OID_Tag IT_OIDT_pkcs3;

public static final IT_OID_Tag IT_OIDT_pkcs7;
public static final IT_OID_Tag IT_OIDT_pkcs7_data;
public static final IT_OID_Tag IT_OIDT_pkcs7_digest;
public static final IT_OID_Tag IT_OIDT_pkcs7_encrypted;
public static final IT_OID_Tag IT_OIDT_pkcs7_enveloped;
public static final IT_OID_Tag IT_OIDT_pkcs7_signed;
public static final IT_OID_Tag
        IT_OIDT_pkcs7_signedAndEnveloped;

public static final IT_OID_Tag IT_OIDT_pkcs9;
public static final IT_OID_Tag IT_OIDT_pkcs9_challengePassword;
public static final IT_OID_Tag IT_OIDT_pkcs9_contentType;
public static final IT_OID_Tag IT_OIDT_pkcs9_countersignature;
public static final IT_OID_Tag IT_OIDT_pkcs9_emailAddress;
public static final IT_OID_Tag IT_OIDT_pkcs9_extCertAttributes;
public static final IT_OID_Tag IT_OIDT_pkcs9_messageDigest;
public static final IT_OID_Tag IT_OIDT_pkcs9_signingTime;
public static final IT_OID_Tag
        IT_OIDT_pkcs9_unstructuredAddress;
public static final IT_OID_Tag IT_OIDT_pkcs9_unstructuredName;

public static final IT_OID_Tag
        IT_OIDT_private_key_usage_period;

public static final IT_OID_Tag IT_OIDT_rc2_cbc;
public static final IT_OID_Tag IT_OIDT_rc2_cfb64;
public static final IT_OID_Tag IT_OIDT_rc2_ecb;
public static final IT_OID_Tag IT_OIDT_rc2_ofb64;
public static final IT_OID_Tag IT_OIDT_rc4;
public static final IT_OID_Tag IT_OIDT_rsa;
public static final IT_OID_Tag IT_OIDT_rsadsi;
public static final IT_OID_Tag IT_OIDT_rsaEncryption;

public static final IT_OID_Tag IT_OIDT_sha;
public static final IT_OID_Tag IT_OIDT_sha1;
public static final IT_OID_Tag IT_OIDT_sha1WithRSAEncryption;
public static final IT_OID_Tag IT_OIDT_shaWithRSAEncryption;
public static final IT_OID_Tag IT_OIDT_stateOrProvinceName;
```

```
  public static final IT_OID_Tag IT_OIDT_subject_alt_name;
  public static final IT_OID_Tag IT_OIDT_subject_key_identifier;
  public static final IT_OID_Tag IT_OIDT_UNKNOWN;
  public static final IT_OID_Tag IT_OIDT_X500;
  public static final IT_OID_Tag IT_OIDT_X509;
};
```

See Also `IE.Iona.OrbixWeb.SSL.IT_OID`

IT_OID_Tag.ASNOidToITOid()

Synopsis `public static final IT_OID_Tag ASNOidToITOid(int ASNOid[]);`

Description This method converts an ASN.1 object identifier to the equivalent OrbixSSL object identifier.

Parameters

`ASNOid[]`	A specified ASN.1 object identifier.

ASNOidToITOid()

Synopsis `public static final IT_OID_Tag ASNOidToITOid(String ASNOid);`

Description This method converts an ASN.1 object identifier to the equivalent OrbixSSL object identifier.

Parameters

`ASNOid`	A specified ASN.1 object identifier.

IT_OID_Tag.toString()

Synopsis `public String toString();`

Description This method obtains a string representation of the object. It overrides `toString()` in class `Object`.

Return Value Returns the string representation of the object.

Class IE.Iona.OrbixWeb.SSL.IT_PublicKeyAlgorithm

Synopsis This class defines the a public key algorithm used for authentication purposes.

Java

```
class IE.Iona.OrbixWeb.SSL.IT_PublicKeyAlgorithm {
public:
   public static final IT_PublicKeyAlgorithm IT_RSA;
};
```

IT_PublicKeyAlgorithm.IT_RSA

Synopsis

```
public static final IT_PublicKeyAlgorithm IT_RSA;
```

Description OrbixSSL uses Rivest Shamir Adleman (RSA) public key cryptography for authentication purposes.

Class IE.Iona.OrbixWeb.SSL.IT_PublicKeyInfo

Synopsis Public key information is contained in this class. In particular, this class maintains the methods used for accessing the public key's exponent and modulus, and the algorithm used to generate the key. It also provides a method to convert to an instance of `java.security.PublicKey`.

Java

```
class IE.Iona.OrbixWeb.SSL.IT_PublicKeyInfo{
public:
   public IT_PublicKeyInfo(java.security.PublicKey key);
   public byte[] convert(IT_Format f);
   public IT_PublicKeyAlgorithm getAlgorithm();
   public java.math.BigInteger getExponent();
   public BigInteger getModulus();
   public int length(IT_Format f);
   public java.security.PublicKey toPublicKey();
};
```

IT_PublicKeyInfo.IT_PublicKeyInfo()

Synopsis `public IT_PublicKeyInfo(java.security.PublicKey key);`

Description This method constructs a public key based on the `java.security.PublicKey` provided.

Parameters

`key`	A public key.

IT_PublicKeyInfo.convert()

Synopsis `public byte[] convert(IT_Format f);`

Description This method converts to an alternate data format.

Parameters

`f` The format of the required conversion. (Currently, no formats are supported.)

Return Value Returns an array of bytes that store the result of the conversion. Returns `null` if the required conversion is not supported.

IT_PublicKeyInfo.getAlgorithm()

Synopsis `public IT_PublicKeyAlgorithm getAlgorithm();`

Description This method returns the algorithm used to generate the public key.

IT_PublicKeyInfo.getExponent()

Synopsis `public java.math.BigInteger getExponent();`

Description This method returns the public key exponent.

IT_PublicKeyInfo.getModulus()

Synopsis `public BigInteger getModulus();`

Description This method returns the public key modulus. (Currently, this is not implemented.)

IT_PublicKeyInfo.length()

Synopsis `public int length(IT_Format f);`

Description This method returns the number of bytes required to store the result of converting to the format specified.

Parameters

`f`	The format of the required conversion. (Currently, no formats are supported.)

Return Value Returns the number of bytes required to store the result of the conversion. Returns `-1` if the required conversion is not supported.

IT_PublicKeyInfo.toPublicKey()

Synopsis `public java.security.PublicKey toPublicKey();`

Description This method converts to an instance of `java.security.PublicKey`.

Class IE.Iona.OrbixWeb.SSL.IT_SecCommsCategory

Synopsis This class contains constants to specify whether an interface or server is secure or not.

Java

```
class IE.Iona.OrbixWeb.SSL.IT_SecCommsCategory {
public:
   public static final IT_SecCommsCategory IT_COMMS_CAT_INSECURE;
   public static final IT_SecCommsCategory IT_COMMS_CAT_SECURE;
};
```

See Also `IE.Iona.OrbixWeb.SSL.IT_CommsSecuritySpec`

IT_SecCommsCategory.IT_COMMS_CAT_INSECURE

Synopsis

```
public static final IT_SecCommsCategory IT_COMMS_CAT_INSECURE;
```

Description This option allows insecure communications.

IT_SecCommsCategory.IT_COMMS_CAT_SECURE

Synopsis

```
public static final IT_SecCommsCategory IT_COMMS_CAT_SECURE;
```

Description This option allows secure communication.

Class IE.Iona.OrbixWeb.SSL.IT_Signature

Synopsis This class contains information on a certificate signature and the algorithm used to generate it.

Java

```
class IE.Iona.OrbixWeb.SSL.IT_Signature {
public:
      public IT_Signature(IT_SignatureAlgType);
      public IT_SignatureAlgType getSignatureAlgType();
};
```

IT_Signature.IT_Signature()

Synopsis `public IT_Signature(IT_SignatureAlgType);`

Description This method constructs a signature generated by the specified signature algorithm.

Parameters

`IT_SignatureAlgType`	A specified signature algorithm.

IT_Signature.getSignatureAlgType()

Synopsis `public IT_SignatureAlgType getSignatureAlgType();`

Description This method retrieves the algorithm generated by the specified signature.

Parameters

`IT_SignatureAlgType`	A specified signature algorithm.

Class IE.Iona.OrbixWeb.SSL.IT_SignatureAlgType

Synopsis This class contains a list of algorithms used to generate signatures.

Java

```
class IE.Iona.OrbixWeb.SSL.IT_SignatureAlgType {
public:
     public static final IT_SignatureAlgType IT_SIG_MD5_WITH_RSA
};
```

IT_SignatureAlgType.IT_SIG_MD5_WITH_RSA

Synopsis

```
public static final IT_SignatureAlgType IT_SIG_MD5_WITH_RSA;
```

Description This value represents an algorithm used to generate signatures.

Class IE.Iona.OrbixWeb.SSL.IT_SSL

Synopsis

Class `IE.Iona.OrbixWeb.SSL.IT_SSL` is the primary interface to OrbixSSL. For example, it provides methods to load an application's certificate and private key, it allows you to configure a comprehensive security policy for an application, and it allows you to introduce customized validation of certificates. Before using the other methods of this class, you must call the method `IT_SSL.init()`.

Java

