FXintegrate On-Demand Developer’s Guide

Version 1.3 June 11, 2002

Currenex, Incorporated
3565 Haven Avenue
Menlo Park, California 94025

1 Introduction

The FXintegrate On-Demand service enables Straight Through Processing (STP) for securities trading by providing a seamless and complete post-trade integration service. By extending the functionality of Currenex's FXtrades service with an automated trade and settlement capturing service, members save time and reduce errors, significantly increasing FX trading and settlement efficiency.

Using the FXintegrate On-Demand service, Currenex members can create automated clients that poll and retrieve trade and settlement information through a flexible and secure interface. Members have the ability to retrieve new trade information to update back-office systems, retrieve previously downloaded information for reconciliation, and even automatically convert Currenex file formats to member formats by providing a standard XSLT stylesheet.

1.1 Document Scope

This developer's guide is intended for readers who will implement HTTPS client software that will connect and download trade and settlement information from the FXintegrate On-Demand service. It is an in depth manual that explains all the functions and components of the FXintegrate On-Demand service and presents examples on how to interface with the service to achieve the desired functionality.

For this reason, it is assumed that the reader is familiar with the concept of application programming, data communication, data security, digital certificates, XML and the HTTPS protocol.

2 System Architecture

2.1 Overview

The main interface to the FXintegrate On-Demand service is a Java servlet that receives connections forwarded from a mutually authenticating SSL enabled web server. Members using any HTTPS V3 compliant client can establish connection with the web server and make requests to the File Delivery servlet.

After establishing a connection and authenticating to the service (details explained in following sections), the client can then format XML based requests to retrieve trade or settlement details information.

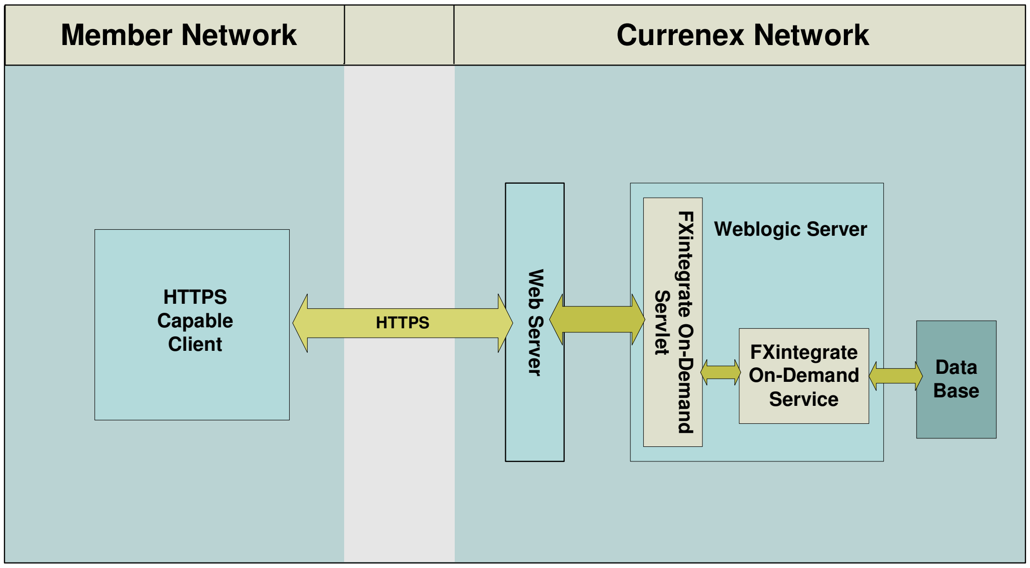

Figure 1. FXintegrate On-Demand service overview

2.2 Delivery Mechanism

The method of delivering trade and settlement information from FXtrades to member clients is HTTPS (HTTP over SSL). Currenex maintains a mutually authenticated SSL web server that listens on port 443. A member client can connect to this service over the HTTPS protocol with a valid private key and digital certificate (certificate request procedures explained in following sections) to request trade or settlement information via an HTTP POST request.

2.3 Request/Response Process Flow

1. Member connects to the FXintegrate On-Demand service via a mutually authenticated SSL session.
2. During session establishment, the service will authenticate the member based on the digital certificate presented. The session will be terminated and appropriate error response returned for any abnormal conditions encountered during session establishment.
3. The connecting member, through an HTTP POST action, sends a XML request to the service. If a transformation is requested, a XSLT stylesheet is also sent as part of the HTTP POST as a separate parameter (parameter names are provided in the following section).
4. The service receives and processes the request. If a formatting error is detected in the request, an appropriate error response is returned. Otherwise the request is processed and the trade or settlement information is returned to the requesting member. If a transformation is requested, the service will convert the response from the Currenex's XML format to the format specified by the XSLT stylesheet.
5. The service terminates the connection.

2.4 Request/Response Parameters

As previously stated, the communication protocol between FXtrades and the member's requesting application is HTTP over SSL. Requests and responses passed between the systems are XML based.

