



Financial products Markup Language

# **FpML<sup>®</sup> 5 User Guide**

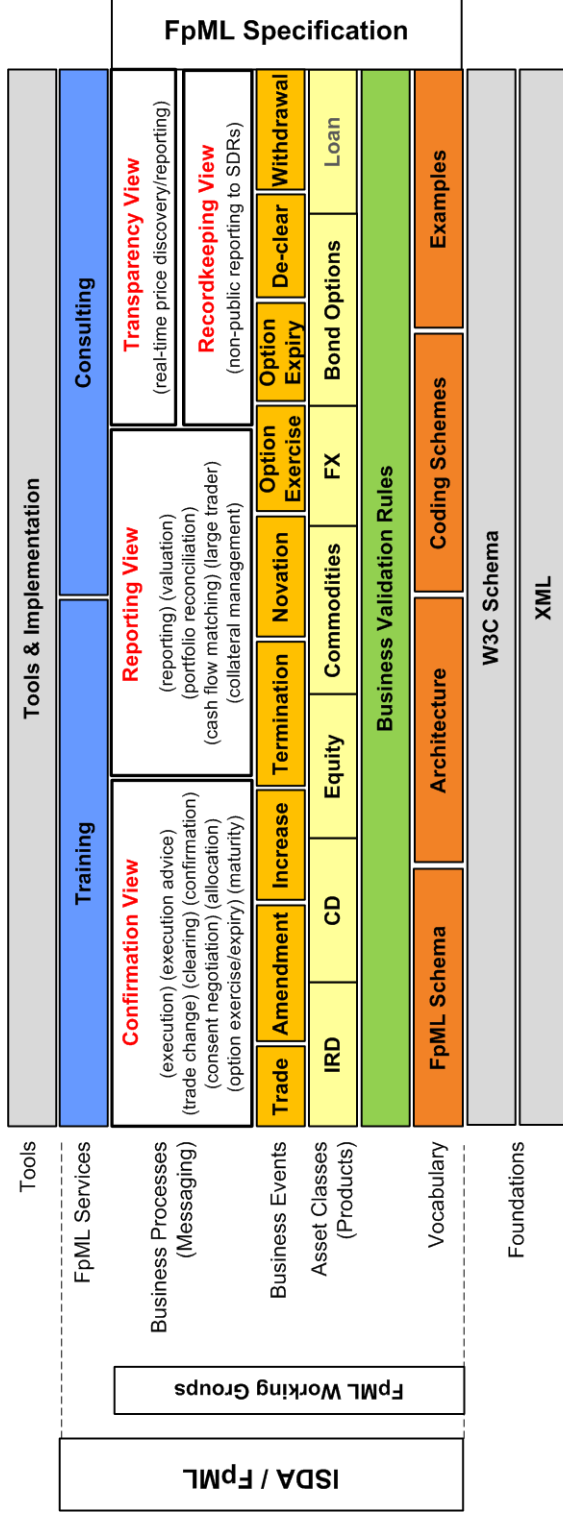
---

**2012 Edition**

**ISDA<sup>®</sup>** | Safe,  
Efficient  
Markets

# FpML at a Glance

The following diagram illustrates the scope of the FpML 5 specification in relation to various dimensions such as technology, industry coverage, governing bodies, and implementation activities.





Financial products Markup Language

# **FpML<sup>®</sup> 5 User Guide**

---

**2012 Edition**

**ISDA<sup>®</sup>** | Safe,  
Efficient  
Markets

# **FpML 5 User Guide 2012 Edition**

Published June 2012

Copyright © 2012 by

INTERNATIONAL SWAPS AND DERIVATIVES ASSOCIATION, INC.  
360 Madison Avenue, 16<sup>th</sup> floor  
New York, NY 10017

FpML® is a registered trademark of the International Swaps and Derivatives Association, Inc.  
ISDA® is a registered trademark of the International Swaps and Derivatives Association, Inc.

# Contents

1. Introduction.....	1
1.1. History of FpML .....	1
1.2. About this User Guide .....	1
1.2.1. Purpose .....	1
1.2.2. Audience .....	2
1.2.3. Content Overview .....	2
1.2.4. Applicability .....	2
1.2.1. Materials for Download .....	3
1.3. Drivers for FpML 5 .....	3
1.4. FpML Documentation .....	4
1.5. Related Resources .....	4
1.5.1. XML Documentation .....	5
1.5.2. XML Terminology and Tools .....	5
1.5.3. Web-based Resources .....	6
2. Key Changes Compared to FpML 4 .....	7
2.1. Main Functionality Changes.....	7
2.2. Views.....	7
2.3. The Use of Multiple Namespaces.....	9
2.4. Multiple Root Elements.....	9
2.5. Enhanced Messaging Framework.....	10
2.6. Other Improvements.....	11
2.6.1. Boolean .....	11
2.6.2. Underlyers.....	11
2.6.3. Refactoring FX Products.....	12
2.6.4. Payments.....	12
2.6.5. Removed Deprecated Structures.....	12
2.6.6. Removed Contract .....	12
2.6.7. Account and Roles – Removed tradeSide.....	13
2.6.8. Other Changes.....	13
2.7. Impact.....	14
2.7.1. Impact of Views.....	14
2.7.2. Impact on Extensions .....	14
2.7.3. Impact of Multiple Roots and Generic Processes .....	14
2.7.4. Migrating Messages from 4.x to 5.x .....	15
3. How to Use FpML – Sample Applications.....	17
3.1. Introduction .....	17
3.1.1. Purpose .....	17
3.1.2. Overview.....	17
3.2. Straight-Through Processing (STP) Data Transfer.....	18
3.2.1. Objective.....	18
3.2.2. Requirements .....	18
3.2.3. Messages and Structures Used.....	18
3.2.4. Tools and Technology.....	18
3.3. Automatic Confirmation.....	20
3.3.1. Objective.....	20
3.3.2. Requirements .....	20
3.3.3. Messages Used.....	20
3.3.4. Customizations / Restrictions .....	21
3.3.5. Validation .....	21
3.3.6. Tools and Technologies .....	21

## FpML 5 User Guide 2012 Edition

3.4.	Derivatives Clearing Organization .....	22
3.4.1.	Objective.....	22
3.4.2.	Requirements .....	22
3.4.3.	Messages Used.....	22
3.4.4.	Customizations / Restrictions .....	23
3.4.5.	Validation .....	23
3.4.6.	Tools and Technologies .....	23
3.5.	Internal Trade Archive .....	24
3.5.1.	Objective.....	24
3.5.2.	Structures Used.....	24
3.5.3.	Tools .....	24
3.5.4.	Issues/Notes .....	25
3.6.	External Trade Repository (for Regulatory Reporting).....	26
3.6.1.	Objective.....	26
3.6.2.	Requirements .....	26
3.6.3.	Views and Business Processes Used.....	26
3.6.4.	Tools .....	27
3.7.	Intra-day Activity Reconciliation .....	28
3.7.1.	Objective.....	28
3.7.2.	Messages Used.....	28
3.7.3.	Reconciliation Approaches .....	28
3.8.	Inventory Reconciliation .....	29
3.8.1.	Objective.....	29
3.8.2.	Messages/Structures Used .....	29
3.8.3.	Matching/Reconciliation Approaches.....	29
3.9.	Bulk Reporting Applications.....	30
3.9.1.	Objective.....	30
3.9.2.	Messages/Structures Used .....	30
3.9.3.	Tools and Technologies .....	30
4.	How to Write FpML – Examples.....	33
4.1.	Introduction .....	33
4.2.	Basic Example: Forward Payment .....	33
4.2.1.	Introduction.....	33
4.2.2.	Required Data Attributes .....	33
4.2.3.	Developing the FpML.....	34
4.3.	IR Swap Confirmation Message Example.....	39
4.3.1.	Introduction.....	39
4.3.2.	Required Data Attributes .....	39
4.3.3.	Developing the FpML.....	40
4.4.	Public Reporting of a Commodity Swap .....	49
4.4.1.	Introduction.....	49
4.4.2.	Required Data Attributes .....	49
4.4.3.	Developing the FpML.....	49
4.5.	Non-public Reporting of an Equity Option .....	52
4.5.1.	Introduction.....	52
4.5.2.	Required Data Attributes .....	52
4.5.3.	Developing the FpML.....	52
4.6.	Clearing of an FX forward .....	55
4.6.1.	Introduction.....	55
4.6.2.	Required Data Attributes .....	55
4.6.3.	Developing the FpML.....	55

## FpML 5 User Guide 2012 Edition

5.	The Organization of FpML .....	57
5.1.	Introduction .....	57
5.1.1.	Purpose .....	57
5.1.2.	Overview.....	57
5.2.	Architecture .....	57
5.2.1.	Architecture Principles.....	57
5.2.2.	Schema and Instance Documents.....	58
5.2.3.	Architectural Changes Between Versions.....	61
5.3.	Document / Messaging Framework.....	63
5.3.1.	Sample Message Flows.....	63
5.3.2.	Generic Messaging / Multi-Event Workflows .....	65
5.3.3.	Views .....	65
5.3.4.	More Information.....	65
5.4.	Other Top Level Structures .....	66
5.4.1.	Party.....	66
5.4.2.	Trade.....	66
5.4.3.	Trade Header.....	67
5.4.4.	Portfolio .....	68
5.5.	Building Block Components .....	68
5.5.1.	Currency and Location Related Components .....	69
5.5.2.	Date-related Components.....	70
5.5.3.	Payment .....	72
5.6.	FpML Validation Framework.....	72
5.6.1.	What Does Validation Add? .....	72
5.6.2.	What Information Does Validation Provide?.....	74
5.6.3.	How Does Validation Work Together with the Specification?.....	76
6.	The FpML Product Framework .....	77
6.1.	Introduction .....	77
6.2.	Derivative Products .....	77
6.2.1.	Product Substitution Framework .....	77
6.2.2.	Product Summary .....	79
6.2.3.	Adding New Products .....	80
6.2.4.	ISDA Product Taxonomy .....	80
6.3.	Underlying Assets .....	81
6.3.1.	Usage .....	81
6.3.2.	Underlying Asset Substitution Framework.....	81
6.3.3.	Summary of Underlying Assets .....	82
7.	The FpML Messaging Framework .....	83
7.1.	Messages .....	83
7.1.2.	Message Naming Convention / Patterns .....	84
7.1.3.	Message Correlation .....	85
7.2.	Business Processes .....	85
7.3.	Views.....	86
7.4.	The Use of Multiple Namespaces.....	87
7.5.	Multiple Root Elements.....	87
7.6.	Generic Messaging / Multi-Event Workflows.....	88
7.7.	Generic (Multi-Event) Flows .....	89
7.8.	Pre-Trade.....	89
7.9.	Confirmation .....	89
7.10.	Reporting .....	90
7.11.	Recordkeeping.....	91
7.12.	Transparency .....	91

## FpML 5 User Guide 2012 Edition

7.13.	Shared Messages .....	91
7.14.	Collateral Management .....	92
7.15.	Loan Syndication.....	92
8.	Customizing FpML.....	93
8.1.	Introduction .....	93
8.1.1.	Purpose of the Section .....	93
8.1.2.	Overview.....	93
8.2.	Wrapping .....	93
8.2.1.	Explanation .....	93
8.2.2.	Advantages and Disadvantages.....	94
8.3.	Extending Type Content.....	94
8.3.1.	Example .....	95
8.3.2.	Advantages and Disadvantages.....	96
8.4.	Restricting Type Content.....	96
8.5.	Extending Product Coverage.....	97
8.5.1.	Create a New Type .....	97
8.6.	Extending an Existing Product .....	98
8.7.	Extending Messages .....	99
8.8.	Versioning and Version Migration .....	100



## Copyright Notice

This ISDA document is protected by Copyright Law. No electronic or hard copy document may be reproduced, photocopied or distributed electronically. Contact the ISDA Legal Department at [isda@isda.org](mailto:isda@isda.org) or +1-212-901-6000 for more information.

Additional copies of this user guide may be obtained from the ISDA website [www.isda.org](http://www.isda.org), under the “Bookstore”.

# FpML 5 User Guide 2012 Edition

# 1. Introduction

## 1.1. History of FpML

From the early 1980s when the interest rate swaps market began developing, the privately negotiated derivatives have grown tremendously in volume. According to the report on Derivatives Market Activity from the Bank for International Settlements, the notional principal outstanding of swaps and other over-the-counter (OTC) derivatives, stood at \$95 trillion in 2000, doubled by the end of 2003 (\$197 trillion), and reached \$708 trillion in June 2011. This corresponds to a 20% annualized growth rate over this eleven-year period.

To lower the cost of processing derivatives and thereby increase the profitability of the business, JP Morgan, in 1997, established a research project to develop the methodology by which these instruments can be traded using e-commerce technologies. PricewaterhouseCoopers was brought on board as a resource, and in 1999 the organizations announced a draft standard for interest rate swaps. At that time, other industry firms were contacted and an independent organization – FpML.org – was formed to develop and promote the Financial products Markup Language (FpML) as an XML-based “lingua franca” for derivatives trading.

The FpML standard is freely licensed and is intended to automate the flow of information between derivatives participants, independent of the underlying software or hardware infrastructure, supporting activities related to these transactions.

On November 14, 2001 ISDA and FpML.org announced their intention to integrate the development process of the FpML standard into the ISDA organizational structure. This combined the organizational strengths of ISDA with FpML’s technology base and allowed the FpML standard to be leveraged using the membership base and experience ISDA has built up since its formation in the mid 80s. The change is an indication of the increased importance of operations, automation and straight-through processing for the ISDA membership.

Expansion of the FpML standard to new products (e.g., correlation swaps, commodities) and business areas (e.g., bulk position reporting, incremental regulatory reporting, clearing, collateral management, syndicated loan) and adoption by new users has continued under ISDA’s sponsorship. Because of the rapid expansion of the standard and applications of the standard, and because of the increasing number of newcomers to FpML, there has been a growing need to provide introductory material to help users new to FpML understand how to use the standard. This user guide to FpML is intended to address this need. In particular, this edition covers regulatory reporting and clearing functions that are needed to be implemented under various regulatory mandates.

## 1.2. About this User Guide

### 1.2.1. Purpose

The user guide to FpML provides guidance to implementers about how FpML may be used by derivative market participants. It suggests applications for FpML, describes at a high level how to write FpML, and provides guidance on related topics.

## FpML 5 User Guide 2012 Edition

The user guide is complementary to the FpML standard specification, and is not a replacement for that specification. It describes specific instances and examples of using FpML, where the FpML specification provides a comprehensive reference and full schema description. In case of any discrepancy between the documents, the FpML standard specification shall be regarded as correct.

### 1.2.2. Audience

This user guide is intended to be used primarily by technologists with some exposure to XML and some familiarity with OTC derivatives. Following are some suggestions on sections of particular interest to people in different roles:

- Technology managers: Section 3, *Application Examples* would be particularly relevant. Section 8, Customizing FpML may also be relevant.
- Business analysts/developers: Section 4, *Writing FpML* should be useful as an introduction to how FpML looks.
- Technical architects/standards developers: All sections, with a particular emphasis on sections 5 to 8.


### 1.2.3. Content Overview

The user guide is organized as follows:

- **Section 1** provides general information and how to best approach this user guide.
- **Section 2** contains information for people and firms looking to upgrade their implementation from FpML 4.
- **Section 3** contains examples of different applications that might use FpML, with a discussion about which features of FpML could be used. For each potential application, there are references to particularly relevant parts of the user guide and the FpML Specification, to help users quickly find related information.
- **Section 4** provides some examples of how to write simple FpML instance documents, to illustrate in basic terms how FpML is constructed.
- **Section 5** highlights some of FpML's key architectural underpinnings, and describes some cross-product components in FpML.
- **Section 6** briefly describes how FpML instruments are specified, and describes the instrument extension mechanism.
- **Section 7** covers support for business processes in FpML 5.
- **Section 8** introduces how to customize FpML for business-specific requirements.

### 1.2.4. Applicability

The FpML user guide has been developed for FpML 5. In most cases, concepts, applications, and examples described in the user guide could be supported by previous versions of FpML. The examples were developed based on version 5.3; some examples may work with earlier 4.x versions with modifications.

 FpML 4	A separate user guide (FpML 4 User Guide 2010 Edition) covers the latest changes introduced in version 4. An electronic version of the publication is available for download from the ISDA Bookstore ( <a href="http://www.isda.org">www.isda.org</a> ).
---	--

### 1.2.1. Materials for Download



The examples developed in this user guide and other materials are available for download at:  
<http://www.fpml.org/userguide/fpml5ug2012k7>

### 1.3. Drivers for FpML 5

*“The FpML organization has been looking at ways of improving the specification for some time. One of the things we have always encouraged within FpML is people sending us feedback on their implementations. Over time we have received a number of comments on the FpML issues list identifying features of the grammar which either made it harder to use or which people felt could do with some redesign.*

*A few years ago, we started the process of reviewing FpML, looking at it from the architectural, product, business process and messaging standpoints to see if we could come up with ways to address these issues. We came up with pragmatic solutions to some them. These solutions modify FpML slightly; new features have been introduced as well as some incompatibilities, but overall version 5 maintains much of the structure and conventions of previous versions.*

*We haven’t made needless changes to the definitions of products; we haven’t made FpML look considerably different from the way it does in FpML 4, but we believe the changes that we’ve introduced increase the consistency in the design of FpML. The way we go about defining products and messages is considerably improved. Within the business process protocols, where previously there were a few gaps and a few missing messages, we believe that the new pattern-based approach enables us to produce grammars in which there are no missing messages and where there’s a greater consistency between the message sets themselves (e.g., having learnt one, you are able to transfer knowledge on to the others).*

*Collectively, the changes made in version 5 prepare FpML for the next generation of challenges, for example, providing the new messages needed for the derivatives market to address regulatory changes.”*

Andrew Jacobs, chair of the FpML Architecture Working Group

FpML 5 introduces a number of technical changes. It eliminates the use of some complex XML features that people found confusing. Documents are simpler in their structure. Techniques and features were introduced that enable documents to have greater longevity. For example, with the new schemas it is possible to process documents against future compatible versions of the schema. This was not possible before because they were tightly bound to a specific schema. This feature is useful in the task of migrating across time or supporting multiple versions simultaneously.

Some of the technical changes introduced in version 5, such as the use of multiple root elements and namespaces, make FpML more aligned with existing financial standards such as ISO 20022 and FIXML. Convergence between financial standards will continue being a key factor in driving future development of FpML.

## FpML 5 User Guide 2012 Edition

The FpML Standard follows a strict set of *change control guidelines* that dictate the types of changes that are allowed in a particular version.

- Minor releases (e.g., FpML 4.8 and 4.9) are intended to be backward compatible and cannot include major changes (e.g., we cannot remove elements, we cannot change names). Architectural changes are limited because of the backward compatibility requirement.
- More significant changes are allowed in major releases (e.g., FpML 4.0 and 5.0). In a major version, functionality cannot be removed. From this perspective functionality is a superset of what was there before. However, introducing backward incompatible changes is permissible. That is the reason why the architectural and technical changes discussed above were introduced in FpML 5.0 as opposed to a minor 4.x release.

*Section 2 contains a detailed overview of the changes that have been highlighted in this section.*

### **1.4. FpML Documentation**

All published FpML documents can be found on the FpML website (<http://www.fpml.org>).

#### **FpML Specifications** (<http://www.fpml.org/spec>)

- The FpML specifications section contains all the versions of the FpML Specification, including Recommendations and Working Drafts.
- Documentation, schemas and examples are available for download. Note that for version 5.x, there are multiple sets of these for each version, one for each supported view.
- As of this writing, the most recent version FpML Recommendation is version 5.3. The examples in this user guide were developed using FpML 5.3. Version 5.3 introduces important changes that are mentioned in the user guide, where relevant.
- The FpML Specifications section requires a simple, free, one-time registration that grants full access to all published versions of FpML.
- The section also contains the FpML Architecture Specification, the normative reference for the architectural rules under which FpML 5 is developed. The latest version of the architecture is 3.0.

#### **FpML Documents** (<http://www.fpml.org/documents>)

- The FpML Documents section contains technical papers and proposals (e.g., validation, versioning, messaging, extensions)

Other information that can be found on the FpML website includes general background information on FpML schema, events, tools and consulting services. In addition, one can sign up for FpML announcements, find further information about the FpML Working Groups (<http://www.fpml.org/wgroup>), or access the online FpML Subversion Source libraries.

### **1.5. Related Resources**

In addition to the ISDA documentation, there are a variety of other resources that implementers can use to learn about FpML and related technologies. Some of these are described below.

### 1.5.1. XML Documentation

FpML is based on XML, the Extensible Markup Language. The FpML Specification and to a lesser degree the user guide assume familiarity with XML and with XML schema. The following are references to learn more about XML and XML Schema:

- XML
  - The XML specification: <http://w3.org/XML/Core/#Publications>
  - XML tutorial: <http://www.w3schools.com/xml/default.asp>
  - The Guide to the XML Galaxy: <http://www.zvon.org/comp/r/tut-XML.html#>
  - An index of XML publications: <http://www.oasis-open.org/cover/xml.html>
  - Brief article by an XML expert on how XML works: <http://www.xml.com/pub/a/w3j/s3.walsh.html>
  - Two-page quick reference guide to XML (assumes some familiarity with XML): <http://www.mulberrytech.com/quickref/XMLquickref.pdf>
- XML Schema
  - The W3C XML Schema specification: <http://w3.org/XML/Schema>.
  - A tutorial: <http://www.xml.com/pub/a/2000/11/29/schemas/part1.html>
  - Links to XML Schema quick reference guides: <http://www.xml.dvint.com/>

### 1.5.2. XML Terminology and Tools

Following are some common terms and tools used when working with XML:

- **Parser:** A parser is a software tool used to break XML documents into pieces that are convenient to be processed by a computer program. Most XML parsers use one (or both) of the following types of interfaces:
  - **DOM (Document Object Model):** DOM parsers convert the XML document into a tree representation that is convenient for computer programs to process.
  - **SAX (Simple API for XML):** SAX parsers convert the XML document into a sequence of “events” representing the different components in the document. SAX parsers allow more efficient programs to be written, especially for processing large documents, but are less convenient than DOM parsers.

Common XML parser implementations include Apache Xerces (<http://www.apache.org>) for Java, or C++, or Microsoft MSXML ActiveX component.

- **Well-formedness:** An XML document that follows all of the standard XML rules is described as “well-formed”. An XML parser can parse a well-formed document. An XML parser will report a parsing error if the input document is not well-formed.
- **Validating Parser:** A parser that can validate an XML document against a schema or DTD. Most modern XML parsers can validate, although validation is not used for some performance-intensive applications.
- **XSLT (Extensible Style Language - Transformations):** A template-driven (pattern matching) language for processing XML. XSLT is particularly strong at reformatting XML, for example into HTML. See <http://www.w3.org/Style/XSL/> for more on the XSLT specification. Common implementations of XSLT include Microsoft MSXSL , Apache Xalan (<http://xml.apache.org/xalan-j/>), Saxon (<http://saxon.sourceforge.net/>).
- **XML Schema:** An expressive, XML-based definition of an XML document’s vocabulary and syntax. FpML uses W3C XML schema to define its structure.
- **DTD (Document Type Definition):** An older, more compact but less expressive (compared to XML Schema) way of defining an XML document’s vocabulary and syntax. Versions of FpML prior to version 4 used DTDs.

## FpML 5 User Guide 2012 Edition

- **Instance Document:** An XML document that contains business data expressed using the syntax rules defined by either a DTD or XML Schema.
- **Namespace:** An identified collection of XML component names. A namespace is typically defined by a schema. Namespaces are used to allow an instance document to specify which vocabulary is meant when a particular name is used.

