FpML Transport Guidelines

Status of this document

This technical paper is a working draft for a set of guidelines for how to transport FpML between organizations. Those guidelines are initially intended to be non-normative recommendations, but over time are intended to evolve into normative specifications.

The FpML Architecture Working Group encourages reviewing organizations to provide feedback as early as possible. Comments on this document should be sent by filling in the form at the following link: http://www.fpml.org/issues. An archive of the comments is available at http://www.fpml.org/issues/

There are also asset class-specific mailing lists; you can join them at the following link:

Join a Working Group at FpML

A list of current FpML Recommendations and other technical documents can be found at

http://www.fpml.org/spec

This document has been produced as part of the FpML Architecture Working Group and is part of the Standards Approval Process.

1 Purpose of this document

This document provides guidelines on how to communicate FpML between firms following a consistent transport model specified by the FpML standards process. While firms that have already developed implementations that do not follow the guidelines may continue to use those, new implementations are strongly encouraged to follow the guidelines in this document. Existing implementations are encouraged to conform to the guidelines when they update their transport mechanisms.

Where the guidelines allow multiple options or choices in how to implement a particular transport, firms are strongly encouraged to implement all of the recommended features of the selected option or level.

2 Transport Mechanisms

FpML-defined transport mechanisms currently include:

- web services using SOAP over HTTP(S)
- file transfer using SFTP (Secure File Transfer Protocol)
- FpML over the FIX session layer.
- FpML over message queues

FpML may also be sent over the SWIFT network. Please consult SWIFT for more information. Following are some links that may be useful:
http://www.swift.com/news/derivatives_solution?lang=en
http://www.swift.com/products_services/derivatives
[to be added: links to IM-Custodian WG? ISITC? Other contact info?]
In addition, this document briefly describes how to embed FpML inside FIXML. Future drafts of these guidelines may cover additional transports as described below.

2.1 Web services (SOAP over HTTP/HTTPS)

2.1.1 Description

A SOAP web service is an HTTP(s) based interface that is defined in a WSDL wrapper. XML requests can be submitted to the web service from a variety of environments. This section defines how to create a web-service based application based on FpML.

2.1.2 Security mechanisms

A variety of security mechanisms are possible for web services applications. Because it is relatively easy for most firms to implement common security mechanisms, FpML does not prescribe which mechanism should be used. However, the following mechanisms have been successfully used to secure FpML web services, and are therefore recommended:

- client certificates
- HTTP(s) basic authentication
- firewall rules (together with other methods)

Proprietary authentication tools have also been used successfully, but are not preferred.

2.1.3 Compression

If the web services framework being used supports compression natively (i.e., so that it is transparent to the client), it is recommended that this be supported for clients that are capable of using it. Non-native compression (i.e. where the client must explicitly compress the message in code) is discouraged for interactive web-services message requests. If a large file must be uploaded to a web-page or URL in more of a batch use scenario, this type of compression may be supported.

2.1.4 Entry points

At least one of the following two entry points should be defined by any FpML web services application.

2.1.4.1 For submitting FpML messages

The following entry point should be provided for a client to submit FpML messages to the FpML web service:

```
public string submitMessage (string fpmldoc);
```

This method will submit the FpML messages to the web service without requiring any immediate processing (ie. responses will be fully asynchronous). The return value will be a request ID that may be used to retrieve or correlate responses

2.1.4.2 For processing FpML messages synchronously

```
public string processMessage (string fpmldoc);
```

This method will submit the FpML message to the web service, perform at least basic processing, and return an FpML message as a response. (That message could be an exception or an acknowledgement or possibly a business response such as an executionNotification or status message, depending on the service.)

