

Secure U2 Web services

Protect your data when exposed through U2 Web services

Skill Level: Intermediate

[Reynal Cocaign \(rcocaign@us.ibm.com\)](mailto:rcocaign@us.ibm.com)

Advisory Software Engineer
IBM

[Nik Kesic \(nikk@us.ibm.com\)](mailto:nikk@us.ibm.com)

Advisory Software Engineer
IBM

11 Sep 2008

With the increasing acceptance and usage of SOA driving business information in the global economy, it has become critical to provide protection, confidentiality, and integrity of sensitive information. The U2 Web Services Developer allows you to publish business functions as Web services and make them available to outside protected network hierarchies. This article takes you on a journey in the world of U2 information security for Information on Demand.

Introduction

The U2 Web Services Development Tool for IBM® UniData® and UniVerse® (U2WSD) allows developers to create Web services quickly from existing subroutines and queries. U2 users embracing Service-Oriented Architecture (SOA) with U2-based applications can provide information as a service using U2WSD. This allows U2-based applications to be consumed by different application environments using a worldwide acceptable open standard. As Web services get increasingly used within business organization, sensitive data can be exposed. It is important to make sure the data is protected. There are many considerations when dealing with Web services security. This article reviews the architecture of U2 Web services, identifies the areas where security can be defined, and how to activate it.

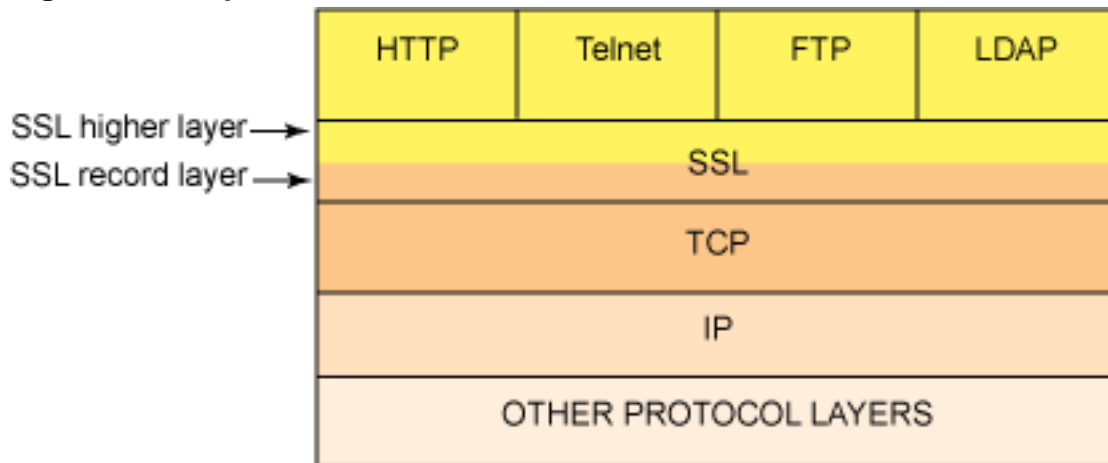
Web services and security

The need for security in Web services differs in each application that relies on the technology. The topology of each application may require different consideration when setting security. In some cases, Secure Socket Layer (SSL) may or may not be enough to address security needs. Outside influences by data governance agencies may define your security journey and level of implementation.

SSL is a transport layer protocol that provides a secure channel between two communicating programs over which application data can be sent securely. It is by far the most widely deployed security protocol used on the World Wide Web. Published Web services are accessed through the Web using the HTTP protocol. When securing using SSL, these Web services are accessed using HTTPS.

The SSL layers encapsulate various application level protocols such as HTTP and Telnet through the Higher Layer and the Transport Layer (TCP/IP) through the Record Layer.