### 1.5.3. Web-based Resources

There are a variety of resources on the web for implementers interested in learning more about FpML and FpML applications. By using a web search tool to look for FpML and related terms, you can find up to date information about solutions related to your application needs. In particular, for most of the applications described in section 3 there are web-based resources. Some of the keywords that can be searched, in addition to “FpML”, include “confirmation service,” “matching messages” and “reconciliation”.

The FpML website (<http://www.fpml.org>) serves as the main source for FpML related information and applications. Along with all published documentation and the specification itself, comprehensive listings of participating organizations and vendors supporting FpML products are available.



## 2. Key Changes Compared to FpML 4

This section contains information for implementers upgrading from FpML 4. A highlight of the architectural and functionality changes introduced in FpML 5 was discussed previously in section 1.3. In the below section, these changes are discussed in detail.

The Architecture Specification version 3.0, the normative reference for the architectural rules under which FpML 5 is developed, is available on the website (<http://www.fpml.org/spec>).

### 2.1. Main Functionality Changes

FpML 5.x introduces support for a number of new applications of FpML. Some of the key changes include:

- Support for bulk reporting. FpML now has the ability to represent bulk reports in XML with multiple trades, where for each trade only the desired information needs to be provided.
- Support for real-time reporting to industry trade repositories (in 5.3 and beyond)
- Support for clearing, including related processes such as approvals (consent negotiation), better support for allocations, netting, etc.

The need to support these types of processes, and issues with FpML version 4.x, caused FpML's designers to implement a number of architectural changes compared to version 4.x. The remainder of this section discusses these architectural changes and some other minor changes.

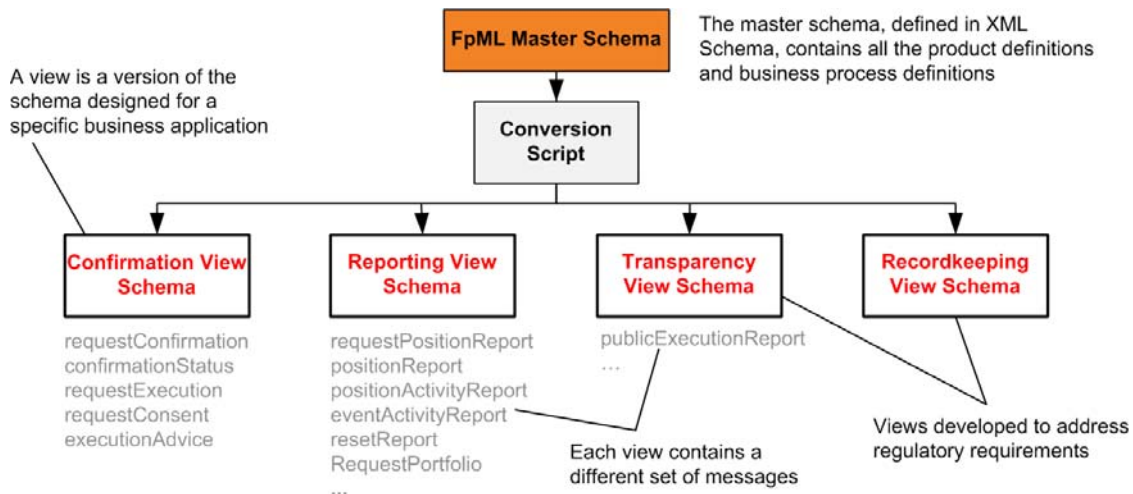
### 2.2. Views

FpML 4.x and earlier versions have a single representation for each product. The product representation was primarily designed from a confirmation perspective (i.e., complete details and a precise description). However, in certain instances, not all the details are known or needed (e.g., pretrade negotiation, summary reporting) making most/all elements optional would make the confirmation too loose. One potential solution is to make additional elements optional; however this could make the confirmation too loose. The use of "views" with different amounts of required detail is another potential solution, and is the one chosen by FpML.

The concept of "views" has been introduced in version 5 and is intended to make FpML easier to use in a number of different business contexts and for different purposes.

FpML maintains a single master schema from which multiple views can be generated. The following diagram illustrates how the different versions of the schema, or views, are generated from the master schema using a conversion script. The master schema contains annotations, with view-specific instructions (e.g., make this element optional in view X, put this element only in view Y)

## FpML 5 User Guide 2012 Edition



- In most cases users of the FpML specifications will use a view specific schema, not the master schema.  
A view is a version of the schema focused on a particular business area or application such as reporting a confirmation. The product representation in each view can be different.
- In versions 5.0-5.2 there are two views:
  - The “**Confirmation**” view: this view is very similar to the FpML representation FpML 4. Trade by trade, confirmation, STP, any sort of messaging apps where you are working on individual trades and need to pull a representation of a trade.  
The “**Reporting**” view: is a looser version of the schema, using the same structure, the same element names but where many elements are mandatory in confirmation view, almost all elements are optional in reporting view. This allows institutions to pick only the fields that need to be represented as part of trade reporting.
- In version 5.3 two additional views are added to address regulatory reporting requirements.
  - The “**Transparency**” view is intended for real-time public reporting, at the time of writing the user guide addresses CFTC rules in 17 CFR Part 43. Additional regulatory reporting requirements (e.g. from European regulators) will be addressed if and when they become available. The view includes a simplified product model without party references, in addition to messages to support reporting from market participants and execution facilities into Swap Data Repositories (SDRs).
  - The “**Recordkeeping**” view is intended for non-public reporting into SDRs, as required by the CFTC rules in 17 CFR Part 45, and is intended to support additional regulatory requirements in the future. It includes a product model that is a little more flexible than Confirmation view, but which supports all of the fields in Confirmation view.
- All the views also include the “standard” product (standardProduct), which allows reporting on trades that can be fully represented with a product ID (plus size and price information), and a “generic” product (genericProduct), for reporting on trades that cannot otherwise be reported on in the FpML schema.

## 2.3. The Use of Multiple Namespaces

- In FpML 4.x and earlier, each FpML version had its unique namespace. For example:

FpML 4.7 REC <http://www.fpml.org/2009/FpML-4-7>

FpML 4.8 REC <http://www.fpml.org/2010/FpML-4-8>

FpML 4.9 REC <http://www.fpml.org/2010/FpML-4-9>

- The FpML 5.x grammar is distributed as multiple XML schemas, each of which is specialized to suit a particular set of related business processes. This allows product and other business object representations to be adjusted to each usage (e.g. strict for confirmation, looser for reporting). Each view has a different namespace to distinguish between the different types of applications.

<http://www.fpml.org/FpML-5/confirmation> (confirmation view)

<http://www.fpml.org/FpML-5/reporting> (reporting view)

<http://www.fpml.org/FpML-5/recordkeeping> (recordkeeping view)

<http://www.fpml.org/FpML-5/transparency> (transparency view)

- Starting in version 5.0, minor FpML versions share the same namespace. For example, the namespace for the confirmation view is the same for FpML 5.0, 5.1, 5.2, and 5.3.

<http://www.fpml.org/FpML-5/confirmation> (shared namespace across minor versions)

Multiple namespaces/forward compatibility enables documents to have greater longevity. For example, with the new 5.x schemas it is possible to process documents against future compatible versions of the schema. This wasn't possible before because documents were tightly bound to a specific schema (e.g., only version 4.9 is tied to the namespace <http://www.fpml.org/2010/FpML-4-9>). This feature is useful in the task of migrating towards future versions or supporting multiple versions simultaneously.

## 2.4. Multiple Root Elements

Earlier versions of FpML used <FpML> as the root of all FpML documents. FpML 5.x now uses different element names to distinguish between message types.

- The removal of common FpML root element.

Type substitution on the root element (`xsi:type`) is no longer used to select the message type and its associated content model. The <FpML> element has been removed from the schema. The use of multiple root elements is easier to understand than type substitution at the root level and certain tools also had problems processing `xsi:type`.

<requestExecution>, <executionAdvice>, <clearingStatus>, <positionReport> are just a few of the many root elements available in FpML 5.x. See the *FpML Messaging Framework* insert (folded at the end of this book) for a complete list of root element names available.

## FpML 5 User Guide 2012 Edition

- Version 5 uses the name of the root element name to express message type.

In FpML 5.x every message has been given its own global element which can appear as a document root element. Business process related messages are typically only defined in one of the FpML process namespaces but some common messages used for error handling are defined in multiple namespaces (e.g. messageRejected).

- The FpML 'version' attribute has been renamed to 'fpmlVersion'.

To make it easier to search for the start of FpML content within an encapsulating XML document, the version attribute has been renamed to make it more distinctive.

### ***2.5. Enhanced Messaging Framework***

FpML 5 introduces the idea of **Generic Business Processes** to address limitations in the version 4 framework. This new framework:

- Consistently implements a set of general principles to make it easier to use the messages to implement real business processes
- Adjusts the representation of parties, accounts, and roles to clarify the purpose of messages and the roles of the parties within a message.

Following are some of the issues in the FpML 4 messaging framework that the version 5 framework seeks to correct:

- Incomplete message set – in many cases not all messages required to implement a business process are defined in the FpML 4 message set. In particular, many requests lack acknowledgements, exception responses, correction capabilities, and in some cases normal responses are missing. Many generic business processes (such as confirmation) have different levels of completeness depending on the specific event that is being covered.
- Inconsistent message correlation – different implementations use different features of the FpML 4 framework to link successive messages together, making them incompatible.
- Unclear processes – it is not always clear how the version 4 messages are to be combined together to fully implement a business process. For example, which message should be used to acknowledge or respond to a request isn't always defined. While this could in theory be addressed through documentation, because the message set is incomplete, this is difficult to do.

## FpML 5 User Guide 2012 Edition

In version 4.x, there were a set of messages for each business process (i.e., distinct messages are necessary to request a trade increase, a trade amendment, a trade amendment).

In FpML 5, one set of messages (e.g., execution, confirmation, allocation) can be reused, for many trading events (different payloads) using the same flow of messages. The messages can be applied in a number of different contexts. For example, the same <requestExecution> message can be used to execute an increase, an amendment, a termination, a novation, a trade.

Using generic business processes offers a number of important benefits:

- Implementers can learn a set of messages once and reuse them in different contexts (using different payloads)
- Generic processes improve consistency across post-trade events and make it easier to ensure all necessary messages are present.
- It greatly reduces the number of messages required to provide full coverage, simplifies the structure of messages and makes it easier to use FpML messaging.

### ***Messaging Correlation***

Message correlation is essential to link successive messages together. As noted earlier, there were issues with inconsistent message correlation in version 4.x. In FpML 5.x, there is a single, well-defined way to link successive messages (such as corrections or retractions of requests or notifications). Successive messages are “correlated” (linked together) using a new, explicit correlationId element. The correlation ID is assigned by the initiator. Subsequent responses use the correlation ID to link back to the original request.

## **2.6. Other Improvements**

In addition to the major architectural changes outlined above, there are a number of long overdue improvements which make the standard more consistent and easier to use.

### **2.6.1. Boolean**

In earlier versions of FpML (mostly CDS), booleans were often represented by the presence or absence of an optional empty element. This approach was ambiguous as it conflicted with short/long forms of the products. In version 5.x the use of optional empty element as synonym for Boolean has been eliminated and replaced by xsd:Boolean type.

### **2.6.2. Underlyers**

Some underlyers have been refactored for consistency. Bond and Index are now correctly derived from UnderlyingAsset (instead of ExchangeTraded type). A new curveInstrument substitution group has been created for Deposit, FxRateAsset, RateIndex, SimpleCreditDefaultSwap, SimpleFra, SimpleIRSwap.

See the *FpML 5 Underlying Assets* table at the end of this user guide, and also on page 82, for the full list of underlyers.

### 2.6.3. Refactoring FX Products

FX coverage has been refactored and expanded to make it more consistent with other FpML product representations and to facilitate further development. As a result of this work the following original 4.x FX model's issues were addressed:

- A set of reusable components that facilitate product development were defined. Existing and future FX products can leverage these building blocks to ensure the FX model is coherent and easy to maintain, as per FpML best practices
- The existing coverage was extended to include Dual Currency Deposits.
- Rationalized the models' constraints (Use of grammar to bring related data together. Better use of XML schema to simplify the validation rules.)

Since FpML 5.1 Recommendation, the following FX products are covered:

- Basic FX Products
  - FX Spot and FX Forward (including non-deliverable settlements, or NDFs)
  - FX Swap
- Simple FX Option Products (including, features, cash and physical settlement)
  - FX options
    - European and American
    - Averaging
    - Barriers
  - Digital Options
- Option Strategies (multiple simple options)

In addition, support for the following money market instrument is also provided:

- Term Deposits (including features)
  - Money Market Deposits
  - Dual Currency

### 2.6.4. Payments

Payment and premium structures have been standardized based on SimplePayment type. Settlement information was removed from payments and products.

### 2.6.5. Removed Deprecated Structures

Deprecated structures which had to be maintained between minor 4.x versions for backward compatibility reasons have been removed in FpML 5.

### 2.6.6. Removed Contract

## FpML 5 User Guide 2012 Edition

In FpML 4.2, support was added for a set of messages intended for communication between investment managers and custodians via a ‘Closed User Group’ on the SWIFT Network.

Originally the CUG notification messages were based on the standard FpML ‘trade’ structure but in 2006 a late change to the schema added a new concept, a ‘contract’, into the model and the CUG messages were updated to make use of it while the rest of the schema remained ‘trade’ based.

Since 4.2, FpML has had two structures -trade and contract- that basically describe the same information but where each is tied to a specific set of messages. This change has created an inconsistency within the model. Contract has been removed in 5.x.

The motivation for introducing ‘contract’ was *“to differentiate more clearly between the trading event itself and the resulting contract.* A processing application must in practice assume a ‘trade’ is a ‘contract’ until it discovers that it isn’t because of the operations applied to it. So differentiating between ‘trade’ and ‘contract’ turned out to be neither practical nor useful.

A study of ISDA documentation has also shown that the term ‘transaction’ is more commonly used in legal documents and across the financial markets as a whole the words ‘trade’, ‘deal’, ‘contract’ and ‘transaction’ are often used interchangeably.

### 2.6.7. Account and Roles – Removed tradeSide

The new messaging framework adjusts the representation of parties, accounts, and roles to clarify the purpose of messages and the roles of the parties within a message.

- **Removed TradeSide** - The 4.x tradeSide structure has been replaced by a simpler implementation adding a new relatedParty element to the partyTradeInformation. The role element within the relatedParty defines the list of roles as code list. This allows easier customization of the roles than using the tradeSide structure.
  - The buyer/seller party references now include optional account references (buyerAccountReference and sellerAccountReference)
  - Similarly, the payer/receiver model includes optional account references (payerAccountReference and receiverAccountReference)
- **Parties Account / Roles** - The existing representation for parties, accounts and roles version 4 was complicated and difficult to work with. In order to simplify the structure and make the relationships clearer the Account structure was moved outside of Party and given a mandatory reference to its beneficiary and an optional reference to its servicer
  - ServicingParty – party that services/supports the account
  - accountBeneficiary – owner of the account/funds

### 2.6.8. Other Changes

There were a few other changes to product representations including adjusted dates and miscellaneous product refactoring.

## **2.7. Impact**

The architectural changes introduced in version 5 will have an impact on the way users need to approach the FpML standard from a planning and implementation perspective. Here are some of the key areas users should be aware of.

### **2.7.1. Impact of Views**

- FpML users must decide which view (schema) to use for a given application/system. Business processes are generally contained in a single view. Often the view can be selected based on the types of messages that the user wants to support.
- Once the view is selected, instance documents should be closely compatible with previous FpML versions
- Choosing a looser view (e.g. reporting) allows more flexibility regarding the data that can be included in an instance document, however, this comes at the expense of the effectiveness of schema validation.

### **2.7.2. Impact on Extensions**

Extensions to the FpML schema will be impacted by the introduction of views.

- Extensions will need to import the appropriate views (e.g. `<xsd:import namespace="http://www.fpml.org/FpML-5/confirmation" ...>`).
- Extensions applicable to multiple views will need to be duplicated

### **2.7.3. Impact of Multiple Roots and Generic Processes**

Using generic processes offers many benefits as described in section 2.4. This comes with a drawback as systems need to look inside messages to see what type of payload is carried by the message. As a result, it may make it slightly harder to route or report on messages by event type.

You cannot look for the `<FpML>` element anymore. Instead, you can look for an element containing the `fpmlVersion` attribute, or a root element in an FpML namespace.



### 2.7.4. Migrating Messages from 4.x to 5.x

Firms that have implemented FpML 4.x messages internally or between trading partners can benefit from the improved messaging framework in version 5. The changes between the two versions are significant. To help the migration process, a mapping table showing the relationship between 4.x and 5.x messages is available for download at <http://www.fpml.org/documents/FpML-message-mapping-4x-vs-5x.xls>.

The following diagram illustrates a few rows of the mapping table. An implementation using a requestAmendmentConfirmation message in FpML 4.9, for example, should instead use the requestConfirmation message available in the confirmation view of FpML 5. This requestConfirmation message would carry an *amendment* business event payload.

Version 4.9			Version 5.3			
Business Process	Schema File	Message Type	Root Element	Event	Schema File	View
confirmation	fpml-posttrade-confir	RequestAmendmentConfir	requestConfirmation	amendment	fpml-confirmation-processes-	confirmation
notification	fpml-contract-notifica	ContractCreated	executionAdvice	trade	fpml-confirmation-processes-	confirmation
notification	Not Available	Not Available	requestExecution	event group	fpml-confirmation-processes-	confirmation
confirmation	fpml-posttrade-confir	RequestNovationConfir	requestConfirmation	novation	fpml-confirmation-processes-	confirmation



## 3. How to Use FpML – Sample Applications

### 3.1. Introduction

#### 3.1.1. Purpose

This section provides examples of generic types of applications that users may wish to implement using FpML. All of these applications are known to have been implemented in some form, and many have been implemented by a number of firms. For each application, there are some suggestions about which FpML messages or structures to use, as well as a brief discussion of issues the implementer may face and resources for addressing them.

#### 3.1.2. Overview

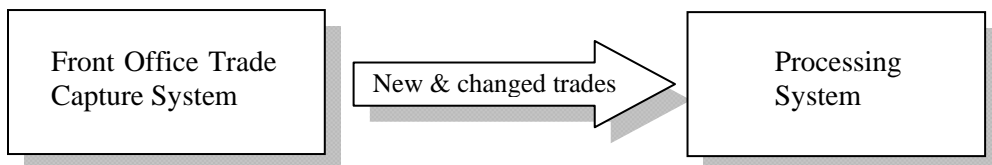
The example applications covered in this section include:

- A straight-through-processing (STP) data feed from a front office system to a processing system.
- A multilateral confirmation service.
- A central counterparty derivatives clearing organization.
- An internal trade repository/archive.
- An external trade archive/data repository to support regulatory reporting.
- Activity reconciliation between related systems/parties.
- Position reconciliation between related systems/parties.
- Bulk reporting applications.

## 3.2. Straight-Through Processing (STP) Data Transfer

### 3.2.1. Objective

A common application for FpML is to send trade information automatically from front office systems to downstream processing systems, to reduce re-keying efforts, error rates, and operational risk.



### 3.2.2. Requirements

To support the application, the following is required:

- The ability to send new, modified, and cancelled trades from the front office to the processing system.
- The ability to record trade details for a wide variety of products.
- The ability to support internal data, such as revenue allocation information.

### 3.2.3. Messages and Structures Used

To support the workflow requirements, the FpML Execution messages can be used. These messages are described in Section 7.

The product coverage is addressed by the FpML product model, outlined in Section 6. If additional products or product characteristics are required, these can be customized as described in Sections 6 and 8.

Additional data items not covered by FpML can be covered through extensions as described in Section 8.

### 3.2.4. Tools and Technology

To implement this application, the following is needed:

- The ability to produce FpML in the upstream system. This can be done in a variety of ways and in different technical environments. Some techniques suited for this include:
  - “Print” statements. Since XML is just text, it can be produced by most programming languages.
  - Constructing XML via a DOM (Document Object Model) tree. This allows the XML to be constructed in memory and then produced as one operation.
  - Using various FpML API frameworks, for example code generated from the schema.
- The ability to transmit FpML from the upstream system to the downstream system. Different ways of doing this include:
  - File transfer, e.g., using shared network drives or file transfer software such as FTP.
  - Message oriented middleware, such as IBM Websphere MQ.

## FpML 5 User Guide 2012 Edition

- Direct socket connections.
- Web services, such as SOAP over HTTP.
- The ability to extract the business information from the FpML instance document in the downstream system. This is typically done in one of the following ways:
  - By “parsing” the FpML using an off the shelf XML parser, and writing custom code to process the data.
  - By using a data extraction or conversion program to convert the data into a format recognizable by the downstream system, such as a delimited file format. The extractors or converters can be written using technologies such as XSLT.

### 3.3. Automatic Confirmation

#### 3.3.1. Objective

To provide automatic confirmation of a trade, there are two models in general use in the industry: matching and affirmation. The model described here and diagrammed below is a confirmation matching model; some services may implement an affirmation model, in which one party affirms (agrees to) the other party's trade. In the affirmation model there is only one representation of the trade. The FpML specifications detail messages and business process descriptions for both models.

In this application, each party sends trade confirmation requests to the confirmation service; the service matches the requests and sends resulting status updates to the parties, indicating whether the trades have been confirmed.



#### 3.3.2. Requirements

Following are the high level requirements for the application:

- The ability for parties to request confirmations, modify the information and resubmit when there is a problem; and cancel confirmation requests in case of error.
- The ability to tightly control the list of valid trades/products to reduce the probability of error or unintended mismatch.
- The ability for the confirmation service to communicate the status of the confirmations (e.g., whether the trades match, have differences (mismatch), or where no matching trade is found).

#### 3.3.3. Messages Used

To make requests to the confirmation service, the following FpML messages can be used:

- **Request Confirmation:** Initiates a trade confirmation process
- **Request Confirmation Retracted:** Stops a trade confirmation process

For confirmation services to return trade statuses, the following messages can be used:

- **Confirmation Acknowledgement:** Indicates receipt of a confirmation request
- **Confirmation Status:** Is used to send matching results. This response message returns the status of an event that has been submitted for matching (e.g., Alleged, Matched, Mismatched, Unmatched)
- **Confirmation Agreed:** Indicates a confirmation acceptance has been received from both sides.
- **Confirmation Disputed:** Indicates that the confirmation requester doesn't accept the proposed matching results. One party is disputing the trade.

These messages are described in Section 7.

### 3.3.4. Customizations / Restrictions

To restrict the range of acceptable products, and to add service-specific details, the schema can be customized to add fields and/or eliminate options. See Section 8 for more information on how to customize FpML.

### 3.3.5. Validation

To validate that the confirmation message meets basic FpML business rules and possibly platform-specific business rules, the FpML validation framework can be used. This allows specific business constraints to be enforced for all input messages. See Section 5 for more information.

### 3.3.6. Tools and Technologies

To implement this application, parties will need the following:

- The ability to create the FpML, as described above in section 3.2.4. Parties need to ensure that the generated FpML meets the confirmation service's specific requirements and constraints.
- The ability to transmit the FpML to the confirmation service. The mechanism to do this will typically be defined by the service, and may include technologies such as:
  - Custom Application Programming Interfaces (APIs), typically based on sockets over dedicated networks
  - Message-oriented middleware on dedicated networks
  - Web services

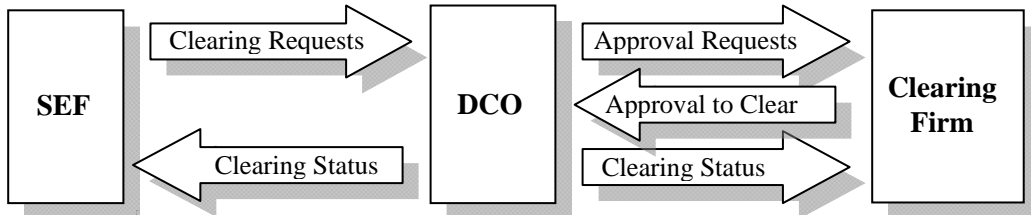
Services will need to create the following, among other items:

- A message delivery infrastructure
- A trade validation mechanism
- A trade matching and persistence mechanism

## 3.4. Derivatives Clearing Organization

### 3.4.1. Objective

Recent regulatory changes have increased the need for central counterparty clearing organizations for OTC derivative trading. These Derivatives Clearing Organizations (DCOs) need a way for trades to be submitted for clearing, processes for approving clearing and possibly allocating trades to multiple accounts, as well as various ways to report back status, cleared trades, and periodic processing results, such as margin requirements. FpML supports all of these interfaces.



