Nasdaq Calypso

FpML Integration Guide

Version 17 – Version 18

Revision 1.0
January 2024
Approved

Document History

Revision	Published	Summary of Changes
1.0	January 2024	First edition for version 17 and version 18

This document describes the FpML interface.

Table of Contents

Overview

This document describes the Calypso FpML API setup and usage. The FpML API is used to translate native Calypso trades into FpML trade documents. These FpML documents can then be embedded within an FpML envelope, either specified by the users particular interface, or by an external interface such as MarkitWire, CME, or LCH.

The FpML API is intended to be used as a utility API, and not directly as an Interface in itself. For example, if exceptions occur during translation, it returns a set of exceptions which can be read by the calling interface and handled in whichever way is appropriate for that interface's use case.

In addition, although standard trade header and party elements are created, many interfaces will likely have to customize those tags according to their messaging requirements. However, the product section is intended to be translated in a standard and universal manner which can be used by any interface that follows the FpML standard.

1.1 Calypso FpML API Features

1.1.1 Supported

The FpML API supports the following trade types:

- Vanilla Swaps
  - Compounding
  - IMM
  - Fees
  - Stub Periods
  - Cash Settle Info (Break Clause)
  - Forward Starting
- Amortizing Swaps (Bullet / Schedule)
- Basis Swaps
- ZC Swaps (Rate / Fixed Amount)
- OIS (using standard Calypso booking method)
- FRA
- Swaption
  - European, American and Bermudan exercise
  - Swaption Early Termination procedure

1.1.2 Not Supported

The following product flavors are currently not supported by the FpML API:

- Vanilla Swaps with
  - Amortization types other than Bullet / Schedule

– Customized Cashflows

1.2 Installation

The FpML API is a part of the Data Uploader. All subsequent instructions assume that all applicable Data Uploader installation steps have been completed successfully.

Please refer to Calypso Data Uploader documentation for specific installation and configuration information.

NOTE: You must install and configure the Data Uploader prior to using the FpML API.

Setup Requirements

2.1 Calypso Mapping Window

The Calypso Mapping Window is required to map the Calypso values to the incoming and outgoing FpML values for importing and generating FpML.

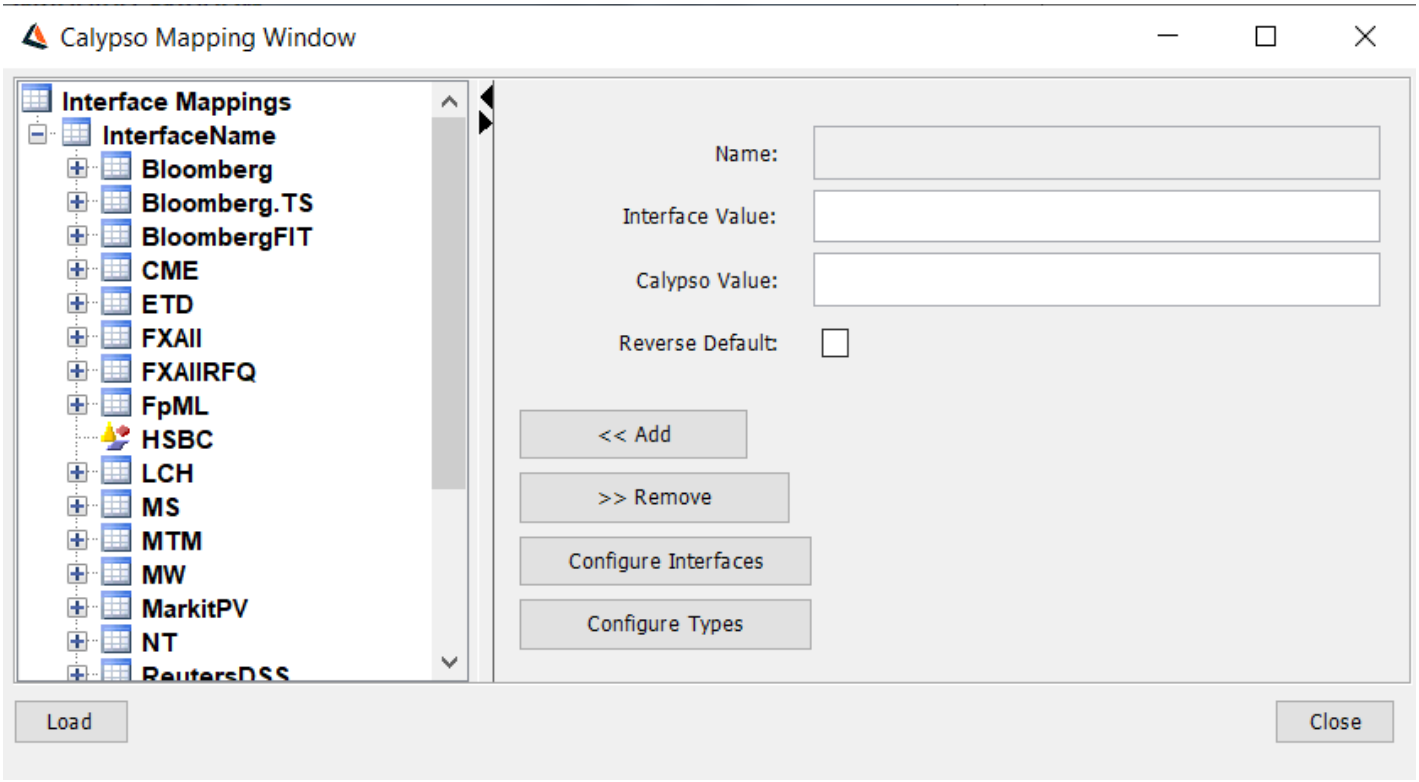
Two Level Mapping

The FpML API uses a two-level mapping system for mapping Calypso values to FpML values. This allows you to share FpML mappings across various interfaces when they are common – e.g. DateRoll, DayCount – and at the same time, override them for interfaces where they are different.

NOTE: Several standard FpML mappings are provided for you out-of-the-box, when you run the **'fpml'** schema category.