Figure 1. SSL protocol stack



Understanding the limits of SSL within a Web services architecture

SSL is a technology that allows client-server to communicate over a secure connection. SSL addresses the following security considerations:

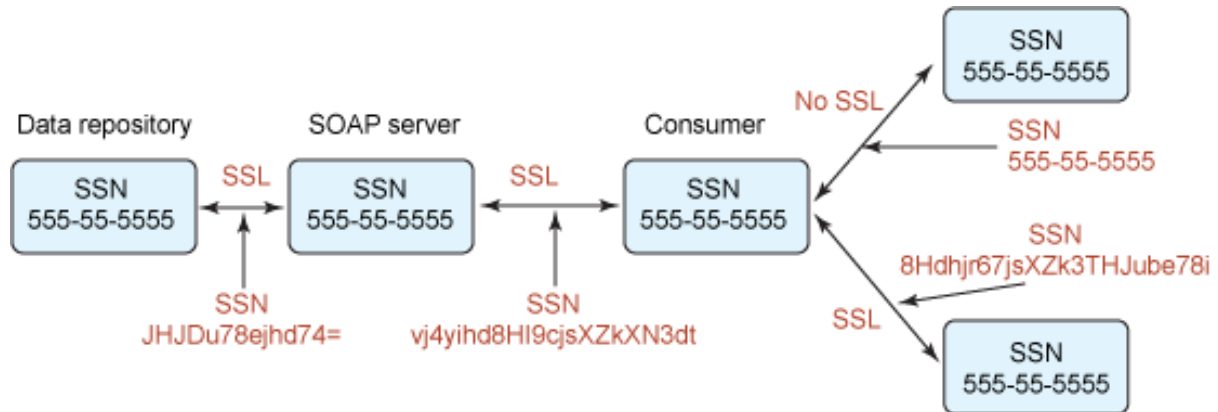
- **Authentication** - The server presents the client with a set of credentials in the form of a certificate which is used to verify the identity of the server.
- **Confidentiality** - SSL responses are encrypted so that data cannot be deciphered by third parties as it transfers between the client and the server on a network.
- **Integrity** - Guarantees the data is not being modified as it is being passed

between the client and the server on a network.

SSL encrypts packets between peer-to-peer TCP/IP connections. Once the encrypted data reaches either end point in a client/server environment, the data becomes decrypted and readily visible. You can have situations where Web services can be created to call other Web Services. In this situation, you may have other connections using the data received. Do not encrypt the data while your connection from the SOAP server to the client is secure.

Figure 2 shows the flow of data through Web services when using SSL. The important point to note is that the data is clear text once it reaches the end point of the SSL connection. The social security number (SSN) is in clear text at the SOAP server level as well as once it reaches the initial consumer of the Web service, and the reverse is true.

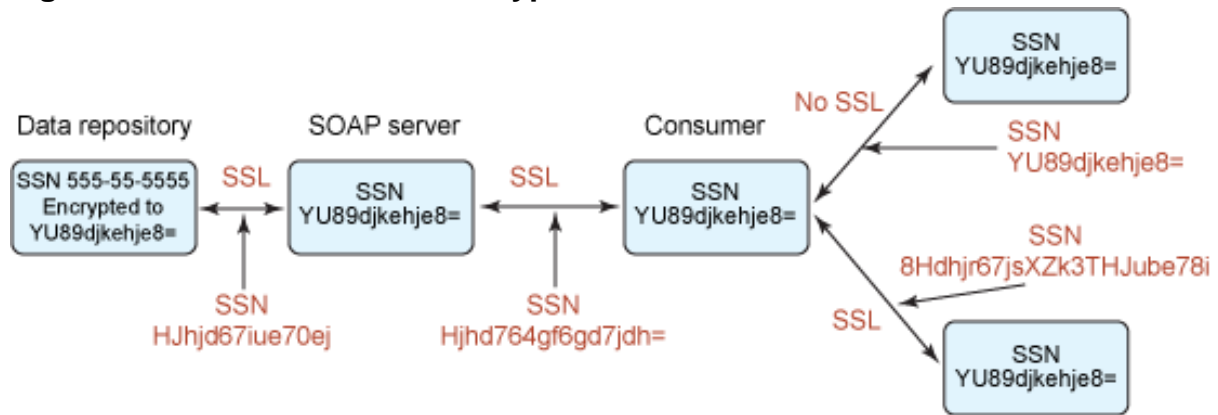
Figure 2. SSL data flow



Besides having a secure connection between your server and your client (Web service consumer), it may be necessary to encrypt the data elements within the XML message (social security number, credit card number, and so on).

Figure 3 highlights the flow of data through Web services when using SSL combined with encrypting the data in motion (end-to-end solution, rather than point-to-point).