### 3.4.2. Requirements

Following are the high level requirements for the application:

- Provide a way for Swap Execution Facilities (SEFs), matching services, and market participants to submit trades for clearing.
- Provide a way to report status on these trades.
- Provide a way for DCOs to request approval from clearing member firms to take on trades.
- Provide a way for DCOs to report that a trade has been cleared.
- Provide a way for DCOs to report on open positions and on settlement amounts, and other regulator reports.

### 3.4.3. Messages Used

To make requests to the confirmation service, the following FpML business processes can be used:

From “Confirmation” view:

- **clearing** – allows requests for clearing, acknowledgements and exceptions, and clearing confirmed or clearing refused messages to be sent.;
- **consent** – allows DCOs to request consent from clearing member firms to clear a trade
- **allocation** – allows firms to request that trades be allocated and for DCOs to report back status, exceptions, etc.
- **others** – there are other messages that can be used, including for example option expiry/notification, or credit event notification.

From “Reporting” view, there are a number of reports that can be used, for example:

- **Position report** – lists open positions at a given point in time; can also be used to report settlement amount, NPVs, margin calculations, etc.



- **Position Activity report** – lists changes in positions over a time period (added, modified, removed)
- **EventActivityReport** – lists specific events that have occurred over a time period (e.g., novations, netting)
- **Reset Report** – lists rate fixings and the affected trades
- **TradeCashFlowsAsserted** – allows calculations of settlement amounts to be detailed.

These messages are described in Section 7.

### 3.4.4. Customizations / Restrictions

To restrict the range of acceptable products, and to add service-specific details, the schema can be customized to add fields and/or eliminate options. See Section 8 for more information on how to customize FpML.

### 3.4.5. Validation

To validate that the confirmation message meets basic FpML business rules and possibly platform-specific business rules, the FpML validation framework can be used. This allows specific business constraints to be enforced for all input messages. See Section 5 for more information.

### 3.4.6. Tools and Technologies

To implement this application, parties will need the following:

- The ability to create the FpML, as described above in section 3.2.4. Parties need to ensure that the generated FpML meets the clearing service's specific requirements and constraints.
- The ability to transmit the FpML to the clearing service. The mechanism to do this will typically be defined by the service, and may include technologies such as:
  - Custom Application Programming Interfaces (APIs), typically based on sockets over dedicated networks
  - Message-oriented middleware on dedicated networks
  - Web services

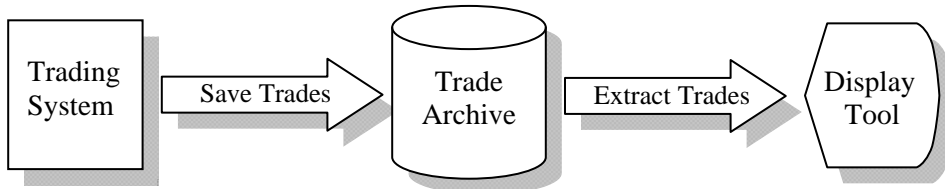
Services will need to create the following, among other items:

- A message delivery infrastructure
- An approval process
- A trade validation mechanism
- Valuation and margining calculation and reporting
- Possibly a netting and netting report process

## 3.5. Internal Trade Archive

### 3.5.1. Objective

Another possible application is the use of FpML to store trades, e.g., to a file system or database. The flexibility of FpML allows a variety of trades with different details to be saved without ongoing database maintenance. Also, FpML is well suited for saving multiple versions of trades; each trade version can be stored in a different file or database record. A simple application for this type of trade archive allows trades to be pulled from the repository and displayed.



### 3.5.2. Structures Used

In this application, although it is possible to represent the trades using messages, there is less benefit in doing so, because the application is not fundamentally a messaging application. If messages are desired, the A2A messages described in Section 3.2 can be used. Alternatively, the FpML “dataDocument” format, a non-message format, can be used. See Section 5 for more details on dataDocument.

In addition, there may be some desire to organize or group trades into collections of related trades, e.g., based on underlying, trading desk, counterparty. If so, the FpML “Portfolio” structure, described in Section 5, can be used to record portfolio contents.

### 3.5.3. Tools

To implement this type of application, the following is needed:

- The ability to generate FpML from the source system. This can be done in a variety of ways, as described above.
- The ability to store and organize the files. Different ways of doing this include:
  - Using a basic file system-based storage scheme
  - In a relational database, with one “blob” (or large text buffer) record per trade version stored in the database
  - Using an XML document management tool
- A retrieval mechanism. This is usually tied to the storage mechanism, but may in addition offer features such as:
  - The ability to query trades based on content
  - The ability to display multiple versions of trades, and possibly to compare them
  - Web-based navigation or query tools
- A mechanism for displaying the trades. This can be done in several ways, such as:
  - A web browser-based display tool such as the *FpML Editor/Viewer*, available from the FpML website
  - A Java application based display application

## FpML 5 User Guide 2012 Edition

- A system for rendering the trades into a report format (e.g., into PDF files) as required

### 3.5.4. Issues/Notes

- One concern often cited by developers with the use of XML (and especially FpML) for this type of application is the large file size. However, this concern is usually not a serious limitation, for the following reasons:
  - FpML trade representations are typically not really that large relative to modern technology. A typical parametric view of a trade is about 10-20KB. 100,000 trades will require about 1-2 GB of storage. 100 versions each of 100,000 trades will require about 100 GB. This will fit on a single large disk.
  - If this is too much (for example, because additional information such as cash flows is being kept, or because storage is limited), FpML is highly compressible. A single trade is likely to compress by a factor of 4 to 10, and more if it is large.
  - A large collection of trades and trade versions is likely to compress by a factor of at least 10 and up to 100 fold or more when compressed as a single unit, depending on the commonality between trades. Also, one compressed large archive file typically uses computer file systems more efficiently than a large number of small files. This technique is useful for creating and distributing archival snapshots of large numbers of trades. For example, a flash drive could hold a dealer's entire current derivatives inventory, possibly even with cash flows, when represented as a compressed archive.
- There are tools on the marketplace for comparing XML or FpML instance documents and reporting differences between them. For reporting differences between successive versions of FpML instance documents, a number of XML differencing tools could be used. For more information, search the web using key words "XML differencing" or "XML diff".

## 3.6. External Trade Repository (for Regulatory Reporting)

### 3.6.1. Objective

Recent regulations require market participants and facilities in many jurisdictions to report OTC derivative trading activity to external trade repositories, for analysis by the regulators, and possibly for public reporting.

As opposed to with an internal trade repository, in this case it is the messaging interface that is the most important place that FpML can add value. A consistent interface allows firms to use the same interface to report to multiple trade repositories with minimal rework, and ensures that the trade repository interface is in alignment with the industry needs.

A workflow diagram, available at <http://www.fpml.org/documents/OTC-Derivatives-Trade-Flow-to-SDR.pdf>, illustrates, at a high level, the interaction between the different organizations and the types of FpML messages they would be exchanging.

### 3.6.2. Requirements

Some of the requirements that this type of service interface will need to meet include:

- support real time public activity reporting
- support non-public event-driven reporting as well as periodic snapshots
- support for specifically mandated surveillance fields
- ability to verify or dispute repository contents
- ability to supply valuations for positions
- ability to withdraw trades from the repository
- consistent trade and product identification system
- support for multiple regulators globally

### 3.6.3. Views and Business Processes Used

To support these requirements, FpML has developed two views.

**Transparency** view is used for real-time public reporting, and contains a stripped down trade representation intended to fully support only standard product types. The main business process used include:

- **public execution** – Specifically the publicExecutionReport, as well as acknowledgement, exception, and retraction messages.
- **verification** – Specifically the verificationStatusNotification and related messages.

**recordkeeping** view is used for non-public regulatory reporting, and supports a full trade and product representation, while allowing a less comprehensive report. The main business processes used include:

- **nonpublic execution** – Specifically the nonpublicExecutionReport, as well as acknowledgement, exception and retraction messages.
- **verification** – Specifically the verificationStatusNotification and related messages.
- **valuation** – Specifically the valuationReport message and related messages.

## FpML 5 User Guide 2012 Edition

There are many surveillance fields supported mostly in the “partyTradeInformation” block in the tradeHeader. Trades are identified using an enhanced (for version 5.3) partyTradeIdentifier to hold the “Unique Swap Identifier” (USI). This version of the partyTradeIdentifier allows the USI namespace to be represented in the “issuer” element. The “reportingRegime” structure allows information to be provided about how this message is applicable to various regulatory reporting regimes and supervisors/regulators.

The product taxonomy is represented using the ISDA product taxonomy modeled in FpML, and held in the <productType> element. FpML published a coding scheme holding these values. A typical product type would be represented as:

```
<productType productTypeScheme="http://www.fpml.org/coding-scheme/product-type">InterestRate:IRswap:Fixed-Float</productType>
```

### 3.6.4. Tools

In addition to the usual tools for generating and processing FpML, there are a couple of tools under development to help with SDR reporting.

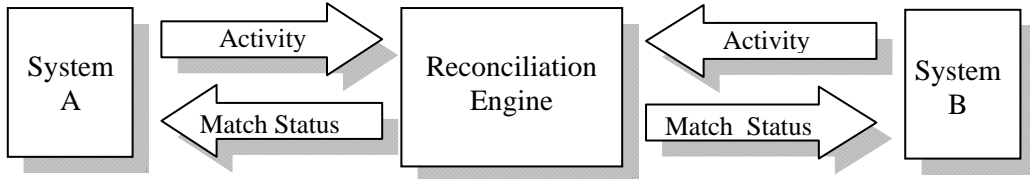
The product taxonomy is being defined precisely using mostly XPath based rules to classify FpML trades into products. A sample XSLT script is able to test trades against these rules.

A similar script is expected to be provided to validate surveillance fields.

## 3.7. Intra-day Activity Reconciliation

### 3.7.1. Objective

FpML can be used to reconcile intra-day activity as recorded in one system with that in another. This type of check could be used to monitor manual dual-keying processes, as a check on automated STP processes, or as a check between separate entities. In general, the idea is to provide a real-time electronic comparison of state changes across systems.



### 3.7.2. Messages Used

Following are some of the messages that can be used to implement this workflow:

- From source systems to reconciliation engine:
  - Request Confirmation
  - Request Confirmation (indicating a correction)
  - Request Confirmation Retracted
- From reconciliation engine to source systems:
  - Confirmation Status

See Section 7 for more information about the matching messages.

### 3.7.3. Reconciliation Approaches

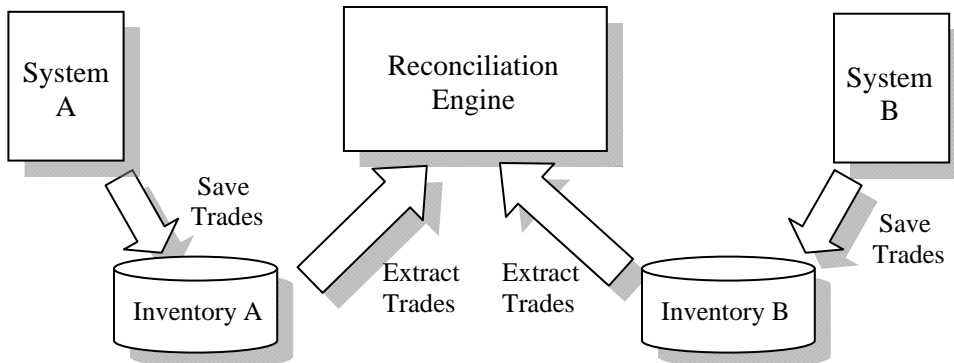
To implement the reconciliation engine, firms have employed different strategies, including:

- Internally developed systems: Some firms choose to develop custom FpML matching and reconciliation technology.
- Third party systems: Some firms choose to acquire and deploy third party matching/reconciliation technology, including:
  - XML differencing: There are some generic XML tools for comparing documents that can be used.
  - FpML-specific tools: There are some FpML-specific or FpML-aware tools that can be used for matching/reconciling trades represented in FpML.

## 3.8. Inventory Reconciliation

### 3.8.1. Objective

Even when automated intra-day reconciliation and confirmation processes exist, one might reconcile the current trade inventory as recorded in one system with that in another. This can be done on a periodic or incremental basis. The objective is to confirm that the trade population in the two places is within tolerable differences, if not identical.



### 3.8.2. Messages/Structures Used

In this application, the key is representing the trade economics; the workflow/activities are not important. For this reason, the preferred formats used to save the trades into FpML files are similar to those used in the Trade Archive application discussed above.

In addition, it is common to organize the trades in the inventories into partitions or portfolios. This allows the reconciliation to work on smaller subsets of the inventory at a time. This typically provides performance and reliability advantages.

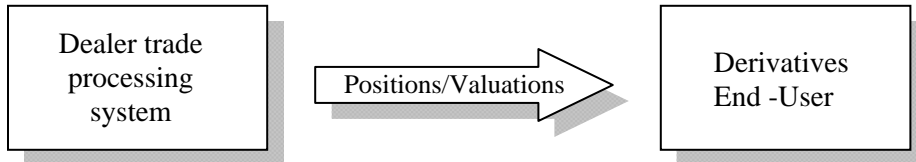
### 3.8.3. Matching/Reconciliation Approaches

Matching and reconciliation approaches are similar to those for intra-day reconciliation, but the message processing and management requirements are simpler. On the other hand, the data volume requirements are typically much higher.

## 3.9. Bulk Reporting Applications

### 3.9.1. Objective

Yet another use of FpML might be to periodically (e.g., daily) generate reports of trade inventories and valuations for specific counterparties and send these to the counterparties to support processes such as collateral calls. Typically these reports will include multiple trades from multiple product types. Below is an example of this type of feed:



### 3.9.2. Messages/Structures Used

The **PositionReport** message fully supports this application by incorporating trade as well as valuation information. The trade data content is modeled reusing the existing FpML trade and product representations, discussed in Sections 5 and 6. The valuation information is modeled using the FpML Pricing and Risk framework, discussed in Section 7.

In addition to this report, there are a number of other reports that can be used, for example:

- The **positionActivityReport** can be used to report on changes in positions (e.g. new, modified, removed) positions over a period of time.
- The **eventActivityReport** can be used to report on trading events (such as trades, amendments, early terminations) over a period of time.
- The **exposureReport** can be used to report on aggregate exposures, such as market value, counterparty risk, for example. One key use of this report is the CFTC Part 20 Large Trader Position Reporting report.
- The **resetReport** can be used to report on rate fixings and the affected trades.
- There are also various other valuation reports, cash flow and portfolio reconciliation reports, for example.

### 3.9.3. Tools and Technologies

To implement this application you will need technologies similar to those described above in Section 3.2. However, because this type of application is typically a periodic (e.g., daily) process, there are some differences in the types of suitable technologies. Following are some of the technologies that are likely to be required:

- The ability to generate FpML, as discussed in previous sections.
- The ability to transmit the FpML from dealers to clients. This could be done in a variety of ways, including:
  - File transfer
  - Download from a secure website
  - Some form of secure electronic mail, e.g., PGP / GPG
- The ability to validate the FpML. This could be done using technologies including:
  - Custom XSLT style sheets



## FpML 5 User Guide 2012 Edition

- Business rules written using the FpML validation framework and implemented on a commercially available validation processor
- The ability to read the incoming FpML. This could be done by converting the file to a format suitable for processing in existing systems, for example via:
  - XSLT-based style sheets
- An XPath-based data extractor written in Java or C++, for example.



## 4. How to Write FpML – Examples

### 4.1. Introduction

This section demonstrates some of the key FpML concepts by building two examples that demonstrate how FpML is formed and why it is structured the way it is.

The first example (section 4.2 Forward Payment) demonstrates some basic aspects of FpML by representing a trade with very few economic details.

The second example (section 4.3 IR Swap) concerns a vanilla interest rate swap confirmation message, and demonstrates how a somewhat more complex FpML instance document is formed.

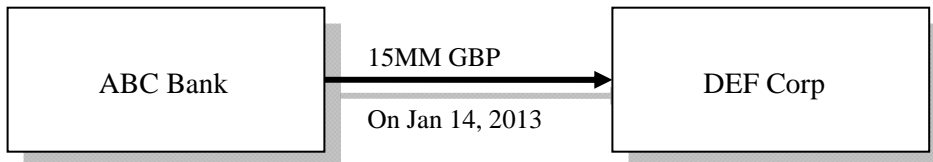
Other examples include

- real-time public reporting of a commodity swap using Transparency view
- non-public regulatory reporting of an equity option using Recordkeeping view
- clearing of an FX option using Confirmation view
- confirmation of an IR swap using Confirmation view

### 4.2. Basic Example: Forward Payment

#### 4.2.1. Introduction

On Jan. 14, 2012, ABC Bank agrees to make a payment to DEF Corporation in one year's time, on Jan. 14, 2013. This date will be adjusted if it falls on a holiday. The payment will be 15,000,000 GBP.



In this example we develop an FpML 5.3 document that represents this transaction.

#### 4.2.2. Required Data Attributes

The following attributes are required to represent this transaction:

- The parties to the trade: represented by SWIFT BIC codes ABCDUS33 and DEFGGB2L.
- The trade date: 2012-01-14
- ABC's reference number: A001.
- The type of trade: "bullet payment"
- Who's paying: ABC
- The payment amount: 15,000,000
- The payment currency: GBP
- The payment date: 2013-01-14
- Date adjustment rules: "Following" convention, New York and London banking days

## 4.2.3. Developing the FpML

### 4.2.3.1. Root Element

All XML documents have what is called a “root” element or “document” element.

In FpML 5, every message has a different root element. E.g., `dataDocument`, `requestConfirmation`, `executionNotification`, `resetReport`, `requestMargin`, `requestValuationReport`, just to name a few. Throughout the user guide we’ll mostly use `dataDocument` as an example.

(The full list of available root elements is described in section 7 or in the *Messaging Framework* insert at the end of this user guide.)

Tags in XML are defined using angle brackets (“<” and “>” signs). There is an opening tag (<mytag>) followed by a matching closing tag (</mytag>).

The following is a simplified version of the starting and ending tags:

```
<dataDocument fpmlVersion="5-3"
    xmlns=" http://www.fpml.org/FpML-5/confirmation">
    <!-- party and trade details will go here -->
</dataDocument>
```

- The **dataDocument** keyword says that this is the top (“root”) of the FpML instance document.
- The **fpmlVersion** attribute says that we are using FpML. With the removal of the <FpML> root element in version 5.x, the `fpmlVersion` attribute may be used to flag the start of the FpML document. The FpML namespace may be used as well to indicate the start of the FpML document.
- `dataDocument` is available in all views. In this case, we are using the “Confirmation” view, as indicated by the namespace (**xmlns** attribute).
- Some additional attributes (not shown here) will specify exactly how the “FpML” namespace is defined. This will link to the FpML schema.
- The <!-- and --> symbols surround an XML comment; the comment is ignored by the receiving program.
- The </dataDocument> tag ends the FpML.



FpML 4

FpML 4.x and earlier versions use <FpML> as the single root element.

The additional attributes found in the root element are discussed in more detail in Section 5.

#### 4.2.3.2. Parties

Second, model the participants to the trade. In FpML, the participants are represented by the “party” element.

Following is an FpML representation of the two parties in this transaction:

```
<dataDocument fpmlVersion="5-3" ...>
  <party id="abc">
    <partyId>ABCDUS33</partyId>
  </party>
  <party id="def">
    <partyId>DEFGGB2L</partyId>
    <partyName>DEF Corp</partyName> <!-- optional -->
  </party>
</dataDocument>
```

- The **party** keyword introduces a party definition
- The attribute, **id** = “**abc**” creates an identifier that can be used throughout the document for referring to this party
- The **partyId** keyword specifies the party’s ID (default is a SWIFT BIC code, but other coding schemes can be used).
- The **partyName** keyword, can optionally be used to hold the legal name of the party.

The **party** element is also discussed in Section 5.

#### 4.2.3.3. Trade

Next, represent the business transaction itself. In FpML, this is represented by a “trade” element with a “tradeHeader” that provides some reference information:

```
<dataDocument fpmlVersion="5-3" ...>
  <trade>
    <tradeHeader>
      <partyTradeIdentifier>
        <partyReference href="abc"/>
        <tradeId tradeIdScheme="http://a.com/t-id">A001</tradeId>
      </partyTradeIdentifier>
      <tradeDate>2012-01-14</tradeDate>
    </tradeHeader>
    <!-- product info skipped -->
  </trade>
  <-- party information omitted -->
</dataDocument>
```

- The **trade** keyword introduces the trade.
- The **tradeHeader** keyword introduces identifying (non-economic) details.
- The **partyTradeIdentifier** block holds identifying information specific to party ABC.
- The **tradeId** element hold’s party A’s reference number for this trade. The **tradeIdScheme** attribute specifies the domain values defined by party A.

- The **tradeDate** tag says when the trade was done.

The “trade” element is discussed in more detail in Section 5.

Please note that in FpML, the “party” elements typically go at the end of the FpML, after the trades. The locations of an *href* and its corresponding *id* in the instance document do not matter since XML references can work forward or backward.

So far from the list of required data elements, we have represented the following:

- The parties: ABCDUS33 and DEFGGB2L, in the “party” elements,
- The trade date: 2012-01-14; and
- ABC’s reference number: A001

The trade date and reference number appear in the “tradeHeader” element.

### 4.2.3.4.Product

Next, represent the actual trade economics; in FpML this is called the “product”. The product for a single payment in FpML is called a “bullet payment” and is represented by the “bulletPayment” tag.

```
<dataDocument fpmlVersion="5-3" ...>
  <trade>
    <!-- trade header skipped -->
    <bulletPayment>
      <payment>
        <!-- payment details go here -->
      </payment>
    </bulletPayment>
  </trade>
  <-- party information omitted -->
</dataDocument>
```

- **bulletPayment** is a kind of FpML “product”; this specifies what type of trade is being done.
- Different products (e.g., swap, fxSwap, equityOption) could go here. See Section 5 for more details on product substitution.
- **bulletPayment** is a product that consists only of a single “payment”.
- Note that **payment** can be used in other places in FpML as well. For example, in option premium payments.

Products are covered in more detail in Section 6.

Next, specify the amount and direction of the payment. This is done by filling in part of the <payment> element, specifically the paying and receiving parties, and the amount of the payment:

## FpML 5 User Guide 2012 Edition

```
<dataDocument fpmlVersion="5-3" ...>
  <trade>
    <!-- trade header skipped -->
    <bulletPayment>
      <payment>
        <payerPartyReference href="abc"/>
        <receiverPartyReference href="def"/>
        <paymentAmount>
          <currency>GBP</currency>
          <amount>15000000.00</amount>
        </paymentAmount>
        <!-- payment date information skipped -->
      </payment>
    </bulletPayment>
  </trade>
  <-- party information omitted -->
</dataDocument>
```

- **payerPartyReference** and **receiverPartyReference** specify respectively which party pays and which party is on the receiving side. (ABC pays, DEF receives).
- They link to (reference) the party IDs (i.e., abc and def) created earlier.
- **paymentAmount** introduces the size of the payment.
- **currency** and **amount** provide details on the payment.