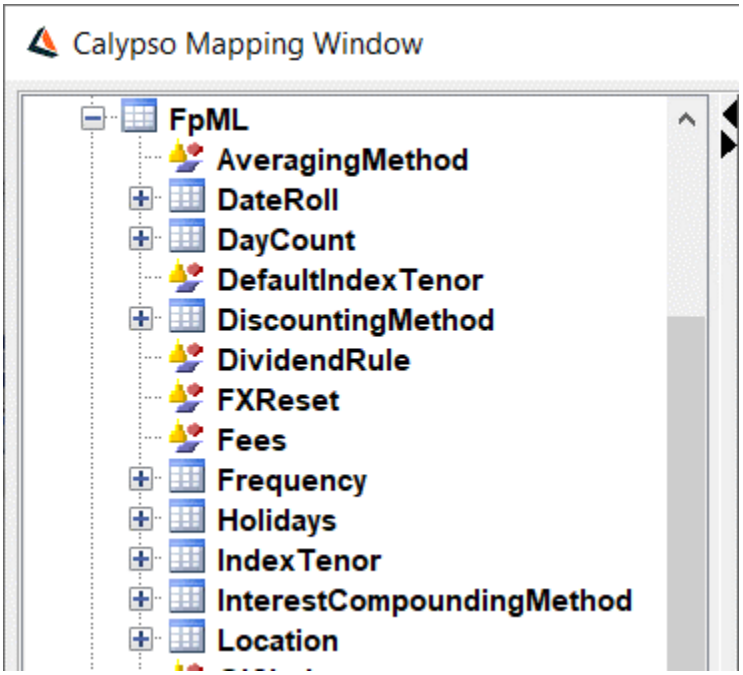
Level 1: Interface level mappings, e.g. LCH, CME, MW, etc.

Mappings for FpML values can be done at the interface level. The interface name is passed in when the FpML API call is made.



Level 2: FpML Level Mappings

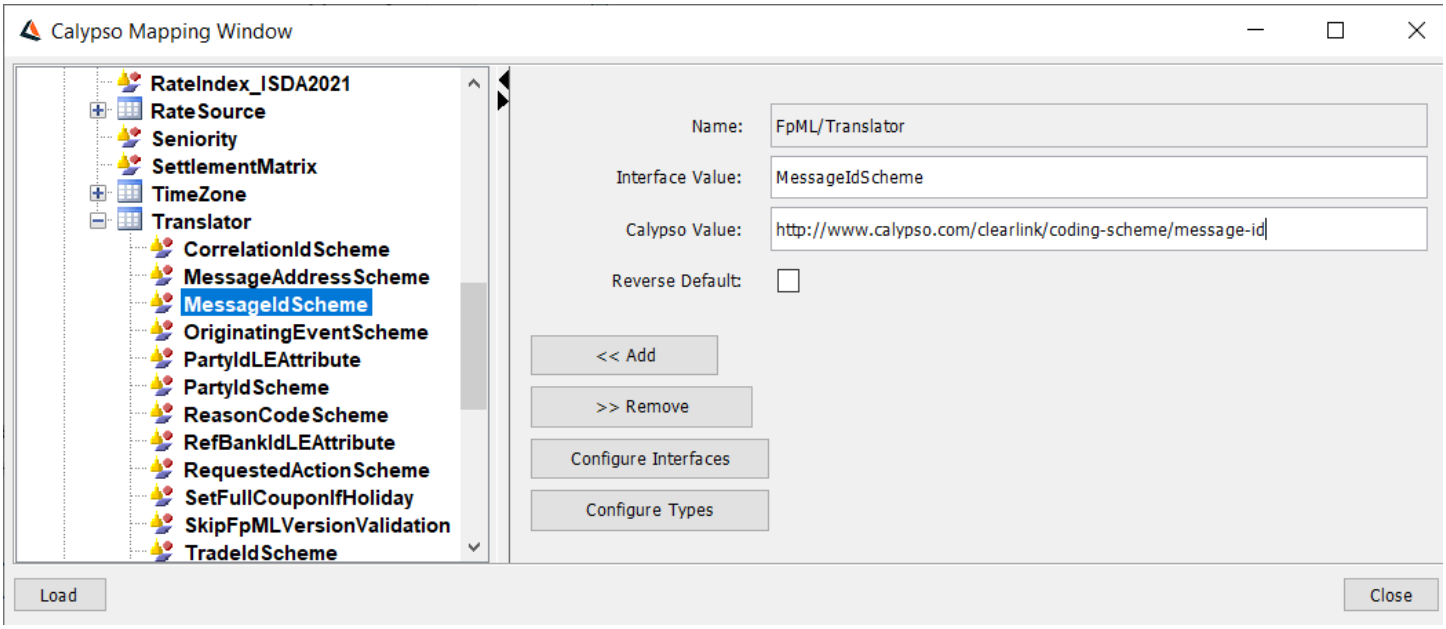
If a mapping is not found at the interface level, the default mapping from the FpML interface is used.



The mapping described here is the FpML mapping.

2.1.1 Mapping Type FpML/Translator

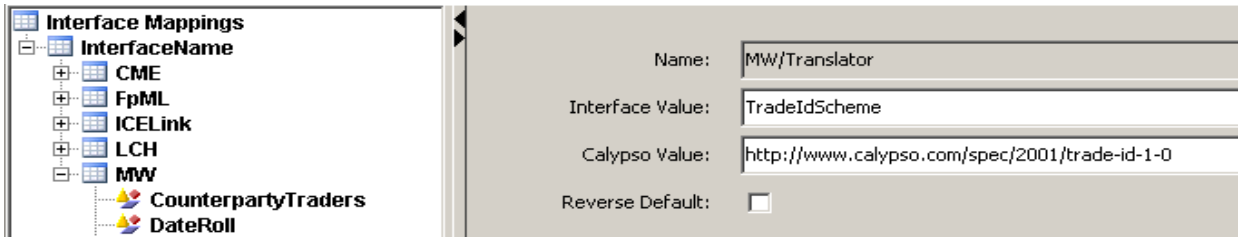
These mappings define values for the attributes used in FpML generation. These attributes should normally be defined at the Interface level, though ones which are common across interfaces can be defined at the FpML level. These attributes are described in detail below and should be setup prior to using the FpML API.



- TradeIdScheme

  This value is used to define the tradeIdScheme attribute within the <tradeId> tag in the <tradeHeader> section of the FpML document.

```
<tradeHeader>
    <partyTradeIdentifier>
        <partyReference href="partyA"/>
        <tradeId tradeIdScheme="http://www.calypso.com/spec/2001/trade-id-1-0">51110</tradeId>
    </partyTradeIdentifier>
    <partyTradeIdentifier>
        <partyReference href="partyB"/>
    </partyTradeIdentifier>
    <tradeDate>2013-06-03</tradeDate>
</tradeHeader>
```