Figure 3. SSL data flow with encryption in motion



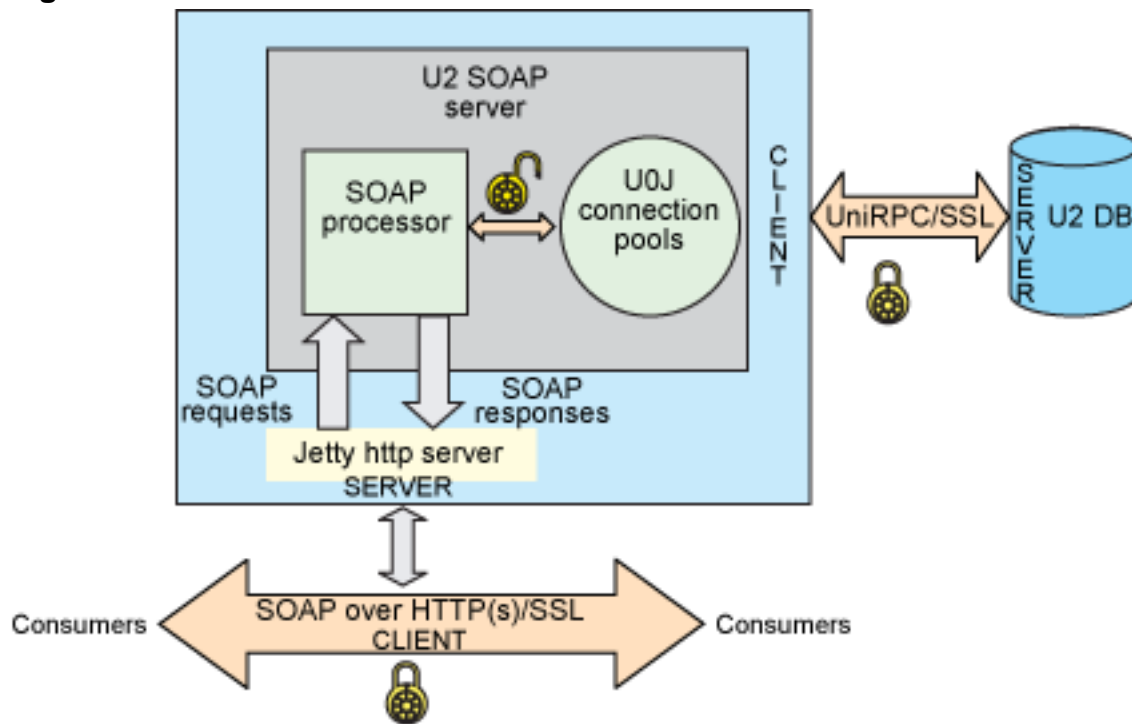
Once the data is received on the consumer side, the information is still encrypted and can only be accessed by a method together with the credentials to unlock and decrypt the information. Should the encrypted data fall in the wrong hands, the data would still be protected.

U2 Web service architecture

The U2 Web Services Developer (U2WSD) is used to configure U2 SOAP servers and develop U2 Web services. The U2 SOAP server sits between the Web service consumers (client) and the U2 database server.

Figure 4 shows the architecture of the U2 Web services:

Figure 4. U2 Web services architecture



The U2 SOAP server connects to the U2 database server using UniObjects for Java (UOJ). The socket connection to the database server is established through the unirpc daemon. The server component of U2 SOAP server is built using Jetty. Jetty is an open source Web server implemented entirely in Java technology. Within the U2 SOAP server, Jetty is only able to serve SOAP requests.

SSL and U2 Web services

SSL can be enabled on both sides of a U2 SOAP server:

- On the consumer to U2 SOAP server side, the U2 SOAP server acts as an SSL server.
- On the U2 SOAP server to U2 database side, the U2 SOAP server acts as an SSL client.

It is important to have a clear understanding of which side acts as an SSL server and which side acts as an SSL client. The SSL setup is different depending on the role performed by each side. [Figure 4](#) shows the U2 SOAP server architecture, highlighting which sides act as clients and servers.

For SSL to function correctly, the SSL server must have its own digital certificate (often called Server Certificate) installed. The SSL client must have the root certificate of the server it connects to installed in its certificate store (trust store). The root certificate is a certificate that can verify the issuer of the SSL server certificate.