```
// Java
class IE.Iona.OrbixWeb.SSL.IT_SSL {
public:
   public static synchronised IT_SSL init() throws INITIALIZE;
   public static synchronized IT_SSL init(ORB orb)
      throws INITIALIZE;

   public static boolean isSSLInstalled();

   public synchronized void setValidateClientCertCallback
         (IT_ValidateX509CertCB cb);
   public synchronized void setValidateServerCertCallback
         (IT_ValidateX509CertCB cb);
   public IT_X509Cert getPeerCert(Socket socket)
         throws IT_SSLException;
   public IT_X509Cert getPeerCert(Object obj)
         throws IT_SSLException;
   public IT_X509Cert getPeerCert(Request req)
         throws IT_SSLException;

   public IT_SSLCipherSuite getNegotiatedCipherSuite(Object obj)
         throws IT_SSLException;
   public IT_SSLCipherSuite getNegotiatedCipherSuite(Request req)
         throws IT_SSLException;
   public IT_SSLCipherSuite getNegotiatedCipherSuite(Socket s)
         throws IT_SSLException;

   public synchronized int getInvocationPolicy();
   public void setInvocationPolicy(int pol)
         throws IT_SSLException;
   public void specifySecurityForInterfaces
           (IT_CommsSecuritySpec specList[]);
   public void specifySecurityForServers
           (IT_CommsSecuritySpec specList[]);

   public synchronized IT_SSLCipherSuite[] specifyCipherSuites
           (IT_SSLCipherSuite[]);

   public synchronized void setApplicationCertChain
      (IT_X509CertChain) throws IT_SSLException;
   public synchronized void setApplicationCertChain
      (IT_X509Cert certChain[]) throws IT_SSLException;
   public synchronized IT_X509Cert[] loadCertChain
      (String, IT_Format) throws IT_X509BadCertException,
      IOException, FileNotFoundException, IT_SSLException;

   public synchronized boolean getClientAuthentication();
   public synchronized boolean setClientAuthentication(boolean b);

   public synchronized int getMaxChainDepth();
   public synchronized int setMaxChainDepth(int depth);

   public synchronized void addTrustedCert(IT_X509Cert cert)
           throws IT_SSLException;
   public synchronized void addTrustedCert(byte derData[])
      throws IT_X509BadCertException, KeyManagementException,
      IT_SSLException;
   public synchronized void addTrustedCert(String file,
      IT_Format f) throws IT_X509BadCertException,
      KeyManagementException, IOException, FileNotFoundException,
      IT_SSLException;

   public synchronized int getCacheOptions();
   publice sysnchronized void setCacheOptions(int opts);

   public synchronized void setPrivateKeyPassword
           (String password);
   public synchronized void setRSAPrivateKeyFromDER
           (byte derData[]) throws IT_SSL_Exception;
   public synchronized void setRSAPrivateKeyFromFile
           (String file, IT_Format f) throws IT_SSLException;
};
```

IT_SSL.addTrustedCert()

Synopsis

```
public synchronized void addTrustedCert(IT_X509Cert cert)
        throws IT_SSLException;
```

Description

This method adds a certificate to the list of CA certificates. Certificates issued by the owners of one of the trusted certificates will be acceptable to the application.

Parameters

`cert`	The certificate of a trusted CA.

Exceptions

Throws an `IT_SSLException` exception if there is a problem adding the certificate.

See Also

`addTrustedCert()` in class `IE.Iona.OrbixWeb.SSL.IT_SSL`

IT_SSL.addTrustedCert()

Synopsis

```
public synchronized void addTrustedCert(byte derData[])
   throws IT_X509BadCertException, KeyManagementException,
   IT_SSLException;
```

Description

This method adds a certificate to the list of CA certificates.

Parameters

`derData`	The certificate data in DER encoded format.

Exceptions

Throws an `IT_X509BadCertException` exception if `derData` does not yield a valid certificate.

Throws a `KeyManagementException` exception if the public key contained in `derData` is invalid.

Throws an `IT_SSLException` if there is a problem adding the certificate to the list.

See Also

`IE.Iona.OrbixWeb.SSL.IT_SSL.addTrustedCert()`

IT_SSL.addTrustedCert()

Synopsis

```
public synchronized void addTrustedCert(String file,
   IT_Format f) throws IT_X509BadCertException,
   KeyManagementException, IOException, FileNotFoundException,
   IT_SSLException;
```

Description This method adds a certificate to the list of CA certificates.

Parameters

`file` The path to the file containing the application's certificate data.

`f` The format of the data in the file. For example:

`IT_Format.IT_FMT_PEM` (PEM format).

`IT_Format.IT_FMT_DER` (DER encoding).

Exceptions Throws an `IT_X509BadCertException` exception if the data contained in `file` yields a corrupt or invalid certificate.

Throws a `KeyManagementException` exception if the data contained in `file` yields a corrupt or invalid public key.

Throws an `IOException` exception if `file` cannot be used.

Throws a `FileNotFoundException` exception if `file` cannot be located.

Throws an `IT_SSLException` if there is a problem adding the certificate to the list.

See Also `IE.Iona.OrbixWeb.SSL.IT_SSL.addTrustedCert()`

IT_SSL.getCacheOptions()

Synopsis

```
public synchronized int getCacheOptions();
```

Description This method obtains the current setting for the OrbixSSL cache options. Cache options are contained in the returned integer as a bitwise `OR` combination.

Return Value Returns the current setting for the OrbixSSL cache.

See Also `IE.Iona.OrbixWeb.SSL.IT_SSL.setCacheOptions()`
`IE.Iona.OrbixWeb.SSL.IT_SSLCacheOptions`

IT_SSL.getClientAuthentication()

Synopsis `public synchronized boolean getClientAuthentication();`

Description This method is used to determine whether client certificate authentication is enabled or not.

Return Value This method returns `true` to signify that client authentication is enabled. Otherwise, it returns `false`.

See Also `IE.Iona.OrbixWeb.SSL.IT_SSL.setClientAuthentication()`

IT_SSL.getInvocationPolicy()

Synopsis `public synchronized int getInvocationPolicy();`

Description This method obtains the invocation policy settings for an OrbixSSL application. When called, it returns the current invocation policy settings for how clients and servers can accept and create SSL connections. The invocation policy for an OrbixSSL application specifies, for example, whether clients support or require SSL for incoming and outgoing connections.

Return Value The integer returned is a bitwise `OR` of options in class `IE.Iona.OrbixWeb.SSL.IT_SSLInvocationOptions`.

See Also `IE.Iona.OrbixWeb.SSL.IT_SSLInvocationOptions`
`IE.Iona.OrbixWeb.SSL.IT_SSL.setInvocationPolicy()`

IT_SSL.getMaxChainDepth()

Synopsis `public synchronized int getMaxChainDepth();`

Description This method returns the maximum depth allowed for certificate chains. Applications can change the maximum certificate chain depth by calling `setMaxChainDepth()`.

Return Value Returns a numeric value specifying the allowed maximum depth of the certificate chain.

See Also `IE.Iona.OrbixWeb.SSL.IT_SSL.setMaxChainDepth()`

IT_SSL.getNegotiatedCipherSuite()

Synopsis

```
public IT_SSLCipherSuite getNegotiatedCipherSuite(Object obj)
        throws IT_SSLException;
```

Description

This method allows OrbixSSL applications to query the ciphersuite that was chosen for connection to the specified peer. It does this by requesting the ciphersuite used by the SSL session which is associated with the specified remote object, `obj`.

Parameters

`obj`	A remote object.

Return Value

Returns the SSL ciphersuite associated with `obj` if the ciphersuite is known and available. Otherwise, it returns `null`.

Exceptions

Throws an `IT_SSLException` exception if there is a problem returning the negotiated ciphersuite. The exception can be queried to find the specific error code. Possible error codes include `IT_SSL_ERR_INSECURE_CONNECTION` and `IT_SSL_ERR_NO_CONNECTION`. For further information on error codes, refer to class `IE.Iona.OrbixWeb.SSL.IT_SSLException`.

See Also

`specifyCipherSuites() in class IE.Iona.OrbixWeb.SSL.IT_SSL`

IT_SSL.getNegotiatedCipherSuite()

Synopsis

```
public IT_SSLCipherSuite getNegotiatedCipherSuite(Request req)
        throws IT_SSLException;
```

Description

This method allows OrbixSSL applications to query the ciphersuite that was chosen for connection to the specified peer. It does this by requesting the ciphersuite used by the SSL session associated with the specified request.

Parameters

`req`	A request received from a connection.

Return Value

Returns the SSL ciphersuite associated with `req` if the ciphersuite is known and available. Otherwise, it returns `null`.

Exceptions Throws an `IT_SSLException` exception if there is a problem returning the negotiated ciphersuite. The exception can be queried to find the specific error code. Refer to class `IE.Iona.OrbixWeb.SSL.IT_SSLException` on page 105 for further information on error codes.

See Also `IE.Iona.OrbixWeb.SSL.IT_SSL.specifyCipherSuites()`

IT_SSL.getNegotiatedCipherSuite()

Synopsis

```
public IT_SSLCipherSuite getNeogotiatedCipherSuite(Socket s)
        throws IT_SSLException;
```

Description This method allows OrbixSSL applications to query the ciphersuite that was chosen for connection to the specified peer. It does this by requesting the ciphersuite used by the SSL session associated with the specified socket.

Parameters

`s`	A socket associated with a connection.

Return Value Returns the SSL ciphersuite associated with `s` if the ciphersuite is known and available. Otherwise, it returns `null`.

Exceptions Throws an `IT_SSLException` exception if there was a problem returning the negotiated ciphersuite. The exception can be queried to find the specific error code. Refer to class `IE.Iona.OrbixWeb.SSL.IT_SSLException` on page 105 for further information on error codes.

See Also `IE.Iona.OrbixWeb.SSL.IT_SSL.specifyCipherSuites()`

IT_SSL.getPeerCert()

Synopsis

```
public IT_X509Cert getPeerCert(Object obj) throws IT_SSLException;
```

Description This method allows OrbixSSL applications to query peer certificates. The certificate of the peer application is returned by retrieving the peer certificate information associated with the remote object, `obj`.

Parameters

`obj`	A remote object.

Return Value Returns the certificate belonging to the server implementing `obj` if the certificate is available. Otherwise it returns `null`.

Exceptions Throws an `IT_SSLException` exception if there is a problem returning the peer certificate. This exception can be queried to find the specific error code. Refer to class `IE.Iona.OrbixWeb.SSL.IT_SSLException` on page 105 for further information on error codes.

See Also `class IE.Iona.OrbixWeb.SSL.IT_X509Cert`

IT_SSL.getPeerCert()

Synopsis

```
public IT_X509Cert getPeerCert(Request req)
        throws IT_SSLException;
```

Description This method allows an OrbixSSL application to request the certificate of a peer. The certificate of the peer application is returned by retrieving the peer certificate information associated with the specified connection, `req`.

Parameters

`req`	A request received from another application.

Return Value Returns the certificate associated with `req` if the certificate is available. Otherwise it returns `null`.

Exceptions Throws an `IT_SSLException` exception if there is a problem returning the peer certificate. This exception can be queried to find the specific error code. For further information, refer to class `IE.Iona.OrbixWeb.SSL.IT_SSLException` on page 105.

IT_SSL.getPeerCert()

Synopsis