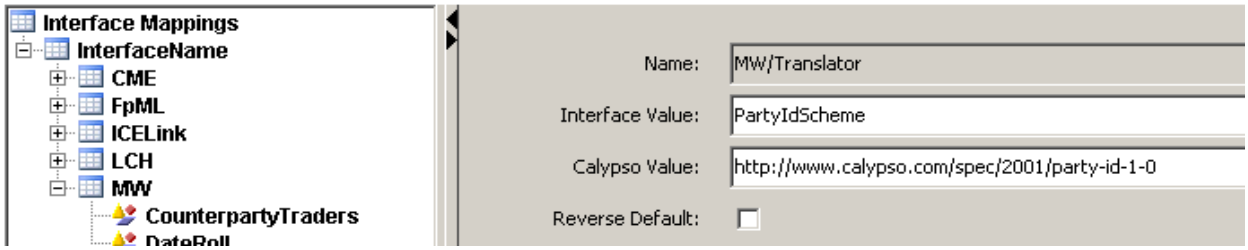
- PartyIdScheme and PartyIdLEAttribute

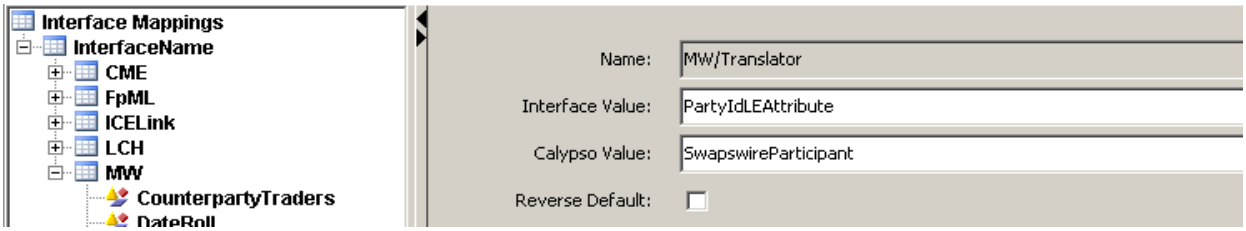
The 'PartyIdScheme' value is used to define the partyIdScheme attribute within the <partyId> tag in the <party> section of the FpML document.

The 'PartyIdLEAttribute' value is used to define the Calypso Legal Entity attribute name. If this mapping is set, the FpML API will look up the value of this LE Attribute on the Calypso Legal Entity for the two trade parties (PO and CP) and set the partyId tag to this value.

Note: The LE Attribute logic assumes a one-to-one mapping between the Calypso Legal Entity and the interface attribute value. This will not work for interfaces which have a parent/child LE look up logic.

```
<party id="partyA">
    <partyId partyIdScheme="http://www.calypso.com/spec/2001/party-id-1-0">CALYPXXXX</partyId>
</party>
<party id="partyB">
    <partyId partyIdScheme="http://www.calypso.com/spec/2001/party-id-1-0">SWAP1234</partyId>
</party>
```

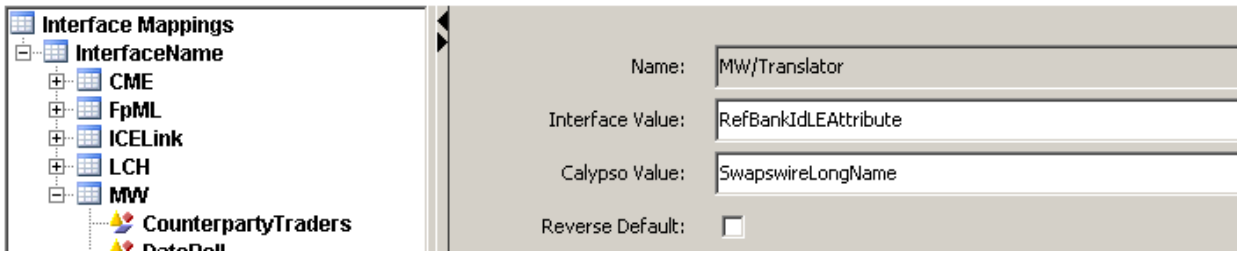




- RefBankIdLEAttribute

The 'RefBankIdLEAttribute' value is used to define the Calypso Legal Entity attribute name for the Reference Banks in the Cash Settle Info screen. If this mapping is set, the FpML API will look up the value of this LE Attribute on the Calypso Legal Entity for each Reference Bank and set the 'referenceBankId' tag (within the <cashSettlementReferenceBanks> tag in the <cashPriceMethod> section of the FpML document) to this value.

```
<cashPriceMethod>
    <cashSettlementReferenceBanks>
        <referenceBank>
            <referenceBankId>Citigroup - New York</referenceBankId>
        </referenceBank>
    </cashSettlementReferenceBanks>
    <cashSettlementCurrency>USD</cashSettlementCurrency>
    <quotationRateType>Mid</quotationRateType>
</cashPriceMethod>
```



- MessageIdScheme

The 'MessageIdScheme' value is used to define the messageIdScheme attribute within the < messageId> tag in the <header> section of the requestConsent FpML document.

```
<requestConsent xmlns="http://www.fpml.org/fpml-5/confirmation" xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
    <header>
        <messageId messageIdScheme="http://www.calypso.com/clearlink/coding-scheme/message-id">112233</messageId>
```

- MessageAddressScheme

The 'messageAddressScheme' value is used to define the messageAddressScheme attribute within the <sentBy> and <sentTo> tags in the < header> section of the requestConsent FpML document.

```
<header>
    <messageId messageIdScheme="http://www.calypso.com/clearlink/coding-scheme/message-id">112233</messageId>
    <sentBy messageAddressScheme="http://www.calypso.com/clearlink/coding-scheme/messageAddress">CALYPSOXXX</sentBy>
    <sendTo messageAddressScheme="http://www.calypso.com/clearlink/coding-scheme/messageAddress">MEGA1234</sendTo>
    <creationTimestamp>0113-07-01T14:53:07</creationTimestamp>
</header>
```

- CorrelationIdScheme

The 'correlationIdScheme' value is used to define the correlationIdScheme attribute within the <correlationId> tag in the requestConsent FpML document.

```
</header>
<isCorrection>false</isCorrection>
<correlationId correlationIdScheme="http://www.calypso.com/clearlink/coding-scheme/correlation-id">776645</correlationId>
<sequenceNumber>1</sequenceNumber>
```

- RequestedActionScheme and OriginatingEventScheme

The 'requestedActionScheme' value is used to define the requestedActionScheme attribute within the <requestedAction> tag in the requestConsent FpML document.

The 'originatingEventScheme' value is used to define the originatingEventScheme attribute within the <originatingEvent> tag in the requestConsent FpML document.

```
<sequenceNumber>1</sequenceNumber>
<requestedAction requestedActionScheme="http://www.calypso.com/clearlink/coding-scheme/requestedAction">Clearing</requestedAction>
<originatingEvent originatingEventScheme="http://www.calypso.com/clearlink/coding-scheme/originatingEvent">Trade</originatingEvent>
<trade>
```

- ReasonCodeScheme

The 'ReasonCodeScheme'value is used to define the 'ReasonCodeScheme' attribute within the <reasonCode> tag in the Exception FpML.

```
    <reason>
        <reasonCode reasonCodeScheme="http://www.calypso.com/clearlink/coding-scheme/reason-code">Code-1</reasonCode>
        <description>Invalid Data</description>
    </reason>
    <reason>
        <reasonCode reasonCodeScheme="http://www.calypso.com/clearlink/coding-scheme/reason-code">Code-2</reasonCode>
        <description>Error in Clearing</description>
    </reason>
</consentException>
```

2.2 Trade Configuration

This section describes any additional trade setup required for the translation of specific trade fields.