In the calling client program, connection/transport errors that prevent accessing these entry points may generate error codes or throw exceptions such as:

- illegal URL/address not found
- timeout
- connection refused
- submission refused (e.g. security issue, like invalid credentials)

2.1.4.3 For retrieving messages

All FpML web services applications should define the following interface method for asynchronously returning/retrieving FpML messages:

```
public string retrieveMessage(boolean wait);
```

The service will return the next message queued for the client in an XML structure that also contains the requestId. (We need to define this structure.) . If no message is currently available and "wait" is true, it will wait for the system-defined timeout. If wait is false the call will not wait if there are no messages available.

Optionally the webservice may also define this variation of the interface:

```
public string retrieveResponseMessage(string requestId, boolean wait);
```

When called with a requestId argument, the service will wait until a response message is available for a specific request, where the requestId was returned by a prior "submitMessage()" request. The return type is similar to the above, ie. an XML structure containing the requestId and the response message. This allows asynchronous requests to be handled in a synchronous manner more easily.

If this variation of the interface is provided, the web service should also provide this interface method:

```
public string retrieveNotificationMessage(boolean wait);
```

This method works as "retrieveMessage", but only retrieves notification messages that aren't as a result of requests. This makes it easier to mix synchronous and asynchronous message retrieval.

2.1.5 Example

- we need to develop an example (probably in Java) of the FpML web service with a WSDL definition.
- we should show
    - request and response
        - successful (ack)
        - unsuccessful (exception)
    - async notification (e.g. status change, trade execution notification, confirmation status)
- We probably will need some kind of pseudo-login process so that if there are multiple clients, responses go to the correct client.

- probably want a short driver program that submits a request, gets a nack, fixes the issue, gets an ack, then gets a subsequent status update, e.g. confirmation status
- ultimately we should have a hosted webservice toy app that can accept requests, validate them against the schema, and return some kind of response (ack/nack and maybe an async notification, like a system notification.)

2.2 SFTP

2.2.1 Description

A file transfer-based communication mechanism based on the Secure File Transfer Protocol and SSH (Secure Shell) can be used transfer FpML files between firms. Typically a client will submit files to a service and then retrieve responses somehow. This section describes how FpML recommends to firms to implement an SFTP-based communications mechanism.

2.2.2 Security mechanism

Service providers implementing FpML transports using SFTP should create an SFTP site that clients can upload ("put") files to, and retrieve ("get") files from. Optionally the service provider can notify clients when a file is available, for example using an email message. When there is a bilateral SFTP-based communication, the SFTP site should normally be run by the larger firm, or the firm offering processing services.

SFTP supports a variety of security mechanisms, including

- username/password,
- client certificates,
- firewalling (restricted IP addresses) (this is not part of SFTP, but is often layered on top of SFTP)

FpML recommends that firms use either

- client certificate-based security for SFTP authentication.[do we need to specify number of bits of encryption, etc., etc about the certs? anything about the process for creating client certs?] or alternatively:
- username/passwords issued using a controlled mechanism.
- optionally, for increased security, firms may set firewall rules to restrict access to specific IPs or IP ranges.
- Additional measures, such as file encryption (such as PGP or GPG encryption) may also be used, but implementers are warned that some clients may find this a barrier to using the service.

2.2.3 Naming Conventions

2.2.3.1 Folder/Directory Layout

- Services should provide a separate account with a separate folder/directory structure for each client.
- Within each client's home account, the following directories should be available:
  - `tmp` - temporary directory for uploading files (optional)
  - `input` - input directory for placing client input to the service
  - `output` - output directory for placing service output to the client
  - `error` - directory for placing input files that cannot be processed in any way (e.g. wrong extension).[optional]
  - optionally services may provide an "`archive`" folder that contains a copy of client submissions that have been processed.