Finally, within the bullet payment the date of the payment and the rules for adjusting it are specified:

```
<!-- FpML and trade header information skipped -->
<payment>
<!-- party and amount info skipped -->
  <paymentDate>
    <unadjustedDate>2013-01-14</unadjustedDate>
    <dateAdjustments>
      <businessDayConvention>FOLLOWING</businessDayConvention>
      <businessCenters>
        <businessCenter>GBLO</businessCenter>
        <businessCenter>USNY</businessCenter>
      </businessCenters>
    </dateAdjustments>
  </paymentDate>
</payment>
<-- rest of trade and party information omitted -->
```

- **paymentDate** introduces the payment date and its adjustment rules
- **unadjustedDate** gives the date prior to any adjustments
- **dateAdjustments** gives the adjustment rules
- **businessDayConvention** specifies what adjustment type is used
- **businessCenters** specify what holiday centers to use, using a SWIFT coding scheme (these are London and New York)

The concept of date adjustment is discussed in more detail in Section 5. This type of “adjustable date” structure, and the “dateAdjustments” contained within it, is widely used throughout FpML.

### 4.2.3.5. Completed Example

Below is the completed FpML corresponding to the previous fragments:

```

<dataDocument fpmlVersion="5-3"...>
  <trade>
    <tradeHeader>
      <partyTradeIdentifier>
        <partyReference href="abc" />
        <tradeId>A001</tradeId>
      </partyTradeIdentifier>
      <tradeDate>2012-01-14</tradeDate>
    </tradeHeader>
    <bulletPayment>
      <payment>
        <payerPartyReference href="abc" />
        <receiverPartyReference href="def" />
        <paymentAmount>
          <currency>GBP</currency>
          <amount>15000000.00</amount>
        </paymentAmount>
        <paymentDate>
          <unadjustedDate>2013-01-14</unadjustedDate>
          <dateAdjustments>
            <businessDayConvention>FOLLOWING</businessDayConvention>
            <businessCenters>
              <businessCenter>GBLO</businessCenter>
              <businessCenter>USNY</businessCenter>
            </businessCenters>
          </dateAdjustments>
        </paymentDate>
      </payment>
    </bulletPayment>
  </trade>
  <party id="abc">
    <partyId>ABCUS33</partyId>
  </party>
  <party id="def">
    <partyId>DEFGB2L</partyId>
  </party>
</dataDocument>

```

The diagram illustrates the structure of the FpML document with three main categories of information:

- Trade Identifying Information:** This category includes the `<tradeHeader>` element, which contains the `<partyTradeIdentifier>` (with `<partyReference href="abc" />` and `<tradeId>A001</tradeId>`) and the `<tradeDate>2012-01-14</tradeDate>`.
- Product Information:** This category includes the `<bulletPayment>` element, which contains the `<payment>` element. The `<payment>` element includes `<payerPartyReference href="abc" />`, `<receiverPartyReference href="def" />`, `<paymentAmount>` (with `<currency>GBP</currency>` and `<amount>15000000.00</amount>`), and `<paymentDate>` (with `<unadjustedDate>2013-01-14</unadjustedDate>` and `<dateAdjustments>` containing `<businessDayConvention>FOLLOWING</businessDayConvention>` and `<businessCenters>` with `<businessCenter>GBLO</businessCenter>` and `<businessCenter>USNY</businessCenter>`).
- Party Information:** This category includes the `<party id="abc">` and `<party id="def">` elements, which contain `<partyId>ABCUS33</partyId>` and `<partyId>DEFGB2L</partyId>` respectively.

### 4.2.3.6. Discussion

The above example illustrates some of the key concepts in writing FpML:

- FpML instance documents are wrapped within a root element that specifies the FpML version, schema location, and other information. The example omits at least one important attribute, the XML namespace (xmlns) attribute, that specifies how to find the FpML schema, which specifies the view. (See Section 5 for more information on linking schemas to FpML instance documents).
- Parties are represented using the `<party>` tag. See Section 5 for more information on parties.
- Transactions are described using the `<trade>` tag, and the `<tradeHeader>` tag is used to provide identifying information. See Section 5 for more information on trades.
- Products are represented by a variety of tags under the `<trade>` tag, following the `<tradeHeader>`. See Section 6 for more information about how products are specified.
- Adjustable dates are frequently used to represent date information. See Section 5 for more information on commonly used date building blocks.
- Payments are represented using a variety of reusable types. See Section 5 for more information on commonly used building blocks such as payment.

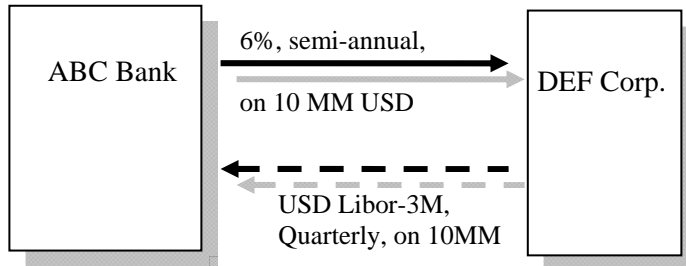


## 4.3. IR Swap Confirmation Message Example

### 4.3.1. Introduction

ABC Bank wishes to send a message to DEF Corp. confirming an interest rate swap. The swap is a 5-year, \$10 million notional fixed/float swap; ABC pays 6% semi-annually, and DEF pays 3 month Libor quarterly.

The diagram below represents the cash flows of the deal:



### 4.3.2. Required Data Attributes

Following are some of the attributes that need to be captured:

- The sender of the message: ABC
- The recipient of the message: DEF
- The parties to the trade: represented by SWIFT BIC codes ABCDUS33 and DEFGGB2L
- The trade date: January 14, 2012
- The effective and termination dates: Jan. 16, 2012 to Jan. 16, 2017
- The date adjustment conventions used for the effective and termination dates and the calculation periods: Modified Following convention
- The business days applicable for the date adjustments: NY business days
- The roll date: calculation periods roll on the 16<sup>th</sup> of the month
- The calculation and payment frequency: semi-annual on the fixed side, quarterly on the floating side
- Rate set in advance or in arrears: rates set in advance (beginning of calculation period)
- Reset adjustments: modified following, using NY and London business days.
- Fixing lag: 2 days (fixing 2 days before rate reset)
- Payment in advance or in arrears: payment in arrears (end of calculation period).
- The notional: 10,000,000 USD
- The fixed rate: 6.00%
- The fixed rate day count fraction: 30/360
- Who is paying fixed (ABC) and float (DEF)
- The floating rate index (a.k.a floating rate option): USD Libor, sourced from Telerate, 3-month tenor
- The floating rate day count fraction: ACT/ACT.ISDA

### 4.3.3. Developing the FpML

#### 4.3.3.1.Headers

##### *FpML root element*

The FpML root element is different than that in the bullet payment example. In this case the FpML instance document type is a “Request Confirmation” message in Confirmation view.

```
<requestConfirmation fpmlVersion="5-3"...>
  <!-- FpML content will go here -->
</requestConfirmation>
```

##### *Message header*

Since this is an FpML message, an FpML message header must be supplied. Among other things, the message header will specify the sender and the recipient.

```
<requestConfirmation fpmlVersion="5-3"...>
  <header>
    <messageId messageIdScheme="http://abc.com/msg">A01</messageId>
    <sentBy>ABCDUS33</sentBy>
    <sendTo>DEFGGB2L</sendTo>
    <creationTimestamp>2012-01-14T15:38:00-00:00</creationTimestamp>
  </header>
</requestConfirmation>
```

- The **messageId** provides an identifier that can be used to refer to the message, for example in a reply
- The **sentBy** field says who sent the message
- The **sendTo** field says who the message was sent to, i.e., the recipient
- The **creationTimestamp** says when the message was created. It is expressed as a date, time, and time zone (offset from UTC)

The message header is discussed in more detail in Section 5.

## *Message correlation and sequencing*

Message correlation is essential to link successive messages together. By principle, there should be a single, well-defined way to link successive messages (such as corrections or retractions of requests or notifications). This should not rely on message or transport level information, but rather use business-level information.

```
<requestConfirmation fpmlVersion="5-3" ...>
  <!--header -->
  <isCorrection>false</isCorrection>
  <correlationId>123</correlationId>
  <sequenceNumber>1</sequenceNumber>
  <onBehalfOf>
    <partReference href="party1"/>
  </onBehalfOf>
  ...
</requestConfirmation>
```

- The **isCorrection** indicates whether the message corrects a previous message.
- The **correlationId** field defines a unique identifier to be used by all subsequent messages related to the same process.
- The **sequenceNumber** field provides the order by which the related messages are linked together.

## *On Behalf Of*

The neutral view principle of FpML when combined with some of the notifications for post-trade processes results in situations where a third party, such as a custodian, cannot easily tell which side of the trade he is supposed to be processing. The *onBehalfOf* field clarifies the party for whom the trade should be processed.

## *Parties, Trade and Trade Header*

The party, trade and trade header information is similar to that for a bullet payment, except for the values of the fields (e.g., the reference number), and therefore are not repeated here. Please see the bullet payment example in the previous section for more detail on how to fill these elements.

### 4.3.3.2.Swap Product

The interest rate swap itself is represented with the FpML “swap” keyword in the product slot of the trade. A swap can contain several swap streams in FpML. The typical number of swap streams is 2, as in this example. Each stream represents a payment stream from one party to the other.

```
<requestConfirmation fpmlVersion="5-3" xmlns="
http://www.fpml.org/FpML-5/confirmation">
  <trade>
    <!-- trade header skipped -->
    <swap>
      <swapStream>
        <!-- fixed stream details go here -->
      </swapStream>
      <swapStream>
        <!-- float stream details go here -->
      </swapStream>
    </swap>
  </trade>
  <!-- party information omitted -->
</requestConfirmation>
```

In this example, the first swap stream will represent the fixed payments from ABC to DEF, while the second will represent the floating payments from DEF to ABC.

### 4.3.3.3.Fixed Stream

Each stream starts with elements indicating which party is paying the cash flows represented by the stream. In the case of the fixed stream, DEF is paying and ABC is receiving:

```
<swapStream>
  <payerPartyReference href="def" />
  <receiverPartyReference href="abc" />
  <!-- etc... -->
```

Key remaining components of fixed interest rate streams include:

- **Calculation period dates**, specifying when calculation periods occur;
- **Payment dates**, specifying when payments occur; and
- **Calculation period amounts**, specifying the amount of the payments.

## Calculation Period Dates

The “calculationPeriodDates” component specifies when the calculation periods occur, including the effective and termination dates, the frequency, and the roll convention:

```
<calculationPeriodDates id="fixDates">
  <effectiveDate>
    <unadjustedDate>2012-01-16</unadjustedDate>
    <dateAdjustments>
      <businessDayConvention>NONE</businessDayConvention>
    </dateAdjustments>
  </effectiveDate>
  <terminationDate>
    <unadjustedDate>2017-01-16</unadjustedDate>
    <dateAdjustments>
      <businessDayConvention>MODFOLLOWING</businessDayConvention>
      <businessCenters id="busCtrs">
        <businessCenter>USNY</businessCenter>
      </businessCenters>
    </dateAdjustments>
  </terminationDate>
  <calculationPeriodDatesAdjustments>
    <businessDayConvention>MODFOLLOWING</businessDayConvention>
    <businessCentersReference href="busCtrs"/>
  </calculationPeriodDatesAdjustments>
  <calculationPeriodFrequency>
    <periodMultiplier>6</periodMultiplier>
    <period>M</period>
    <rollConvention>16</rollConvention>
  </calculationPeriodFrequency>
</calculationPeriodDates>
```

The above specifies:

- The **effective date**: 2012-01-16, with no adjustments.
- The **termination date**: 2017-01-16, adjusted using the modified following convention, using New York business days.
- The **adjustments** to be applied for each calculation period: modified following convention, and the same business centers as the termination date, i.e., New York business center.
- The **calculation frequency**: every 6 months, with a roll date on the 16<sup>th</sup> of the month.

## *Payment Dates*

The payment dates structure specifies how frequently and when the stream cash flows are paid. In this case, the payments are semi-annually, at the end of each calculation period.

```
<paymentDates>
  <calculationPeriodDatesReference href="fixDates" />
  <paymentFrequency>
    <periodMultiplier>6</periodMultiplier>
    <period>M</period>
  </paymentFrequency>
  <payRelativeTo>CalculationPeriodEndDate</payRelativeTo>
  <paymentDatesAdjustments>
    <businessDayConvention>MODFOLLOWING</businessDayConvention>
    <businessCentersReference href="busCtrs" />
  </paymentDatesAdjustments>
</paymentDates>
```

- The **calculationPeriodDatesReference** element explicitly links the payments to the fixed stream calculation periods.
- The **paymentFrequency** element defines that payments occur every 6 months.
- The **payRelativeTo** element states that payments are made at the end of the calculation period.
- The **paymentDatesAdjustments** indicates that payment dates are to be adjusted using the modified following convention, using the set of business centers defined above (i.e., New York business days).

## *Calculation Period Amounts*

The calculation period amounts element specifies how much money is paid during each calculation period.

```
<calculationPeriodAmount>
  <calculation>
    <notionalSchedule>
      <notionalStepSchedule>
        <initialValue>10000000.00</initialValue>
        <currency>USD</currency>
      </notionalStepSchedule>
    </notionalSchedule>
    <fixedRateSchedule>
      <initialValue>0.06</initialValue>
    </fixedRateSchedule>
    <dayCountFraction>30/360</dayCountFraction>
  </calculation>
</calculationPeriodAmount>
```

- The **calculation** element wraps the calculation details.
- The **notionalSchedule** element specifies the notional. In this case, it is 10,000,000 USD, and does not change during the life of the swap.
- The **fixedRateSchedule** element specifies the fixed rate that is paid, 6%.

- The **dayCountFraction** element specifies that the day count fraction used is 30/360.

#### **4.3.3.4.Floating Stream**

With all the elements for the fixed stream defined we can now turn to the definition of the floating stream. A number of elements are similar to those in the fixed stream. There are however, several new elements describing the rules for the periodic rate reset; also, the calculation amounts are different than in the fixed stream.

#### ***Elements similar to fixed stream***

Many of the elements in the floating stream are very similar to those in the fixed stream, and therefore are not shown here:

- The “payerPartyReference” and the “receiverPartyReference” are similar to those in the fixed stream, except that the parties are reversed in role.
- The calculation period dates element is very similar to that in the fixed stream. The only significant difference is that the calculation frequency is 3 Months (i.e., quarterly) instead of 6 Months.
- The payment dates element is also very similar to that in the fixed stream, except that the frequency is quarterly and the payments are based on the floating calculation periods.

## Reset Dates

One element that is new in the floating stream is the `resetDates` element, which defines rules for resetting the floating rate. Within it, the `fixingDates` element describes how the rates are observed relative to the reset dates.

```
<resetDates id="resetDates">
  <calculationPeriodDatesReference href="fltDates" />
  <resetRelativeTo>CalculationPeriodStartDate</resetRelativeTo>
  <fixingDates>
    <periodMultiplier>-2</periodMultiplier>
    <period>D</period>
    <dayType>Business</dayType>
    <businessDayConvention>NONE</businessDayConvention>
    <businessCenters>
      <businessCenter>GBLO</businessCenter>
      <businessCenter>USNY</businessCenter>
    </businessCenters>
    <dateRelativeTo href="resetDates" />
  </fixingDates>
  <resetFrequency>
    <periodMultiplier>3</periodMultiplier>
    <period>M</period>
  </resetFrequency>
  <resetDatesAdjustments>
    <businessDayConvention>MODFOLLOWING</businessDayConvention>
    <businessCentersReference href="busCtrs" />
  </resetDatesAdjustments>
</resetDates>
```

- The **resetDates** element describes resetting rules, and the `id` attribute allows it to be referenced elsewhere in the document.
- The **calculationPeriodDatesReference** explicitly indicates which calculation periods these resets are related to, i.e., the floating calculation period dates.
- The **resetRelativeTo** says that the resets are done at the beginning of each calculation period (“in advance”).
- The **fixingDates** element describes how the rates are observed relative to the reset dates.
  - There is a 2 business day lag from rate observation to reset (it is expressed as -2 days because fixing is before resetting).
  - The business days used for determining the lag are New York and London business days.
  - The fixing dates are relative to the reset dates. See Section 4 for more on the “dateRelativeTo” element.
- The rate **resetFrequency** is quarterly (i.e., there is no averaging).
- Reset dates are **adjusted** using the modified following convention and the previously defined business centers (i.e., New York).



### *Floating calculation amounts*

The calculation amounts element in the floating stream is similar in terms of notional, but is different in terms of the floating rate calculation. In this case the floating rate calculation is quite simple.

```
<calculationPeriodAmount>
  <calculation>
    <notionalSchedule>
      <notionalStepSchedule>
        <initialValue>10000000.00</initialValue>
        <currency>USD</currency>
      </notionalStepSchedule>
    </notionalSchedule>
    <floatingRateCalculation>
      <floatingRateIndex>USD-LIBOR-Telerate</floatingRateIndex>
      <indexTenor>
        <periodMultiplier>3</periodMultiplier>
        <period>M</period>
      </indexTenor>
    </floatingRateCalculation>
    <dayCountFraction>ACT/ACT.ISDA</dayCountFraction>
  </calculation>
</calculationPeriodAmount>
```

- The **notionalSchedule** element is the same as in the fixed stream.
- The **floatingRateCalculation** element describes how the floating rate is computed.
- The **floatingRateIndex** specifies the index (ISDA term, “Floating Rate Option”) used for observing the rate. In this case it is USD-LIBOR from Telerate.
- The **indexTenor** element specifies the tenor of the index, in this case 3 Months.
- The **dayCountFraction** element specifies the day count fraction to be used for computing the calculation period, in this case ACT/ACT.ISDA basis.

#### **4.3.3.5.Discussion**

- A number of the building block elements described above are covered in more detail in Section 5, including message headers, trades, trade headers, date adjustments, and the dateRelativeTo element.
- The interest rate swap product itself is covered in detail in the Interest Rates section of the FpML Specification.
- The example represents a vanilla interest rate swap, omitting much of the complexity that is possible in a full FpML swap. Some of the items that were omitted include:
  - Compounding: This is represented by setting the payment frequency less than the calculation frequency, and adding elements to describe how compounding is done.
  - Averaging: This is done by setting the resetting frequency higher than the calculation frequency, and adding elements to specify the averaging method.
  - Stubs: Both long and short stubs can be specified at the beginning of each stream, and short stubs at the end. This is done by specifying extra dates in the calculationPeriodDates element, and by specifying stub rate calculation details in the calculationPeriodAmount element.
  - Uneven notionals (i.e., amortizing, accreting, or roller coaster): This can be done by supplying steps in the notional schedule.

## FpML 5 User Guide 2012 Edition

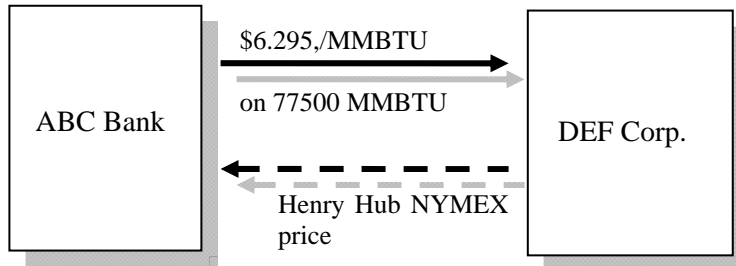
- Uneven fixed rate: This can be done by supplying steps in the fixed rate schedule.
- Rate treatment, spreads, multipliers: A variety of calculations can be specified for floating rates.
- Cash flows: It is possible to list each cash flow explicitly, including the associated calculation periods and rate observations.
- Features: A variety of swap features (a.k.a. “break clauses”) can be specified, including:
  - Mandatory and optional early termination
  - Extension
  - Cancellation

## 4.4. Public Reporting of a Commodity Swap

### 4.4.1. Introduction

ABC Bank wishes to send a message to SDR LLC to publicly report a natural gas swap trade that ABC executed with DEF. The swap is a cash-settled 1-month swap on a total notional quantity of 77,500 MMBTU.

The diagram below represents the cash flows of the deal:



### 4.4.2. Required Data Attributes

Key attributes include:

- Message sent by ABC to SDR
- Product type: Cash settled NG swap
- effective and termination dates: July 1 2006 to July 31 2006
- underlying commodity reference price: NATURAL GAS-HENRY HUB-NYMEX (from ISDA commodity reference price list.), first nearby settlement price on last day of the model
- fixed price: \$6.295/MMBTU
- total quantity (volume): 77,500 MMBTU

### 4.4.3. Developing the FpML

This message needs to be generated using the “publicExecutionReport” message in “Transparency” view.

The message will start as follows:

```

<publicExecutionReport fpmlVersion="5-3"
xmlns=http://www.fpml.org/FpML-5/transparency" ...>
  <header>
    <messageId messageIdScheme="http://abc.com/msg">123</messageId>
    <sentBy>abc</sentBy>
    <sendTo>sdr</sendTo>
    <creationTimestamp>2006-06-30T00:00:00-00:00</creationTimestamp>
  </header>
</publicExecutionReport>
  
```

Subsequent to this, the message header and messaging framework fields must be supplied. See above in the “requestConfirmation” example message for more information.

## FpML 5 User Guide 2012 Edition

After the message framework fields, the sender will supply the payload. This consists of an “originating event” (how the trade originated, in this case as a result of trading activity), and the trade body. Inside the trade there will be a trade identifier (to help the SDR track the trade).

```
<originatingEvent>Trade</originatingEvent>
<trade>
  <tradeHeader>
    <partyTradeIdentifier>
      <issuer>[ID of issuing firm]</issuer>
      <tradeId>[transaction ID issued by that firm]</tradeId>
    </partyTradeIdentifier>
  ...

```

Following this, there will be a number of surveillance fields in the “partyTradeInformation” block:

```
<partyTradeInformation>
  <executionDateTime>xxx</executionDateTime>
  <intentToClear>>false</intentToClear>

```

etc....

Following the end of the trade header, there is a commodity swap element, and some information identifying the asset class and product type, followed by the trade economic details:

```
<commoditySwap>
  <primaryAssetClass>Commodity</primaryAssetClass>
  <productType>Commodity:Energy:Gas:Swap:Cash</productType>
  <effectiveDate>
    <adjustableDate>
      <unadjustedDate>2006-07-01</unadjustedDate>
    </adjustableDate>
  </effectiveDate>
  <terminationDate>
    <adjustableDate>
      <unadjustedDate>2006-07-31</unadjustedDate>
    </adjustableDate>
  </terminationDate>

```

Following this, there is a fixed and a floating leg:

```
<fixedLeg>
  <fixedPrice>
    <price>6.295</price>
    <priceCurrency>USD</priceCurrency>
    <priceUnit>MMBTU</priceUnit>
  </fixedPrice>
  <notionalQuantity>
    <quantityUnit>MMBTU</quantityUnit>
  </notionalQuantity>
  <totalNotionalQuantity>77500.00</totalNotionalQuantity>
</fixedLeg>

```

## FpML 5 User Guide 2012 Edition

The floating leg is similar, but instead of a “fixedPrice” element, there is a “commodity” element that contains the commodity reference price and a few other terms:

```
<floatingLeg>
  <commodity>
    <instrumentId instrumentIdScheme="...">NATURAL GAS-HENRY HUB-
NYMEX</instrumentId>
    <specifiedPrice>Settlement</specifiedPrice>
    <deliveryDates>FirstNearby</deliveryDates>
  </commodity>
```

There is also a “calculation” element that contains a “pricingDates” structure that allows the day distribution to be reported:

```
<calculation>
  <pricingDates>
    <dayDistribution>Last</dayDistribution>
  </pricingDates>
</calculation>
```

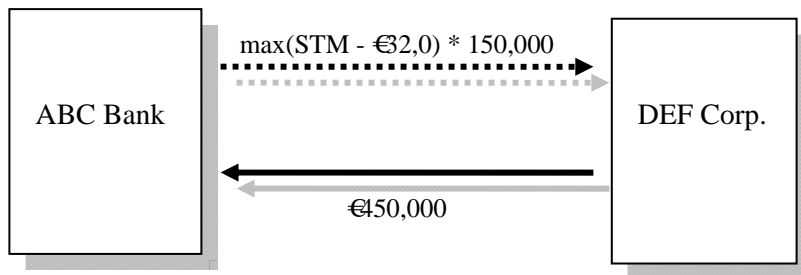
The completed example is in FpML’s example set in Transparency view, in the generated-products/commodity-derivatives folder – it is example #1.