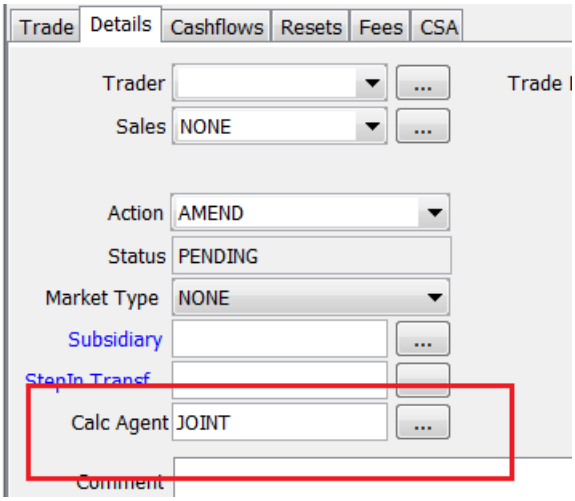
For the most part, it should be obvious to users of the FpML API which GUI fields map to which FpML tags. However, in certain cases FpML requires a specific domain of values to be used for a particular FpML tag, and the Calypso GUI either does not support this as a domain or does not support the field in the same way.

For such cases, we have determined a 'default' way that users can specify these values on the Calypso trade so that they can be properly translated by the FpML API.

2.2.1 Cash Settle Info

Calculation Agent

For the Cash Settle Info calculation agent, we use the Calc Agent field from the trade screen Details tab.



In FpML this field can either point to the PO or CP party references, or a specific set of hard-coded values.

In Calypso, the Calc Agent field can contain either a Legal Entity short code, or a manually entered value. This fits well with the FpML valid values.

The following are the valid values for the Calc Agent field, to be properly translated to FpML:

- 'JOINT' (both the PO and CP parties are the calculation agent)
- The actual LE shortname of the PO or CP (if only one or the other are the calculation agent)
- 'ExercisingParty'
- 'NonExercisingParty'
- 'AsSpecifiedInMasterAgreement'
- 'AsSpecifiedInStandardTermsSupplement'

Below are examples of how 'JOINT' and 'ExercisingParty' would each be translated into FpML.

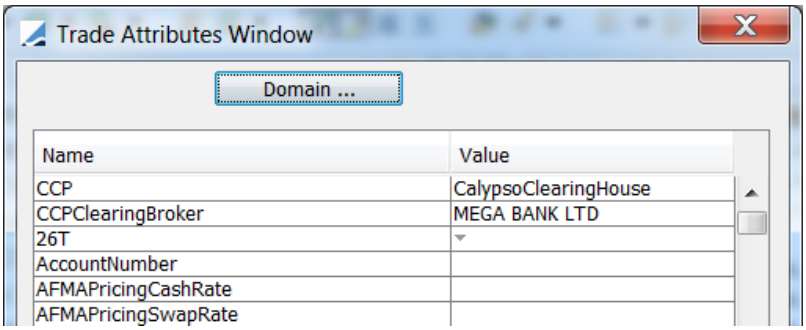
JOINT

```
<calculationAgent>
    <calculationAgentPartyReference href="partyA"/>
    <calculationAgentPartyReference href="partyB"/>
</calculationAgent>
```

ExercisingParty

```
<calculationAgent>
    <calculationAgentParty>ExercisingParty</calculationAgentParty>
</calculationAgent>
```

2.2.2 Trade Keywords



<sentBy> and <sendTo> tags in the <header> section of envelope FpML e.g. requestConsent etc. are populated from the trade keywords.

For populating <sentBy>, value of the keyword *'CCP'* from the trade is used to lookup a Calypso LE with that short code. If found, it will use the value of the specified partyIdAttribute on that Calypso LE as the value for the tag.

For populating < sendTo >, value of the keyword *'CCPClearingBroker'* from the trade is used to lookup a Calypso LE with that short code. If found, it will use the value of the specified partyIdAttribute on that Calypso LE as the value for the tag.

If no attribute or LE are found, it will use the keyword value instead. If no keyword with that name exists, it will add a warning to the exceptions vector.

```
<header>
    <messageId messageIdScheme="http://www.calypso.com/clearlink/coding-scheme/messageAddress">111111</messageId>
    <sentBy messageAddressScheme="http://www.calypso.com/clearlink/coding-scheme/messageAddress">CALYPSOXXX</sentBy>
    <sendTo messageAddressScheme="http://www.calypso.com/clearlink/coding-scheme/messageAddress">MEGA1234</sendTo>
    <creationTimestamp>2013-08-27T20:49:06.001Z</creationTimestamp>
</header>
```

FpML API Usage

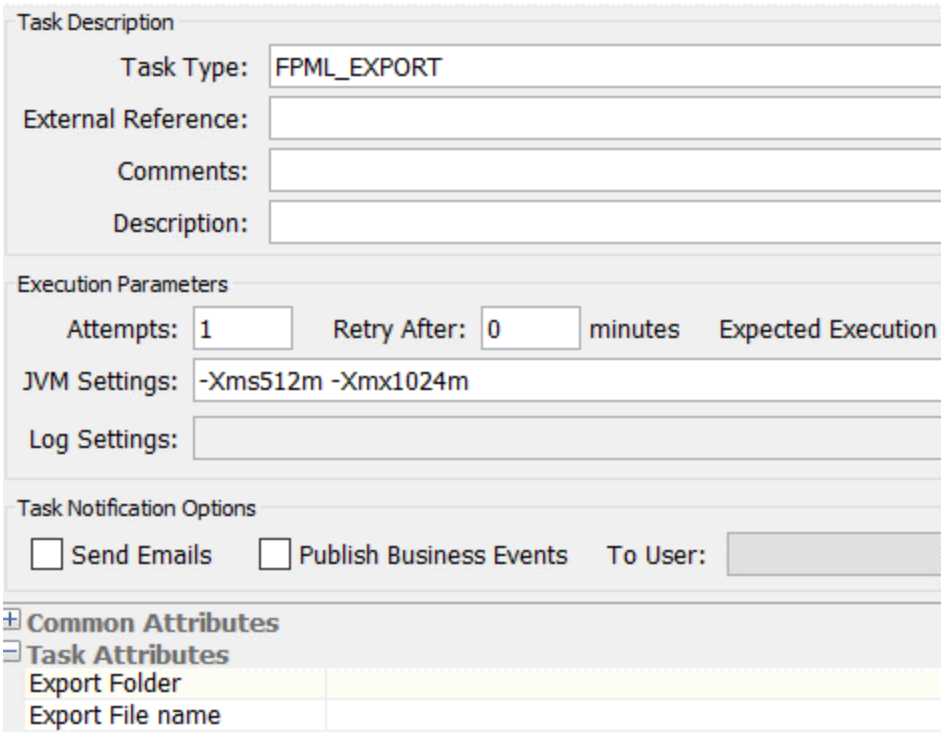
This section describes the actual usage of the FpML API.