2.2.3.2 File Naming Convention

- Each FpML message should be contained in a separate file with a name that is unique and ideally ascending in intended processing order. Ideally this file name will be based on a uniquely identifiable component of the message, such as the message ID, or the timestamp. FpML does not define any processing order semantics for SFTP message transfer.
- To support bulk submission of multiple files, FpML recommends that services provide support for the following archive/compression methods:
  - Zip files with the .zip suffix. The file name of the zip file should be a unique ascending alphanumeric value (possibly date/time based) with the "`.zip`" suffix.
  - (optional) Tar files, optionally compressed with Gzip. The file name of the archive file should be a unique ascending alphanumeric value (possibly date/time based) with the "`.tar`" or "`.tar.gz`" suffix.
  - Public key encryption such as PGP or GPG may also used.

2.2.4 Communication mechanism/protocols,

2.2.4.1 Message Submission by clients

To submit messages to the service, the clients should:

1) log on securely to the SFTP site/account
2) upload each file to the "tmp" directory using FTP commands like "put", or if no tmp directory is provided, using a temporary file extension.
3) when each file is uploaded, move/rename it to the "input" directory, or to use a permanent file extension such as ".xml" or ".zip"

To process messages received from clients, services should:

1) poll the "`input`" folder periodically
2) when files are identified, they should be processed according to their file types as follows:
   a. `.xml` files should be treated as FpML. They should be moved into a temporary directory for processing read into the application, and optionally copied to an archive folder once processed.
   b. archive files (`.zip`, `.tar`, `.tar.gz`) should be moved into a temporary folder, decompressed/unpacked, and processed in turn. Optionally the archive file could be copied to an `archive` folder. If the archive cannot be extracted, the file should be moved to the "`error`" folder.
   c. unrecognized file types should be moved to the "`error`" folder.

If the `archive` folder is provided, the service may choose to periodically (e.g. daily, weekly) prune out the archive if client has not already done so.

2.2.4.2 Message Retrieval by clients

To return files to clients, services should:

1) create the files in a temporary directory
2) move the files in to the client's "`output`" directory as a single "move" or "rename" operation. Ensure that each new file is named with a unique, ideally ascending filename that ideally corresponds to something in the message, such as the message ID or timestamp.

Clients should:

1) periodically poll the "`output`" directory
2) download the file using SFTP commands, e.g. get.
3) delete the file from the "output" directory once downloaded.

If the service generates large output files, such as periodic reports, it may compress these using one of the above methods and require clients to decompress the files once downloaded.

- response methods
  - we should define that FpML response messages and async notifications should be supported, i.e. there should be a way for a client to retrieve FpML responses, such as acks, exceptions, status messages, and notifications.
- bulk transfer, compression and encryption methods
  - The following archive/compression mechanisms should be supported:
    - zip - for all platforms
    - gzip - for Unix/Linux based platforms
    - tar - for Unix/Linux based platforms
    - PGP - optional
  - Where a large number of messages need to be returned at once, these can be archived into a single file for space/download time reasons.

2.2.5 Demonstration

We should develop a demonstration of FpML over SFTP.