The FXintegrate On-Demand service expects one required name-value pair as parameter for the HTTP request and an optional name-value pair if a transformation is expected. The expected name for the required request parameter is "downloadRequestXML". The following is an example HTTP request to the FXintegrate On-Demand service.

```
POST /fxintegrate/download HTTP/1.1
Accept: text/html, text/xml, image/gif, image/jpeg
Accpet-Language: en-us
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/4.5
Host: member.company.com

downloadRequestXML=<?xml...><DownloadRequest>...</DownloadRequest
>
```

NOTE: The above request is a sample only. It does not show the correct URI and does not show a complete request body. It is also one of two posting formats, details of the two formats are explained in the next section.

To ensure the server parses the value of the "downloadRequestXML" parameter properly, the request value must be URL encoded before it is sent to Currenex.

If the requester specifies the optional XSLT Transform element (refer to next section for details), an extra parameter must be specified in the request to include the actual XSLT stylesheet. The expected name for the XSLT transform stylesheet parameter is "xsltStylesheet".

The response will be an XML file with the header content-type set to "text/xml" if the XML format is requested. If the CSV format is requested, the response will be a file containing the Currenex standard comma separated value file format and the content-type will be set to "text/html". The details of how to request each response format are explained in the next section. The following is an example response from the FXintegrate On-Demand service for a request with download format being XML.

```
HTTP/1.1 200 OK
Connection: close
Content-Length: ###
Content-Type: text/xml
Date: Fri, 12 Jan 2001 11:07:42 GMT
Server: Netscape/iPlanet 4.0.x


<?xml version="1.0"?>
<CurrenexDownload>
  <Trades>
    <party>
       ...
    </party>
    <trade>
       ...
    </trade>
  </Trades>
</CurrenexDownload>
```

NOTE: The above response is a sample only. It does not contain a complete response and some header information may be omitted.

2.4.1 Request Content-Types

The FXintegrate On-Demand service can receive HTTP POST request in two different formats.

One is the standard "application/x-www-form-urlencoded" content-type where parameters appear in the POST body as name-value pairs. An example of this method is displayed in the previous section. The value, in a name-value pair, should be URL encoded to ensure proper escape characters are included for special characters.

The second is the "multipart/form-data" content-type where the XML request and XSLT stylesheet can be uploaded as a file. This allows the FXintegrate On-Demand service user to store standard XML requests and/or XSLT stylesheet in a file and have a simple interface that can post the entire file content in the "multipart/form-data" format to the FXintegrate On-Demand service. Below is sample HTML for a form that can post both the XML request and the XSLT stylesheet in the "multipart/form-data" format.

```
<html>
<head>
 <title>FXintegrate On-Demand User Application</title>
</head>
<body>

<h3>Select the XML request and XSLT styelsheet to upload</h2>
```

```
<form method="post" enctype="multipart/form-data"
action="https://localhost:444/weblogic/download">
<table cellpadding="5" cellspacing="0" border="0">
<tr>
        <td>Request File</td>
        <td><input type="file" accept="*.xsl"
name="downloadRequestXML"></td>
</tr>
<tr>
        <td>XSLT File</td>
        <td><input type="file"
name="xsltStylesheet"></td>
</tr>
<tr>
        <td align="center" colspan="2"><input type="submit"></td>
</tr>
</table>
</form>
</body>
</html>
```

2.4.2 Request

2.4.2.1 Format

The FXintegrate On-Demand request format is described in the XML DTD and HTML documentation that should have been supplied with this document. If you have not received a copy of those files please contact a Currenex representative.

In using the FXintegrate On-Demand service, you are free to use any method to construct the XML request as long as the request conforms to the specified DTD.

2.4.2.2 Sample Requests

End of day reconciliation:

The following is an example showing a request for an end of day reconciliation on February 1, 2001. The time range is from 00:00:00am on February 1, 2001 to 11:59.59pm February 1, 2001. The response to this output would return the entire trade history for this particular day, including all new trades, amended trades, rolled trades and cancelled trades.

```
<?xml version="1.0" encoding="UTF-8"?>

<DownloadRequest version="1.0" type="Trade">
 <All>
   <DateRange>
     <BeginDate>20010201000000Z</BeginDate>
     <EndDate>20010201235959Z</EndDate>
   </DateRange>
 </All>
```

```
</DownloadRequest>
```

Latest changes since last download:

The following request shows an example on how to create a request that would be used to retrieve all trades, new and modified, since the previous auto-download action. At the end of each auto-download session, the FXintegrate On-Demand service will update the member's auto-download time. When the next download procedure is performed, the download period will be from the member's last download timestamp to the current time of request.

```
<?xml version="1.0" encoding="UTF-8"?>

<DownloadRequest version="1.0" type="Trade">
  <NewAndModified/>
</DownloadRequest>
```

Lastest changes since last download with translation:

The following requests the same information as the previous example. In addition, it requests the server to translate the XML download into a different format specified by the <Transform> element. Within the <Transform> element, the MimeType (content-type) and FileName (content-disposition) elements are also exercised.

```
<?xml version="1.0" encoding="UTF-8"?>

<DownloadRequest version="1.0" type="Trade">
  <NewAndModified/>
</DownloadRequest>
<Transform>
  <XSLT/>
  <MimeType>text/html</MimeType>
  <FileName insertDate=true>trade.html</FileName>
</Transform>
```

For the above request to be valid, an additional xsltStylesheet value must also be included in the http POST request. The resulting response will include two additional http header fields:

```
Content-Type: text/html
Content-Disposition: attachement; filename=trade20010512_123445.html
```

2.4.3 Response

2.4.3.1 Format

The FXintegrate On-Demand response format is described in the XML DTD and HTML documentation that should have been supplied with this document. If you have not received a copy of those files please contact a Currenex representative.

2.4.3.2 Trades vs. Events

The FXintegrate On-Demand service supports two different modes of operation. These allow trade data to be downloaded and used in different ways. Each of these is described below

Trade Download
Trade download mode downloads the current status of the trades requested as of the time of the download. It represents a snapshot of the state of each trade at a particular point in time. It is intended to be used in situations where the current state of the trade is important such as end of day reconciliation. Trade downloads can only be requested by time period or trade ID.

Event Download
Event download mode downloads events that change the state of a trade over its life. It is used to communicate changes to trades as they occur ensuring that data remains consistent across Currenex and trade capture systems. Event downloads can be used to pass trade updates in real-time to other systems, ensuring that data in downstream systems is always current. Event downloads can be requested by time period, by trade ID or as all events since the last download. Event workflows are described in Section 3

The choice of download mode is up to the user and will depend on how the data is to be used. Note that the mode selection is on a per request basis so nothing prevents one from using both of them if required. This is, however, not generally recommended.

2.4.3.3 Error Responses

In the case where an error occurred during the processing of client request, the response XML will include an Error element. This element, as the DTD defines, includes an ErrorCode and an ErrorDescription element. The following table lists the possible error codes and their corresponding error descriptions.

Possible Response Errors		
Code	**Message**	**Description**
100	Invalid XML request	This error is returned when the request XML does not conform to the request DTD.
120	Unsupported request version	The value specified in the version attributes of the DownloadRequest element is not supporte.
130	GET not supported	HTTPS File Download service only supports HTTP POST action.
180	The HTTP post data cannot be parsed	In the case where the post data is in form of multi-part mime, this error indicates that the data is not in the expected format and the server could not parse the data.
200	Internal error occurred during retrieval of session certificate	The user or process' certificate cannot be retrieved from the server. Please contact Currenex member service.

Possible Response Errors		
Code	**Message**	**Description**
210	Session certificate not present	A client certificate was not presented to the FXintegrate On-Demand service.
220	Session certificate not properly formatted	The certificate retrieved from the server is invalidly formatted. Please contact Currenex member service.
230	Session certificate does not match user certificate	The certificate presented to the service is different than the user's certificate in the database.
240	User certificate not found	The user's certificate cannot be located in the database.
250	Invalid user certificate	The user's certificate from the database is invalidly formatted. Please contact Currenex member service.
260	Invalid login id	The login ID in the user certificate does not exist in the database.
270	Download request denied	The user or process connected to the download service does not have the proper permission required.
280	Requested download format is not supported	The value specified in the downloadFormat attribute of the DownloadRequest element is not supported.
290	Invalid begin date	The begin date is invalid.
300	Invalid end date	The end date is invalid.
310	Invalid date range	The date range specified is invalid.
320	XSLT Stylesheet Not Specified	A XSLT transform was requested, but a xslt stylesheet was not included in the post.
330	Error occurred during XSLT transformation	An error occurred when processing the response XML through the XSLT processor. Please contact Currenex member service.
340	XSLT transformer configuration error	The XSLT processor on the server side was configured incorrectly. Please contact Currenex member service.
350	Internal error occurred	A server side internal system error occurred. Please contact Currenex member service.
600	Client HTML could not be generated	If using the Currenex FXtrades application to perform download, this is an error that could be generated when performing a page GET operation. This is an internal service error please contact Currenex Member Services.
601	User SessionTicket could not be validated	If using the Currenex FXtrades application to perform download, this is an error that could be generated when performing a page GET operation. This means the user was not properly logged in to the system prior to using the download application. Please log out and try again.

Possible Response Errors		
Code	**Message**	**Description**
603	The maximum number of download requests has been exceeded. Please try again later.	The download service can handle certain number of concurrent downloads. This error is displayed when this threshold has been reached. Please try again at a later time.
604	Event based download not enabled	In Order to do an StpEvent based query, the client must have StpEvents enabled. Please contact Currenex Member Services to enable this permission.

2.5 Security

FXintegrate On-Demand relies on the secured hypertext transfer protocol for secure communication and authentication. A Currenex member connects to the FXintegrate On-Demand service via HTTP over SSL on the standard port of 443. The Currenex server is configured for mutual authentication via digital certificates.