SSL is a peer-to-peer protocol, which means that both ends must be configured to enable SSL communication. The next section of this article reviews how to configure each side of the SSL connection in order to have a secure U2 SOAP server.

Configuring the U2 database server

As you have seen in the architecture of the U2 SOAP server, the connection to the database server is established using UOJ. This means that the U2 database server must be set up to secure incoming connections from UOJ. The first step in configuring the database is to create a server Security Context Record (SCR). A SCR contains all the SSL-related properties necessary for the server to establish a secured connection with an SSL client. The properties include the server's private key, the server certificate, the client authentication flag, authentication strength, and the defined trusted entities.

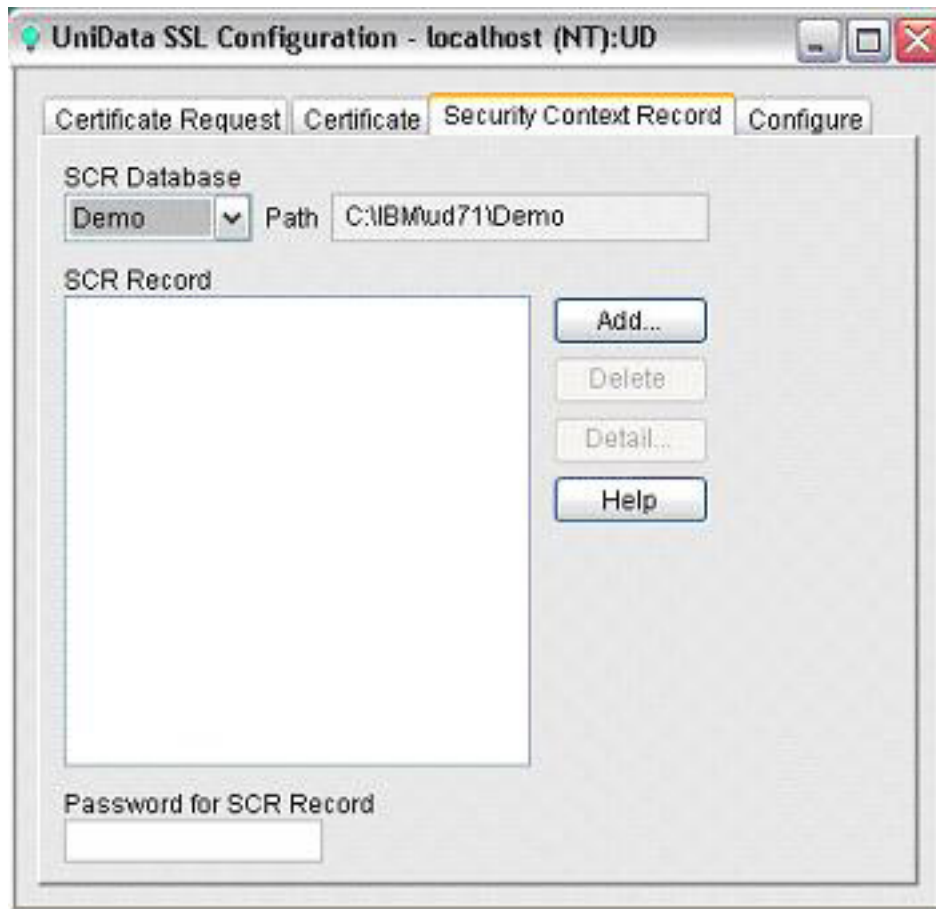
There are two ways to create an SCR record:

- UniAdmin
- U2 Security API from a BASIC program

Let's take a look at how to create an SCR using UniAdmin. This assumes that you already have a valid certificate for your U2 database server.

Once you are connected to your U2 database server using UniAdmin, select the SSL Configure option. The U2 SSL configuration window appears. Select the Security Context Record tab in order to create an SCR for UOJ.

Figure 5. Create Security Context Record using UniAdmin



To create an SCR:

1. Select the account where you want to create the SCR from the SCR database list.
2. Click the **Add** button to add an SCR.
3. Enter an ID for the SCR in the Security Context Record ID box.
4. Select the appropriate version for the SCR record in the Version box.
5. Define the SCR for the server.
6. Specify the server authentication depth and rules.
7. Specify the certificate path rule to obtain the certificate for the server.
8. Specify the Private Key file and Private Key password.
9. Specify the password for the SCR.