## 4.5. Non-public Reporting of an Equity Option

### 4.5.1. Introduction

ABC Bank wishes to send a message to SDR LLC to non-publicly report (for regulatory recordkeeping purposes) an equity option it has sold to DEF. The option is an American call option with a size of 150,000 options, with an entitlement of 1 share each, on STMMicroelectronics N.V. ordinary shares, with a strike of €32.00. In return DEF pays a premium of €405,000.

The diagram below represents the cash flows of the deal:



### 4.5.2. Required Data Attributes

Key attributes include:

- option style, expiration date: American, September 27, 2005
- option type: Call
- underlying equity: STM-FP
- size of option (entitlement, number of options): 150,000 options of 1 share
- strike: €32.00

In addition, there are a number of surveillance fields, including:

- execution timestamp
- whether the trade is to be cleared
- what regulatory reporting regime the trade is intended for.

See the full FpML example for details of these fields.

### 4.5.3. Developing the FpML

This message needs to be generated using the “nonpublicExecutionReport” message in “Recordkeeping” view.

The message will start as follows:

```
<nonpublicExecutionReport fpmlVersion="5-3"
xmlns=http://www.fpml.org/FpML-5/recordkeeping ...>
<header>
  <messageId messageIdScheme="http://abc.com/msg">123</messageId>
  <sentBy>abc</sentBy>
```

## FpML 5 User Guide 2012 Edition

```
<sendTo>sdr</sendTo>
<creationTimestamp>2011-01-01T00:00:00-00:00</creationTimestamp>
</header>
</nonpublicExecutionReport>
```

Subsequent to this, the message header and messaging framework fields must be supplied. See above in the “requestConfirmation” example message for more information.

After these message framework fields, the sender will supply the payload. This consists of an “originating event” (how the trade originated, in this case as a result of trading activity), and the trade body. Inside the trade there will be a trade identifier (to help the SDR track the trade).

```
<originatingEvent>Trade</originatingEvent>
<trade>
  <tradeHeader>
    <partyTradeIdentifier>
      <issuer>[ID of issuing firm]</issuer>
      <tradeId>[transaction ID issued by that firm]</tradeId>
    </partyTradeIdentifier>
    ...
```

Following this, there will be a number of surveillance fields in the “partyTradeInformation” block:

```
<partyTradeInformation>
  <executionDateTime>xxx</executionDateTime>
  <intentToClear>>false</intentToClear>
  etc...
```

The “reportingRegime” element holds information about what reporting regimes (such as Dodd Frank reporting, ODRF, the Hong Kong Trade Repository, MiFID, etc.) this trade is reported under. This includes the supervisory body (e.g. regulator), the purpose(s) of the message, and the role of the party.

```
<reportingRegime>
  <name>DoddFrankAct</name>
  <supervisorRegistration>
    <supervisoryBody>CFTC</supervisoryBody>
  </supervisorRegistration>
  <reportingRole>ReportingParty</reportingRole>
  <reportingPurpose>RealTimePublic</reportingPurpose>
  <reportingPurpose>PrimaryEconomicTerms</reportingPurpose>
  <mandatorilyClearable>>false</mandatorilyClearable>
</reportingRegime>
```

Documentation provided by SDR implementations can provide more detail on how this is to be populated.

## FpML 5 User Guide 2012 Edition

The equity option itself is held in a element called “equityOptionTransactionSupplement” for typical SDR reporting applications. It typically must contain the asset class and a product ID following the ISDA product taxonomy coding scheme:

```
<equityOptionTransactionSupplement>  
  <primaryAssetClass>Equity</primaryAssetClass>  
  <productType>Equity:Option:PriceReturnBasicPerformance:SingleName  
  </productType>
```

Following this, it will contain references for the buyer and the seller of the option, the type of option (put or call), the underlyer (the STMicroelectronics stock), the exercise details, the strike, the number of options and option entitlement, and the premium, as well as optionally many other details of the product.

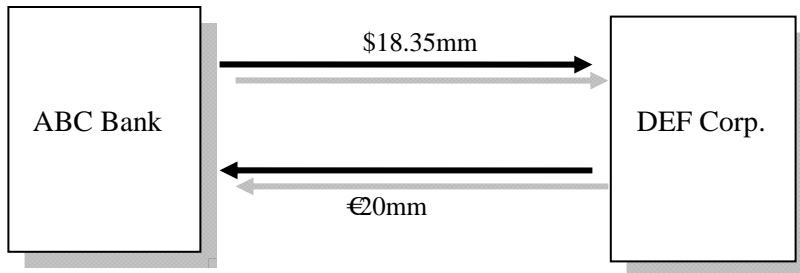
Please see record-ex20 in Recordkeeping view in the “products” folder to see the completed example.



## 4.6. Clearing of an FX forward

### 4.6.1. Introduction

SEF Corp (a swaps execution facility) wishes to send a message to ClearCo LLC (a derivatives clearing organization) requesting that an FX forward be cleared. The FX forward is €20mm sold by DEF, in return for \$18.35mm from ABC.



### 4.6.2. Required Data Attributes

Key attributes of the trade include:

- the value date: Dec. 21, 2001.
- notional amount and currency: €20,000,000 sold by DEF to ABC in return for \$18,350,000 from ABC to DEF.
- It is also possible to supply the forward exchange rate implied by these amounts

### 4.6.3. Developing the FpML

To support clearing requests, the FpML “Confirmation” view must be used, and for this request we need the “requestClearing” message.

```

<requestClearing fpmlVersion="5-3" xmlns=http://www.fpml.org/FpML-5/confirmation" ...>
  <!-- FpML content will go here -->
</requestClearing>
  
```

Subsequent to this, the message header and messaging framework fields must be supplied. See above in the “requestConfirmation” message example for more information.

## FpML 5 User Guide 2012 Edition

Following this, the trade being cleared must be supplied. This trade will have FpML trade header information as usual, followed by an FpML representation of an FX forward. This will list the two currency amounts that are exchanged (together with which party is paying and receiving each currency) and the value date:

```
<fxSingleLeg>
  <exchangedCurrency1>
    <payerPartyReference href="party2"/>
    <receiverPartyReference href="party1"/>
    <paymentAmount>
      <currency>EUR</currency>
      <amount>20000000</amount>
    </paymentAmount>
  </exchangedCurrency1>
  <exchangedCurrency2>
    <payerPartyReference href="party1"/>
    <receiverPartyReference href="party2"/>
    <paymentAmount>
      <currency>USD</currency>
      <amount>18350000</amount>
    </paymentAmount>
  </exchangedCurrency2>
  <valueDate>2001-12-21</valueDate>
</fxSingleLeg>
```

The example #7 in the Confirmation view “business-processes/clearing” folder contains the complete message. In addition, that example supplies the optional forward exchange rate information.

## 5. The Organization of FpML

### 5.1. Introduction

#### 5.1.1. Purpose

This section summarizes key concepts underlying the design of the FpML standard, in particular some of the architectural principles. In addition, it describes some of the more important cross-product structures that are used widely throughout the standard.

#### 5.1.2. Overview

The section is organized as follows:

- Architecture: a summary of FpML architecture principles and how FpML is built on XML.
- Document/messaging structure: an introduction to FpML messaging.
- Key transaction structures: key structures used to represent trades in FpML.
- Reusable components: commonly reused building block components.
- Validation: an introduction to the FpML validation framework.

### 5.2. Architecture

#### 5.2.1. Architecture Principles

The way that FpML 5 uses XML is documented in the FpML Architecture 3.0 available at <http://www.fpml.org/spec>.

Some key points that the FpML Architecture defines include:

- How the data is represented in XML. For example, business data should be held in XML element content, not in XML attributes. There is also coverage for “schemes”, which allow list values to be described.
- Naming conventions. For example, element and attribute names should be in “lowerCamelCase” (mixed capitalization, starting with a lower case).
- How to reference within and between documents. This is typically done with “id” and “href” attributes that link to them.
- How versioning is to be represented. This is done in part using the “version” attribute.
- How namespaces are to be used.
- How extensions are to be created.

Following is an example of how XML can be written, following the FpML architectural rules:

```
<lowerCamelCaseElement>  
  <anotherElementWithData>DataInElement</anotherElementWithData>  
  <anElementReference href="target"/>  
  <theTargetElement id="target">Some Data</theTargetElement>  
</lowerCamelCaseElement>
```

Following is an example of well-formed XML that doesn't follow the FpML architectural rules:

```
<ILLEGAL_ELEMENT_NAME>  
  <BadElemName BadAttribute="Data Not Allowed Here"/>  
</ILLEGAL_ELEMENT_NAME>
```

The errors in the last example include:

- Illegal naming conventions for elements and attributes (all should be lowerCamelCase)
- Use of attributes to hold business data

### 5.2.2. Schema and Instance Documents

So far in this user guide we have focused on examples of FpML instance documents. In XML terms, these are called “instance” documents. These documents are instances (or if you prefer, examples) of documents whose structure is defined by a “schema” document. The schema defines the rules for building an instance document, i.e., “grammar” and “vocabulary”, or “syntax”. In much of the FpML specification documentation, the emphasis is on the schema, as opposed to the instances. This is because the instances will be different for every use, while the schema is the same for all.

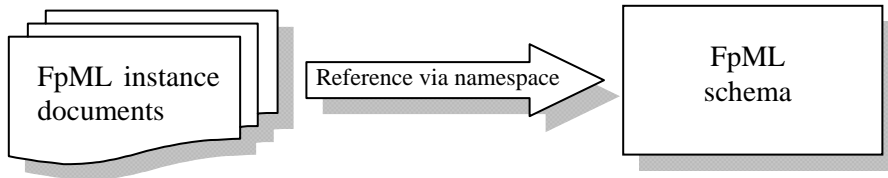
The schema and the instances are linked as follows: The root element in an FpML instance document has an “xmlns” (XML namespace) attribute, not described so far, that specifies the “namespace” (i.e., the vocabulary) that is used by the instance. The example below shows all of the attributes that the FpML root element will typically contain:

```
<dataDocument  
  fpmlVersion="5-3"  
  xmlns= "http://www.fpml.org/FpML-5/confirmation"  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xsi:schemaLocation="http://www.fpml.org/FpML-5/confirmation fpml-main-5-3.xsd">
```

- The **fpmlVersion** attribute is an FpML attribute to define the FpML version that was used to create the document.
- The **xmlns** attribute specifies the namespace that is followed by the document. This is the official link between the instance document and the schema. The XML document processor is supposed to use this namespace to find the schema that was used. In the above example, the FpML 5 confirmation view namespace (i.e., <http://www.fpml.org/FpML-5/confirmation>) is specified as the default namespace. Any element without a namespace prefix (e.g., <payment>, <currency>) is therefore considered to be in the FpML 5 confirmation view namespace. Different views may adjust product definitions to suit particular business processes or stage of the trade life cycle.
- The **xmlns:xsi** attribute defines the xsi namespace, so the parser will be able to understand what to do with the **xsi:schemaLocation** attribute described below (i.e., to recognize it as a special, pre-defined attribute).
- The **xsi:schemaLocation** attribute is an *optional* attribute that gives a hint to the XML document processor of where to find the schema document. Note that the XML document processor is free to ignore this and use its own version of the schema file.

## FpML 5 User Guide 2012 Edition

The relation of these documents can be diagrammed as follows:



Two standard schema-related namespaces worth describing include the following:

- For XML schema itself, “<http://www.w3.org/2001/XMLSchema>”. This namespace is used to indicate that the associated elements are being used to define an XML schema. This namespace is usually identified with the prefix “xsd” or “xs”. It is normally found only in XML schema definition documents, not in instance documents.
- For XML schema instances, “<http://www.w3.org/2001/XMLSchema-instance>”. This namespace is used within instance documents for hints or advice to XML document processors about how to use the XML schema associated with the instance document. It is usually identified with the prefix “xsi”. It is normally found in instance documents and not in schema documents. FpML uses the xsi namespace to tell the parser which message type to validate the document against.

The FpML schema is also divided into a number of smaller subschema files for maintainability:

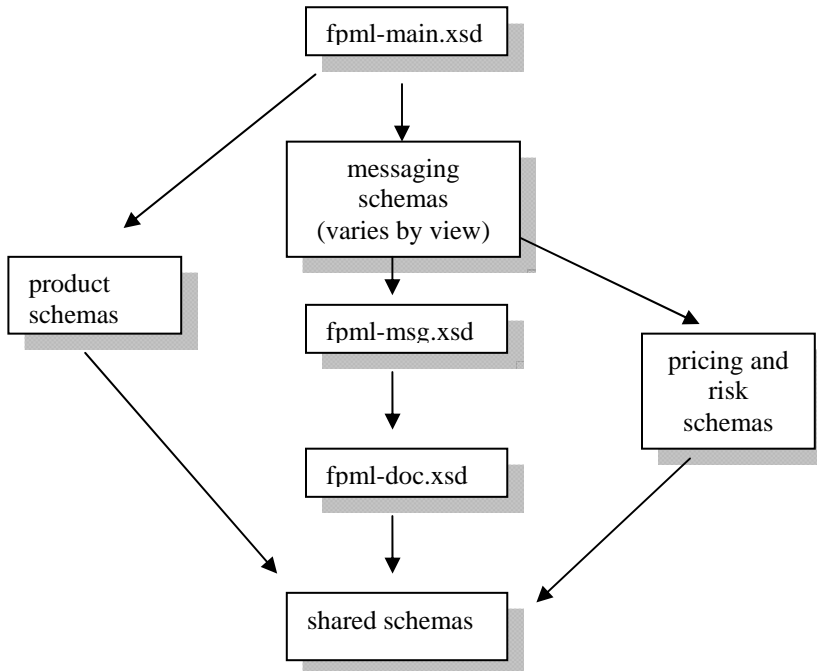
- The overall collection of subschema files is defined in `fpml-main-5-x.xsd`.
- Top level structures like document and trade are defined in `fpml-doc-5-x.xsd`.
- Shared components and enumeration types are defined in `fpml-shared-5-x.xsd` and `fpml-enum-5-x.xsd`.
- Messaging is defined in `fpml-msg-5-x.xsd` and in several business process specific schema files, discussed in Section 7.
- Derivative products are defined in a variety of asset class-specific files (e.g., `-irs-`, `-cd-`, `-eqd-`, `-fx-`), and underlying assets are defined in `fpml-asset-5-x.xsd`.

where x stands for the minor FpML version. E.g., FpML 5.3.

This organization is documented in more detail in the first two sections of any version of the FpML specification (i.e., Section 1 – *Introduction and Overview*, Section 2- *FpML Overview*).

## FpML 5 User Guide 2012 Edition

Following is a summary of the dependencies of the different FpML sub-schemas.



Note that the available messaging schemas will vary by view. Some of the view-specific messaging subschemas include:

- fpml-confirmation-processes.xsd (confirmation)
- fpml-recordkeeping-processes.xsd (recordkeeping)
- fpml-reporting.xsd, fpml-collateral-processes.xsd, fpml-reconciliation.xsd (reporting)
- fpml-transparency-processes.xsd (transparency)

### 5.2.3. Architectural Changes Between Versions

Over the years, FpML’s architecture has changed slightly. The following gives an overview of the more significant changes that have occurred between the different versions. More detailed information on these changes and the rationale for them can be found in the respective versions where this change was first used.

- Up to version 3.0, the FpML syntax was defined in a document called a DTD (Document Type Definition). This format is more compact than XML schema, but not as powerful nor as descriptive. With the introduction of schema, there were a large number of changes in the internal organization of the schema.
- Top level structural changes. Between versions 1 and 3, a couple of structural changes were introduced, in particular:
  - In version 1.0, products were wrapped in a “product” tag. This was dropped from subsequent versions.

#### *Version 1.0:*

```
<trade>
  <tradeHeader> . . . </tradeHeader>
  <product>
    <swap> . . . </swap>
  </product>
</trade>
```

#### *Version 2.0+:*

```
<trade>
  <tradeHeader> . . . </tradeHeader>
  <swap> . . . </swap>
</trade>
```

- Up to version 2.0, parties were within the trade element. They were moved out of the trade element in version 3.0.

#### *Versions 1.0 – 2.0:*

```
<trade>
  <tradeHeader> . . . </tradeHeader>
  <swap>. . .</swap> <!-- or "product", or other product -->
  <party id="abc">
    <partyId>ABC123</partyId>
  </party>
</trade>
```

#### *Versions 3.0 and 4.x:*

```
<trade>
  <tradeHeader> . . . </tradeHeader>
  <swap> . . . </swap> <!-- or other product -->
</trade>
<party id="abc">
  <partyId>ABC123</partyId>
</party>
```

- There have been some changes in the handling of list values :
  - Up to version 3.0, all of these were done using FpML “schemes”, with scheme defaults in the FpML root element. Schemes look like the following:

## FpML 5 User Guide 2012 Edition

```
<currency currencyScheme="http://schemeURI">USD</currency>
```

- From version 4.0 onward, some of the smaller and more stable schemes were moved to XML Schema “enumerations”. Also, scheme defaults were removed from the FpML header element and put into the schema. Enumerations look like the following:

```
<businessDayConvention>FOLLOWING</businessDayConvention>
```

- Intra-document referencing syntax has changed slightly:
  - *Versions 1.0 – 2.0:*  
`<payerPartyReference href="#abc" />`
  - *Versions 3.0 – 4.x:*  
`<payerPartyReference href="abc" />`
- From version 4.0 onward, the majority of the elements are defined locally to avoid name collisions.
- Each FpML version has its unique namespace.

FpML 4.0 REC	<a href="http://www.fpml.org/2003/FpML-4-0">http://www.fpml.org/2003/FpML-4-0</a>
FpML 4.1 REC	<a href="http://www.fpml.org/2004/FpML-4-1">http://www.fpml.org/2004/FpML-4-1</a>
FpML 4.2 REC	<a href="http://www.fpml.org/2005/FpML-4-2">http://www.fpml.org/2005/FpML-4-2</a>
FpML 4.3 REC	<a href="http://www.fpml.org/2007/FpML-4-3">http://www.fpml.org/2007/FpML-4-3</a>
...	
FpML 4.8 REC	<a href="http://www.fpml.org/2010/FpML-4-8">http://www.fpml.org/2010/FpML-4-8</a>
FpML 4.9 REC	<a href="http://www.fpml.org/2010/FpML-4-9">http://www.fpml.org/2010/FpML-4-9</a>

- Starting in version 5.0, minor FpML versions share the same namespace. Each view has a different namespace to distinguish between the different types of applications.

FpML 5.0 REC	<a href="http://www.fpml.org/FpML-5/confirmation">http://www.fpml.org/FpML-5/confirmation</a>	(confirmation view)
	<a href="http://www.fpml.org/FpML-5/reporting">http://www.fpml.org/FpML-5/reporting</a>	(reporting view)
FpML 5.1 REC	<a href="http://www.fpml.org/FpML-5/confirmation">http://www.fpml.org/FpML-5/confirmation</a>	(no change)
	<a href="http://www.fpml.org/FpML-5/reporting">http://www.fpml.org/FpML-5/reporting</a>	(no change)
...		
FpML 5.3 REC	<a href="http://www.fpml.org/FpML-5/confirmation">http://www.fpml.org/FpML-5/confirmation</a>	(no change)
	<a href="http://www.fpml.org/FpML-5/reporting">http://www.fpml.org/FpML-5/reporting</a>	(no change)
	<a href="http://www.fpml.org/FpML-5/recordkeeping">http://www.fpml.org/FpML-5/recordkeeping</a>	(new view)
	<a href="http://www.fpml.org/FpML-5/transparency">http://www.fpml.org/FpML-5/transparency</a>	(new view)

- The FpML root element was removed in version 5.0 and replaced by multiple root elements. Type substitution on the root element (xsi:type) is no longer used to select the message type and its associated content model. The <FpML> element has been removed from the schema in favor of distinct root elements. Multiple roots is easier to understand than type substitution at the root level and certain tools also had problems processing xsi:type.



### 5.3. Document / Messaging Framework

FpML instance documents are all based on the “Document” type, in other words the “FpML” element is defined to be of type “Document”. There are two main classifications (subtypes) of Documents:

- Data documents are documents that contain only data, not messages. These are relatively unstructured and are intended primarily for non-messaging uses or uses within proprietary messaging frameworks. They are represented using the “DataDocument” type.
- Message documents are intended to be used for communicating between firms or systems. They are represented using a variety of types, as described below.

Following are common characteristics of messages:

- Message documents are divided into three main types:
  - Notification messages are used to send unsolicited information.
  - Request messages are used to ask for something to be done.
  - Response messages are used to reply to “Request” messages.
- All messages have a message header. A message header contains information about the sender, the recipient, and various pieces of message identification information. The interest rate swap example in Section 4.3.3.1 shows an example of a simple message header. In addition, the message header can hold information such as:
  - “copyTo” addresses, for recipients that should get a copy of information.
  - An “inReplyTo” element, which can be used to indicate the message that this replies to.
  - An “expiryTimestamp”, for indicating when the message should be ignored as too old.
  - A digital signature, for authenticating the message sender.

#### *Sample message header*

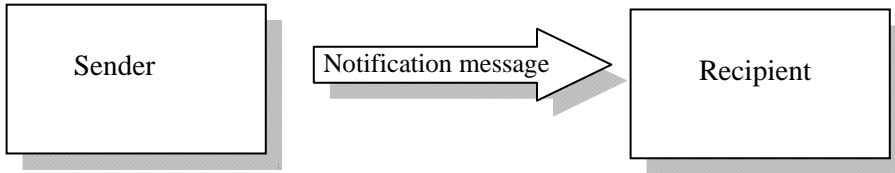
```
<executionNotification fpmlVersion="5-3"...>
  <header>
    <messageId messageIdScheme="http://abc.com/msg">A1</messageId>
    <inReplyTo messageIdScheme="http://def.com/msg">B2</inReplyTo>
    <sentBy>ABCDUS33</sentBy>
    <sendTo>DEFGGB2L</sendTo>
    <creationTimestamp>2012-06-02T15:38:00-00:00</creationTimestamp>
    <expiryTimestamp>2012-06-02T18:38:00-00:00</expiryTimestamp>
  </header>
  <!-- message content goes here -->
</executionNotification>
```

Section 7 of this user guide provides a more detailed overview of the messaging framework.

#### 5.3.1. Sample Message Flows

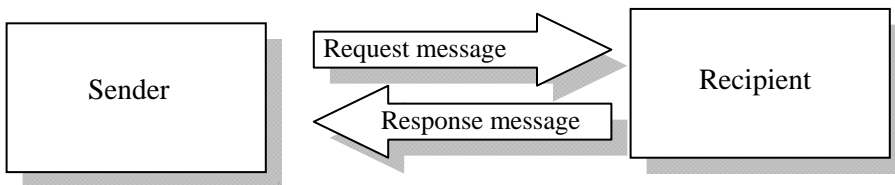
##### *Notification Message Flow:*

The standard notification message flow is a one way flow from the sender to the recipient. However, notification messages may have been triggered by a previous message, so they are allowed (but not required) to contain an “inReplyTo” element to identify that original message.