To connect to the service, the user must present a digital certificate that's signed by the Currenex CA. This certificate must contain the requester's login ID in the Common Name field.

Upon receiving the digital certificate during initial connection handshake, the FXintegrate On-Demand service will verify that the certificate is indeed signed by the Currenex CA and is within its validity period. The service will then retrieve the client's information from the repository and verify that the presented certificate matches the certificate of the client that is stored in the repository. The final step is to ensure the connecting client has the required permission to perform HTTPS file download.

When all checks are affirmative, the client request is processed by the service and corresponding result is returned to the client. Any security related errors would result in an error response being sent back to the client.

For details on obtaining a valid certificate from Currenex, please refer to the section on Digital Certificate.

3 Business Process Architecture

3.1 Events

STP events are created to signal important transitions in a trade's life. These events are used by the download interface to communicate trade changes to a party that is integrating their system with FXTrades. Each trade has two counterparties and both can request events or not. The actions of one party do not affect the other.

Each event has a type that reflects the nature of the transition a trade has experienced.

STP Events	
Event Code	**Description**
NEWT	This event is created when a trade is executed as the result of a price discovery process. The event can occur at most once in the history of a trade and may never occur (see the following). This event is created when a trade is executed or when an order is filled.
AMND	This event occurs when an existing trade is amended. It will have a new trade ID and will contain a reference back to the original trade.
CANC	This event is created when an existing trade is cancelled outright. The trade ID in this event will refer to the trade being cancelled. Once a trade has been cancelled no other events can occur on the trade.
ROLL	This event occurs when a trade is rolled forward. Each ROLL contains a new, unique trade ID and a reference back to the original trade. A trade may be rolled forward into more than one new trade. For each of these new trades a ROLL event will be generated that refers back to the original trade. It is possible for rolled trades to themselves be rolled.
ALOC	This event is similar to the ROLL in that it is one of many children of a parent trade. ALOC events occur when a trade is allocated. As with ROLL events each one will contain a unique trade ID and a reference to the originating trade.
UDFA	Defined fields can be altered anytime during the history of a trade. When they are altered a UDFA event records the fact that the change occurred. Many UDFA events can occur on a trade. Each UDFA event will contain the trade ID of the trade being modified. New trade ID's are not assigned for a defined field change.
CNDF	This is a special event necessitated by the behavior of defined fields. The letters stand for *cancelled with incomplete defined fields*. See the following discussion for further explanation.
SETL	This event occurs when the settlement instructions have been finalized for a trade. It contains the trade ID of the trade to which the settlement instructions belong. It will only be downloaded if settlement instructions have been requested.

3.2 Workflow

As trades within the FXtrades system undergo changes in response to user actions, those changes are communicated via FXintegrate On-Demand through the events described above. The following workflow rules apply to all events:

- NEWT, AMND, ROLL and ALOC are beginning events for trades. Each of these events will contain a new, unique trade ID. Those that are the result of changes to other trades (AMND, ROLL, ALOC) will contain references to the original trades.
- AMND starts a new trade and points back to its originating trade. This is a 1 to 1 relationship.
- ROLL and ALOC start new trades and point back to an originating trade. There may be many ROLLs or ALOCs for a single originating trade. Hence these are 1 to many events.
- An ALOC trade can have NO further events.
- CANC and CNDF end a trade's life. Such a trade cannot be amended, rolled or allocated.
- UDFA events can occur at any time in a trade's history and are not managed with a new trade ID.
- A CNDF event will be received for any trade that is cancelled, rolled, amended or allocated before its defined fields have been set. This is to ensure that some record of the original trade is sent even though it may contain no defined fields.
- NEWT, AMND, ROLL events will not be downloaded until any defined fields attached to the referenced trade have been set. This ensures that data required by downstream systems is not missing when a trade is downloaded.
- A trade cannot be allocated until its defined fields have been set.

3.3 Examples

The following examples illustrate the sequence of events that will occur for a given set of actions on a particular trade.

New Trade
NEWT

Allocated Trade
NEWT
ALOC
ALOC

Rolled Trade
NEWT
ROLL
ROLL
ROLL

Cancelled Trade
NEWT
CANC

Amended Trade
NEWT
AMND

New Trade with UDF's, cancelled before UDF's are set
CNDF

New Trade with UDF's, rolled before UDF's are set
CNDF
ROLL
ROLL

4 Requirements

4.1 Account Creation

Prior to using the FXintegrate On-Demand service, the appropriate member account must first be created. Please contact Currenex Member Services to request an HTTPS account. Upon account creation, the member will receive an HTTPS user ID and account PIN. The user ID and account PIN will both be used during the digital certificate request process, described below.

4.2 Digital Certificate

In order to connect to the FXintegrate On-Demand service, the connecting member must present a valid certificate that is issued by the Currenex Certification Authority (CA). This section will describe the process required to request an authentication certificate from Currenex.

Since the actual trade and settlement download process will be performed by an application, the name and email address for this account should be the application administrator's information. At the completion of the registration process (the process described in Account Creation), the Member Service representative will notify the application administrator with the login ID and a certificate request PIN.