3.1 Exporting Trades

You can export trades using the FPML_EXPORT scheduled task or using the FpML REST APIs.

For info on the REST APIs, please refer to Calypso REST APIs documentation.

3.1.1 FPML_EXPORT Scheduled Task



Export Folder - Enter the path where the report will be generated.

Export File name - Enter the file name of the generated report.

The FpML API produces only the Data Document portion of the FpML document, which includes the trade (tradeHeader & product) and party sections of the FpML standard.

Each interface will add their particular FpML envelope wrapped around the Data Document and add any additional customizations they may require for the tradeHeader and party tags.

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
- <FpML xmlns="http://www.fpml.org/FpML-5/confirmation" xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" fpmlVersion="5-3">
  - <trade>
    - <tradeHeader>
      + <partyTradeIdentifier>
      + <partyTradeIdentifier>
        <tradeDate>2013-06-12</tradeDate>
      </tradeHeader>
    + <swap>
    </trade>
    <party id="partyA"/>
    <party id="partyB"/>
</FpML>
```


3.1.2 FpMLUtil Wrapper Class

The FpML API provides the wrapper class 'FpMLUtil' for making calls to the API for creating FpML from a Calypso Trade. The following two functions are provided by this class (the specifics of each parameter are described in subsequent sections):

```
/**
 * This method returns a FpML string for the Calypso Trade passed in.
 * @param calypsoTrade - trade object which is translated to FpML
 * @param fpmlMap - Hash Map pre-populated with required arguments
 * @param interfaceName - interface name used for mapping
 * @param fpmlVersion - version of FpML to create
 * @param exception - UploadBOException vector to add exceptions in translation
 * @return
 */

String translate(Trade calypsoTrade, HashMap<String, Object> fpmlMap,  String interfaceName, String fpmlVersion,
Vector<UploadBOException> exception);

/**
 * This method returns a FpML JAXB object for the Calypso Trade passed in.
 * @param calypsoTrade - trade object which is translated to FpML
 * @param fpmlMap – Hash Map pre-populated with required arguments
 * @param interfaceName - interface name used for mapping
 * @param fpmlVersion - version of FpML to create
 * @param unmarshaller - Unmarshallar object required to generate a JAXB object from a FpML string
 * @param exception - UploadBOException vector to add exceptions in translation
 * @return
 */

Object translate(Trade calypsoTrade, HashMap<String, Object> fpmlMap, String interfaceName, String fpmlVersion,
Unmarshaller unmarshaller, Vector<UploadBOException> exception);
```

FpML Map

Whereas the 'Translator' mapping type is for values which remain the same across all trades being translated for a particular interface, the FpML Map (Hash Map) is for values which change for each trade being translated.

The FpML Map should be pre-populated with values for the keys defined in the 'FpMLMapOutgoingConstants' wrapper class:

```
/**
* constant to set PO HRef in the FpML
*/
public static final String FPML_MAP_POHREF = "POHRef";


/**
* constant to set CP HRef in the FpML
*/
public static final String FPML_MAP_CPHREF = "CPHRef";
```

The 'POHRef' and 'CPHRef' values are used to define the HRef IDs to be used for the Calypso Trade PO and CP, when setting the party reference tags required in the FpML.

```
        </swapStream>
        <swapStream id="floatingLeg">
            <payerPartyReference href="partyB"/>
            <receiverPartyReference href="partyA"/>
            <calculationPeriodDates id="floatingCalcPeriodDates">
                <effectiveDate>
                    <unadjustedDate>2013-06-05</unadjustedDate>
                    <dateAdjustments>

    <party id="partyA">
        <partyId partyIdScheme="http://www.calypso.com/spec/2001/party-id-1-0">CALYPXXXX</partyId>
    </party>
    <party id="partyB">
        <partyId partyIdScheme="http://www.calypso.com/spec/2001/party-id-1-0">SWAP1234</partyId>
    </party>
```

UploadBOException

The UploadBOException Vector *must* be initialized prior to calling the FpML API, so that the API can populate it with exceptions that arise during translation.

After the translate function returns, the caller should check the Vector to see if any exceptions were generated. If there are, the caller is free to deal with them however they wish. Typical uses could include publishing the

exceptions to the Task Station, if the interface is already using workflow messages to track its progress, or the caller can loop through the exceptions and test for error codes that affect their use case.

> **Note:** Type and Source are set to default values on the exception; you will want to customize those values when publishing the exception to the Task Station so that it is grouped with your workflow type.

Unmarshaller

If the caller is using jaxb for their internal logic, it is assumed that they will be using their own jaxb classes generated from their particular schema, which can be based on FpML extensions. Hence, the unmarshaller being passed in must be created from the caller's JAXBContext which has been initialized to their jaxb packages.

> **Note:** The FpML API will aquire a lock on the unmarshaller when performing the actual unmarshal, in case it has been declared static by the caller.

Please note that because the FpML API produces just the Data Document portion of the FpML, your JAXB object must define the Data Document as an acceptable root tag. This can be accomplished using a binding file when generated the JAXB. A sample of how this can be done is included in Appendix A.

In addition, if you find that when marshaling from your JAXB object to a string, the namespace prefix for the FpML tags is being set to ns1 or ns2, you will need to define a binding to force the namespace prefix to be empty for the FpML schema, which is the standard. Please see Appendix A for more details.

3.1.3 Logging

To see logging messages for the FpML API, you need to set the following log categories:

- UPLOADER: Set this to see logging for the Data Uploader utility functions used by the FpML API.
- CalypsoToFpML: Set this to see logging for the FpML API translation from the Calypso Trade object to FpML.

3.1.4 Sample Code

The following sample code shows a typical usage of the FpML API to generate a FpML string or JAXB object from a Calypso Trade. It is assumed in this sample that you have already retrieved the Calypso Trade object from the Calypso Data Server.