The document "Securing U2 Database Server" (available for [download](#)) goes through these steps. It provides a walk-through with screen shots of the procedure.

Once the SCR is created, you need to configure SSL for UOJ. You can use the UniAdmin "Configure" option from the SSL configure to assign the newly created SCR to the service used by UOJ (either udcs or uvcs).

Configuring the U2 UOJ client

As you have seen, the U2 SOAP server is a Java application that includes both Jetty to handle the Web services requests and UOJ that connects to the database server to handle the request.

When connecting to the database, the U2 SOAP server acts as a client. Since the U2 SOAP server is a Java program, the standard Java *keystore* is used to store certificates. The Java keystore is manipulated by the standard Java Runtime Environment (JRE) *keytool* utility.

When an SSL client is making connection to the U2 SOAP server, it will send its server certificate to the client for verification. The client must have access to the root certificate of the server certificate to be able to accomplish the task.

You should use the keytool utility to add the root CA certificate of the U2 database server you are connecting to in the standard cacerts keystore of the JRE used by the SOAP server. The cacerts keystore is located in the installation directory of the JRE under the lib/security.

To find out your JRE home directory on Windows, select **Start > Programs > IBMU2 > Web Tools > U2 Web Services Developer**, then right-click and choose **Properties**. In the Target field you can find the JRE home path (for example, "C:\IBM\UniDK\JRE2\jre").

Note

"changeit" is the default password for the default "cacerts" keystore. Your administrator may have changed it. You must have the correct password to import the certificate into this file.

You can install the root CA certificate in the cacerts keystore using the following steps:

1. Copy the root certificate on the machine where the U2 SOAP server will run.
2. Change directory to the JRE home directory.
3. Change directory to the lib\security directory.

4. Import the root certificate using the following command:

Listing 1. Importing a root certificate to the cacert keystore

```
keytool -import -file path_to_cert\U2DBroot.cer -alias U2uojkey  
-keypass "my password"  
-keystore cacerts -storepass "changeit"
```

Once the root certificate is installed on the UOJ client, you should be able to perform an SSL connection to the U2 database,

The document "Securing U2 Database Server" (available for [download](#)) goes through these steps. It provides a walk-through with screen shots of the procedure.

Configuring the U2 SOAP server

After configuring the UOJ SSL client, you need to configure the U2 SOAP server. In this case, the U2 SOAP server is acting as an SSL server.

For this task, you need to create a Java keystore to store the server certificate. Once again, the U2 SOAP server is a Java application, and the management of the SSL configuration (certificates) is handled by the utility provided with the JRE.

First, you need to create a Java keystore. The keystore is created using the keytool command. For example:

Listing 2. Creating a Java keystore

```
keytool -genkey -keyalg RSA -keysize 1024 -alias u2sskey -keypass "my password"  
-keystore u2sskeystore -storepass passphrase
```

You have to decide which algorithm and key size to use. The ones used in Listing 2, above, are shown as an example. The value for `passphrase` should be your own password for the keystore.

You then have to load the certificates (server certificate and root certificate) in the keystore you just have created.

1. At DOS prompt, change directory to the location where you have created the keystore in the step above.
2. Import the root certificate:

Listing 3. Importing the SOAP server root certificate

```
keytool -import -file u2ssroot.cer -alias "U2ssrootkey" -keypass "my password"
-keystore u2sskeystore -storepass passphrase
```

3. Import the server certificate:

Listing 4. Importing the SOAP server certificate

```
keytool -import -file u2SoapServer.cer -alias "u2sskey" -keypass "my password"
-keystore u2sskeystore -storepass passphrase
```

The document "Securing U2 SOAP server" (available for [download](#)) goes through these steps. It provides a walk-through with screen shots of the procedure.

Configuring the U2 Web Service Explorer

As part of the U2 Web Services Developer, you have a tool called the U2 Web Service Explorer that allows you to test the services you have developed.

When using this testing tool, the U2 Web Services Developer becomes an SSL client to the U2 SOAP server.

As you have seen before when an SSL client is making a connection to the U2 SOAP server, the U2 SOAP server sends its server certificate to the client for verification. The client must have access to the root certificate to be able to accomplish the task.