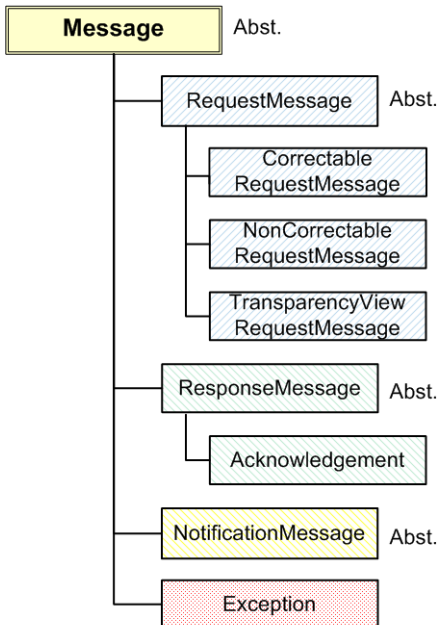
***Request/Response Message Flow:***

The standard request message flow is a two-way, request-response interaction. Typically, the response will indicate the message that is being responded to in the “inReplyTo” field.



***Base messages:***

The following diagram shows the main types of messages (e.g., request / response / notification). All FpML messages are derived from one of these base message types.



### ***Message Naming Convention:***

Messages follow this naming convention:

- requestXxx
- xxxAcknowledgement
- xxxException
- requestXxxRetracted
- xxx[Status] or xxx[Response]

For the consent negotiation process, for example, the messages include:

- requestConsent – initiating request
- consentAcknowledgement - acknowledgement
- consentException - exception
- requestConsentRetracted – retraction of the request
- consentGranted - response
- consentRefused – response

See the Messaging Framework insert at the end of the user guide for a full list of messages.

### **5.3.2. Generic Messaging / Multi-Event Workflows**

In FpML 5, most workflows are designed to work consistently for a number of events, including new trades and post-trade events such as novations, amendments, and terminations. For example, one set of messages (e.g., execution or confirmation or allocation) can be reused, for many trading events (different payloads e.g., trade, amendment, option) using the same flow of messages. The messages can be applied in a number of different contexts. For example, the same <requestExecution> message can be used to execute an increase, an amendment, a termination, a novation, a trade.

### **5.3.3. Views**

'Views' are flavors of the FpML schema that adjust product definitions to suit particular business processes or stage of the trade life cycle. There are four views in FpML 5.3. Starting with FpML 5, each release of the specification is published as a set of schemas, one for each view. Each view has its own unique namespace URI. The idea of 'views' has existed since the inception of FpML but all the initial design and development focused on just the product 'confirmation' view. See sections 2.2 or 7.3 for more detailed information regarding FpML views.

### **5.3.4. More Information**

The FpML specification provides more information in the Business Process Architecture section. It discusses the message types and structures in detail and describes several business processes using interaction diagrams.

It is important to note that the FpML Messaging framework is message transport independent. In other words, it is not linked to any specific messaging transport protocol. Thus it can be used with any message transport, from simple ones like e-mail or HTTP, to message-oriented middleware. Section 7 of this user guide provides a more detailed overview of the messaging framework.

## 5.4. Other Top Level Structures

Some of the key, top level structures in an FpML instance document include:

- Party
- Trade
- Trade Header
  - partyTradeIdentifier
  - partyTradeInformation
- Portfolio

### 5.4.1. Party

- The Party element is used to identify a participant in a transaction. This participant does not need to be a principal to the transaction. Instead, it could be a broker, arranger, agent, etc.
- Since version 4.2 the Party structure contains information about accounts.
- Until version 4.1, the roles of the parties were typically identified using party references. Since version 4.2 a complete set of roles are defined within the tradeSide structure (see following Trade section).
- For example, the following parties may be defined in an FpML instance document:

```
<party id="abc">
  <partyId>ABCDUS33</partyId>
  <account id="abcacc">
    <accountId>1234345</accountId>
  </account>
</party>
<party id="def">
  <partyId>DEFGGB2L</partyId>
  <account id="defacc">
    <accountId>789123</accountId>
  </account>
</party>
```

### 5.4.2. Trade

- The Trade element is used to hold transaction information. The Trade element includes
  - Identification information in a tradeHeader.
  - A product, providing economic details of the transaction
  - Optional additional information in subsequent slots, including items such as:
    - otherPartyPayments, for fees and commissions
    - brokers
    - calculation agents
    - collateral
    - documentation
    - governing law
    - allocations

*Example trade structure, with many (not all) slots filled*

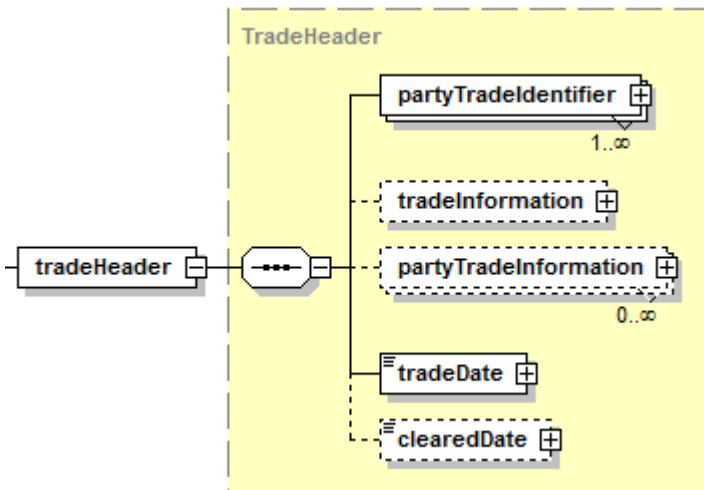
```

<trade>
  <tradeHeader>
    <partyTradeIdentifier>
      <partyReference href="abc"/>
      <tradeId>A001</tradeId>
    </partyTradeIdentifier>
    <tradeDate>2012-01-14</tradeDate>
  </tradeHeader>
  <!-- product info skipped -->
  <collateral>
    <!-- collateral info goes here -->
  </collateral>
  <documentation>
    <!-- documentation info goes here -->
  </documentation>
  <governingLaw>GBEN</governingLaw>
</trade>

```

### 5.4.3. Trade Header

The tradeHeader is a structure within the trade meant to capture trade-related information which is not product specific.



#### 5.4.3.1. PartyTradeIdentifier

The partyTradeIdentifier is a structure defining one or more trade identifiers allocated to the trade by a party. A link identifier allows the trade to be associated with other related trades, e.g. trades forming part of a larger structured transaction. It is expected that for external communication of trade there will be only one tradeId sent in the document per party.

Regulations have introduced the requirement for swaps market participants and utilities to report trades using a shared “unique swap identifier” (USI), also known as the Unique Transaction Identifier (UTI) in other jurisdictions.

The USI/UTI is captured in FpML using a two-field representation within the partyTradeIdentifier. For example,

```
<partyTradeIdentifier>
  <issuer issuerScheme="http://www.fpml.org/coding-
  scheme/external/cftc/issuer-identifier">101ABCD123</issuer>
  <tradeId tradeIdScheme="http://www.fpml.org/coding-scheme/external/unique-
  transaction-identifier">12345678901234567890123456789012</tradeId>
</partyTradeIdentifier>
```

The *tradeId* is an issuer-specific transaction identifier. The *issuer* element identifies the organization that issued the trade identifier. Both fields combined constitute the USI/UTI.

The issuer coding scheme URI (<http://www.fpml.org/coding-scheme/external/cftc/issuer-identifier>) is specified to reference a CFTC-created identifier to be used as a prefix for the firm that issues the USI.

### 5.4.3.2. PartyTradeInformation

The PartyTradeInformation is a type defining party-specific additional information that may be recorded against a trade. The structure is intended to capture, in particular, surveillance fields that would be used by various regulators to monitor systemic risk.

In the recordkeeping view the structure is mandatory and available under *trade\tradeHeader\partyTradeInformation*. whereas is is optional in the transparency, and reporting views.

In the transparency view, a subset of surveillance fields from the PartyTradeInformation is made available within a party-independent *trade\tradeHeader\tradeInformation* block.

An example of a surveillance field is the *reportingRegime*. It allows the organization to specify which if any relevant regulators or other supervisory bodies this is relevant for, and what reporting rules apply.

Refer to the PartyTradeInformation type definition within the fpml-doc.xsd subschema for the full list, the definition and usage of the different surveillance fields.

### 5.4.4. Portfolio

The portfolio structure allows a collection of trades to be grouped together, identified by party trade identifier:

```
<portfolio id="port1">
  <tradeId id="t1" tradeIdScheme="http://b.com/tids">12</tradeId>
  <tradeId id="t2" tradeIdScheme="http://b.com/tids">23</tradeId>
  <tradeId id="t3" tradeIdScheme="http://b.com/tids">34</tradeId>
  <tradeId id="t4" tradeIdScheme="http://b.com/tids">45</tradeId>
  <tradeId id="t5" tradeIdScheme="http://b.com/tids">56</tradeId>
</portfolio>
```

## 5.5. Building Block Components

FpML has a large number of building block components that are used widely throughout the schema. The following types of components are described in more detail in this section:

- Currency and location related components
- Date-related components
- Payment

There are of course many other shared components. Some of these, not described here, include components for identifying legal entities and instruments and components for describing settlement instructions and documentation.

### 5.5.1. Currency and Location Related Components

There are a number of building block components for holding currency and location-related information. Some of these include:

- Currency
- Money
- Business Center, Business Centers
- Exchange ID
- Address

Several of these have been used in earlier examples. “Currency” is used to hold a currency code (by default an ISO currency code), while “Money” holds a currency and an amount. In the following, “paymentAmount” is of type “Money”:

```
<paymentAmount>
  <currency>USD</currency>
  <amount>1000000</amount>
</paymentAmount>
```

BusinessCenter holds a city or other business location, and BusinessCenters holds a collection of these. The default coding scheme for BusinessCenter is a four character SWIFT code in which the first two characters represent the country, and the last two represent the city. The following example represents London (GB + LO) and New York (US + NY) holidays:

```
<businessCenters id="pmtBusCtrs">
  <businessCenter>GBLO</businessCenter>
  <businessCenter>USNY</businessCenter>
</businessCenters>
```

This collection can be referenced by a BusinessCentersReference:

```
<businessCentersReference href="pmtBusCtrs"/>
```

This allows one set of business centers to be defined and then reused throughout a document. Most places where a list of business centers can be provided, a reference can be used instead.

For identifying exchanges, the “ExchangeId” type is used, e.g.,

```
<exchangeId>NYSE</exchangeId>
```

FpML does not currently standardize the values used in this scheme, so it is up to implementers to decide the coding scheme that they wish to use for this element.

The “Address” type and types contained within it can be used to identify locations. An example address is:

```
<routingAddress>
  <streetAddress>
    <streetLine>123 E 4 St.</streetLine>
  </streetAddress>
```

```
<city>New York</city>
<state>NY</state>
<country>US</country>
<postalCode>10003</postalCode>
</routingAddress>
```

### 5.5.2. Date-related Components

A number of components in FpML are used frequently for representing dates and date-related information. Some are described in the following sections.

#### 5.5.2.1. Adjustable date, business day conventions

The Adjustable Date type holds an unadjusted date and adjustment rules (which are held in the “BusinessDayAdjustments” type). A number of elements throughout the schema are defined to be of type “AdjustableDate”, or of type “BusinessDayAdjustments”. The adjustments rules include a business day convention and a list of business centers. The following example states that the termination date is January 14, 2015, subject to the *Following* business day convention, using the payment business days defined in this FpML instance document:

```
<terminationDate>
  <unadjustedDate>2015-01-14</unadjustedDate>
  <dateAdjustments>
    <businessDayConvention>FOLLOWING</businessDayConvention>
    <businessCentersReference href="pmtBusCtrs" />
  </dateAdjustments>
</terminationDate>
```

In addition to the “AdjustableDates” type, there are several similar types that hold variations on this. These include:

- **AdjustableDate2:** Date adjustments can be omitted or referenced
- **AdjustableDates:** Holds several adjustable dates, using the same adjustment rules.
- **AdjustableOrRelativeDate(s):** Holds either AdjustableDate(s) or Relative Date(s) (described below).

#### 5.5.2.2. Period, Frequency, Offset

In addition to date types that allow specific dates to be defined, there are date types that allow time periods to be defined. These include:

- **Period:** A defined number of periods.
- **Frequency:** A defined number of period but used for specifying payment or calculation frequencies at which the value T (Term) is applicable.
- **Offset:** A defined number of periods, additionally allowing Business or Calendar to be specified when the offset is in Days.

Period is commonly used to hold tenors, expressed in numbers of days, weeks, months, or years. The following expresses that the index tenor is 3 months:

```
<indexTenor>
  <periodMultiplier>3</periodMultiplier>
  <period>M</period>
</indexTenor>
```



The following example states that the payment days offset is 5 business days:

```
<paymentDaysOffset>
  <periodMultiplier>5</periodMultiplier>
  <period>D</period>
  <dayType>Business</dayType>
</paymentDaysOffset>
```

### 5.5.2.3.Date Relative To

The “dateRelativeTo” element allows a reference to a specific date to be included in a structure. This allows an unambiguous definition of the base date used to compute a relative date. The following states that something is to be defined relative to the trade date:

```
<tradeDate id="TradeDt">2012-01-15</tradeDate>
```

Then:

```
<dateRelativeTo href="TradeDt" />
```

### 5.5.2.4.Relative Dates

Leveraging the Offset and dateRelativeTo components described above, it is possible to define a date as being calculated relative to another date.

The RelativeDateOffset extends the Offset to include:

- The date adjustments used to compute the relative date.
- An explicit reference to the date upon which the relative date is based.

An example of this is the following:

```
<fixingDates>
  <periodMultiplier>-2</periodMultiplier>
  <period>D</period>
  <dayType>Business</dayType>
  <businessDayConvention>NONE</businessDayConvention>
  <businessCenters id="fixingBusinessCenters0">
    <businessCenter>EUTA</businessCenter>
  </businessCenters>
  <dateRelativeTo href="resetDates0"/>
</fixingDates>
```

This states that fixing dates are 2 Target (EUTA) Business Days before the reset dates identified by “resetDates0”. No adjustment is necessary because “business” days are used for the offset.

RelativeDateOffset is used by the “RelativeDate” and “RelativeDates” type to allow a date or a series of dates to be defined relative to another date or date series.



Info

A technical paper on *financial date calculations in FpML*, developed by the Validation Working Group is available for download from <http://www.fpml.org/documents/technical.html>  
This document describes how to perform calculations with dates and intervals to derive new dates and schedules.

### 5.5.3. Payment

Another structure used frequently throughout FpML is the “Payment” structure. It is used to represent a payment of a specified amount of money from one party to another at a specified time. It uses many of the components described above. This type is applied for defining payments such as the bullet payment in Section 4.2, premium payments, commission payments, and others.

A payment consists of:

- References to the payer and receiver
- The amount
- The unadjusted (and optionally adjusted) payment date
- Optional categorization and settlement information

An example payment is the following:

```
<premium>
  <payerPartyReference href="partyA"/>
  <receiverPartyReference href="partyB"/>
  <paymentAmount>
    <currency>EUR</currency>
    <amount>100000</amount>
  </paymentAmount>
  <paymentDate>
    <unadjustedDate>2012-08-30</unadjustedDate>
    <dateAdjustments>
      <businessDayConvention>FOLLOWING</businessDayConvention>
      <businessCenters>
        <businessCenter>EUTA</businessCenter>
      </businessCenters>
    </dateAdjustments>
  </paymentDate>
</premium>
```

Description: Party A pays Party B EUR 100,000 on August 30, 2012; subject to the Following convention using Target business days.

## 5.6. FpML Validation Framework

The FpML business rules validation framework allows business constraints to be applied to FpML instance documents to ensure that they are meaningful. This section discusses how FpML business rules validation relates to XML Schema validation, what is provided with the FpML validation framework, and how validation enhances the FpML specification.

### 5.6.1. What Does Validation Add?

To understand what FpML validation does, it will be useful to follow an example document as it passes through successive levels of increasingly stringent checking. In the following, we consider the case of a payment that will be made on January 14, 2012, subject to adjustment by the Following business day convention.

#### 5.6.1.1. Well-formedness

A first attempt to represent this payment might be as follows:

```
Payment-date: January 14, 2012
Adjustment: Following
```

## FpML 5 User Guide 2012 Edition

An XML parser will report errors with this document, as it is not well-formed XML (in fact, it is not XML at all). To resolve these errors, the document could be changed as follows:

```
<PaymentDate adjustment="Following">January 14, 2012</PaymentDate>
```

This is well-formed XML, and a non-validating XML parser will be able to parse this.

However, this is not FpML, so an FpML-based application trying to process this document will not be able to understand it, because the document structure doesn't use FpML terminology and structuring concepts.

### 5.6.1.2. Syntactic Validation

Continuing with the above example, the document could be restructured into schema-valid FpML as follows:

```
<paymentDate>  
  <unadjustedDate>2012-01-14</unadjustedDate>  
  <dateAdjustments>  
    <businessDayConvention>FOLLOWING</businessDayConvention>  
  </dateAdjustments>  
</paymentDate>
```

This document will now pass FpML schema validation when processed with a validating XML parser. This type of document can be described as “syntactically” valid or “schema valid”. It says that the document uses names and structures in accordance with those defined in the schema.

### 5.6.1.3. Semantic Validation

A document can be syntactically valid and still not completely understandable, because some business meaning is missing or incorrect. In the preceding example, the sample document meets the requirements of the FpML schema, but the adjustments cannot be understood fully because the business days to be used for the adjustments are not specified. The FpML schema does not require the business days to be specified, because if the business day convention is “None” the business days are meaningless. However, in this case the business adjustments can only be fully understood if the business days are specified.

This type of validation is called “semantic” validation - it determines whether the FpML instance document is understandable. There are a series of business rules that an FpML instance document must pass to be meaningful. These business rules are in addition to the syntax rules defined by the schema.

The corrected document is as follows:

```
<paymentDate>  
  <unadjustedDate>2012-01-14</unadjustedDate>  
  <dateAdjustments>  
    <businessDayConvention>FOLLOWING</businessDayConvention>  
    <businessCentersReference href="primaryBusinessCenters"/>  
  </dateAdjustments>  
</paymentDate>
```

### Another example: comparing dates

Below is another example illustrating the need for semantic validation. Consider two dates, the trade effective date and termination date.

- Effective date: January 15, 2017
- Termination date: January 15, 2012

A trade effective date that starts after the termination date does not make business sense and is obviously invalid data.

In FpML, these dates would be represented as shown below.

```
<effectiveDate>
  <unadjustedDate>2017-01-15</unadjustedDate>
  <dateAdjustments>
    <businessDayConvention>NONE</businessDayConvention>
  </dateAdjustments>
</effectiveDate>
<terminationDate>
  <unadjustedDate>2012-01-15</unadjustedDate>
  <dateAdjustments>
    <businessDayConvention>NONE</businessDayConvention>
  </dateAdjustments>
</terminationDate>
```

This document would pass FpML schema validation (these are indeed valid dates, taken individually) but would not be valid from a semantic/business standpoint. An FpML validation rule could be defined to disallow this combination of dates.

### 5.6.2. What Information Does Validation Provide?

FpML validation rules are defined in the Validation Architecture section of the Specification <http://www.fpml.org/spec> (See section 4 of any version of the specification.)

The FpML business validation rules define a large number of constraints on FpML instance documents similar to the above. For each constraint, the following is provided:

- An English-language version of the rule
- A technical expression of the rule
- Optionally, examples of documents that pass the rule (valid test cases)
- One or more examples of documents that fail the rule (invalid test cases)

## FpML 5 User Guide 2012 Edition

Below is a figure illustrating one of the many validation rules defined in the specification. Rule ird-14 defined for interest rate products provides a way to compare dates and ensure that the termination date must occur after the effective date.

<b>ird-14 (Mandatory)</b>
<u>English Description:</u> Context: CalculationPeriodDates (complex type) If <code>effectiveDate</code> exists and If <code>terminationDate</code> exists, then <code>terminationDate/unadjustedDate</code> must be after <code>effectiveDate/unadjustedDate</code>
<u>XPath Description:</u> Context: CalculationPeriodDates [exists( <code>effectiveDate</code> )] [exists( <code>terminationDate</code> )] <code>terminationDate/unadjustedDate</code> gt <code>effectiveDate/unadjustedDate</code>
Test cases: [ <a href="#">Invalid</a> ] [ <a href="#">Invalid</a> ]

An FpML implementation that implements rule ird-14, for example, as part of its business validation layer, would be able to detect the problem discussed in the previous subsection.

A number of implementations of the rules have been developed in a variety of languages.

The rules that have been defined fall into a variety of categories. Some of the types of rules that have been defined include the following:

- **Date constraints:** dates must be in a particular order or pattern. For example, termination date must be after effective date.
- **Structural constraints:** if A is present, then B must also be. For example, if a stub period is defined, the payment amount calculations for it must also be defined.
- **Data value based constraints:** if element A has a given value, then .... For example, if the business day convention isn't NONE, the business centers must be supplied.
- **Special cases:** unusual handling / requirements for particular specialized products/markets. For example, the SFE (Sydney Futures Exchange) has special date roll rules.

In addition, the FpML specification allows an FpML instance document to record which validation rule set(s) it is asserted to follow. This is done with the <validation> tag as follows:

```
<requestConfirmation fpmlVersion="5-3" . . . >
  <header>
    <messageId messageIdScheme="http://abc.com/msg">A01</messageId>
    <sentBy>ABCDUS33</sentBy>
    <sendTo>DEFGGB2L</sendTo>
    <creationTimestamp>2012-01-14T15:38:00-00:00</creationTimestamp>
  </header>
  <validation validationScheme="http://abc.com/rules">IRD</validation>
  <validation validationScheme="http://abc.com/rules">DEF-MSTR</validation>
</requestConfirmation>
```

This states that the document is believed to be valid according to ABC's IRD and DEF-MSTR rule sets. Presumably this is ABC's rules for interest rate products and its rules for its master agreement with DEF.

This information may be used by a recipient to process the message, but in many cases the recipient will apply its own validation policies independent of the sender's.

### **5.6.3. How Does Validation Work Together with the Specification?**

The FpML Validation Rules defined in the specification are intended to be respected by all applications generating FpML, and as guidance to applications processing FpML on what validation rules to implement. These validation rules are considered to be required to be followed by an FpML application.

In addition, custom validation rules can also be defined in a similar way for other purposes. These might include:

- enforcing system constraints
- enforcing business policies
- enforcing rules defined by master agreements
- enforcing platform-specific policies

In this way, implementers of FpML-based applications can leverage tools using the FpML validation framework to identify and process validation constraints.

Since FpML 4.2, including all 5.x versions, validation rules have been and are published as a part of the FpML specification. Versions prior to FpML 4.2 have the validation rules and framework included in a separate package.

## 6. The FpML Product Framework

### 6.1. Introduction

This section describes FpML’s framework for identifying financial products, both derivatives and simpler financial products. It is intended to explain how the type of product is specified and how new products are added, but not to explain any particular product in detail. The section is organized into two subsections:

- Derivative products
- Underlying instruments

### 6.2. Derivative Products

#### 6.2.1. Product Substitution Framework

Derivative products form the core of what FpML is typically used to represent. They represent OTC derivatives products such as interest rate swaps, swaptions, FRAs, credit default swaps, equity swaps, commodity swaps, equity options, commodity options and FX options.

Where the trade element in FpML represents the non-economic information about a transaction (e.g., identifying information, documentation, etc..), the “Product” type represents the economic details of a transaction. While the possible “trade” details are similar for most or all transactions, the “Product” details differ depending on the type of financial product that was traded.