```
// Code above this has retrieved the Calypso Trade and assigned it to a
// variable named calypsoTrade of type com.calypso.tk.core.Trade


try {


// Interface Name
String interfaceName = "MW";
```

```
// FpML version
String fpmlVersion = "5-3";

// Populate FpML Map (Hash Map) with required values
HashMap<String, Object> fpmlMap = new HashMap<String, Object>();
fpmlMap.put(FpMLMapOutgoingConstants.FPML_MAP_POHREF, "partyA");
fpmlMap.put(FpMLMapOutgoingConstants.FPML_MAP_CPHREF, "partyB");



// *** FpML String ***

// Initialize UploadBOException Vector
Vector<UploadBOException> exceptionsForString = new Vector<UploadBOException>();

// Call String translate function
String strFpml53 = FpMLUtil.translateCalypsoToFpml(calypsoTrade, fpmlMap, interfaceName,
fpmlVersion, exceptionsForString);

// Check the UploadBOException Vector in case of translation errors
       if( !Util.isEmpty(exceptionsForString) ) {
              Log.debug(LOG, "Errors in translating trade to FpML 5-3 string.");
       } else {
              Log.debug(LOG, "Successfully translated trade to FpML 5-3 string.");
}

// *** FpML JAXB ***

// Initialize UploadBOException Vector
Vector<UploadBOException> exceptionsForJaxb = new Vector<UploadBOException>();

// Call Object translate function
DataDocument fpmlDoc53 = (DataDocument) FpMLUtil.translateCalypsoToFpml(calypsoTrade, fpmlMap,
interfaceName, fpmlVersion, myJaxbContext.createUnmarshaller(), exceptionsForJaxb);

// Check the UploadBOException Vector in case of translation errors
```

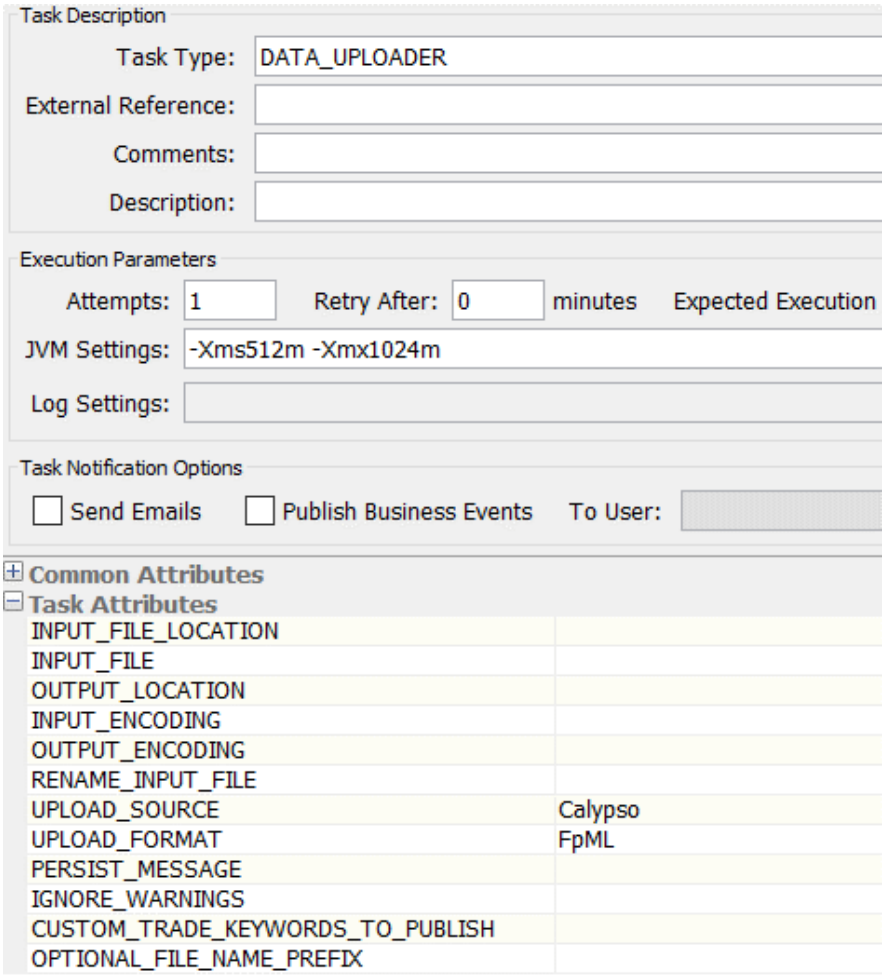
```
        if( !Util.isEmpty(exceptionsForJaxb) ) {
                Log.debug(LOG, "Errors in translating trade to FpML 5-3 jaxb.");
        } else {
                Log.debug(LOG, "Successfully translated trade to FpML 5-3 jaxb.");
}
} catch (Exception e) {
        Log.error(LOG, "Error occurred running test program. See log for more details.", e);
}
```

3.2 Importing Trades

You can import trades using the DATA_UPLOADER scheduled task, using the Data Uploader window or using the FpML REST APIs.

For info on the REST APIs, please refer to Calypso REST APIs documentation.

3.2.1 DATA_UPLOADER Scheduled Task



Task Attributes

- INPUT_FILE_LOCATION – Location of the file to be imported.
- INPUT_FILE – File name.
- OUTPUT_LOCATION – Location of the converted file.
- INPUT_ENCODING – Encoding scheme to be used for the file to be imported.
- OUTPUT_ENCODING – Encoding scheme to be used for the converted file.
- RENAME_INPUT_FILE – Select true to rename the imported file so that you know it has been imported.
- UPLOAD_SOURCE – Select the source of the file to be imported.
- UPLOAD_FORMAT – Select the format of the file to be imported.

You can also enter regular expressions for the attributes INPUT_FILE_LOCATION, INPUT_FILE, OUTPUT_LOCATION.

Examples:

INPUT_FILE_LOCATION = ${user.home}\datauploader\input

INPUT_FILE = Futures_[0-9]{1}.xml

OUTPUT_LOCATION = ${user.home}\datauploader\output

Upon import, the following files are created:

In INPUT_FILE_LOCATION – "<file name>.finished" to indicate that the file has been imported.

In OUTPUT_LOCATION:

- "<fine name>_result.xml" to show the items successful imported.
- "<file name>_rejected.xml" to show the rejected items.

The rejected file is created if Publishers=File,RejectedFile in "datauploader.properties".

3.2.2 Data Uploader Window

Source – Select the source of the file.

Format - Based on the selected source, format can be selected accordingly.

Appendix A: JAXB Customizations

This section contains details on how to customize the JAXB classes produced by the XJC compiler, which will be required if you wish to use JAXB with the results returned by the FpML API.

The details in this section assume you are using the XJC compiler on the command line to produce your JAXB files. Please adapt accordingly if you are using a build tool such as maven or gradle.

Note: There is a known issue with gradle that you must force the XJC compilation step to use the maven jaxb2 plugin so that it picks up the correct version of JAXB (2.2.5 or higher) to generate the correct bindings as specified in the binding file.

4.1 Prerequisites

For both customizations listed below, you will need the following jars on the classpath when generating the JAXB classes:

- jaxb2-basics-<version>.jar
- jaxb2-basics-runtime-<version>.jar
- jaxb2-basics-tools-<version>.jar
- commons-beanutils-<version>.jar
- commons-lang-<version>.jar
- commons-logging-<version>.jar

4.2 Data Document Customization

As mentioned previously in this document, because the FpML API produces just the Data Document portion of the FpML, your JAXB object must define the Data Document as an acceptable root tag. To do so you will need three additional pieces.

4.2.1 Binding File

You will need to create a binding file with the details below. The important details to note are the 'annox' namespace defined, as well as the 'annox' tags. Please set the schemaLocation value to the name of the xsd file where the Data Document element is defined.