```
public IT_X509Cert getPeerCert(Socket socket)
        throws IT_SSLException;
```

Description This method allows OrbixSSL applications to query peer certificates. The certificate of the peer application is returned by retrieving the peer certificate information associated with the socket (`socket`) for a particular connection.

Parameters

`socket` The socket over which this application is communicating to its peer. A certificate can be returned only if `socket` is associated with an SSL connection.

Return Value Returns the peer certificate if available. Otherwise it returns `null`.

Exceptions Throws `IT_SSLException` if there is a problem returning the peer certificate. This exception can be queried to find the specific error code. For further information, refer to class `IE.Iona.OrbixWeb.SSL.IT_SSLException` on page 105.

See Also `class IE.Iona.OrbixWeb.SSL.IT_X509Cert`

IT_SSL.init()

Synopsis

```
public static synchronised IT_SSL init() throws INITIALIZE;
```

Description This method is responsible for initializing the class. It must be called by the application before any communications take place and before invoking any other `IT_SSL` methods.

This method creates and initializes an instance of `IT_SSL` and makes it available as `_CORBA.Orbix.SSL`. All subsequent SSL related operations should be called on this instance, which is returned by `init()`.

You must call `ORB.init()` before calling `IT_SSL.init()`.

Return Value Returns an instance of `IT_SSL` that you can use to call `IT_SSL` methods.

Exceptions Throws an `INITIALIZE` exception if there is an error during initialization. Possible causes of initialization failure include SSL being unavailable or disabled, or `ORB.init()` not being called. The exception message contains explanatory text.

IT_SSL.init()

Synopsis

```
public static synchronized IT_SSL init(ORB orb)
                                             throws INITIALIZE;
```

Description This method is an alternative to the version of init() that takes no parameters. One of these methods must be called by the application before any communications take place and before invoking any other `IT_SSL` methods.

This method creates and initializes an instance of `IT_SSL`. It associates the `IT_SSL` object with a particular `ORB` object. You can use this approach with OrbixWeb 3.1 and later. The `IT_SSL` object associated with each `ORB` is entirely independent of any other.

You must call `ORB.init()` before calling `IT_SSL.init()`.

Return Value Returns an instance of `IT_SSL` that you can use to call `IT_SSL` methods.

Exceptions Throws an `INITIALIZE` exception if there is an error during initialization. Possible causes of initialization failure include SSL being unavailable or disabled, or `ORB.init()` not being called. The exception message contains explanatory text.

IT_SSL.loadCertChain()

Synopsis

```
public synchronized IT_X509Cert[] loadCertChain
   (String file, IT_Format f) throws IT_X509BadCertException,
   IOException, FileNotFoundException, IT_SSLException;
```

Description This method loads a certificate chain from a file. You can then use this certificate chain to identify your application. To do this, pass the returned array to `setApplicationCertChain()`. You can also use this function to load the certificates of trusted CAs before calling `addTrustedCA()`.

This function supports files in PKCS#12 format. This format is commonly used by web browsers.

Parameters

`file`	The path to the file containing the application's certificate data.
`f`	The format of the data in the file. For example: `IT_Format.IT_FMT_PEM` (PEM format). `IT_Format.IT_FMT_DER` (DER encoding). `IT_Format.IT_FMT_PKCS12` (PKCS#12 format).

Return Value Returns an array of certificates representing the certificate chain read from file.

Exceptions Throws an `IT_X509BadCertException` exception if the data contained in `file` yields a corrupt or invalid certificate.

Throws an `IOException` exception if `file` cannot be used.

Throws a `FileNotFoundException` exception if `file` cannot be located.

Throws an `IT_SSLException` if there is a problem creating the certificate array.

IT_SSL.isSSLInstalled()

Synopsis

```
public static boolean isSSLInstalled();
```

Description This method indicates if SSL security is available to your OrbixWeb application.

Return Value Returns `true` if SSL security is available. Otherwise, it returns `false`.

IT_SSL.setApplicationCertChain()

Synopsis

```
public synchronized void setApplicationCertChain
        (IT_X509Cert certChain[]) throws IT_SSLException;
```

Description This method sets the application certificate and specifies a chain of CA certificates that sign the application certificate. Element `0` of `certChain` must be the application certificate. Each subsequent certificate must belong to the CA that issued the previous certificate.

Calling the method `setApplicationCertChain()` overwrites the current certificate chain. The private key, however, will be retained. If the peer certificate in `certChain` is associated with a private key other than that currently specified, reset the key using one of the private key methods.

Parameters

`certChain`	The certificate chain for the application.

Exceptions Throws an exception of type `IT_SSLException` if each certificate in the chain, starting with the second, does not belong to the CA that issued the previous certificate.

IT_SSL.setApplicationCertChain()

Synopsis

```
public synchronized void setApplicationCertChain
         (IT_X509CertChain certChain) throws IT_SSLException;
```

Description

This method uses an `IT_X509CertChain` object to set the application certificate chain.

Calling the method `setApplicationCertChain()` overwrites the current certificate chain. The private key, however, will be retained. If the peer certificate in `certChain` is associated with a private key other than that currently specified, reset the key using one of the private key methods.

Parameters

`certChain`	The certificate chain for the application.

Exceptions

Throws an exception of type `IT_SSLException` if each certificate in the chain, starting with the second, does not belong to the CA that issued the previous certificate.

IT_SSL.setCacheOptions()

Synopsis

```
public synchronized void setCacheOptions(int opts);
```

Description

This method configures the OrbixSSL session caching mechanism. Caching may be disabled entirely, enabled for clients only, enabled for servers, or enabled for both clients and servers.

See Also

`IE.Iona.OrbixWeb.SSL.IT_SSL.getCacheOptions()`
`IE.Iona.OrbixWeb.SSL.IT_SSLCacheOptions`

IT_SSL.setClientAuthentication()

Synopsis

```
public synchronized boolean setClientAuthentication(boolean b);
```

Description

This method enables or disables authentication of client certificates by a server. A server requests a peer certificate chain from a client only if this method is set to `true`. This method is primarily used by servers, but can be used by clients to authenticate any callbacks they receive.

Parameters

b   Setting this parameter to `true` enables client certificate authentication. Setting this parameter to `false` disables client certificate authentication.

Return Value This method returns the previous setting for client certificate authentication.

See Also `IE.Iona.OrbixWeb.SSL.IT_SSL.getClientAuthentication()`

IT_SSL.setInvocationPolicy()

Synopsis `public void setInvocationPolicy(int pol) throws IT_SSLException;`

Description This method specifies client and server access to the security policy, how clients and servers can accept and create SSL connections, and whether clients support or require SSL for incoming and outgoing connections. Applications have separate control with respect to using OrbixSSL security to establish connections and with respect to using OrbixSSL security to accept connection attempts.

The `setInvocationPolicy()` method sets the invocation policy for an OrbixSSL application as secure, insecure, or a combination of both. Using `specifySecurityForServers()` or `specifySecurityForInterfaces()`, you can make the invocation policy generally secure with specific exceptions. Similarly, you can make the invocation policy generally insecure but secure for specified servers and interfaces.

You can specify only one connect option in `pol`. Specifying more than one causes an exception to be thrown.

Parameters

pol   An integer value which is the bitwise `OR` combination of the following `IT_SSLInvocationOptions` flags:

`IT_SECURE_ACCEPT`

`IT_INSECURE_ACCEPT`

`IT_SPECIFIED_INSECURE_CONNECT`

`IT_INSECURE_CONNECT`

`IT_SECURE_CONNECT`

`IT_SPECIFIED_SECURE_CONNECT`

The options are explained as follows:

- `IT_SECURE_ACCEPT`

  This option means that the server will accept SSL connections. If the `IT_INSECURE_ACCEPT` option is not also specified, it will only accept SSL connections and reject non-SSL connections. It rejects non SSL connections by sending a `org.omg.CORBA.NO_PERMISSION` exception to the initiator and closing the connection.

- `IT_INSECURE_ACCEPT`

  This option means that the server is capable of accepting connections from non-SSL clients. If `IT_SECURE_ACCEPT` and `IT_INSECURE_ACCEPT` are both specified, the server will serve both secure and insecure clients. This type of server offers an optional connection authentication, privacy and integrity to clients that wish to avail of it. It should not be specified for servers whose services are regarded as sensitive and to which access should be restricted.

- `IT_SECURE_CONNECT`

  This option means that the client is capable of initiating SSL connections. Target servers should have a secure invocation policy. If this is not the case, an `org.omg.CORBA.NO_PERMISSION` exception will be thrown.

- `IT_SPECIFIED_INSECURE_CONNECT`

  For some secure client applications it may be too restrictive to allow only secure connections to all servers. When you choose this option, your attempts to connect to specified insecure interfaces or to specified insecure servers will be allowed. Refer to `specifySecurityForInterfaces()` on page 151 and `specifySecurityForServers()` on page 152 for further information.

- `IT_SPECIFIED_SECURE_CONNECT`

  This option means that the client try to communicate insecurely with all servers except when connecting through explicitly specified secure interfaces, or explicitly specified secure servers. When this option is specified, the client also attempts to use SSL when the server's IOR indicates that it requires SSL.

Note: This currently is only possible if the client uses a server IOR that contains a `TAG_SSL_SEC_TRANS` structure, indicating that the server supports or requires SSL. OrbixSSL automatically includes this tag in IORs that are generated by SSL servers.

- `IT_INSECURE_CONNECT`

  This option indicates that your client is capable of initiating insecure connections and that the client side of the application has no security requirements.

Exceptions Throws an `IT_SSLException` exception if more than one connect option is specified by `pol`.

See Also

```
IE.Iona.OrbixWeb.SSL.IT_SSLInvocationOptions()
IE.Iona.OribxWeb.SSL.IT_SSL.setClientAuthentication()
IE.Iona.OrbixWeb.SSL.IT_SSL.specifyCipherSuites()
IE.Iona.OrbixWeb.SSL.IT_SSL.specifySecurityForInterfaces()
IE.Iona.OrbixWeb.SSL.IT_SSL.specifySecurityForServers()
```

IT_SSL.setMaxChainDepth()

Synopsis