The basic trade structure is as follows:

```
<trade>
  <tradeHeader> . . . </tradeHeader>
  <product> . . . </product>
  <!-- more trade information -->
</trade>
```

However, after version 1.0 the “product” keyword described above never actually appears in the FpML instance document. Instead, the “product” tag is an “abstract” tag that is a member of a “substitution group”. In practical terms, this means that instead of putting in a “product” tag, you can put in any element that is a member of the “product” substitution group.

For example, “creditDefaultSwap” is a member of this group, as is “equityOption”. That means that either of the following is allowed in FpML:

```
<trade>
  <tradeHeader> . . . </tradeHeader>
  <creditDefaultSwap> . . . </creditDefaultSwap>
  <!-- more trade information -->
</trade>
```

Or:

```
<trade>
  <tradeHeader> . . . </tradeHeader>
  <equityOption> . . . </equityOption>
  <!-- more trade information -->
</trade>
```

The first signals that the trade is a credit default swap, while the second signals that the trade is an OTC equity option.



Info

See the *FpML Products Framework* insert at the end of the user guide for an overview of all the products and their base types.

A softcopy can be downloaded at  
[Http://www.fpml.org/documents/FpML5-products-framework.pdf](http://www.fpml.org/documents/FpML5-products-framework.pdf)

### Strategies

In addition, there is a special kind of product called a “Strategy” that combines other products. This allows the creation of complex structures by combining existing products.

The following is an example of a strategy, combining an FX option with an FX forward to create a delta-hedged option:

```
<trade>
  <tradeHeader> . . . </tradeHeader>
  <strategy>
    <productType>Delta-Hedge</productType>
    <fxSimpleOption>
      <productType>European FX Option</productType>
      <buyerPartyReference href="DEF"/>
      <sellerPartyReference href="ABC"/>
      <!-- more fx option details here -- >
    </fxSimpleOption>
    <fxSingleLeg>
      <!-- FX spot/forward details here -->
    </fxSingleLeg>
  </strategy>
```



## 6.2.2. Product Summary

The following table contains a summary of the products covered by FpML, along with the FpML versions that the products were introduced in:


	Asset Class	Product	Product Variants	Since
	n/a	Strategy		3.0
		<b>genericProduct</b> (formerly nonSchemaProduct in 5.0 and 5.1) (to represent an OTC derivative transaction whose economics are not fully described using an FpML schema.) <b>standardProduct</b> (to represent a standardized OTC derivative transaction whose economics do not need to be fully described using an FpML schema because they are implied by the product ID)		5.2 (RV) 5.3
Interest Rate Derivatives	IRD	bulletPayment*		2.0
		capFloor		2.0
		fra		1.0
		swap	break clauses (cancelable, extendible, early termination), asset swap (since 4.2), inflation swap (since 4.2) Brazilian swap (since 4.4)	
		swaption	American, European, Bermuda, Cash/Physical	2.0
Foreign Exchange	FX	fxSingleLeg	Spot, Forward, Non-Deliverable Forwards	3.0
		fxSwap		3.0
		fxOption (fxSimpleOption in 3.x/4.x)	Knock-in and knock-out options, Side averaging rate option*, barrier option*	3.0
		fxDigitalOption*		3.0
		termDeposit*	Dual Currency Deposit	4.0
Credit Derivativ	CD	creditDefaultSwap	CDS index (since 4.1), CDS Basket (since 4.2), Loan CDS (since 4.3), CDS on Mortgage (since 4.3)	4.0
		creditDefaultSwapOption		4.3
Equity Derivatives	Equity	equityOption*	various option features/exercise types	3.0
		equityOptionTransactionSupplement		4.1
		brokerEquityOption*		4.1
		equityForward		4.1
		returnSwap (formerly equitySwap)		4.0^
		equitySwapTransactionSupplement		4.1
	Dividend	dividendSwapTransactionSupplement		4.3
	Variance	varianceSwap*		4.3
		varianceSwapTransactionSupplement		4.3
		varianceOptionTransactionSupplement		4.6
Correlation	correlationSwap		4.3	
	Bond Options	bondOption	bond, convertible bond	4.3
Commo-dities		commoditySwap	financially and physically settled (4.6)	4.5
		commodityForward	bullion	4.6
		commodityOption	financially-settled, forwards	4.8
		commoditySwaption	physically-settled options (formerly in the commodityOption from 4.8 until 5.2)	5.3

\* Product not supported in the *Transparency* view of version 5.

^ Indicates that the product was first defined in this version as equitySwap, however, the structure was changed substantially in 4.1 and specially 4.2, when it was renamed returnSwap. New implementations are strongly advised to use 4.2 or above for this product.

The following equity products should be modeled as follows in the transparency view:

- equityOption -> use equityOptionTransactionSupplement
- brokerEquityOption -> use equityOptionTransactionSupplement
- varianceSwap -> use varianceSwapTransactionSupplement)

	<p>The FX product model has been extensively enhanced and refactored in FpML 5.1.</p> <p>New implementations should consider using the latest FpML 5.x FX framework instead of the 4.x model.</p>
---	---

FpML 5

### 6.2.3. Adding New Products

To create a new product in your own namespace that can be used in an FpML trade, you should do the following:

- Create a type derived from the “Product” type, e.g., “MyProduct”.
- Create a new global element, e.g., “myProduct”, whose type is “MyProduct”. Make that element a member of the “fpml:product” substitution group.

See Section 8 for more information on extending FpML.

### 6.2.4. ISDA Product Taxonomy

As part of the ongoing effort to improve the OTC derivatives infrastructure, ISDA developed an implementation plan mid-2011 to define a standardized taxonomy (classification) for OTC derivatives. The ISDA OTC Taxonomies support regulatory mandates to increase transparency through public and regulatory reporting. The taxonomies can be downloaded from the ISDA website at <http://www.isda.org/otc-taxonomies-and-upi/>

The final taxonomies are included in the FpML data standard to facilitate the reporting process. The Product Taxonomy Scheme (<http://www.fpml.org/coding-scheme/product-taxonomy>) based on the latest version of the ISDA Product Taxonomy is set as the default scheme for the productType element in FpML. E.g.,

```
<requestConfirmation>
...
<trade>
...
  <swap>
    <productType>InterestRate:IRSwap:FixedFloat</productType>
    ...
    <assetClass>InterestRate</assetClass>
```

## 6.3. Underlying Assets

### 6.3.1. Usage

In addition to representing complex derivative products, FpML has a representation of a fairly large number of simple, standardized financial instruments. These instruments, called “UnderlyingAssets” in FpML, can be used for a variety of purposes:

- As underlying assets in various derivatives, including:
  - Equity options
  - Equity swaps
  - Asset swaps
- As reference obligations in credit default swaps
- For a variety of purposes in pricing and risk, including:
  - For describing curve inputs
  - For describing benchmark asset prices

It is also expected that use of the underlying assets will increase in future versions of FpML.

### 6.3.2. Underlying Asset Substitution Framework

The underlying asset framework is very similar to the product framework. In places where underlying assets are used, a substitution group can allow the asset to be substituted as required. However, underlying assets are different from “products” in the sense that all are derived from UnderlyingAsset rather than Product. In addition, UnderlyingAsset defines some standard data fields available for all assets.

For example, “equity” is an FpML underlying asset, and here is a use of “equity” as a basket component:

```
<basketConstituent>
  <equity>
    <instrumentId instrumentIdScheme="http://www.fpml.org/coding-
      scheme/external/instrument-id-bloomberg">TIT.ME</instrumentId>
    <description>Telecom Italia spa</description>
    <currency>EUR</currency>
    <exchangeId exchangeIdScheme="http://www.fpml.org/coding-
      scheme/external/exchange-id-MIC">Milan Stock Exchange</exchangeId>
  </equity>
  <constituentWeight>
    <openUnits>432000</openUnits>
  </constituentWeight>
</basketConstituent>
```

However, if instead a “bond” (a different FpML underlying asset) were desired, the FpML might look like:

```
<basketConstituent>
  <bond>
    <instrumentId
      instrumentIdScheme="http://www.fpml.org/spec/2002/instrument-id-
      ISIN-1-0">JP310860A032</instrumentId>
    <couponRate>0.0213</couponRate>
    <maturity>2011-03-08</maturity>
  </bond>
  <constituentWeight>
    <openUnits>432000</openUnits>
  </constituentWeight>
</basketConstituent>
```

### 6.3.3. Summary of Underlying Assets

The following table summarizes the underlying assets available in FpML. All of these assets are defined in the FpML asset subschema, e.g., fpml-asset-5-3.xsd.

<b>Underlying Asset</b>	<b>Description</b>
bond	a security typically delivering interest coupon payments and requiring the repayment of a principal amount at its maturity
cash	an asset in monetary form, typically held in a bank account
commodity	a commodity underlying asset
convertibleBond	a bond that can under specified circumstances be converted into equity (e.g., common stock) in the issuer
deposit	a term deposit, a money market instrument of fixed duration yielding a specific interest rate
equity	an ownership share in an entity, typically common stock
exchangeTradedFund	a fund whose units can be traded on an equity exchange
future	identifies the underlying asset when it is a listed future contract (a standardized, daily-settled contract traded on an exchange for the purchase or sale of an asset at some specified date in the future)
fx	identifies a simple underlying asset type that is an FX rate. Used for specifying FX rates in the pricing and risk
index	an asset whose value is based on the value of a set of instruments, typically equities
loan	an underlying asset that is a loan
mortgage	a mortgage backed security
mutualFund	a pooled investment vehicle that takes positions in a variety of financial instruments, typically equities
rateIndex	an interest rate index, such as USD LIBOR
simpleFra	a simple, benchmark Forward Rate Agreement
simpleIrSwap	a simple, benchmark Interest Rate Swap
simpleCreditDefault Swap	a simple, benchmark Credit Default Swap

## 7. The FpML Messaging Framework

This section describes how business processes are represented in FpML 5, summarizing them into several categories, depending on where they happen within the trade lifecycle.

Beyond this introduction, the FpML specification provides more information in the Business Process Architecture section. It discusses the message types and structures in detail and describes several business processes using interaction diagrams. Developers interested in better understanding these business process definitions are directed to that documentation.

### 7.1. Messages

Message documents are intended to be used for communicating between firms or systems. They are represented using a variety of types, as described below.

Following are common characteristics of messages:

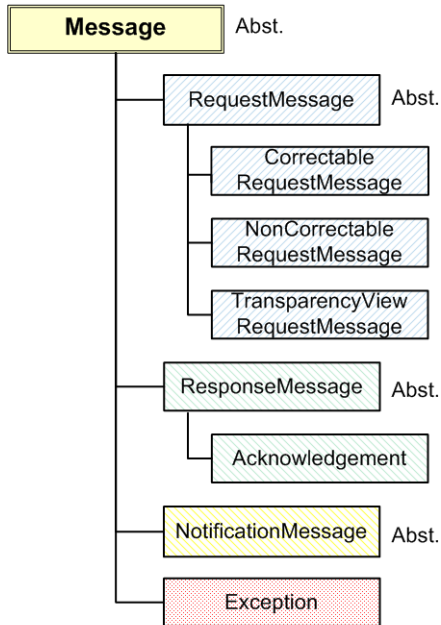
- Message documents are divided into three main types:
  - Notification messages are used to send unsolicited information.
  - Request messages are used to ask for something to be done.
  - Response messages are used to reply to “Request” messages.
- All messages have a message header. A message header contains information about the sender, the recipient, and various pieces of message identification information.

#### *Sample message header*

```
<executionNotification fpmlVersion="5-3"...>
<header>
<messageId messageIdScheme="http://abc.com/msg">A1</messageId>
<inReplyTo messageIdScheme="http://def.com/msg">B2</inReplyTo>
<sentBy>ABCDUS33</sentBy>
<sendTo>DEFGGB2L</sendTo>
<creationTimestamp>2012-06-12T15:38:00-00:00</creationTimestamp>
<expiryTimestamp>2012-06-12T18:38:00-00:00</expiryTimestamp>
</header>
<!-- message content goes here -->
</exectutionNotification>
```

### 7.1.1.1. Base Messages

The following diagram shows the main types of messages (e.g., request / response / notification). All FpML messages are derived from one of these base message types.



Three new messages subtypes of Request Message have been introduced in FpML 5:

- Correctable Request Message: defines the content model for a request message that can be subsequently corrected or retracted.
- Non Correctable Request Message: defines the content model for a request message that cannot be subsequently corrected or retracted.

### 7.1.2. Message Naming Convention / Patterns

FpML defines a general pattern of messages (Request or Notification, Acknowledgement, Exception, Retraction and optionally, Response/Status) to ensure consistent processes across trades and post-trade events, observable completion, consistent message correlation and retraction and consistent error reporting.

Messages follow this naming convention, where Xxx represents the name of the process:

- requestXxx
- xxxAcknowledgement
- xxxException
- requestXxxRetracted
- xxx[Status] or xxx[Response]

For the consent negotiation process, for example, the messages include:

- requestConsent – initiating request
- consentAcknowledgement – acknowledgement
- consentException – exception

- requestConsentRetracted – retraction of the request
- consentGranted – response
- consentRefused – response

See the *FpML Messaging Framework* insert (at the end of the user guide) for a full list of supported messages.

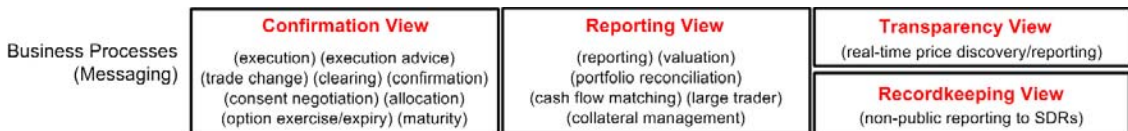
### 7.1.3. Message Correlation

Message correlation is essential to link successive messages together. As noted earlier, there were issues with inconsistent message correlation in version 4.x. In FpML 5.x, there is a single, well-defined way to link successive messages (such as corrections or retractions of requests or notifications). Successive messages are “correlated” (linked together) using a new, explicit correlationId element. The correlation ID is assigned by the initiator. Subsequent responses use the correlation ID to link back to the original request.

## 7.2. Business Processes

Messages are divided into a series of business processes. Because of the large number of messages, the messages are divided into a number of categories, according to whether the messages are sent prior to trade execution, during the confirmation process, or afterward.

The following diagram illustrates the (processes) and the category (i.e. “view”) where they can be found (more information on views in the next section)



For example, consent negotiation (getting permission to do something) is a business process. For each business process, in general the following messages will be available:

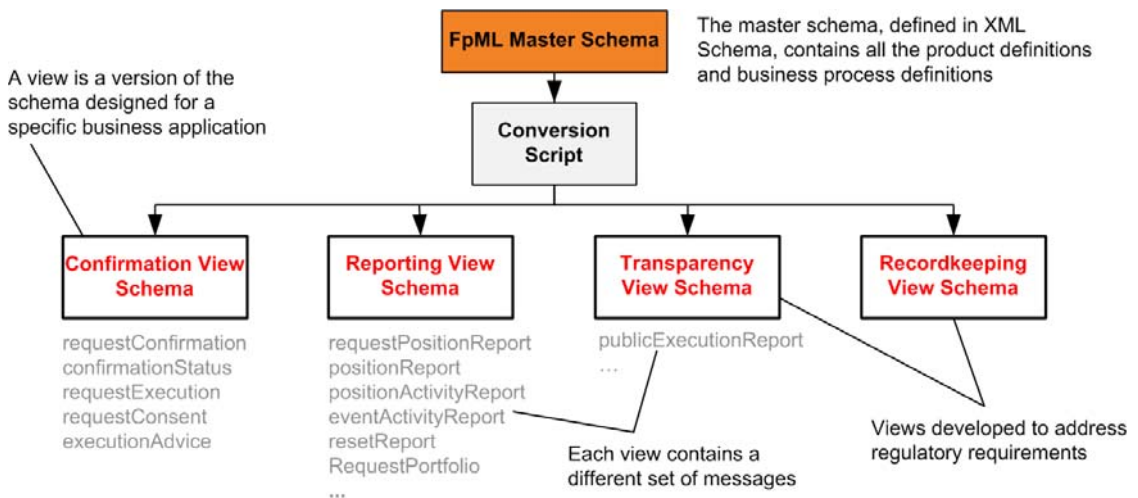
- process request or notification message – to initiate the process. In some cases the process may be initiated by a notification message rather than a request message.
- process acknowledgment message – to acknowledge that the request or notification was received.
- process exception message – to report that a request or notification cannot be processed.
- process request/notification retracted message – to withdraw the original request or notification.
- possibly, process-specific response messages

### 7.3. Views

A view is a version of the schema focused on a particular business area or application. Views are intended to provide multiple product representations, each one of them appropriate for the business process where it is defined.

The **rationale** for the concept of "views" is to provide a consistent representation of key information across many types of business process, while allowing the set of mandatory and optional data to vary between processes. For example, a firm reporting on an interest rate swap may not provide information such as: payment date and reset date definitions on the floating side, or the business day adjustments that were used, etc. However, all of these pieces of information are crucial for confirming that swap once it is traded. So for confirmation view we want these pieces of information to be mandatory, while they are optional for pre-trade view.

FpML maintains a single master schema from which multiple views can be generated. The following diagram illustrates how the different versions of the schema, or views, are generated from the master schema using a conversion script. The master schema contains annotations, with view-specific instructions (e.g., make this element optional in view X, put this element only in view Y)



- End users will use a view-specific schema, not the master schema.
- In version 5.1 there are two views:
  - The “**Confirmation**” view: This view is intended to be used for confirming the precise details of contracts and post-trade business events. This view is intended to have similar characteristics to the FpML versions 1-4 product representation, i.e. a very detailed product representation capturing the details needed for a transaction confirmation.
  - The “**Reporting**” view: This view is intended to be used for reporting trading and business activities and positions (including as part of STP flows), as well as processes such as reconciliation. This view has a moderately loose product representation; it requires key economic information such as the notional, key dates, and parties, but leaves other information optional.



## FpML 5 User Guide 2012 Edition

- In version 5.3, two additional views are added to address regulatory reporting:
  - The “**Transparency**” view is intended to be used for reporting price forming information about executed transactions to the public by reporting parties and execution platforms. This view is intended to provide only the key product economics that are appropriate for standardized transactions, and not customizations and detailed information not required for price discovery, for example details of date adjustment rules and the like..
  - The “**Recordkeeping**” view is intended to be used for reporting the Primary Economic Terms of derivative transactions to Swaps Data Repositories from entities including market participants, execution platforms, and clearing or confirmation services. This view is intended to have similar characteristics to the FpML "confirmation view" product representation, i.e. a very detailed product representation capturing the details needed for a transaction valuation; it may not include all documentation and legal terms.

### 7.4. The Use of Multiple Namespaces

- The FpML 5.x grammar is distributed as multiple XML schemas, each of which is specialized to suit a particular set of related business processes. This allows product, and other business object, representations to be adjusted to each usage (e.g. strict for confirmation, looser for reporting). Each view has a different namespace to distinguish between the different types of applications.

<http://www.fpml.org/FpML-5/confirmation> (confirmation view)  
<http://www.fpml.org/FpML-5/reporting> (reporting view)  
<http://www.fpml.org/FpML-5/recordkeeping> (recordkeeping view)  
<http://www.fpml.org/FpML-5/transparency> (transparency view)

- Starting in version 5.0, minor FpML versions share the same namespace. For example, the namespace for the confirmation view is the same for FpML 5.0, 5.1, 5.2 and 5.3.

<http://www.fpml.org/FpML-5/confirmation> (shared namespace across minor versions)  
<http://www.fpml.org/FpML-5/transparency> (introduced in 5.3, shared onward)

This feature enables documents to have greater longevity. For example, with the new 5.x schemas it is possible to process documents against future compatible versions of the schema. This feature is useful in the task of migrating across time or supporting multiple versions simultaneously.

### 7.5. Multiple Root Elements

Different element names are used to distinguish between different message types. <requestExecution>, <executionAdvice>, <clearingStatus>, <positionReport> are just a few of the many root elements available in FpML 5.x. See the *FpML Messaging Framework* insert (at the end this user guide) for a complete list of root element names available.

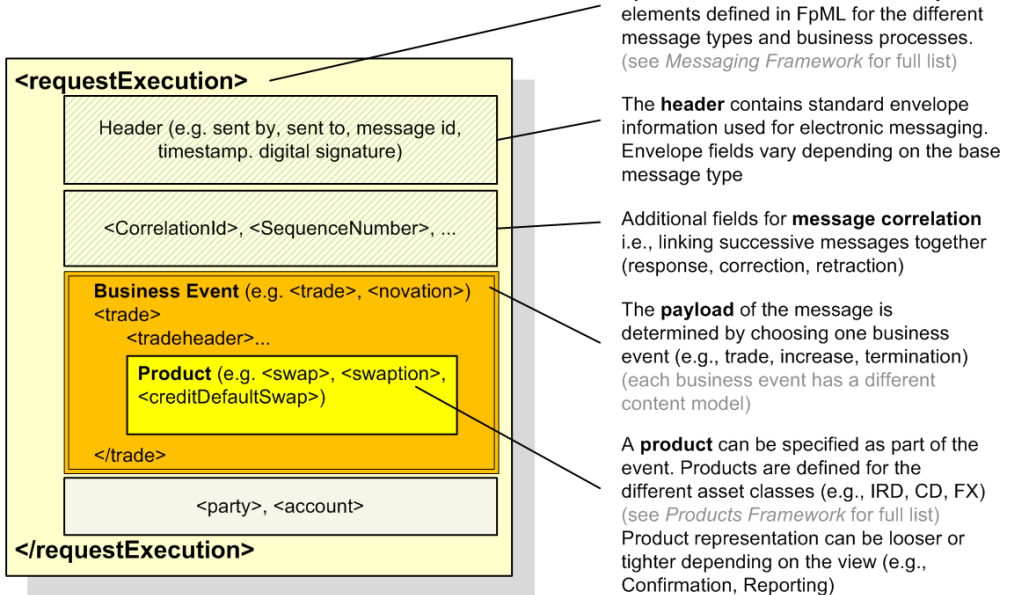
## 7.6. Generic Messaging / Multi-Event Workflows

FpML 5 introduces the idea of **Generic Business Processes** to address limitations in the version 4 framework.

In FpML 5, most workflows are designed to work consistently for a number of events, including new trades and post-trade events such as novations, amendments, and terminations. For example, one set of messages (e.g., execution or confirmation or allocation) can be reused, for many trading events (different payloads e.g., trade, amendment, option) using the same flow of messages. The messages can be applied in a number of different contexts. For example, the same `<requestExecution>` message can be used to execute an increase, an amendment, a termination, a novation, a trade.

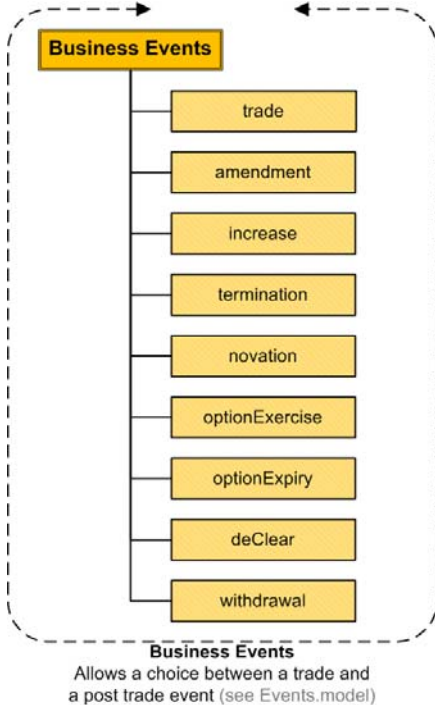
The following diagram illustrates this concept of a generic message carrying different payloads.