```
<?xml version="1.0" encoding="US-ASCII" ?>
<jxb:bindings version="1.0" xmlns:jxb="http://java.sun.com/xml/ns/jaxb"
              xmlns:xs="http://www.w3.org/2001/XMLSchema"
              xmlns:xjc="http://java.sun.com/xml/ns/jaxb/xjc"
              xmlns:annox="http://annox.dev.java.net"
              jxb:extensionBindingPrefixes="xjc">

    <jxb:bindings schemaLocation="fpml-doc-5-3.xsd" node="/xs:schema">
        <!-- Annotate the following classes with XmlRootElement -->
        <jxb:bindings node="xs:complexType[@name='DataDocument']">
            <annox:annotate>
                <annox:annotate annox:class="javax.xml.bind.annotation.XmlRootElement" name="FpML" />
            </annox:annotate>
        </jxb:bindings>
    </jxb:bindings>

</jxb:bindings>
```

4.2.2 Dependencies

You will need to add two additional jars to the classpath:

- jaxb2-basics-annotate-<version>.jar
- annox-<version>.jar

4.2.3 XJC Command Line

On the XJC command line, you will need to add the following arguments:

- '-extension'
- '-Xannotate'
- Specify the binding file as '-b <binding file>'

4.3 FpML Namespace Customization

As mentioned previously in this document, if you experience namespace issues when marshaling your jaxb to a string, you will need to force the namespace prefix to be empty for the FpML schema. To do so you will need three additional pieces.

4.3.1 Binding File

You will need to create a binding file with the details below. [This binding file builds on the one described in the Data Document section above.] The important details to note are the 'namespace' namespace defined, as well as the 'namespace' tag. Again, please set the schemaLocation value to the name of the xsd file where the FpML schema is defined.

```
<?xml version="1.0" encoding="US-ASCII" ?>
<jxb:bindings version="1.0" xmlns:jxb="http://java.sun.com/xml/ns/jaxb"
              xmlns:xs="http://www.w3.org/2001/XMLSchema"
              xmlns:xjc="http://java.sun.com/xml/ns/jaxb/xjc"
              xmlns:annox="http://annox.dev.java.net"
              xmlns:namespace="http://jaxb2-commons.dev.java.net/namespace-prefix"
              jxb:extensionBindingPrefixes="xjc">

    <jxb:bindings schemaLocation="fpml-doc-5-3.xsd" node="/xs:schema">
        <!-- Force the namespace prefix to be empty -->
        <jxb:bindings>
            <namespace:prefix name="" />
        </jxb:bindings>

        <!-- Annotate the following classes with XmlRootElement -->
        <jxb:bindings node="xs:complexType[@name='DataDocument']">
            <annox:annotate>
                <annox:annotate annox:class="javax.xml.bind.annotation.XmlRootElement" name="FpML" />
            </annox:annotate>
        </jxb:bindings>
    </jxb:bindings>

</jxb:bindings>
```

4.3.2 Dependencies

You will need to add one additional jar to the classpath:

- jaxb2-namespace-prefix-<version>.jar

4.3.3 XJC Command Line

On the XJC command line, you will need to add the following arguments:

- '-extension'
- '-Xnamespace-prefix'
- Specify the binding file as '-b <binding file>'

4.4 Envelope Support in FpML

FpML Utility supports Envelopes (Messages) for business processes in FpML Messages.

Currently following envelopes are supported in FpML Utility –

- requestConsent
- consentAcknowledgement
- consentException

- clearingAcknowledgement
- clearingConfirmed
- clearingRefused

FpML Map

Additional data needs to be specified in FpML Map for the Envelope

```
     /**
       * constant to set Envelope Type in the FpML
       */
       public static final String FPML_MAP_ENVELOPE_TYPE = "EnvelopeType";


       You need to set 'EnvelopeType' in the Map by using predefined values from the class
       EnvelopeType  e.g.


       fpmlMap.put(FpMLMapOutgoingConstants.FPML_MAP_ENVELOPE_TYPE,
                           EnvelopeType.REQUEST_CONSENT);
       *Note* if FPML_MAP_ENVELOPE_TYPE property is not specified in the FpML Map, by  default
DataDocument Fpml will be returned back.



       /**
        * constant to set Message Id in Message Header in envelope the FpML
        */
       public static final String FPML_MAP_MESSAGE_ID = "MessageId";



       /**
        * constant to set ReplyTo Message Id in Message Header in envelope              * FpML
        */
       public static final String FPML_MAP_REPLYTO_MESSAGE_ID =
                                       "ReplyToMessageId";



       /**
        * constant to set Message Id in Message Header in envelope the FpML
```

```
        */
       public static final String FPML_MAP_CORRELATION_ID = "CorrelationId";



       /**
        * constant to exclude trade details from envelope FpML
        * Must be set to a variable of type Boolean
        */
       public static final String FPML_MAP_EXCLUDE_TRADE_DETAILS =
                                  "ExcludeTradeDetails";



       /**
        * constant to set ReasonMap<ReasonCode, Description> in Exception                * envelope
FpML
        * Must be set to a variable of type HashMap<String, String>
        * In this HashMap, key is 'ReasonCode' and value is 'Description'
        */
       public static final String FPML_MAP_REASON_CODE_MAP = "ReasonCodeMap";
```

4.5 FpML Extension

The following business lifecycle objects are supported in FpML.

- Request Consent
- Clearing Confirmed
- Clearing Refused
- Clearing Status

The Calypso version of each object is created in the Calypso FpML schema to provide the following features.

- Ability to pass trade keywords.
- Ability to pass custom attributes (name value pairs to be used in translation)
- Ability to pass mapping source (same FpML can be mapped based on the source passed here)

```
<xsd:complexType name="RequestConsent">
    <xsd:complexContent>
        <xsd:extension base="RequestConsent">
            <xsd:sequence>
                <xsd:element name="CalypsoCustomData" type="calypso:CalypsoCustomData" minOccurs="0" maxOccurs="1">
                    <xsd:annotation>
                        <xsd:documentation>to pass attributes via the fpml for customization</xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

CalypsoCustomData contains provision to pass Trade Keywords and Attributes and the Mapping Source.

```
<xsd:complexType name="CalypsoCustomData">
    <xsd:sequence>
        <xsd:element name="Source" type="xsd:string"/>
        <xsd:element name="TradeKeywords" type="calypso:TradeKeywords" maxOccurs="1"/>
        <xsd:element name="Attributes" type="calypso:Attributes" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="Attributes">
    <xsd:sequence>
        <xsd:element name="Attribute" type="calypso:Attribute" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="TradeKeywords">
    <xsd:sequence>
        <xsd:element name="Keyword" type="calypso:Attribute" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="Attribute">
    <xsd:sequence>
        <xsd:element name="Name" type="xsd:string"/>
        <xsd:element name="Value" type="xsd:string"/>
    </xsd:sequence>
```

The FpML API is designed in such a way to provide support to both the business objects.