```
public synchronized int setMaxChainDepth(int depth);
```

Description This method allows individual applications to set or change the maximum depth allowed for certificate chains. During an SSL handshake, any peer certificate chains that exceed the specified depth causes the handshake to fail and an exception to be thrown.

Parameters

`depth`	Numeric value specifying the current acceptable maximum certificate chain depth.

Return Value Returns the previous maximum certificate chain depth setting.

See Also `IE.Iona.OrbixWeb.SSL.IT_SSL.getMaxChainDepth()`

IT_SSL.setPrivateKeyPassword()

Synopsis

```
public synchronized void setPrivateKeyPassword(String password);
```

Description

This method specifies the password used to decrypt the private key. Private keys stored in PEM format may be stored in encrypted form and loaded using the method `setRSAPrivateKeyFromFile()`. If the file is found to be encrypted, this password is used to decrypt it.

This method must be called before using `setRSAPrivateKeyFromFile()` with encrypted keys.

Parameters

`password`	The password used to decrypt the private key.

See Also

`IE.Iona.OrbixWeb.SSL.IT_SSL.setRSAPrivateKeyFromFile()`

IT_SSL.setRSAPrivateKeyFromDER()

Synopsis

```
public synchronized void setRSAPrivateKeyFromDer
        (byte[] derData[]) throws IT_SSLException;
```

Description

This method allows you to specify the private key to an OrbixSSL application. Private keys are used by OrbixSSL applications for authentication purposes.

Parameters

`derData`	The private key data in DER encoded format.

Exceptions Throws an `IT_SSLException` exception if `derData` does not yield a valid key.

See Also `IE.Iona.OrbixWeb.SSL.IT_SSL.setRSAPrivateKeyFromFile()`

IT_SSL.setRSAPrivateKeyFromFile()

Synopsis

```
public synchronized void setRSAPrivateKeyFromFile
        (String file, IT_Format f) throws IT_SSLException;
```

Description This method allows you to directly specify the private key to an OrbixSSL application. Private keys are used by OrbixSSL applications for authentication purposes.

Parameters

`file`	The path to the file containing the private key data. If the file contains bad key data, an `IT_SSLException` is thrown.
`f`	The format of the data in the file. For example: `IT_Format.IT_FMT_PEM` (PEM format). `IT_Format.IT_FMT_DER` (DER encoding).

Exceptions Throws an `IT_SSLException` exception if there was a problem setting the key. For example, if the data contained in `file` yields a corrupt or invalid private key.

See Also `IE.Iona.OrbixWeb.SSL.IT_SSL.setRSAPrivateKeyFromFile()`

IT_SSL.setValidateClientCertCallback()

Synopsis

```
public Synchronized void setValidateClientCertCallback
        (IT_ValidateX509CertCB cb);
```

Description This method enables you to validate client certificates by specifying an application-level certificate validation method. It allows servers or clients acting as servers to validate the peer certificate chain and to decide if a connection should be established. You can register methods to process server or client certificates separately, or the same method for both.

Passing `null` to this method disables client certificate validation.

Parameters

`cb`	This class implements interface `IT_ValidateX509certCB`. The method `validateCert()` is used to validate peer certificates.

See Also `IE.Iona.OrbixWeb.SSL.IT_SSL.setValidateServerCertCallback()`

IT_SSL.setValidateServerCertCallback()

Synopsis

```
public synchronized void setValidateServerCertCallback
        (IT_ValidateX509CertCB cb);
```

Description This method is used to validate server certificates. It specifies an application-level certificate validation method for server certificates, and allows clients or servers acting as clients to validate the peer chain and decide whether the connection should be established.

Passing `null` to this method disables server certificate validation.

Parameters

`cb`	This class implements interface `IT_ValidateX509certCB`. The method `validateCert()` is used to validate peer certificates.

See Also `IE.Iona.OrbixWeb.SSL.IT_SSL.getPeerCert()`
`IE.Iona.OrbixWeb.SSL.IT_SSL.setValidateServerCertCallback()`

IT_SSL.specifyCipherSuites()

Synopsis

```
public synchronized IT_SSLCipherSuite[] specifyCipherSuites
        (IT_SSLCipherSuite suite[]);
```

Description An application uses this method to specify the set of ciphersuites that it is prepared to use. By default, all ciphersuites defined in class `IE.Iona.OrbixWeb.SSL.IT_SSLCipherSuite` are enabled. Applications that require a more focused set of ciphersuites to be made available, however, should use `IT_SSL.specifyCipherSuites()`.

Parameters

`suite`	The set of ciphersuites to be used.

Return Value Returns the set of ciphersuites that will be used.

See Also `IE.Iona.OrbixWeb.SSL.IT_SSL.getNegotiatedCipherSuite()`

IT_SSL.specifySecurityForInterfaces()

Synopsis

```
public void specifySecurityForInterfaces
         (IT_CommsSecuritySpec specList[]);
```

Description This method allows clients and servers acting as clients to specify particular security requirements for interfaces. This method is used with the invocation policies `IT_SSLInvocationOptions.IT_SPECIFIED_INSECURE_CONNECT` and `IT_SSLInvocationOptions.IT_SPECIFIED_SECURE_CONNECT`.

Note: This method is applicable only when a connection to a server is being established. Once a connection to a server has been established, this connection can be used to access other interfaces in that server without reference to the list of specified interfaces. The main use anticipated for this method is to provide a means to allow insecure connections to be established through a specified insecure interface.

Parameters

`specList`	An array specifying interfaces and their associated security category.

See Also class `IE.Iona.OrbixWeb.SSL.IT_CommsSecuritySpec`
class `IE.Iona.OrbixWeb.SSL.IT_SecCommsCategory`

IT_SSL.specifySecurityForServers()

Synopsis

```
public void specifySecurityForServers
        (IT_CommsSecuritySpec specList[]);
```

Description

This method allows clients and servers acting as clients to specify particular security requirements for servers. This method is used with the invocation policies `IT_SSLInvocationOptions.IT_SPECIFIED_INSECURE_CONNECT` and `IT_SSLInvocationOptions.IT_SPECIFIED_SECURE_CONNECT`.

Parameters

`specList`	An array specifying servers and associated security categories.

See Also

`IE.Iona.OrbixWeb.SSL.IT_CommsSecuritySpec`
`IE.Iona.OrbixWeb.SSL.IT_SecCommsCategory`

Class IE.Iona.OrbixWeb.SSL.IT_SSLCacheOptions

Synopsis This class sets the current settings for the OrbixSSL session cache options. Caching can be enabled for clients only, enabled for servers only, enabled for both clients and servers, or disabled.

Java

```
class IE.Iona.OrbixWeb.SSL.IT_SSL {
public:

        public static final int IT_SSL_CACHE_CLIENT;
        public static final int IT_SSL_CACHE_NONE;
        public static final int IT_SSL_CACHE_SERVER;
};
```

See Also `IE.Iona.OrbixWeb.SSL.IT_SSL.setCacheOptions()`
`IE.Iona.OrbixWeb.SSL.IT_SSL.getCacheOptions()`

IT_SSLCacheOptions.IT_SSL_CACHE_CLIENT

Synopsis `public static final int IT_SSL_CACHE_CLIENT;`

Description This value means that there is to be SSL caching for OrbixSSL clients only. It may be combined with `IT_SSL_CACHE_SERVER` to enable caching for clients and servers.

IT_SSLCacheOptions.IT_SSL_CACHE_NONE

Synopsis `public static final int IT_SSL_CACHE_NONE;`

Description This value means that there is to be no SSL session caching.

IT_SSLCacheOptions.IT_SSL_CACHE_SERVER

Synopsis

```
public static final int IT_SSL_CACHE_SERVER;
```

Description

This value means that there is to be SSL caching for OrbixSSL servers only. It may be combined with `IT_SSL_CACHE_CLIENT` to enable caching for clients and servers.

Class IE.Iona.OrbixWeb.SSL.IT_SSLCipherSuite

Synopsis

This class maintains a list of ciphersuites supported by OrbixSSL. Using methods defined in class `IE.Iona.OrbixWeb.SSL.IT_SSL`, these ciphersuites can be enabled or disabled.

The list of ciphersuites supported by OrbixSSL is as follows:

```
IT_SSLV3_RSA_WITH_RC4_128_SHA
IT_SSLV3_RSA_WITH_RC4_128_MD5
IT_SSLV3_RSA_WITH_3DES_EDE_CBC_SHA
IT_SSLV3_RSA_WITH_DES_CBC_SHA
IT_SSLV3_RSA_EXPORT_WITH_DES40_CBC_SHA
IT_SSLV3_RSA_EXPORT_WITH_RC4_40_MD5
```

All of these ciphersuites comprise the following components:

- Specification of the key exchange algorithm.

  RSA certificates are useful for key exchanges as RSA is a widely used public-key algorithm that can be used for either encryption or digital signing.

- Specification of cipher to be used.

  Permitted ciphers are taken from the following list: `RC2`, `RC4`, `DES`, `3DES_EDE`, `CBC`.

- Specification of the hash algorithm to be used.

  Permitted hashes include `MD5` and `SHA`.

Only specific combinations of these options are available as listed, and one combination is referred to as a `CipherSuite`.

Java

```
class IE.Iona.OrbixWeb.SSL.IT_SSLCipherSuite {
public:
   public String toString();
   public static final IT_SSLCipherSuite
           IT_SSLV3_RSA_WITH_RC4_128_SHA;
   public static final IT_SSLCipherSuite
           IT_SSLV3_RSA_WITH_RC4_128_MD5;
   public static final IT_SSLCipherSuite
           IT_SSLV3_RSA_WITH_3DES_EDE_CBC_SHA;
```