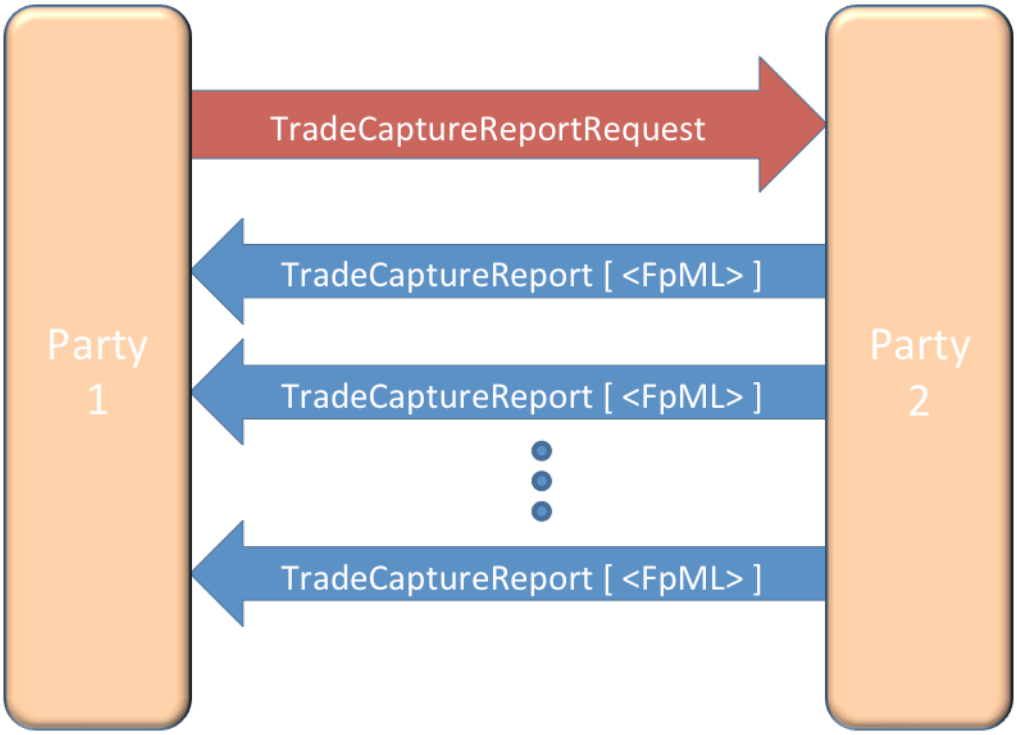
- Do we need some kind of a registration/login process to set up the client-specific logins? Perhaps to start we should just set up several and allow clients to pick one.
- What technology should we use for developing the SFTP code?
  - Spring for Java?
  - Jsch or Apache commons-vfs
    - http://kodehelp.com/java-program-for-uploading-file-to-sftp-server/
  - sshj
  - see this for various suggestions: http://stackoverflow.com/questions/14617/java-what-is-the-best-way-to-sftp-a-file-from-a-server

2.3 FpML over FIX

- description
  - Embed FpML in an pure FIX (tag = value) transport.
  - In FIX there is support to encapsulate FpML since version 4.4. The XML_nonFIX tag allows including FpML as a string of characters.
  - There are two different approaches in FIX regarding the use of embedded FpML:
    1. Embed FpML within the FIX Session Layer only and have all the business content in FpML. For example:

8=FIX.4.4^9=7718^35=n^34=41^49=PROTRD^52=20100920-23:06:17.536^56=ALUNO02212=7660^213=<?xml version="1.0" encoding="utf-8"?>
<requestConfirmation xmlns="http://www.fpml.org/FpML-5/confirmation" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" fpmlVersion="5-6" xsi:schemaLocation="http://www.fpml.org/FpML-5/confirmation ../../fpml-main-5-6.xsd">
<header>
<messageId messageIdScheme="http://www.example.com/messageId">MS/2006/04/02/15-12</messageId>
<sentBy>PARTYABICXXX</sentBy>
<sendTo>PARTYBBICXXX</sendTo>
<creationTimestamp>2013-04-02T15:38:00Z</creationTimestamp>
</header>
….
^10=200^

    2. FpML encapsulated in FIX Business Messages: Represent only the product definition in FpML and embed it within the FIX Instrument component:
       a. The FIX Instrument component contains a SecurityXML field to embed the product definition in XML format.
       b. There are three tags to embed the product definition in XML:
          i. Tag 1184 = SecurityXMLLen: Must be set if SecurityXML field is specified and must immediately precede it
          ii. Tag 1185 = SecurityXML: XML payload or content describing the Security information.
          iii. Tag 1186 = SecurityXMLSchema: XML Schema used to validate the XML used to describe the Security.