- Plain Vanilla FpML (a message purely based on the FpML Schema without any extensions)

```
<?xml version="1.0" encoding="UTF-8"?>
- <requestConsent xmlns:lch="http://xsd.swapclear.com/clearlink-1-0/confirmation" fpmlVersion="5-2" xmlns:xsi="http://www.w3.org/2001/
  - <header>
      <messageId messageIdScheme="http://www.lchclearnet.com/clearlink/coding-scheme/message-id">#ID#_1</messageId>
      <sentBy messageAddressScheme="http://www.lchclearnet.com/clearlink/coding-scheme/party-id">LCHLGB2L</sentBy>
      <sendTo messageAddressScheme="http://www.lchclearnet.com/clearlink/coding-scheme/party-id">FCM2</sendTo>
      <creationTimestamp>2012-02-22T14:52:16.169+08:00</creationTimestamp>
    </header>
    <isCorrection>false</isCorrection>
    <correlationId correlationIdScheme="http://SEF1/clearlink/coding-scheme/correlation-id">E14_#ID#_RF</correlationId>
    <sequenceNumber>1</sequenceNumber>
    <requestedAction>Clearing</requestedAction>
    <originatingEvent>Trade</originatingEvent>
  - <trade xmlns="http://www.fpml.org/FpML-5/confirmation" xmlns:ns2="http://www.w3.org/2000/09/xmldsig#">
    - <tradeHeader>
      - <partyTradeIdentifier>
          <partyReference href="counterpartyA"/>
          <tradeId tradeIdScheme="http://www.lchclearnet.com/clearlink/coding-scheme/trade-id">E14_CTPTY_#ID#_RF</tradeId>
```

- Calypso FpML (a message which is based on the schema definition defined above)

```
<?xml version="1.0" encoding="utf-8" ?>
- <calypso:requestConsent xsi:type="calypso:RequestConsent" fpmlVersion="5-0" xmlns="http://www.fpml.org/FpML-5/confirmation" xmlns:calypso="http://www.calypso.com/otc-clearing/confirmation" xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  - <header>
      <messageId messageIdScheme="http://www.clearingsvc.com/messageId/Calypso">123</messageId>
      <sentBy messageAddressScheme="http://www.clearingsvc.com/partyId">ANONYMOUS</sentBy>
      <sendTo messageAddressScheme="http://www.clearingsvc.com/partyId">ABC</sendTo>
      <creationTimestamp>2006-01-01T09:01:00+05:00</creationTimestamp>
    </header>
    <isCorrection>false</isCorrection>
    <correlationId correlationIdScheme="http://www.clearingsvc.com/conversationId/OTC">456</correlationId>
    <sequenceNumber>1</sequenceNumber>
    <requestedAction>Clearing</requestedAction>
    <originatingEvent originatingEventScheme="http://www.fpml.org/coding-scheme/originating-event">Trade</originatingEvent>
  - <trade>
    - <tradeHeader>
      - <partyTradeIdentifier>
          <issuer issuerIdScheme="http://www.fpml.org/coding-scheme/external/cftc/issuer-identifier">1010000051</issuer>
          <tradeId tradeIdScheme="http://www.fpml.org/coding-scheme/external/unique-transaction-identifier">SW100000000LCH000060708231704125</tradeId>
        </partyTradeIdentifier>
```

Which contains the Calypso custom data as follows:

```
- <calypso:CalypsoCustomData>
    <calypso:Source>FpML</calypso:Source>
  - <calypso:TradeKeywords>
    - <calypso:Keyword>
        <calypso:KeywordName>name1</calypso:KeywordName>
        <calypso:KeywordValue>value1</calypso:KeywordValue>
      </calypso:Keyword>
    - <calypso:Keyword>
        <calypso:KeywordName>name2</calypso:KeywordName>
        <calypso:KeywordValue>value2</calypso:KeywordValue>
      </calypso:Keyword>
    </calypso:TradeKeywords>
  </calypso:CalypsoCustomData>
</calypso:requestConsent>
```

The following FpML schemes should be used when using the Calypso version of the FpML message.

Scheme Name	Scheme Value	Description
messageId	http://www.fpml.org/coding-scheme/message-id <messageId messageIdScheme="http://www.fpml.org/coding-scheme/message-id">123</messageId>	Saved as Trade Keyword
sentBy	http://www.fpml.org/coding-scheme/party-id <sentBy messageAddressScheme="http://www.fpml.org/coding-scheme/party-id">ANONYMOUS</sentBy>	Saved as Trade Keyword
sendTo	http://www.fpml.org/coding-scheme/party-id <sendTo messageAddressScheme="http://www.fpml.org/coding-scheme/party-id">ABC</sendTo>	Saved as Trade Keyword
CorrelationId	http://www.fpml.org/coding-scheme/correlation-id <correlationId correlationIdScheme="http://www.fpml.org/coding-scheme/correlation-id">456</correlationId>	Saved as Trade Keyword

Scheme Name	Scheme Value	Description
tradeId	http://www.fpml.org/coding-scheme/trade-id <partyTradeIdentifier> <partyReference href="clearing-svc" /> <tradeId tradeIdScheme="http://www.fpml.org/coding-scheme/trade-id">SVC_001</tradeId> </partyTradeIdentifier>	Saved as Trade External Reference
partyReference	clearing-svc <party id="clearing-svc">	Treated as Counterparty of the trade
partyReference	member-firm <party id="member-firm">	Treated as Processing Org of the trade
OriginCode Category Scheme	http://www.fpml.org/coding-scheme/org-type-category <category categoryScheme="http://www.fpml.org/coding-scheme/org-type-category">	Saved as Trade Keyword
PriorUSI issuerIdScheme	http://www.fpml.org/coding-scheme/external/cftc/issuer-identifier <issuer issuerIdScheme="http://www.fpml.org/coding-scheme/external/cftc/issuer-identifier">1013456789</issuer>	Saved as Trade Keyword
USI tradeIdScheme	http://www.fpml.org/coding-scheme/external/unique-transaction-identifier <tradeId tradeIdScheme="http://www.fpml.org/coding-scheme/external/unique-transaction-identifier">LCH_USI_1</tradeId>	Saved as Trade Keyword
Account accountIdScheme	http://www.fpml.org/coding-scheme/account-Id <account id="account1"> <accountId accountIdScheme="http://www.fpml.org/coding-scheme/account-Id">AAAA</accountId> <!-- change this for account --> <servicingParty href="member_firm"/> </account>	Used to identify the position account and derive the book to be used.
OriginatingEvent	http://www.fpml.org/coding-scheme/originating-event <originatingEvent originatingEventScheme="http://www.fpml.org/coding-scheme/originating-event">Trade</originatingEvent>	Used to determine netting / terminate