```
public static final IT_SSLCipherSuite
        IT_SSLV3_RSA_WITH_DES_CBA_SHA;
public static final IT_SSLCipherSuite
        IT_SSLV3_RSA_EXPORT_WITH_DES40_CBC_SHA;
public static final IT_SSLCipherSuite
        IT_SSLV3_RSA_EXPORT_WITH_RC4_40_MD5;
```

See Also specifyCipherSuites() in class IE.Iona.OrbixWeb.IT_SSL

IT_SSLCipherSuite.toString()

Synopsis public String toString()

Description This method overrides toString() in class Object.

Class IE.Iona.OrbixWeb.SSL.IT_SSLException

Synopsis

OrbixSSL can throw exceptions of this type when errors occur. This class contains a list of possible OrbixSSL error codes.

Java

```
class IE.Iona.OrbixWeb.SSL.IT_SSLException {
   public IT_SSLException(int errorCode);
   public int getErrorCode();
   public String getErrorMessage();
   public String toString();

   public static final int IT_SSL_ERR_CERT_NOT_ISSUER;
   public static final int IT_SSL_ERR_HANDSHAKE_TIMEOUT;
   public static final int IT_SSL_ERR_INSECURE_CONNECTION;
   public static final int IT_SSL_ERR_INVALID_OPT_COMBO;
   public static final int IT_SSL_ERR_NO_CONNECTION;
   public static final int IT_SSL_ERR_ORB_NOT_INITIALISED;
   public static final int IT_SSL_ERR_SECURITY_INACTIVE;
   public static final int IT_SSLV_ERR_CERT_CHAIN_TOO_LONG;
   public static final int IT_SSLV_ERR_CERT_HAS_EXPIRED;
   public static final int IT_SSLV_ERR_CERT_NOT_YET_VALID;
   public static final int IT_SSLV_ERR_CERT_SIGNATURE_FAILURE;
}
```

IT_SSLException.IT_SSLException()

Synopsis

```
public IT_SSLException(int errorCode);
```

This method constructs an exception with the specified error code. You can examine the error message associated with the error code by calling `IT_SSLException.getErrorMessage()`.

IT_SSLException.getErrorCode()

Synopsis `public int getErrorCode();`

Description This method returns the error code associated with this exception.

IT_SSLException.getErrorMessage()

Synopsis `public String getErrorMessage();`

Description This method returns the error message associated with this exception.

IT_SSLException.toString()

Synopsis `public String toString();`

Description This method returns a short description of this object. It overrides `toString()` in class `Throwable`.

IT_SSLException.IT_SSL_ERR_CERT_NOT_ISSUER

Synopsis `public static final int IT_SSL_ERR_CERT_NOT_ISSUER;`

This error code signifies the failure of an attempt to add a CA to the end of a certificate chain. This can happen if that CA did not sign the previous certificate in the chain.

IT_SSLException. IT_SSL_ERR_INSECURE_CONNECTION

Synopsis `public static final int IT_SSL_ERR_INSECURE_CONNECTION;`

Description This error code signifies that an attempt was made to make a secure operation call on an insecure connection.

IT_SSLException.IT_SSL_ERR_INVALID_OPT_COMBO

Synopsis `public static final int IT_SSL_ERR_INVALID_OPT_COMBO;`

Description This error code signifies that an illegal combination of options was specified as a parameter to a method. For example, this can occur if more than one connect option is specified to `setInvocationPolicy()`.

IT_SSLException.IT_SSL_ERR_NO_CONNECTION

Synopsis `public static final int IT_SSL_ERR_NO_CONNECTION;`

Description This error code signifies an attempt to invoke an operation where there was no connection.

IT_SSLException.IT_SSL_ERR_ORB_NOT_INITIALISED

Synopsis `public static final int IT_SSL_ERR_ORB_NOT_INITIALISED;`

Description This error code signifies that `ORB.init()` was not called before `IT_SSL.init()`.

IT_SSLException.IT_SSL_ERR_SECURITY_INACTIVE

Synopsis `public static final int IT_SSL_ERR_SECURITY_INACTIVE;`

Description This error code signifies that SSL is not available or not activated.

IT_SSLException.
IT_SSLV_ERR_CERT_CHAIN_TOO_LONG

Synopsis `public static final int IT_SSLV_ERR_CERT_CHAIN_TOO_LONG;`

Description This error code signifies that the certificate chain depth exceeds the maximum specified by `IT_SSL.setMaxChainDepth()`.

IT_SSLException.IT_SSLV_ERR_CERT_HAS_EXPIRED

Synopsis

```
public static final int IT_SSLV_ERR_CERT_HAS_EXPIRED;
```

Description This error code signifies that the certificate expiry date is earlier than the current date.

IT_SSLException. IT_SSLV_ERR_CERT_NOT_YET_VALID

Synopsis

```
public static final int IT_SSLV_ERR_CERT_NOT_YET_VALID;
```

This error code signifies that the date at which the certificate becomes valid is later than the current date.

IT_SSLException. IT_SSLV_ERR_CERT_SIGNATURE_FAILURE

Synopsis

```
public static final int IT_SSLV_ERR_CERT_SIGNATURE_FAILURE;
```

This error code signifies that the signature of a certificate is invalid when decoded using the public key of the following certificate in the certificate chain.

Class IE.Iona.OrbixWeb.SSL.IT_SSLInvocationOptions

Synopsis

This class is used by an OrbixSSL application to provide the invocation options for the invocation policy of an application. The invocation policy for an OrbixSSL application specifies how the application uses SSL to communicate with other applications.

This class contains constants that allow you to specify how clients and servers accept and create OrbixSSL connections. The values detailed in this class are passed to `setInvocationPolicy()` in class `IE.Iona.OrbixWeb.SSL.IT_SSL`.

Note: Applications have separate control with respect to using OrbixSSL security to make connections and to accept connection attempts.

Java

```
class IE.Iona.OrbixWeb.SSL.IT_SSLInvocationOptions {
public:
   public static final int IT_INSECURE_ACCEPT;
   public static final String IT_INSECURE_ACCEPT_STRING;
   public static final int IT_INSECURE_CONNECT;
   public static final String IT_INSECURE_CONNECT_STRING;
   public static final int IT_SECURE_ACCEPT;
   public static final String IT_SECURE_ACCEPT_STRING;
   public static final int IT_SECURE_CONNECT;
   public static final String IT_SECURE_CONNECT_STRING;
   public static final int IT_SPECIFIED_INSECURE_CONNECT;
   public static final String
            IT_SPECIFIED_INSECURE_CONNECT_STRING;
   public static final int IT_SPECIFIED_SECURE_CONNECT;
   public static final String IT_SPECIFIED_SECURE_CONNECT_STRING;
};
```

IT_SSLInvocationOptions.IT_INSECURE_ACCEPT

Synopsis `public static final int IT_INSECURE_ACCEPT;`

Description This option means that the server is capable of accepting connections from insecure clients. It should not be specified for servers whose services are regarded as sensitive and to which access should be restricted.

IT_SSLInvocationOptions.IT_INSECURE_CONNECT

Synopsis `public static final int IT_INSECURE_CONNECT;`

Description This option means that the client is capable of initiating insecure connections.

IT_SSLInvocationOptions.IT_SECURE_ACCEPT

Synopsis `public static final int IT_SECURE_ACCEPT;`

Description This option means the server can accept SSL connections. If `IT_INSECURE_ACCEPT` is also specified, only SSL connections are accepted. In such a case, non SSL connections are rejected by sending a `NO_PERMISSION` exception to the initiator and closing the connection.

IT_SSLInvocationOptions.IT_SECURE_CONNECT

Synopsis `public static final int IT_SECURE_CONNECT;`

Description This option means that the client is capable of initiating SSL connections.

IT_SSLInvocationOptions.
IT_SPECIFIED_INSECURE_CONNECT

Synopsis `public static final int IT_SPECIFIED_INSECURE_CONNECT;`

Description This option allows connections through specified insecure interfaces, or to specified insecure servers.

IT_SSLInvocationOptions. IT_SPECIFIED_SECURE_CONNECT

Synopsis

```
public static final int IT_SPECIFIED_SECURE_CONNECT;
```

Description

This option means that the client communicates insecurely with all servers, except those explicitly specified.

Class IE.Iona.OrbixWeb.SSL.IT_UTCTime

Synopsis This class represents a time value and is used to specify certificate validity. You can convert this type to an instance of `java.util.Date` or to a string.

Java

```
class IE.Iona.OrbixWeb.SSL.IT_UTCTime {
public:
      java.util.Date toDate()
      public Date toDate();
      public String toString();
};
```

IT_UTCTime.toDate()

Synopsis `public Date toDate();`

Description This method converts the time value to an instance of `java.util.Date`.

IT_UTCTime.toString()

Synopsis `public String toString();`

Description This method converts the time value to a string. It overrides `toString()` in class `Object`.

Interface IE.Iona.OrbixWeb.SSL.IT_ValidateX509CertCB

Synopsis

This interface is used to validate client and server certificates. When OrbixSSL completes its validation of a certificate in a certificate chain, it calls the method `validateCert()`. You can implement this method to provide additional certificate validation.

OrbixSSL calls this method once for each certificate in the chain, passing the chain as a parameter and incrementing the chain depth each time. The peer certificate is set at element `0`. Subsequent certificates make up a CA chain.

User implementations of this method may validate the certificate in whatever manner is appropriate to the application. The method should return `IT_CertValidity.IT_SSL_VALID_YES` if the certificate is valid; `IT_CertValidity.IT_SSL_VALID_NO` if the certificate is invalid.

The parameter `systemOpinion` contains the result of OrbixSSL validation. Your custom validation method should examine the value of this parameter before returning a decision on the validity of a certificate.

Java

```
interface IE.Iona.OrbixWeb.SSL.IT_ValidateX509CertCB {
   public IT_CertValidity validateCert
            (IT_CertValidity systemOpinion,
             IT_X509CertChain peerCertChain);
}
```

See Also

```
IE.Iona.OrbixWeb.SSL.IT_SSL.setValidateClientCertCallback()
IE.Iona.OrbixWeb.SSL.IT_SSL.setValidateServerCertCallback()
```

IT_ValidateX509CertCB.validateCert()

Synopsis