1. Create a PKCS10 certificate request, otherwise known as a certificate signing request (CSR), containing the correct login ID and the administrator's email address. The login ID is created at the time when member services created the HTTPS service account. The key pair used to create the PKCS10 request should be an RSA key pair with a 1024 bit key length. The certificate's distinguished name (DN) should be in the format:

   Email=contact@membercompany.com, CN=Member Name, OU=Login - username, OU=Currenex, O=Currenex Inc.

   Example: Email=bob@company.com, CN=Bob Doe, OU=Login – bdoe, OU=Currenex, O=Currenex Inc.

   ***Note**: Username is the login id assigned by Currenex Member Services during Account Creation.*

   ***Note**: There are many ways to create a PKCS10 certificate request. The request can be generated from hardware as well as software based devices. The member is free to generate the request with any toolkit. For details on PKCS10 certificate requests, please refer to the document appendix.*

   The following is an example PKCS10 CSR:

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBtzCCASACAQAwdzELMAkGA1UEBhMCVVMxETAPBgNVBAgTCEFueVN0YXRlMRAw
```

```
DgYDVQQHEwdBbnlDaXR5MRUwEwYDVQQKEwxZb3VyIENvbXBhbnkxFzAVBgNVBAsT
DllvdXIgRGVwYXJ0bWVuMRMwEQYDVQQDEwp5b3VyZG9tYWluMIGfMA0GCSqGSIb3
DQEBAQUAA4GNADCBiQKBgQDLvFcMfxO2EltbdA/IFNX+s8gt/pVduzUtCwDHZIHc
s4/R6dmO8sJs3tiQ9Vd37Uf3qxKGwTBd2HUU2NU6bHz+HZTDQRQVfZ7DcmkRI74h
VS4LCyNjoqhx+osyq4lfUhMZy21YNvNsqujd9ZVG4hz02tLRE2L+kkU8WHOqx8L9
jQIDAQABoAAwDQYJKoZIhvcNAQEEBQADgYEAOwSh9UE3imNShbiq6WkweCNQ9qj4
nRLd/Ihz4F66H+sIIxH4EccB1qLjbcHuDjOhJETZiMqAOK5IcGRWqxbdYuup2JMB
qY0aLglS4pj6bF/mOC7PllmTUeCN5HyUAzhUcE165d5Jdcl49uce8kOsJ/kJM891
SGeaO+wwg+ZQ0ms=
-----END NEW CERTIFICATE REQUEST-----
```

2. Using an IE browser, proceed from the following URL:

   https://enroll.currenex.com

   Fill in the required information, userID and certificate request PIN, paste the PKCS10 request in base64 encoding in to the appropriate text field then click the submit button. The certificate will not be returned immediately, a separate email will be sent to notify the administrator that the certificate has been generated.

3. Once the certificate has been generated, Currenex will send an email to the email address specified in the Email field of the DN in the certificate. This email may or may not include the actual certificate. If not, information will be included in the email instructing the administrator on where and how to retrieve the certificate.

Once the member receives the valid certificate, it should be imported into whatever device the member used to generate the certificate request.

For details on certificate related issues please refer to the Currenex Certificate Practice Statement (CPS).

4.3 HTTPS Client

Connecting to the FXintegrate On-Demand service requires an HTTPS client. Currenex does not impose any restrictions on the type or architecture of the HTTPS client. This client, however, must be capable of performing connections via mutually authenticated SSL sessions.

The key pair used to create the certificate request in the previous section and the Currenex signed certificate must be use to establish the SSL connection. After connecting to the FXintegrate On-Demand service, the client must be able to construct a valid XML request message that conforms to the DTD defined in section 2.2.1 titled “request”.

The client then performs an HTTP POST action to the FXintegrate On-Demand service URL with the XML request as the value for the request parameter “downloadRequestXML”.

Handling of the trade and settlement response from Currenex may or may not be the responsibility of the HTTPS client. Response handling methodology is entirely up to the member and beyond the scope of this document.

5 Reference Client

5.1 Client Overview

Currenex provides a Java based HTTPS client that can communicate with and request trade information from the FXintegrate On-Demand service. This client is provided as a reference implementation only, please read the license agreement before using this client in any environment.

The reference client utilizes Sun's JSSE (Java Secure Socket Extensions) implementation to perform SSL related tasks. JSSE can be obtained from the following URL:

http://java.sun.com/products/jsse

Prior to using the reference HTTPS client, please follow the directions outlined in the following sections to properly configure the reference client before use.

5.2 Client Configuration

5.2.1 Configuration File

Before using the reference HTTPS client, the download.properties file must be properly configured. The download.properties file is used by the client application to initialize configuration parameters. It must reside in the classpath for the program to detect and read its contents. The format of the file is in the standard Java properties file format, where an equal (=) sign separates the name and value pair.

The supported values are listed and explained below.

HTTPS Client Configuration Parameters		
Parameter	**Example**	**Description**
keystorePath	/usr/local/https/keystore	Required field that specifies the path to the keystore file that contains the keys and certificates needed to login to the FXintegrate On-Demand service.
keystorePassword	Some_secret	Required field that specifies the password to unlock the keystore.
host	fxtrades.currenex.net	Required field that specifies the hostname to connect to.

HTTPS Client Configuration Parameters		
Parameter	**Example**	**Description**
port	443	Optional field that specifies the port to connect to. The default value is 443.
path	/fxintegrate/download	Required field that indicates the HTTPS File Deliery service's path.
downloadRequestXMLFilename	/usr/local/https/request.xml	Required field that specifies the file that contains the XML request.
xsltStylesheetFilename	/usr/local/https/stylesheet.xsl	Optional field that specifies the file that contains the XSLT stylesheet.

5.2.2 Keystore

In order to connect to the FXintegrate On-Demand service the connecting client must present a valid digital certificate. This certificate and its associated private/public key pair of the reference implementation reside in a Java keystore.