- response method
  - we should show how the response should be returned
    - FIX only vs
    - FpML embedded in FIX
- we should provide examples of several messages
- we should provide links to FIX documentation of the appropriate message types

2.4 Message Queues (JMS, MQ, MSMQ, AMQP)

2.4.1 Description

Many firms send FpML over a standard message queuing software such as IBM MQ or MS MQ, typically using a wrapper such as JMS. This section provides guidelines and advice on how to implement this type of transport for FpML.

2.4.2 Guidelines

- avoid using message properties for carrying business data that is not in the message payload (it is ok to extract business data to message properties to simplify routing, for example)
- There should be some linkage between message properties and FpML message data - for example, the following correspondences are recommended:
    - Assuming that individual queues are used for each client,
        - the queue that the client writes to to submit messages to the service should be called "x.in", where "x" is the value in the "header/sentBy" element of the FpML message.
        - The queue that the client reads from to retrieve messages from the service should be called "x.out", where "x" is the value in the "header/sentBy" element of the FpML message.
        - The "header/sendTo" should correspond to the service name. Services may use this for message routing where there are multiple services.
        - If desired, key attributes can be extracted from the FpML for routing, such as:
            - message name
            - asset class
            - product tag, productType, or productId
            - fpmlVersion.
        - Where the above is done, key field extraction should preferably be done by the recipient/service rather than required to be done by the submitter/client.

2.4.3 Security mechanisms

Messaging security may be handled by any standard message queuing infrastructure. However:

- If network security is required, VPNs should be supported. Dedicated lines may also be supported if desired.
- Some kind of connection request mechanism could be required/developed. There is current no support for this in FpML.

2.4.4 Sample Code

- We have developed sample code for this in Java that works with Apache QPID (using AMQP messaging.)

2.5 Other transport protocols to consider

The following transports have been considered by the FpML Architecture Working Group. Some of these may be covered in future versions of this document.

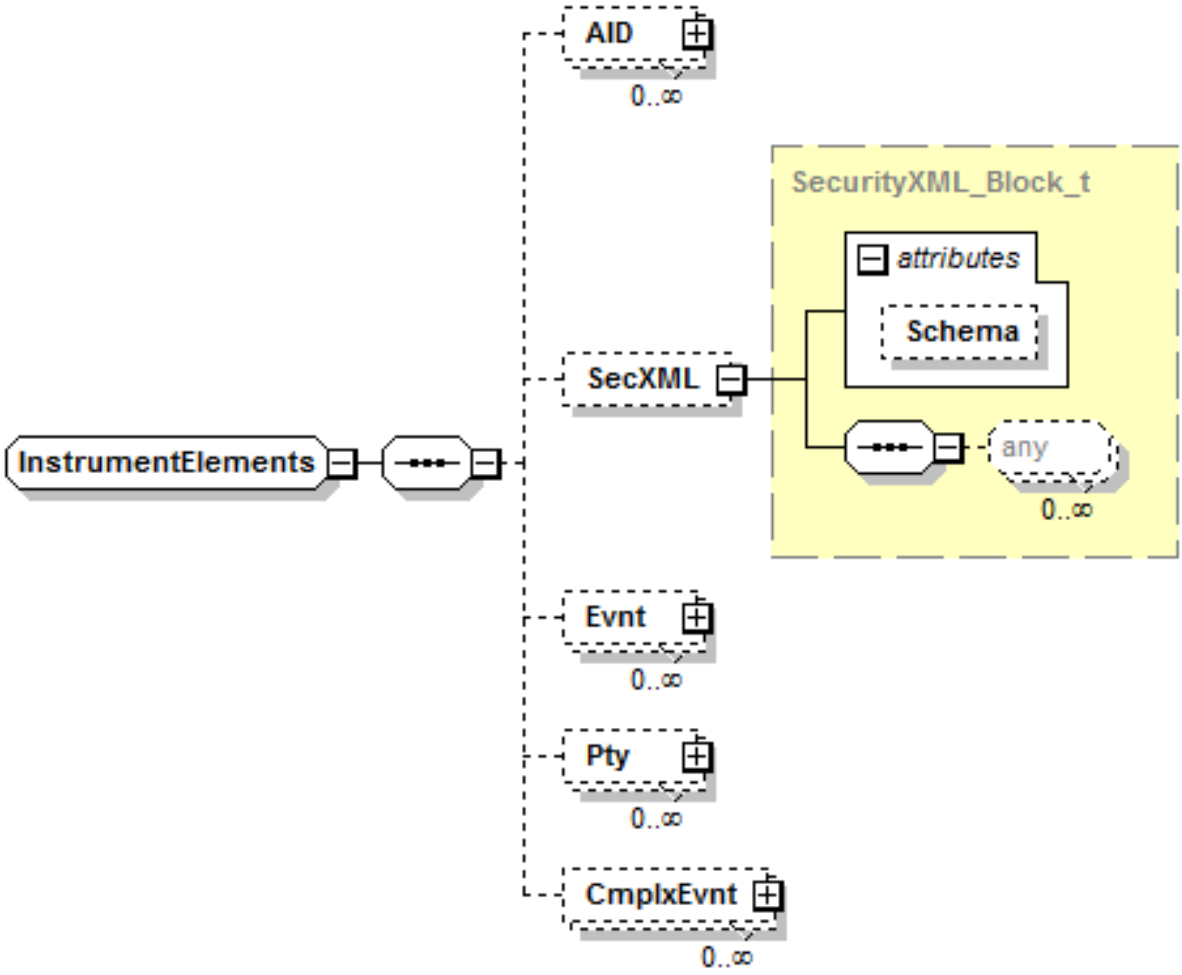
- SMTP - FpML over email - EFET uses SMTP, should FpML consider this too?
- Simple HTTP - file upload and download. We could define a simple way to submit messages using POST and retrieve messages either using GET or POST. (Need to consider caching behavior when polling using GET)
- RESTful HTTP web services - essentially the same as simple HTTP but following the REST (representational state transfer) rules
- WebDAV - more convenient way of using HTTP to communicate, but not so widely adopted/security headaches.