```
public abstract ITCertValidity validateCert
         (IT_CertValidity systemOpinion,
          IT_X509CertChain peerCertChain);
```

Description This method determines the validity of the certificate.

Parameters

`peerCertchain`	The peer certificate chain.
`systemOpinion`	This parameter contains OrbixSSL's opinion of the validity of the certificate.

Return Value Returns `IT_CertValidity.IT_SSL_VALID_YES` if the certificate is deemed valid.

Returns `IT_CertValidity.IT_SSL_VALID_NO` if the certificate is deemed to be invalid.

Returns `IT_CertValidity.IT_SSL_VALID_NO_APP_DECESION` if the validity of the certificate cannot be determined.

Class IE.Iona.OrbixWeb.SSL.IT_X509BadCertException

Synopsis This class is used to indicate bad certificate data.

Java

```
class IE.Iona.OrbixWeb.SSL.IT_X509BadCertException {
public:
      public IT_X509BadCertException();
      public IT_X509CertException(String text);
}
```

IT_X509BadCertException.IT_X509BadCertException()

Synopsis `public IT_X509BadCertException();`

Description This method constructs an exception with the default error message.

IT_X509BadCertException.IT_X509BadCertException()

Synopsis `public IT_X509CertException(String text);`

Description This method constructs an exception with the error message provided.

Parameters

`text`	An error message.

Class IE.Iona.OrbixWeb.SSL.IT_X509Cert

Synopsis

This class provides an interface to a certificate. It is the primary interface for retrieving information about a certificate issuer, the subject's public key, certificate extensions, and other certificate attributes.

Java

```
class IE.Iona.OrbixWeb.SSL.IT_X509Cert {
public:
   public IT_X509Cert(byte certData[])
            throws IT_X509BadCertException;
   public IT_X509Cert(String file, IE_Format filetype)
            throws IT_X509BadCertException,
            java.io.FileNotFoundException, java.io.IOException;
   public byte[] convert(IT_Format f);
   public IT_ExtensionList getExtensions();
   public IT_AVAList getIssuer();
   public IT_UTCTime getNotAfter();
   public IT_UTCTime getNotBefore();
   public java.math.BigInteger getSerialNumber();
   public IT_Signature getSignature();
   public IT_AVAList getSubject();
   public IT_PublicKeyInfo getSubjectPublicKey();
   public int getVersion();
   public int length(IT_Format f);
   public String toString();
};
```

IT_X509Cert.IT_X509Cert()

Synopsis

```
public IT_X509Cert(byte certData[])
        throws IT_X509BadCertException;
```

Description This method constructs an `IT_X509Cert` from the given byte array, which must contain DER-encoded certificate data.

Parameters

`certData`	An X.509 certificate containing certificate data.

Exceptions Throws an `IT_X509BadCertException` exception if `certData` contains invalid certificate data.

IT_X509Cert.IT_X509Cert()

Synopsis

```
public IT_X509Cert(String file, IE_Format filetype)
        throws java.io.FileNotFoundException,
        java.io.IOException, IT_X509BadCertException;
```

Description This method constructs an `IT_X509Cert` from the data in the specified file. Specifying the format of data, the parameter `filetype` takes the value `IT_Format.IT_FMT_PEM` or `IT_Format.IT_FMT_DER`.

Parameters

`file`	A specified file.
`filetype`	A specified file type.

Exceptions Throws an `IT_X509BadCertException` exception if `file` contains invalid certificate data.

Throws a `java.io.FileNotFoundException` exception if `file` cannot be located.

Throws a `java.io.IOException` exception if there is a problem using `file`.

IT_X509Cert.convert()

Synopsis `public byte [] convert(IT_Format f);`

Description This method converts this certificate to the format specified by `f`. If the value of `f` is `IT_Format.IT_FMT_DER`, the returned `byte` array contains the certificate represented as DER-encoded data.

Parameters

`f`	A specified format.

Return Value Returns the certificate converted to the specified format `f`. Returns `null` if the required conversion is not supported.

See Also `IE.Iona.OrbixWeb.SSL.IT_Format`

IT_X509Cert.getExtensions()

Synopsis `public IT_ExtensionList getExtensions();`

Description This method retrieves the list of extensions that this certificate can include. Individual extensions can be retrieved from the returned `IT_ExtensionList` as `IT_Extension` instances. You can then retrieve the extension data from the `IT_Extension` objects.

Returns A populated extension list, if extensions exist. Returns `null` otherwise.

IT_X509Cert.getIssuer()

Synopsis `public IT_AVAList getIssuer();`

Description This method retrieves the distinguished name of the certificate issuer (CA) as an `IT_AVAList` instance. Individual components of the distinguished name (for example, the common name or the organization name) can be retrieved from the `IT_AVAList` instance.

See Also `IE.Iona.OrbixWeb.SSL.IT_AVAList`
`IE.Iona.OrbixWeb.SSL.IT_AVA`

IT_X509Cert.getNotAfter()

Synopsis `public IT_UTCTime getNotAfter();`

Description This method returns the time after which this certificate is invalid.

See Also `IE.Iona.OrbixWeb.SSL.IT_UTCTime`

IT_X509Cert.getNotBefore()

Synopsis `public IT_UTCTime getNotBefore();`

Description This method returns the time before which this certificate is invalid.

See Also `IE.Iona.OrbixWeb.SSL.IT_UTCTime`

IT_X509Cert.getSerialNumber()

Synopsis `public java.math.BigInteger getSerialNumber();`

Description This method returns the serial number of the certificate.

IT_X509Cert.getSignature()

Synopsis `public IT_Signature getSignature();`

Description This method returns the certificate signature as an instance of `IT_Signature`. The algorithm used to generate the signature can be obtained from this instance.

See Also `IE.Iona.OrbixWeb.SSL.IT_Signature`

IT_X509Cert.getSubject()

Synopsis `public IT_AVAList getSubject();`

Description This method retrieves the distinguished name of the entity that this certificate identifies as an `IT_AVAList` instance. Individual components of the distinguished name (common name or organization name, for example) can be retrieved from the `IT_AVAList` instance.

See Also `IE.Iona.OrbixWeb.SSL.IT_AVA`
`IE.Iona.OrbixWeb.SSL.IT_AVAList`

IT_X509Cert.getSubjectPublicKey()

Synopsis

```
public IT_PublicKeyInfo getSubjectPublicKey();
```

Description This method retrieves the public key of the entity that this certificate identifies. The algorithm used to generate the key, the key modulus and exponent can all be retrieved from the returned `IT_PublicKeyInfo` instance. This instance may also be converted to an instance of `java.security.PublicKey`.

See Also `IE.Iona.OrbixWeb.SSL.IT_PublicKey`
`IE.Iona.OrbixWeb.SSL.IT_PublicKeyInfo`

IT_X509Cert.getVersion()

Synopsis

```
public int getVersion();
```

Description This method obtains the X.509 version of the certificate

Return Value Returns the X.509 version of the certificate. In accordance with the X.509 specification, a value of `0` indicates version one, a value of `1` indicates version two and a value of `2` indicates version three.

IT_X509Cert.length()

Synopsis

```
public int length(IT_Format f);
```

Description This method obtains the number of bytes required to store the result of converting this certificate to the format specified by `f`.

Parameters

`pos`	The specified index position of the required extension in this list.

Return Value Returns the number of bytes required to store the result of the conversion. Returns `-1` if the required conversion is not supported otherwise.

See Also `IE.Iona.OrbixWeb.SSL.IT_AVA.length()`

IT_X509Cert.toString()

Synopsis

```
public String toString();
```

Description This method obtains the `String` representation of the certificate, which includes all X.509 certificate attributes. It overrides `toString()` in class `Object`.

Class IE.Iona.OrbixWeb.SSL.IT_X509CertChain

Synopsis This class represents a chain of certificates. The first certificate in the chain is the certificate authenticating the SSL client or server. Each subsequent certificate signs the previous one. An instance of this class is supplied as a parameter to verify certificate callbacks and is used to obtain the peer certificate and its issuer certificates.

Java