Keystore is essentially a database that holds public/private key pairs and their associated X.509 certificates. The keystore can also hold only certificates of entities that it trusts. For example, when connecting to a SSL enabled website, members must trust the server that they are connecting to, the keystore in this case, would hold the certificate of the server that the members are connecting to.

The keystore is used by the reference HTTPS client for two reasons.

- To store the authentication key and certificate required connecting to the FXintegrate On-Demand service.
- To store the FXintegrate On-Demand service’s issuer certificate.

Sun Microsystems provides a tool as part of its standard JDK distribution that can create and manipulate a java keystore. This tool is properly named keytool. For details on keytool and location for documentation from Sun, please refer to the appendix.

Steps must be taken to properly setup the JVM before creating the keystore. The keys and certificates used with FXintegrate On-Demand service must conform to the RSA algorithm. Sun’s JDK by default, does not include support for the RSA algorithm, however, JSSE does. Please follow the instructions given by Sun to properly install JSSE. JSSE installation instructions can be found here:

http://java.sun.com/products/jsse/INSTALL.html

It is very important that section 4, **Register the SunJSSE provider**, be followed closely.

Once JSSE is properly installed, follow the steps below to create and properly initialize a keystore. Then create a certificate request for Currenex CA to sign. Finally, import the Currenex CA certificate and install the Currenex signed member certificate.

Note: The following steps should be executed in order and values inside the { } should be substituted with values that reflect member environment or choices. Please ensure that jsse.jar library is in the classpath before executing the following steps.

**** Depending on the particular machine setup, the command:*
java sun.security.tools.KeyTool …
will sometimes produce an error "keytool error: KeyPairGenerator not available". In the case when this error is encountered, please replace the command with "keytool" and retain the same parameter values.

1. Create the public/private key pair and initial keystore with the follow command:

   ```
   java sun.security.tools.KeyTool –genkey –alias {userId} –keyalg RSA –keysize
   1024 –dname {distinguished_name} –keystore {keystoreName}
   ```

   Alternate command:

   ```
   keytool –genkey –alias {userId} –keyalg RSA –keysize 1024 –dname
   {distinguished_name} –keystore {keystoreName}
   ```

   You will be prompted to enter a password for the keystore object being created, please enter a password. When prompted to enter a second password for the specified alias, please enter the same password or simply press enter to present the same password.

2. Generate PKCS10 certificate request with the following command:

   ```
   java sun.security.tools.KeyTool –certreq –alias {userId} –keystore
   {keystoreName} –file {cert_req_file}
   ```

   Alternate command:

   ```
   keytool –certreq –alias {userId} –keystore {keystoreName} –file {cert_req_file}
   ```

3. The certificate request will be saved in the file {cert_req_file}. Follow the procedure outlined in the section on Digital Certificate to request a signed certificate from Currenex with the previously generated PKCS10.

   Before importing the X.509 certificate generated by Currenex, the Currenex CA root certificate must reside within the client's keystore. To import the Currenex CA certificate, perform the following steps:

   - Go to https://enroll.currenex.com and click INSTALL CA. This will prompt the user to save Currenex CA certificate. Save the certificate to disk.
   - Now import this certificate into the keystore as a trusted CA with the following command:

     ```
     java sun.security.tools.KeyTool –import –alias RootCA –file {filename} –
     keystore {keystoreName}
     ```

Alternate command:

keytool –import –alias RootCA –file {filename} –keystore {keystoreName}

4. Once the certificate is generated and assuming it is stored in a file called cert.file, follow the command below to import the certificate into the keystore.

   java sun.security.tools.KeyTool –import –alias {userId} –file cert.file –keystore {keystoreName}

   Alternate command:

   keytool –import –alias {userId} –file cert.file –keystore {keystoreName}

   When prompted to trust this file, please type “yes”.

5. The last step of the keystore initialization process is to import the signer of the Currenex server certificate. This is the signer of the Currenex fxtrades.currenex.net server certificate which is issued by VeriSign.

   The Currenex test server signer certificate and the production server signer certificate can both be found in the document appendix. The alias of this entry MUST start with TrustedCA. If multiple signer certificates need to be inserted into the keystore, the entries can be aliased as TrustedCA0, TrusedCA1, TrustedCATwo, etc. As long as the alias name begins with TrustedCA, it is considered valid. Store the certificates in a file called {filename} and perform the following command:

   java sun.security.tools.KeyTool –import –alias TrustedCA –file {filename} –keystore {keystoreName}

   Alternate command:

   keytool –import –alias TrustedCA –file {filename} –keystore {keystoreName}

Note: Use the keytool command as instructed above will require the user to enter the keystore password, both on creation and subsequent access. As a shortcoming of the keytool command, passwords are displayed in clear; care should be taken to protect the password.

The content of the keystore can be viewed by typing the following command:

$> keytool –list –keystore {keystoreName}

5.3 Using the Client

To use the client, compile all classes in the fxintegrate package. The compiled class directory or jar file must reside in the classpath. Make sure JSSE is properly installed and all JSSE related jar files are included in the classpath. Also make sure the download.properties file is properly configured and resides in a directory that is in the classpath as well.

To execute the client, run the DownloadClient program from the fxintegrate.demo.download package from the command line.

%>java fxintegrate.demo.download.DownloadClient

Output similar to the follow should be seen:

```
<?xml version="1.0" encoding="UTF-8"?>
<CurrenexDownload>
    <Trades>
        <party id="Currenex_X">
            <typedPartyId>
              <partyId>Currenex</partyId>
              <partyIdType>Currenex</partyIdType>
            </typedPartyId>
          </party>
          <party id="C">
            <typedPartyId>
              <partyId>Demo Customer Three</partyId>
              <partyIdType>Currenex</partyIdType>
            </typedPartyId>
          </party>
          <party id="B">
            <typedPartyId>
              <partyId>Demo Bank Three</partyId>
              <partyIdType>Currenex</partyIdType>
            </typedPartyId>
          </party>
          <trade>
            <tradeHeader>
              <partyTradeIdentifier>
                <partyReference href="C" />
                <tradeId>A200116409HK000</tradeId>
              </partyTradeIdentifier>
              <partyTradeIdentifier>
                <partyReference href="Currenex_X" />
                <tradeId>A200116409HK000</tradeId>
              </partyTradeIdentifier>
              <tradeDateTime>2001-06-13T20:29:22Z</tradeDateTime>
              <trader>
                <partyReference href="C" />
                <userId>dc3user1</userId>
              </trader>
              <trader>
                <partyReference href="B" />
                <userId>db3user1</userId>
              </trader>
              <subFund>Demo Customer Three</subFund>
              <event>
                <eventType>NEWT</eventType>
              </event>
            </tradeHeader>
            <tradeRequest>
              <requesterPartyReference href="C" />
              <buySell>BUY</buySell>
              <specifiedMoney>
                <currency>JPY</currency>
                <amount>1000000.00</amount>
              </specifiedMoney>
              <againstCurrency>USD</againstCurrency>
              <tenor>
                <period>SP</period>
              </tenor>
              <deliveryType>DEL</deliveryType>
            </tradeRequest>
            <product>
              <productType>SP</productType>
              <fxLeg>
                <cashFlow1>
                  <currency>JPY</currency>
                  <amount>1000000.00</amount>
                  <buyerPartyReference href="C" />
                </cashFlow2>
                <valueDate>2001-06-15</valueDate>
                <exchangeRate>
                  <currency1>JPY</currency1>
                  <currency2>USD</currency2>
                  <quoteBasis>currency1percurrency2</quoteBasis>
                  <rate>123.2120000</rate>
                </exchangeRate>
              </fxLeg>
            </product>
          </trade>
    </Trades>
</CurrenexDownload>
```

5.4 Limitations

The HTTPS client reference implementation does not have the ability to handle connections through proxies. Modifications to the reference source would be required to add proxy functionality into the HTTPS client.

The reference client does not provide the ability to automatically poll the FXintegrate On-Demand service for trade information on periodic intervals.

Appendix

Appendix A: XML Response Details

Please refer to response.html in the docs subdirectory for a complete list and explanation on all FXintegrate On-Demand response elements.

Appendix B: PKCS10 Certificate Request

PKCS10 is part of RSA Security's Public Key Cryptography Standard. The PKCS standards cover many aspects of cryptography, including the RSA encryption algorithm itself, message syntax, private key storage, cryptographic token access and others. PKCS10 refers to the syntax used when requesting a X.509 standard digital certificate.

Many tools can generate PKCS10 requests. These range from software-based toolkits to hardware-based cryptographic key storage units and accelerators. In creating a PKCS10 request, Currenex does not impose any restrictions on how a member create it, as long as it conforms to the PKCS10 standard.

For more information on the PKCS10 certificate request syntax, please visit RSA Security's website.

http://www.rsasecurity.com/rsalabs/pkcs/pkcs-10/index.html

Sun's keytool, the software based cryptographic key management tool set, can generate PKCS10 based requests. Please refer to the next section for details.

Appendix C: Keytool

Keytool is tool provided by Sun Microsystems as a part of the Java Development Kit to manage keystores (databases) of private keys and their association with X.509 digital certificates. It can also store and maintain a list of trusted certificates when connecting to a remote client through the Secured Socket Layer (SSL) protocol.

When using the Currenex reference HTTPS client, it is recommended that the member use keytool to create and manage both public/private key pairs as well as creating PKCS10 based certificate requests.

This section will explain the steps required for the member to properly create, initialize and use the keytool to manage the keystore used by the HTTPS client. For detailed information on all aspect of the keytool utility, please refer to the Sun's documentation.

http://java.sun.com/products/jdk/1.2/docs/tooldocs/solaris/keytool.html

Appendix D: Development and Production Signer Certificates

The Currenex development server https://staging.currenex.com and the production server https://fxtrades.currenex.net both utilize server certificates signed by VeriSign. The VeriSign root CA certificate must be imported into the reference implementation's keystore to provide proper trust of Currenex servers.

The following is a base 64 encoding of the VeriSign CA certificate.