3 FpML embedded in FIXML

3.1 FpML in FIXML

[Not really a transport ... move either to a subitem under FIX, or its own area?]

- description
  - Embed FpML in a FIXML messageThe FIXML SecurityXML Component allows encapsulating an FpML product definition within a FIXML business message:



- we should provide examples of FpML in FIX:

```
<?xml version="1.0" encoding="UTF-8"?>
<FIXML xmlns="http://www.fixprotocol.org/FIXML-5-0-SP1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
>
<TrdCaptRpt ReqID="C092CBTEOD20061006" RptID="102994" LastQty="4"
TrdDt="2006-10-06" MtchStat="0" TxnTm="2006-10-06T00:00:00-05:00"
BizDt="2006-10-06" PxTyp="18" TrdTyp="0" MtchID="0000182185" MsgEvtSrc="REG"
LastPx="0.59375">
<Instrmt MatDt="2016-12-20" MMY="201612" StrkPx="105.00" Mult="1000"
Exch="CBT" CFI="OPAXPS" ID="17">
<SecXML Schema="http://www.fpml.org/FpML-5/confirmation">
<dataDocument fpmlVersion="5-0" xmlns="http://www.fpml.org/FpML-5/confirmation"
```

```
xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.fpml.org/FpML-5/confirmation ../fpml-main-5-6.xsd">
<trade>
<tradeHeader>
....
</tradeHeader>
<swap>
<productType>InterestRate:IRSwap:FixedFloat</productType>
...

</swap>
...
</dataDocument>
</SecXML>
</Instrmt>
...
</RptSide>
</TrdCaptRpt>
</FIXML>
```

- submissions
- responses

4 Recommended transports

Following are some guidelines on how to select which transport(s) to implement for a particular application. Many applications will support multiple transports, for example message queues and web services for straight through processing and SFTP for bulk daily reporting.