```
class IE.Iona.OrbixWeb.SSL.IT_X509CertChain {
public:
      public IT_X509CertChain();
      public void add(IT_X509Cert cert);
      public IT_X509Cert getCert(int pos);
      public IT_X509Cert getCurrentCert();
      public int getCurrentDepth();
      public IT_CertError getErrorInfo();
      public int numCerts();
      public String toString();
};
```

See Also `IE.Iona.OrbixWeb.SSL.IT_X509Cert`

IT_X509CertChain.IT_X509CertChain()

Synopsis `public IT_X509CertChain();`

Description This method constructs an empty certificate chain.

IT_X509CertChain.add()

Synopsis `public void add (IT_X509Cert cert);`

Description This method adds the supplied certificate to the end of the list.

Parameters

`cert`	The supplied certificate.

IT_X509CertChain.getCert()

Synopsis `public IT_X509Cert getCert(int pos);`

Description This method obtains the certificate at the specified index in the chain.

Parameters

`pos`	The index position in the chain of the required certificate.

Return Value Returns the certificate at index `pos`, if it is a valid index. Returns `null` otherwise.

IT_X509CertChain.getCurrentCert()

Synopsis `public IT_X509Cert getCurrentCert();`

Description This method returns the certificate that is marked as current in the chain. This certificate is always the one at the current depth. Functionally, this is equivalent to `getCert(getCurrentDepth())`.

IT_X509CertChain.getCurrentDepth()

Synopsis `public int getCurrentDepth();`

Description This method obtains the current depth of the certificate chain.

Return Value Returns the current depth of the certificate chain.

IT_X509CertChain.getErrorInfo()

Synopsis `public IT_CertError getErrorInfo();`

Description This method returns information on the last error associated with the certificate chain.

IT_X509CertChain.numCerts()

Synopsis `public int numcerts();`

Description This method obtains the number of certificates in this chain.

Return Value Returns the number of certificates in this chain.

IT_X509CertChain.toString()

Synopsis `public String toString();`

Description This method provides a detailed string representation of the certificate chain’s content. It overrides `toString()` in class `Object`.

Part V

Appendices

Appendix A
Security Recommendations

Some general recommendations for increasing the security of OrbixSSL applications are as follows:

- Use SSL security for every application where possible. This means specifying `SECURE_DAEMON` as your daemon policy, and using the default invocation policy for all OrbixSSL applications. Under these conditions, no unauthorized applications can access your servers or be accessed by your applications.
- Replace the demonstration certificates that are installed with OrbixSSL. These must be replaced by a set of certificates and private keys that have been securely generated. Refer to Chapter 3 on page 39 for more information.

  You should also change the pass phrases used to protect private keys. Do not reuse the pass phrases that were used for the example private keys.
- Do not enable the default certificate, and do not issue a default certificate for live systems.

  The use of a default certificate is generally not appropriate in a production system because access to the dynamic library of the OrbixSSL version installed on the system would allow any client to use the default certificate, even a client from another machine. The OrbixSSL dynamic libraries in effect contain the default pass phrase that protects the private key of the default certificate.
- If your application requires some interoperability with insecure applications, only allow specifically listed servers and interfaces to be contacted insecurely by your clients. Use secure callbacks for clients wherever possible as this is the default setting for OrbixSSL.
- Where it is necessary for remote insecure clients to contact OrbixSSL servers that are capable of accepting secure and insecure connections, set the daemon policy to `RESTRICTED_SEMI_SECURE_DAEMON` (instead of `SEMI_SECURE_DAEMON`).

- The OrbixSSL installation modifies the existing Orbix binaries so that they can use the Orbix binary certificate for authentication purposes. The permissions on these binaries are readable only by `root`, but executable by everybody. Do not change the permissions to be readable by everybody.
- Use the 128 bit or triple DES cipher suites exclusively where possible. The extra time taken to perform the more secure bulk cipher computations does not impact the overall performance of OrbixSSL applications significantly.

  The security of an SSL application is only as strong as the weakest cipher suite that it is prepared to support. Consider the presence of stronger cipher suites as an optional service for more discerning applications that wish to communicate with your application.
- An RSA key size of at least 1024 bits is recommended for most secure applications. 1024 bit keys are significantly slower to use than 512 bit keys but they greatly increase the security of systems. The use of SSL session caching helps to minimize the number of public key computations.

Appendix B
OpenSSL Utilities

OrbixSSL ships a version of the `openssl` program that is available from the OpenSSL project. OpenSSL is a publicly available implementation of the SSL protocol. Consult the `OpenSSL.cpr` file that is provided with OrbixSSL for information about the copyright terms of OpenSSL.

The `openssl` program consists of a large number of utilities that have been combined into one program. This appendix describes how you use the `openssl` program with OrbixSSL when managing X.509 certificates and private keys.

A number of examples using `openssl` commands are described in Chapter 3, “Managing Certificates”. Read Chapter 3 before consulting this appendix.

This appendix describes four `openssl` utility commands:

`x509` Manipulates X.509 certificates.

`req` Creates and manipulates certificate signing requests, and self-signed certificates.

`rsa` Manipulates RSA private keys.

`ca` Implements a Certification Authority (CA).

Using OpenSSL Utilities

An `openssl` utility command line takes the following form:

```
openssl command arguments
```

For example:

```
openssl x509 -in OrbixCA -text
```

Each command is individually described in this appendix. To get a list of the arguments associated with a particular command, use the `-help` option as follows:

```
openssl command -help
```

For example:

```
openssl x509 -help
```

The x509 Utility Command

In OrbixSSL the `x509` utility command is mainly used for:

- Printing text details of certificates you wish to examine.
- Converting certificates to different formats.

The options supported by the openssl `x509` utility command are as follows:

```
-inform arg         - input format - default PEM
                    (one of DER, NET or PEM)
-outform arg        - output format - default PEM
                    (one of DER, NET or PEM
-keyform arg        - private key format - default PEM
-CAform arg         - CA format - default PEM
-CAkeyform arg      - CA key format - default PEM
-in arg             - input file - default stdin
 -out arg            - output file - default stdout
-serial             - print serial number value
-hash               - print serial number value
-subject            - print subject DN
```

-issuer	- print issuer DN
-startdate	- notBefore field
-enddate	- notAfter field
-dates	- both Before and After dates
-modulus	- print the RSA key modulus
-fingerprint	- print the certificate fingerprint
-noout	- no certificate output
-days arg	- How long till expiry of a signed certificate - def 30 days
-signkey arg	- self sign cert with arg
-x509toreq	- output a certification request object
-req	- input is a certificate request, sign and output
-CA arg	- set the CA certificate, must be PEM format
-CAkey arg	- set the CA key, must be PEM format. If missing it is assumed to be in the CA file
-CAcreateserial	- create serial number file if it does not exist
-CAserial	- serial file
-text	- print the certificate in text form
-C	- print out C code forms
-md2/-md5/-sha1/ -mdc2	- digest to do an RSA sign with

Using the x509 Utility Command

To print the text details of an existing PEM-format X.509 certificate, use the x509 utility command as follows:

```
openssl x509 -in MyCert.pem -inform PEM -text
```

To print the text details of an existing DER-format X.509 certificate, use the x509 utility command as follows:

```
openssl x509 -in MyCert.der -inform DER -text
```

To change a certificate from PEM format to DER format, use the x509 utility command as follows:

```
openssl x509 -in MyCert.pem -inform PEM -outform
DER -out MyCert.der
```

The req Utility Command

The req utility command is used to generate a self-signed certificate or a certificate signing request (CSR). A CSR contains details of a certificate to be issued by a CA. When creating a CSR, the req command prompts you for the necessary information from which a certificate request file and an encrypted private key file are produced. The certificate request is then submitted to a CA for signing.

If the -nodes (no DES) parameter is not supplied to req, you are prompted for a pass phrase which will be used to protect the private key.

Note: It is important to specify a validity period (using the -days parameter). If the certificate expires, applications that are using that certificate will not be authenticated successfully.

The options supported by the openssl `req` utility command are as follows:

-inform arg	input format - one of DER TXT PEM
-outform	arg output format - one of DER TXT PEM
-in arg	inout file
-out arg	output file
-text	text form of request
-noout	do not output REQ
-verify	verify signature on REQ
-modulus	RSA modulus
-nodes	do not encrypt the output key
-key file	use the private key contained in file
-keyform arg	key file format
-keyout arg	file to send the key to
-newkey rsa:bits	generate a new RSA key of 'bits' in size
-newkey dsa:file	generate a new DSA key, parameters taken from CA in 'file'
-[digest]	Digest to sign with (md5, sha1, md2, mdc2)
-config file	request template file
-new	new request
-x509	output an x509 structure instead of a certificate req. (Used for creating self signed certificates)
-days	number of days an x509 generated by -x509 is valid for
-asn1-kludge	Output the 'request' in a format that is wrong but some CA's have been reported as requiring [It is now always turned on but can be turned off with -no-asn1-kludge]

Using the req Utility Command

To create a self signed certificate with an expiry date a year from now, the `req` utility command can be used as follows to create the certificate `CA_cert.pem` and the corresponding encrypted private key file `CA_pk.pem`:

```
openssl req -config ssl_conf_path_name -days 365
-out CA_cert.pem -new -x509 -keyout CA_pk.pem
```

This following command creates the certificate request `MyReq.pem` and the corresponding encrypted private key file `MyEncryptedKey.pem`:

```
openssl req -config ssl_conf_path_name -days 365
-out MyReq.pem -new -keyout MyEncryptedKey.pem
```

The rsa Utility Command

The `rsa` command is a useful utility for examining and modifying RSA private key files. Generally RSA keys are stored encrypted with a symmetric algorithm using a user-supplied pass phrase. The OpenSSL `req` command prompts the user for a pass phrase in order to encrypt the private key. By default, `req` uses the triple DES algorithm. The `rsa` command can be used to change the password that protects the private key and to convert the format of the private key. Any `rsa` command that involves reading an encrypted `rsa` private key will prompt for the PEM pass phrase used to encrypt it.

The options supported by the openssl `rsa` utility command are as follows:

`-inform arg`	`input format - one of DER NET PEM`
`-outform arg`	`output format - one of DER NET PEM`
`-in arg`	`inout file`
`-out arg`	`output file`
`-des`	`encrypt PEM output with cbc des`
`-des3`	`encrypt PEM output with ede cbc des using 168 bit key`
`-text`	`print the key in text`
`-noout`	`do not print key out`
`-modulus`	`print the RSA key modulus`

Using the rsa Utility Command

Converting a private key to PEM format from DER format involves using the `rsa` utility command as follows:

```
openssl rsa -inform DER -in MyKey.der -outform PEM
-out MyKey.pem
```

Changing the pass phrase which is used to encrypt the private key involves using the `rsa` utility command as follows:

```
openssl rsa -inform PEM -in MyKey.pem -outform PEM
-out MyKey.pem -des3
```

Removing encryption from the private key (which is not recommended) involves using the `rsa` command utility as follows:

```
openssl rsa -inform PEM -in MyKey.pem -outform PEM
-out MyKey2.pem
```

Note: Do not specify the same file for the `-in` and `-out` parameters, because this may corrupt the file.

The ca Utility Command

You can use the ca command to create X.509 certificates by signing existing signing requests. It is imperative that you check the details of a certificate request before signing. Your organization should have a policy with respect to the issuing of certificates. Before implementing CAs, refer to Chapter 3 for more information.

The ca command is used to sign certificate requests thereby creating a valid X.509 certificate which can be returned to the request submitter. It can also be used to generate Certificate Revocation Lists (CRLS). For information on the ca -policy and -name options, refer to "The OpenSSL configuration file" on page 194.

To create a new CA using the openssl ca utility command, two files (serial and index.txt) need to be created in the location specified by the OpenSSL configuration file that you are using.

The options supported by the OpenSSL ca utility command are as follows:

```
-verbose          - Talk alot while doing things
-config file      - A config file
-name arg         - The particular CA definition to use
-gencrl           - Generate a new CRL
-crldays days     - Days is when the next CRL is due
-crlhours hours   - Hours is when the next CRL is due
-days arg         - number of days to certify the certificate for
-md arg           - md to use, one of md2, md5, sha or sha1
-policy arg       - The CA 'policy' to support
-keyfile arg      - PEM private key file
-key arg          - key to decode the private key if it is
                  encrypted
-cert             - The CA certificate
-in file          - The input PEM encoded certificate request(s)
-out file         - Where to put the output file(s)
-outdir dir        - Where to put output certificates
```