For this to work, the U2 Web Services Developer needs to be able to access the root certificate. In the configuration, the root certificate for the U2 SOAP server must be installed in the cacerts keystore of the JRE used to start the U2 Web Services Developer.

In the same manner that you had to install the U2 database server root certificate in the cacerts keystore, you install the root certificate of the U2 SOAP server in the cacerts keystore.

1. At the DOS prompt, cd C:\IBM\UniDK\JRE2\jre\lib\security (or your real JRE home directory), import the root certificate:

Listing 5. Importing the U2 SOAP server root certificate to the cacerts keystore

```
keytool -import -file pathtocert\u2ssroot.cer -alias U2ssrootkey
-keypass "my password"
-keystore cacerts -storepass "changeit"
```

The document "Securing U2 SOAP server" (available for [download](#)) goes through these steps. It provides a walk-through with screen shots of the procedure.

Configuring the Windows Certificate Store

If you use any Windows product (IE, .NET, and so on) to consume U2 Web services, you need to install the root certificate of the U2 SOAP server in the Windows Certificate Store. If you obtained your server certificate from a major third party CA, most likely the proper root certificate is already installed in the Windows store. If that is not the case, you will have to configure the Windows Certificate Store as follows:

1. Open an Internet Explorer window.
2. From the tool bar, select **Tools > Internet Options > Content > Certificates...**
3. Click on **Import**, then select **Next**.
4. For file name, specify the path for your root certificate (for example, path_to_cert\u2wsdroot.cer), then select **Next**.
5. Check "**Place all certificates in the following store**", then click on **Browse**.
6. Select "**Trusted Root Certificate Authorities**", **OK**, and **Next**.
7. The wizard should ask you to confirm the content of the certificate. Click **Finish**.
8. The wizard should display a "success" message.
9. Find and select the "**Trusted Root Certificate Authorities**" tab, and verify that the root certificate is indeed installed.
10. Select **OK**, and close the IE windows.

Once this is set up, you should be able to connect to the U2 SOAP server with Windows products using SSL.

The document "Securing U2 SOAP server" (available for [download](#)) goes through these steps. It provides a walk-through with screen shots of the procedure.

Summary of the configuration

As you have seen, there are many steps to configure SSL within the U2 SOAP Server environment in order to secure the U2 Web services. Multiple certificates are in play. Multiple Java keystore have to be specified. **Figure 6** shows the entire configuration of the security components with U2 Web services.

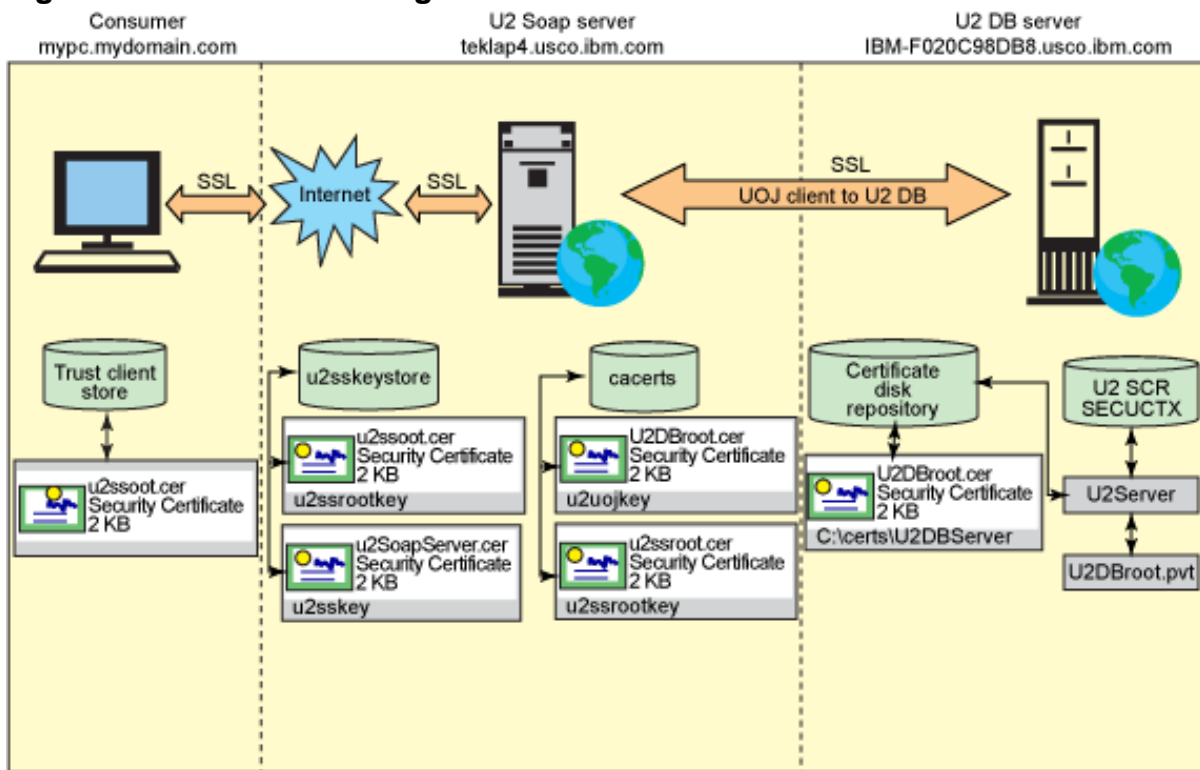
The U2 database server has a Security Context Record (SCR) setup that references the server certificate. The SCR is used to configure the udcs/uvcs rpc service used to connect with UniObjects for Java (UOJ).