```
-----BEGIN CERTIFICATE-----
MIIEMTCCA5qgAwIBAgIQI2yXHivGDQv5dGDe8QjDwzANBgkqhkiG9w0BAQIFADBf
MQswCQYDVQQGEwJVUzEXMBUGA1UEChMOVmVyaVNpZ24sIEluYy4xNzA1BgNVBAsT
LkNsYXNzIDMgUHVibGljIFByaW1hcnkgQ2VydGlmaWNhdGlvbiBBdXRob3JpdHkw
HhcNOTcwNDE3MDAwMDAwWhcNMDQwMTA3MjM1OTU5WjCBujEfMB0GA1UEChMWVmVy
aVNpZ24gVHJ1c3QgTmV0d29yazEXMBUGA1UECxMOVmVyaVNpZ24sIEluYy4xMzAx
BgNVBAsTKlZlcmlTaWduIEludGVybmF0aW9uYWwgU2VydmVyIENBIC0gQ2xhc3Mg
MzFJMEcGA1UECxNAd3d3LnZlcmlzaWduLmNvbS9DUFMgSW5jb3JwLmJ5IFJlZi4g
TElBQklMSVRZIExURC4oYyk5NyBWZXJpU2lnbjCBnzANBgkqhkiG9w0BAQEFAAOB
jQAwgYkCgYEA2IKA6NYZAn0fhRg5JaJlK+G/1AXTvOY2O6rwTGxbtueqPHNFVbLx
veqXQu2aNAoV1Klc9UAl3dkHwTKydWzEyruj/lYncUOqY/UwPpMo5frxCTvzt01O
OfdcSVq4wR3Tsor+cDCVQsv+K1GLWjw6+SJPkLICp1OcTzTnqwSye28CAwEAAaOC
AZAwggGMMA8GA1UdEwQIMAYBAf8CAQAwCwYDVR0PBAQDAgEGMBEGCWCGSAGG+EIB
AQQEAwIBBjAgBgNVHSUEGTAXBgpghkgBhvhFAQgBBglghkgBhvhCBAEwggE1BgNV
HSAEggEsMIIBKDCCASQGC2CGSAGG+EUBBwEBMIIBEzAoBggrBgEFBQcCARYcaHR0
cHM6Ly93d3cudmVyaXNpZ24uY29tL0NQUzCB5gYIKwYBBQUHAgIwgdkwFRYOVmVy
aVNpZ24sIEluYy4wAwIBARqBv1ZlcmlTaWduJ3MgQ2VydGlmaWNhdGlvbiBQcmFj
dGljZSBTdGF0ZW1lbnQsIHd3dy52ZXJpc2lnbi5jb20vQ1BTLCBnb3Zlcm5zIHRo
aXMgY2VydGlmaWNhdGUgJiBpcyBpbmNvcnBvcmF0ZWQgYnkgcmVmZXJlbmNlIGhl
cmVpbi4gU09NRSBXQVJSQU5USUVTIERJU0NMQUlNRUQgJiBMSUFCSUxJVFkgTFRE
LiAoYykxOTk3IFZlcmlTaWduMA0GCSqGSIb3DQEBAgUAA4GBALiMmMMrSPVyzWgN
GrN0Y7uxWLaYRSLsEY3HTjOLYlohJGyawEK0Rak6+2fwkb4YH9VIGZNrjcs3S4bm
fZv9jHiZ/4PC/NlVBp4xZkZ9G3hg9FXUbFXIaWJwfE22iQYFm8hDjswMKNXRjM1G
UOMxlmaSESQeSltLZl5lVR5fN5qu
-----END CERTIFICATE-----
```

To import this certificate, save the text content above in a file and follow the import instructions given in step 5 of section 4.2.2 Keystore.

To verify the validity of the above certificate, the member is free to use any tool to parse the above certificate and contact VeriSign for further details.