```
-infiles....       - The last argument, requests to process
-spkac file        - File contains DN and signed public key and
                   challenge
-preserveDN        - Do not re-order the DN
-batch             - Do not ask questions
-msie_hack          - msie modifications to handle all thos
                    universal strings
```

Note: Most of the above parameters have default values as defined in `openssl.cnf`.

Using the ca Utility Command

Converting a private key to PEM format from DER format involves using the `ca` utility command as shown in the following example. To sign the supplied CSR `MyReq.pem` to be valid for 365 days and create a new X.509 certificate in PEM format, use the `ca` utility as follows:

```
openssl ca -config ssl_conf_path_name -days 365
-in MyReq.pem -out MyNewCert.pem
```

The OpenSSL configuration file

A number of OpenSSL commands (for example, req and ca) take a -config parameter that specifies the location of the OpenSSL configuration file. This section provides a brief description of the format of the configuration file and how it applies to the req and ca commands. An example configuration file is listed at the end of this section.

The openssl.cnf configuration file consists of a number of sections that specify a series of default values which are used by the OpenSSL commands.

[req] Variables

The req section contains the following settings:

```
default_bits = 1024
default_keyfile = privkey.pem
distinguished_name = req_distinguished_name
attributes = req_attributes
```

The default_bits setting is the default RSA key size that you wish to use. Other possible values are 512, 2048, 4096.

The default_keyfile value is default name for the private key file created by req.

The distinguished_name value specifies the section in the configuration file that defines the default values for components of the distinguished name field. The req_attributes variable specifies the section in the configuration file that defines defaults for certificate request attributes.

[ca] Variables

You can configure the file openssl.cnf to support a number of CAs that have different policies for signing CSRs. The -name parameter to the ca command specifies which CA section to use. For example:

```
openssl ca -name MyCa ...
```

This command refers to the CA section [MyCa]. If -name is not supplied to the ca command, the CA section used is the one indicated by the default_ca variable. In the “Example openssl.cnf File” on page 197, this is set to CA_default (which is the name of another section listing the defaults for a number of settings associated with the ca command). Multiple different CAs can be supported in the configuration file, but there can be only one default CA.

Possible [ca] variables include the following:

```
dir:            The location for the CA database

The database is a simple text database containing
the following tab separated fields

status:         A value of ‘R’ - revoked, ‘E’
                -expired or ‘V’ valid
issued date:    When the certificate was
                certified
revoked date:   When it was revoked, blank if not
                revoked
serial number:  The certificate serial number
certificate:    Where the certificate is located
CN:             The name of the certificate

The serial field should be unique as should the
CN/status combination. The ca program checks these
at startup.

certs:          This is where all the previously
                issued certificates are kept
```

[policy] Variables

The policy variable specifies the default policy section to be used if the `-policy` argument is not supplied to the `ca` command. The CA policy section of a configuration file identifies the requirements for the contents of a certificate request which must be met before it is signed by the CA.

There are 2 policies defined in the "Example openssl.cnf File" on page 197: `policy_match` and `policy_anything`.

Consider the following value:

```
countryName = match
```

This means that the country name must match the CA certificate.

Consider the following value:

```
organisationalUnitName  = optional
```

This means that the `organisationalUnitName` does not have to be present.

Consider the following value:

```
commonName = supplied
```

This means that the `commonName` must be supplied in the certificate request.

The `policy_match` section of the example `openssl.cnf` file specifies the order of the attributes in the generated certificate as follows:

```
countryName
stateOrProvinceName
organizationName
organizationalUnitName
commonName
emailAddress
```

Example openssl.cnf File

```
################################################################
# OpenSSL example configuration file.
# This is mostly used for generation of certificate requests.
#################################################################
[ ca ]
default_ca    = CA_default     # The default ca section
#################################################################

[ CA_default ]

dir      =/opt/progress/OrbixSSL1.0c/certs # Where everything is
kept

certs               = $dir # 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/new_certs # default place for new certs
certificate         = $dir/CA/OrbixCA # The CA certificate
serial              = $dir/serial # The current serial number
crl                 = $dir/crl.pem # The current CRL
private_key         = $dir/CA/OrbixCA.pk # The private key
RANDFILE            = $dir/.rand # private random number file
default_days        = 365 # how long to certify for
default_crl_days    = 30 # how long before next CRL
default_md          = md5 # which message digest to use
preserve            = no # keep passed DN ordering

# A few different ways of specifying how closely the request should
# conform to the details of the CA

policy                    = policy_match

# For the CA policy [policy_match]

countryName               = match
stateOrProvinceName       = match
organizationName          = match
organizationalUnitName    = optional
commonName                = supplied
emailAddress              = optional
```

```
# For the ‘anything’ policy
# At this point in time, you must list all acceptable ‘object’
# types

[ policy_anything ]
countryName                = optional
stateOrProvinceName        = optional
localityName               = optional
organizationName           = optional
organizationalUnitName     = optional
commonName                 = supplied
emailAddress               = optional

[ req ]
default_bits               = 1024
default_keyfile            = privkey.pem
distinguished_name         = req_distinguished_name
attributes                 = req_attributes

[ req_distinguished_name ]
countryName                = Country Name (2 letter code)
countryName_min            = 2
countryName_max            = 2
stateOrProvinceName        = State or Province Name (full name)
localityName               = Locality Name (eg, city)
organizationName           = Organization Name (eg, company)
organizationalUnitName     = Organizational Unit Name (eg, section)
commonName                 = Common Name (eg. YOUR name)
commonName_max             = 64
emailAddress               = Email Address
emailAddress_max           = 40

[ req_attributes ]
challengePassword          = A challenge password
challengePassword_min      = 4
challengePassword_max      = 20
unstructuredName           = An optional company name
```

Appendix C
Troubleshooting OrbixSSL

This is a checklist to help you make sure that OrbixSSL is installed and configured correctly:

- Ensure that your application works without OrbixSSL, by disabling all OrbixSSL calls in the application. If the application does not work, OrbixSSL is not causing the problem.
- Check whether your application works using the Default Cert mechanism provided by OrbixSSL. Disable all OrbixSSL calls in the application and specify `IT_ENABLE_DEFAULT_CERT TRUE` in the `itssl.cfg` OrbixSSL policy file. If the application now works, any problem is likely to be caused by either OrbixSSL code in the application, or by the certificate or private key that your application is using.

The rest of the suggestions in this appendix assume that your OrbixSSL code is not disabled.

- Insure that `IT_SSL::init()` is called and the return value checked. Also ensure that the return value of all OrbixSSL functions is carefully examined.
- Set `export IT_SSL_TRACE_LEVEL=1`

  This will give some high level handshake information.
- Set `IT_SSL_TRACEFILE` to point to a debug file for a process. The process can now write additional very detailed SSL debug information to this file. Set `IT_SSL_TRACEFILE` to a different file for each process, so that the output of two processes are not confused.

- Check that the certificates, private keys and passwords are correct. For example:

  ```
  openssl x509 -in MyCert -text
  ```

  This should display the text details of the certificate.

  ```
  openssl rsa -in MyKey -text
  ```

  This should display the text details of the private key, if the private key is encrypted (which it normally should be). You are asked for a pass-phrase – input the pass-phrase that the OrbixSSL application is attempting to use to decrypt the private key.

- Investigate whether the `openssl s_client` or `openssl s_server` utilities provided with OrbixSSL can communicate using the same certificates and keys that they are trying to use with the OrbixSSL applications. If this is not the case then there is a problem with the keys, certificates, or pass-phrases. The customer should recheck them. For example:

  ```
  openssl s_client -ssl3 -host SomeHost
  -port SomeServerPort -CAfile SomeCAFile
  -cert SomeClientCert -debug

  openssl s_server -accept MyServerPort -ssl3 -
  CAfile SomeCAFile -cert SomeClientCert -debug -
  Verify 2
  ```

  The argument `-Verify` enforces client authentication. It is followed by an integer that determines the maximum chain depth allowed. You can also use `-verify` can be instead of `-Verify` which will not reject the connection if a client cert is not available.

  If `openssl_server` is interrupted the port number it was using can become unavailable for a period of time. Simply use another port when trying again. The `openssl s_client` port parameter must change to match.

  There is no support for SSL Version 2.0 in OrbixSSL. It supports SSL Version 3.0 only. It does not issue or accept Version 2.0 `hello` messages. This behavior can be simulated in `openssl s_client` and `openssl s_server` by the use of the `-ssl3` parameter shown above.

  You can also use `openssl s_client` and `openssl s_server` can be used to establish SSL connections with OrbixSSL servers. For example, you can specify the OrbixSSL server port to `openssl s_client`, and it then attempts to handshake with the OrbixSSL server.

You can also use `s_server` to simulate an OrbixSSL server by running it on the SSL port specified in the IOR that an OrbixSSL client uses. Use `IORDump` see the port.

- If you are an experienced progammer, examine the output of operating system diagnostic tools such as `truss` (Solaris) or `trace` (HP-UX) for the client, server and daemon separately.

Summary of Useful Output to Gather

If you have problems with OrbixSSL and must make a support call, he following can be very helpful:

- Separate files for the Daemon, client and server of the following output having specified `IT_SSL_TRACE_LEVEL=1`:

  The `stdout` and `stderr` (for example, `&` on Unix)

  ```
  daemon.out
  client.out
  server.out
  ```

- Separate `IT_SSL_TRACE_FILE` output for the daemon, client and server:

  ```
  daemon.log
  client.log
  server.log
  ```

- Separate `truss` (or `trace`) output for the daemon, client and server. For Multi-threaded applications use `trace -l` on Solaris to show the system calls per thread.

  ```
  daemon.trc
  client.trc
  server.trc
  ```

- The OrbixSSL Security config file `itssl.cfg`
- The root CA file that is referenced by `itssl.cfg`
- If appropriate the certificates and private key files with passwords can be useful, in order to attempt to reproduce the problem exactly.

Note: Do not send us the password and private keys for a Live system!

- If possible the complete source for a minimal test case.
- If this is not possible then include the excerpts of the client and server programs which make OrbixSSL calls.
- A core dump, and a text stack trace, if the problem causes the program to dump core.

Index