UOJ references the cacerts Java keystore. It contains the root certificate of the U2 database server.

The U2 SOAP server has a keystore that contains its server certificate as well as its root certificate.

The consumers of the SOAP server have the root certificate of the U2 SOAP server in their trust store.

Figure 6. Overall SSL configuration for U2 Web Services



Once you have completed the setup of each of the SSL components, you will be able to define the SSL connection properties for your U2 SOAP server. The document "U2 SOAP Server Connection Setup" (available for [download](#)) goes over the setup. It uses the information that has been presented through the configuration of each of the components (U2 Database Server, UOJ, U2 SOAP server, Consumer).

Going beyond SSL

As you saw earlier on, SSL has limits in a Web services environment, as it protects from point to point rather than from end to end. What techniques can you use to further secure your sensitive data? There are two ways to interact with the U2 database when using the U2 Web services:

- Subroutines
- Queries

Subroutines pass input and output parameters through the SOAP requests. In order to protect the data that is returned by the SOAP server, the sensitive one should be encrypted within the BASIC subroutine using the ENCRYPT function.

Listing 6 demonstrates such a technique. You should decide your own encryption algorithm and keys.

Listing 6. Encrypting parameters of subroutines

```

SUBROUTINE GetCustomerSSN (CUSTID, SSN)
* parameters :
* in : CUSTID - Customer ID
* out: SSN - Social Security Number
*Skipping reading customer file. Assigning dummy value to SSNClear variable

SSNClear= "555-55-5555"
ALG="rc2-cbc" ; * 128 bit rc2 algorithm in CBC mode
MYKEY="4142434445464748494A4B4C4D4E4F50"; * HEX - Actual Key
IV="4142434445464748" ; * HEX - Initialization Vector
DATALOC=1 ; * Data in String
KEYLOC=1 ; * Key in String
ACTION=2 ; * Base64 encode after encryption
KEYACTION=3 ; * KEY_ACTUAL_OPENSSL
SALT="" ; * SALT not used
RESLOC=1 ; * Result in String

* encrypt the social security number before sending it back to the Web Service
RET.CODE=ENCRYPT(ALG,ACTION,SSNClear,DATALOC,MYKEY,KEYLOC,KEYACTION,SALT,IV,
SSN,RESLOC)

IF RET.CODE <> 0 THEN
SSN="Error in Encrypt"
END
END

```

For queries, you can specify I-type dictionaries that allow to return encrypted data, rather than clear text for sensitive data. In this case, the query references the I-type dictionaries that perform the encryption. An example of such an I-type is as follows:

Listing 7. Dictionary definition for I-type

```

LIST DICT TEST BY TYP BY @ID TYP LOC CONV NAME FORMAT SM ASSOC 17:07:59 Aug 26 2
008 1
Key..... TYP LOC..... CONV NAME..... FORMAT SM ASSOC.....

@ID          D          0          TEST          10L    S
SSN          D          1          social security
              number
SECURESSN    I    SUBR("SECURES
              SN",SSN)    SECURESSN    10L    S

3 records listed

```

The subroutine called by the I-type SECURESSN uses the SSN dictionary but returns an encrypted value:

Listing 8. Subroutine definition for the I-type

```

SUBROUTINE SECURITYSSN(ANSWER,IN.SSN)
ALG="rc2-cbc" ; * 128 bit rc2 algorithm in CBC
mode
MYKEY="4142434445464748494A4B4C4D4E4F50" ; * HEX - Actual Key
IV="4142434445464748" ; * HEX - Initialization Vector
INSTR="My data string" ; * Data String
DATALOC=1 ; * Data in String
KEYLOC=1 ; * Key in String
ACTION=2 ; * Base64 encode after encryption
KEYACTION=3 ; * KEY_ACTUAL_OPENSSL
SALT="" ; * SALT not used
RESLOC=1 ; * Result in String RESULT
RET.CODE=ENCRYPT(ALG,ACTION,IN.SSN,DATALOC,MYKEY,KEYLOC,KEYACTION,SALT,IV,
ANSWER,RESLOC)
IF RETURN.CODE <> 0 THEN
SSN="Error in Encrypt"
END
RETURN
END

```

Since the encryption performs in the basic subroutine in both cases references a standard encryption, you can use cryptographic facilities from outside U2 to perform the decryption. As long as the application that decrypts these encrypted fields knows about cipher to use, the key, IV, and salt, it will be able to decrypt the information passed back to the application.

If the application is not aware of the security credentials of the information, then the encrypted data will be kept in its encrypted form.

Examples of these concepts are presented in the article "[U2 encryption in an open technology world](#)" (developerWorks, June 2004).

Summary

This article has taken you on a journey securing the three tiers that make up a typical U2 Web services deployment. You should have also seen that security extends beyond SSL with the probable need to encrypt data elements before the

data is in motion. This article also provided examples to further demonstrate the practical working aspects of securing peer-to-peer communication end points.

Downloads

Description	Name	Size	Download method
Securing the U2 SOAP Server	secureU2SoapServerConsumerU2ss-document1.pdf	2,778KB	HTTP
Securing U2 Database Server	secureU2SoapServerU2ss-U2DB-document2.pdf	961KB	HTTP
U2 SOAP Server Connection Setup	U2SoapServerSetupDocument3.pdf	2,382KB	HTTP
SSL Configuration for U2 Web Services	3tierss-v2.pdf	58KB	HTTP

[Information about download methods](#)

Resources

Learn

- "[U2 encryption in an open technology world](#)" (developerWorks, April 2006): Walk through the four worlds of encryption covering C#, Java, U2 BASIC, and OpenSSL shell scripting and see that encryption of data produced is interchangeable amongst them.
- [Java KeyTool User Guide](#): Find documentation, example code, and ancillary files for keytool.
- "[U2 Web Services Developer](#)" (IBM, August 2008): Learn more about U2 Web Services Developer (PDF).
- "[UniData: Using UniAdmin](#)" (IBM, August 2008): Gain a better understanding of UniAdmin and the tasks which the tool enables you to perform (PDF).
- "[UniData: Security Features](#)" (IBM, August 2008): Learn how to use implement SSL security (PDF).
- "[UniData: UniBasic Extensions](#)" (IBM, August 2008): Gain a better understanding of the UniBasic extensions (PDF).
- [developerWorks Information Management zone](#): Learn more about Information Management. Find technical documentation, how-to articles, education, downloads, product information, and more.
- Stay current with [developerWorks technical events and webcasts](#).
- [Technology bookstore](#): Browse for books on these and other technical topics.

Get products and technologies

- Build your next development project with [IBM trial software](#), available for download directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Participate in [developerWorks blogs](#) and get involved in the developerWorks community.

About the authors

Reynal Cocaign

Reynal Cocaign works for the IBM U2 Technical Support group located in Denver, Colorado. Reynal brings 13 years of experience in technical support. He specializes

in Web technologies and U2 Basic extensions.

Nik Kesic

Nik Kesic works for the IBM U2 Client Support group in Denver, Colorado. Nik's career has spanned consultancy, high-level support, enterprise architecture, and training. He has generated collateral and training material, and has published articles on Web enablement using IBM IDE Tools, sockets, XML, SOAP, SSL, and encryption.