Structure of a typical FpML message: e.g., `requestExecution`



## 7.7. Generic (Multi-Event) Flows

Each view contains a different set of business processes that are applied to the following events. All events use the same messages to support the processes.



## 7.8. Pre-Trade

Pre-trade business processes are not supported in FpML 5.

Some draft *request for quote* messages have been developed in the master schema, however, they haven't received enough review from the community. As a result, the Pre-trade view has not been officially published in FpML 5.

## 7.9. Confirmation

The Confirmation view supports the following business processes:

- Execution
- ExecutionAdvice
- Trade Change
- Consent Negotiation
- Confirmation
- Clearing
- Allocation
- Option Exercise/Expiry

For example, the following messages are available for these processes:

### Execution Process

- requestExecution
- requestExecutionRetracted
- executionAcknowledgement
- executionException

- executionNotification
- executionRetracted
- Confirmation Process
- requestConfirmation
- requestConfirmationRetracted
- confirmationAcknowledgement
- confirmationException
- confirmationRefused
- confirmationStatus
- confirmationDisputed

### 7.10. Reporting

The FpML reporting capability is designed to allow institutions to report details about derivative portfolios with a variety of levels of granularity, in a format consistent with the normal messaging and confirmation specification.

The reporting capability is intended to support a number of use cases, including but not limited to:

- Regulatory reporting
- Reporting between counterparties for reconciliation purposes
- Reporting from service providers to clients on portfolios maintained by the service.

Some of the main reports include:

- **Position Report** - The position report is intended to allow a firm to report the current status of a portfolio of positions.
- **Position Activity Report** - The position activity report is a variation of the Position Report that is specifically targeted to reporting changes in position, for example changes between two points in time.
- **Event Activity Report** - The Event Activity Report allows a firm to report on a series of related or unrelated business events, such as trades, amendments, novations, or terminations. These events will typically be applicable for a specific context (e.g. a single account or product) over a specific time period (e.g. a day or month). It does not allow valuation information or servicing events to be reported.
- **Reset report**, added in FpML 5.0, allows a firm to report on the fixing of a floating rate index, and to indicate the positions that are affected by the fixing.
- **Cashflow Matching** - The FpML standard for cashflow matching is aimed at providing a messaging standard to support the ISDA guideline developed by the ISDA Operations Committee.

The Reporting view supports the following business processes:

- Reporting
- Valuation
- Portfolio Reconciliation
- Cash Flow Matching
- Collateral Management (see next section)

For example, the following valuation messages are defined:

- requestValuationReport
- valuationReport

- valuationReportAcknowledgement
- valuationReportException

### **7.11. Recordkeeping**

For the recordkeeping view, a set of messages are defined for non-public execution reporting:

- **nonpublicExecutionReport**
- nonpublicExecutionReportRetracted
- nonpublicExecutionReportAcknowledgement
- nonpublicExecutionReportException

The non-public execution report message is provided for reporting parties and agents (such as execution platforms) to report on the execution of trades to Swaps Data Repositories for regulatory reporting purposes. The non-public execution report contains a complete representation of the economic terms of each product.

- **valuationReport** – allows valuation information to be supplied for a number of trades at the same time. This valuation information can include mark to markets, quotations, risk measures, and other observed or calculated values.
- A set of **verification messages**, shared across multiple views are also available. These messages allow a firm to either verify or dispute records contained within an SDR. See “shared messages” section below.

### **7.12. Transparency**

For the transparency view, the following messages are defined for public execution reporting:

- **publicExecutionReport**
- publicExecutionReportRetracted
- publicExecutionReportAcknowledgement
- publicExecutionReportException

The public execution report message is provided for reporting parties and agents (such as execution platforms) to report on the execution of trades to the public, via Swaps Data Repositories or third party messaging services. The public execution report contains a simplified version of the product model covering only the commonly used terms that affect pricing.

- A set of **verification messages**, shared across multiple views are also available. See “shared messages” section below.

### **7.13. Shared Messages**

A number of messages are shared across multiple views where applicable:

- **messageRejected** – an exception message sent to inform another system that some exception has been detected.

## FpML 5 User Guide 2012 Edition

- **requestRetransmission** – a message to request that a message be retransmitted. The original message will typically be a component of a group of messages, such as a portfolio or a report in multiple parts. (not available in the transparency view)
- **requestEventStatus /eventStatusResponse** – a set of messages allowing one party to query the status of one event (trade or post-trade event) previously sent to another party.
- **serviceNotification** – a notification messages that allow a service to report information on status and other alerts to users.
- **verificationStatusNotification/Acknowledgement/Exception** – verification status messages support the ability for users to dispute and verify positions in the SDR, attached is the reference to the regulation that they derived this requirement from.



Info

See the *FpML Messaging Framework* insert at the end of the user guide for the full list of messages available for the different views.

The softcopy can be downloaded at  
[Http://www.fpml.org/documents/FpML5-messaging-framework.pdf](http://www.fpml.org/documents/FpML5-messaging-framework.pdf)

### 7.14. Collateral Management

The FpML standard has been extended to cover the following collateral management processes:

- Margin Call
- Collateral Substitution
- Interest payment

Collateral messages were introduced in version 5.1 and were based on the requirements defined by the ISDA Collateral Committee in the November 12, 2009 publication Standards for the Electronic Exchange of OTC Derivative Margin Calls

([http://www.isda.org/c\\_and\\_a/pdf/Electronic-Messaging.pdf](http://www.isda.org/c_and_a/pdf/Electronic-Messaging.pdf)).

### 7.15. Loan Syndication

FpML versions 5.0-5.3 do not include support for Syndicated Loans. A new, refactored loan framework is being developed and may be included in a later version of FpML.

The Loan messaging framework continues to be developed in FpML 4.x series but is being refactored in FpML 5. Firms who have not started implementation should consider loan support in FpML 5.

## 8. Customizing FpML

### 8.1. Introduction

#### 8.1.1. Purpose of the Section

This section discusses techniques for adjusting FpML to better meet specific application requirements. This includes adding information not included in FpML as well as preventing FpML features from being used when they are not required.

Many of the techniques described here are described in more detail in the FpML Architecture Specification 3.0, section 6 (<http://www.fpml.org/spec>).

These techniques apply to FpML 4.x and 5.x.

#### 8.1.2. Overview

The section is organized as follows:

- **Wrapping:** a traditional approach for extending FpML.
- **Type extension:** a more modern approach for extending FpML.
- **Type restriction:** a more modern approach for constraining FpML.
- **Extending product coverage:** detail on adding product coverage.
- **Extending message/business process coverage:** detail on adding messages.
- **Migrating extensions:** a discussion on managing extensions to FpML as FpML evolves.

## 8.2. Wrapping

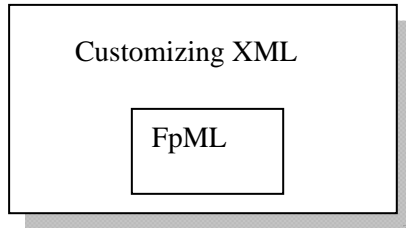
### 8.2.1. Explanation

Wrapping is a traditional technique for extending FpML. It has been used by a number of firms implementing FpML since the earliest versions of FpML.

In wrapping, the FpML is contained within an XML “wrapper” that contains elements that are not available in FpML but that are required for the application. Below is a simplified example:

```
<abcML version="0.1">
  <additionalData>123</additionalData>
  <moreAdditionalData>456</moreAdditionalData>
  <trade>
    <dataDocument fpmlVersion="5-3"...>
      <!-- detail omitted -->
    </trade>
    <!-- parties omitted -->
  </dataDocument>
</abcML>
```

The following diagram illustrates the “wrapping” approach for extending FpML:



### 8.2.2. Advantages and Disadvantages

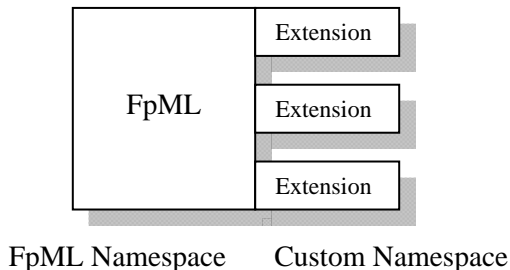
Some of the advantages and disadvantages of the “wrapping” approach include:

- Pros
  - This approach is straightforward and simple.
  - It has been used widely for several years by a number of dealers and has been proved to be successful.
  - The distinction between customized and standard FpML is clear.
- Cons:
  - It is not easy to add elements to structures that are repeated more than once.
  - Linking extensions to the original data can be hard.
  - This approach does not support restricting the contents of the FpML.

Because of the disadvantages listed above, the FpML Architecture 3.0 does not recommend using the “wrapping” approach for new implementations.

### 8.3. Extending Type Content

The recommended approach for extending FpML is to extend the types defined by FpML with new elements that are in the customizing firm’s namespace. This approach is documented in the FpML Architecture 3.0 document in Section 6.4.





### 8.3.1. Example

In FpML, the “AdjustableDate” type is defined as having an *unadjusted date* and *date adjustments*. Let’s assume that ABC, a firm implementing an FpML-based application, would like to add a new element, “adjustedDate”, to that type.

The type extension would be defined in the ABC’s schema (AbcML) as follows:

```
<xsd:schema
  xmlns = "http://www.abc.com/AbcML"
  xmlns:fpml = "http://www.fpml.org/FpML-5/confirmation"
  xmlns:dsig = "http://www.w3.org/2000/09/xmlsig#"
  targetNamespace = "http://www.abc.com/AbcML"
  xmlns:abc = "http://www.abc.com/AbcML"
  xmlns:xsd = "http://www.w3.org/2001/XMLSchema"
  elementFormDefault = "qualified"
  attributeFormDefault = "unqualified">

<xsd:import namespace= "http://www.fpml.org/FpML-5/confirmation"
  schemaLocation="fpml-main-5-3.xsd" />

  <!-- ***** Extension of AdjustableDate ***** -->

<xsd:complexType name = "AdjustableDate">
  <xsd:annotation>
    <xsd:documentation>ABC extension to FpML Adjustable Date
      type</xsd:documentation>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base = "fpml:AdjustableDate">
      <xsd:sequence>
        <xsd:element name="adjustedDate" type="xsd:date"
          minOccurs="0" >
          <xsd:annotation>
            <xsd:documentation xml:lang="en">The date after
              adjustment using the adjustment conventions.
            </xsd:documentation>
          </xsd:annotation>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
```

Then an AbcML document could use the new type in existing FpML products by using the “xsi:type” attribute to indicate that the new type is being used. The new element is in the “abc” namespace, so it requires an appropriate prefix. For example, assuming that the AbcML schema were being used and the “abc” prefix pointed to that namespace, in a payment the user could do the following:

```
<dataDocument fpmlVersion = "5-3"
  xmlns = "http://www.fpml.org/FpML-5/confirmation"
  xmlns:abc = "http://www.abc.com/AbcML"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance">
  ...

  <payment>
    <payerPartyReference href="abc"/>
    <receiverPartyReference href="def"/>
    <paymentAmount>
      <currency>GBP</currency>
      <amount>15000000.00</amount>
    </paymentAmount>
    <paymentDate xsi:type="abc:AdjustableDate">
      <unadjustedDate>2012-01-17</unadjustedDate>
      <dateAdjustments>
        <businessDayConvention>FOLLOWING</businessDayConvention>
        <businessCenters>
          <businessCenter>GBLO</businessCenter>
          <businessCenter>USNY</businessCenter>
        </businessCenters>
        <abc:adjustedDate>2012-01-18</abc:adjustedDate>
      </dateAdjustments>
    </paymentDate>
  </payment>
```

### 8.3.2. Advantages and Disadvantages

- Pros
  - Allows extension to existing FpML elements in-place.
  - Supports extending repeated elements.
  - Extensions are closely linked to the original FpML, making it easy to determine the relation of the extension to the original FpML.
  - Extensions are clearly documented in the instance documents because they are in a different namespace.
  - It is relatively easy to migrate extensions to a new version of FpML.
  - This approach is a recommended approach in the FpML 3.0 Architecture.
- Cons
  - Requires the use of multiple namespaces and namespace-aware applications.
  - XPath expressions may be slightly longer due to the need for namespace prefixes.

### 8.4. Restricting Type Content

In many cases, specific implementations may wish to restrict the ability of users to use specific elements or options. To implement these restrictions, implementers may use the schema “redefine” capability to remove optional elements, make optional elements mandatory, etc. This technique is described in detail in the FpML Architecture Specification 3.0, in Sections 6.5 to 6.7.

## 8.5. Extending Product Coverage

The first step in creating a new product is the creation of a new product type based on the FpML “Product” type.

### 8.5.1. Create a New Type

First, create a new product type based on the FpML “Product” type.

In the following example we create a type called “Forward” that is a forward purchase/sale of an FpML underlying asset (such as a bond or a stock). Note that this product is simplified compared what you might require for a real implementation, e.g., it does not include any support for settlement date or amount, ignores commissions and accrued interest, for example. However, it does convey some of the information one might require for such a product, such as the buyer and seller, the asset, the quantity, the price (and provision for the price units, etc...), and the forward sale date.

```
<xsd:complexType name = "Forward">
  <xsd:annotation>
    <xsd:documentation>A forward transaction on an
      underlying asset.</xsd:documentation>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base = "fpml:Product">
      <xsd:sequence>
        <xsd:group ref="fpml:BuyerSeller.model"/>
        <xsd:element ref="fpml:underlyingAsset" />
        <xsd:element name="quantity" type="xsd:decimal"/>
        <xsd:element name="price" type="xsd:decimal" />
        <xsd:group ref="fpml:QuotationCharacteristics.model"/>
        <xsd:element name="fwdDate" type="fpml:AdjustableDate"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Next create a new global element that is a member of the “product” substitution group. This will allow the newly created product to be included in FpML trades and other places that require FpML products:

```
<xsd:element name="forward" type="Forward"
  substitutionGroup="fpml:product" />
```

To use this product, reference it as “abc:forward”, for example:

```
<trade>
  <tradeHeader>
    <!-- details deleted -->
  </tradeHeader>
  <abc:forward>
    <buyerPartyReference href="abc" />
    <sellerPartyReference href="def" />
    <bond>
      <!-- details omitted -->
    </bond>
    <abc:quantity>100</abc:quantity>
    <abc:price>100.5</abc:price>
    <quoteUnits>PctOfParValue</quoteUnits>
    <abc:fwdDate>
      <unadjustedDate>2012-01-14</unadjustedDate>
      <dateAdjustments>
        <!-- details omitted -->
      </dateAdjustments>
    </abc:fwdDate>
  </abc:forward>
</trade>
```

Note that in the above example, the document is assumed to have its default namespace to be the FpML namespace, so no prefixes are required for FpML elements. This includes any elements that are included using FpML groups (such as the BuyerSeller.model or the QuotationCharacteristics.model), or referencing FpML substitution groups (such as the underlyingAsset substitution group).

A different example is provided in the FpML Architecture Specification, Section 6.2.

### **8.6. Extending an Existing Product**

The process for extending an existing product is similar to that described above, except that instead of extending “fpml:Product”, one extends the specific product, for example “fpml:Swap”. Then one has the choice of either creating a new member of the “product” substitution group, as described above, or not bothering, and just using the “xsi:type” attribute to indicate that a proprietary extension was used. In the first approach, the trade would have the following structure:

```
<trade>
  <tradeHeader>
    <!-- details omitted -->
  </tradeHeader>
  <abc:swap>
    <!-- details omitted -->
  </abc:swap>
</trade>
```

In the second approach, the trade would look something like this:

```
<trade>
  <tradeHeader>
    <!-- details omitted -->
  </tradeHeader>
  <swap xsi:type="abc:Swap">
    <!-- details omitted -->
  </swap >
</trade>
```

In either case the extended elements would normally be in the “abc” namespace and therefore would require a namespace prefix.

This topic is also discussed in the FpML Architecture Specification, Section 6.2.

### **8.7. Extending Messages**

To create a new message, one must create a new message Complex Type extending one of the existing base message types, namely:

- NotificationMessage
- RequestMessage
- ResponseMessage

For example, assume that one wants to create a new notification message called “Trade Report”, with a report date and a collection of trades, plus the associated parties. The message definition would be as follows:

```
<xsd:complexType name = "TradeReport">
  <xsd:complexContent>
    <xsd:extension base = "fpml:NotificationMessage">
      <xsd:sequence>
        <xsd:element name="reportDate" type="xsd:date"/>
        <xsd:element name="trade" type="fpml:Trade"
          maxOccurs="unbounded"/>
        <xsd:element name="party" type="fpml:Party"
          maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Define a global element based on the new type:

```
<xsd:element name="tradeReport" type="TradeReport"/>
```

## FpML 5 User Guide 2012 Edition

Then you can use the new message tradeReport of type TradeReport (based on the FpML notification message) e.g.:

```
<abc:tradeReport fpmlVersion = "5-3"
  xmlns = "http://www.fpml.org/FpML-5/reporting"
  xmlns:abc = "http://www.abc.com/AbcML"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance">

  <header>
    <messageId messageIdScheme="http://abc.com/ms">A1</messageId>
    <sentBy>ABCDUS33</sentBy>
    <sendTo>DEFGGB2L</sendTo>
    <creationTimestamp>2012-01-14T15:38:00-00:00</creationTimestamp>
  </header>
  <abc:reportDate>2012-01-01</abc:reportDate>
  <abc:trade>
    <!-- trade details omitted -->
  </abc:trade>
  <abc:trade>
    <!-- trade details omitted -->
  </abc:trade>
  <abc:party>
    <!-- party details omitted -->
  </abc:party>
  <abc:party>
    <!-- party details omitted -->
  </abc:party>
</abc:tradeReport>
```

Another example of this is provided in the FpML Architecture Specification, Section 6.3

### **8.8. Versioning and Version Migration**

When customizing FpML, one needs to be aware that both FpML and the customizations will evolve over time. In the proprietary schema, a mechanism for tracking the version of the customizations needs to be specified, in addition to the mechanisms specified in FpML to record the FpML version.

To migrate to a new version of the customizations without changing the FpML version, update the customized schema and its version mechanisms, preserving the references to FpML unchanged.

To migrate to a new version of FpML, one will normally issue a new version of the customized schema that references the updated version of FpML.









## FpML 5 Underlying Assets

Underlying Asset	Description
bond	a security typically delivering interest coupon payments and requiring the repayment of a principal amount at its maturity
cash	an asset in monetary form, typically held in a bank account
commodity	a commodity underlying asset
convertibleBond	a bond that can under specified circumstances be converted into equity (e.g., common stock) in the issuer
deposit	a term deposit, a money market instrument of fixed duration yielding a specific interest rate
equity	an ownership share in an entity, typically common stock
exchangeTradedFund	a fund whose units can be traded on an equity exchange
future	identifies the underlying asset when it is a listed future contract (a standardized, daily-settled contract traded on an exchange for the purchase or sale of an asset at some specified date in the future)
fx	identifies a simple underlying asset type that is an FX rate. Used for specifying FX rates in the pricing and risk
index	an asset whose value is based on the value of a set of instruments, typically equities
loan	an underlying asset that is a loan
mortgage	a mortgage backed security
mutualFund	a pooled investment vehicle that takes positions in a variety of financial instruments, typically equities
rateIndex	an interest rate index, such as USD LIBOR
simpleFra	a simple, benchmark Forward Rate Agreement
simpleIrSwap	a simple, benchmark Interest Rate Swap
simpleCreditDefault Swap	a simple, benchmark Credit Default Swap

(See page 82)

## Useful Links

### FpML Specification

(schema, examples, documentation)

<http://www.fpml.org/spec/>

### FpML Roadmap

(timing and coverage of past and upcoming versions)

<http://www.fpml.org/roadmap/roadmap.pdf>

### FpML at a Glance

<http://www.fpml.org/documents/FpML5-at-a-glance.pdf>

### FpML Products Framework

<http://www.fpml.org/documents/FpML5-products-framework.pdf>

### FpML Messaging Framework

<http://www.fpml.org/documents/FpML5-messaging-framework.pdf>

### FpML Messages Mapping from 4.x to 5.x

<http://www.fpml.org/documents/FpML-message-mapping-4x-vs-5x.xls>

## FpML 5 Products (See page 79)

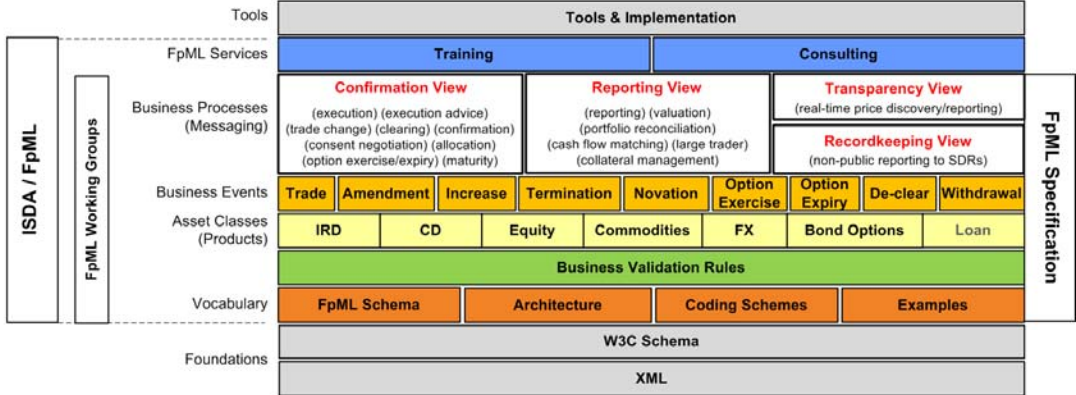
	Asset Class	Product	Product Variants	Since
	n/a	Strategy		3.0
		<b>genericProduct</b> (formerly nonSchemaProduct in 5.0 and 5.1) (to represent an OTC derivative transaction whose economics are not fully described using an FpML schema.) <b>standardProduct</b> (to represent a standardized OTC derivative transaction whose economics do not need to be fully described using an FpML schema because they are implied by the product ID)		5.2 (RV) 5.3
Interest Rate Derivatives	IRD	bulletPayment*		2.0
		capFloor		2.0
		fra		1.0
		swap	break clauses (cancelable, extendible, early termination), asset swap (since 4.2), inflation swap (since 4.2) Brazilian swap (since 4.4)	
		swaption	American, European, Bermuda, Cash/Physical	2.0
Foreign Exchange	FX	fxSingleLeg	Spot, Forward, Non-Deliverable Forwards	3.0
		fxSwap		3.0
		fxOption (fxSimpleOption in 3.x/4.x)	Knock-in and knock-out options, Side averaging rate option*, barrier option*	3.0
		fxDigitalOption*		3.0
		termDeposit*	Dual Currency Deposit	4.0
Credit Derivativ	CD	creditDefaultSwap	CDS index (since 4.1), CDS Basket (since 4.2), Loan CDS (since 4.3), CDS on Mortgage (since 4.3)	4.0
		creditDefaultSwapOption		4.3
Equity Derivatives	Equity	equityOption*	various option features/exercise types	3.0
		equityOptionTransactionSupplement		4.1
		brokerEquityOption*		4.1
		equityForward		4.1
		returnSwap (formerly equitySwap)		4.0
		equitySwapTransactionSupplement		4.1
	Dividend	dividendSwapTransactionSupplement		4.3
	Variance	varianceSwap*		4.3
		varianceSwapTransactionSupplement		4.3
		varianceOptionTransactionSupplement		4.6
Correlation	correlationSwap		4.3	
	Bond Options	bondOption	bond, convertible bond	4.3
Commo-dities		commoditySwap	financially and physically settled (4.6)	4.5
		commodityForward	bullion	4.6
		commodityOption	financially-settled, forwards	4.8
		commoditySwaption	physically-settled options (formerly in the commodityOption from 4.8 until 5.2)	5.3

\* Product not supported in the *Transparency* view of version 5.

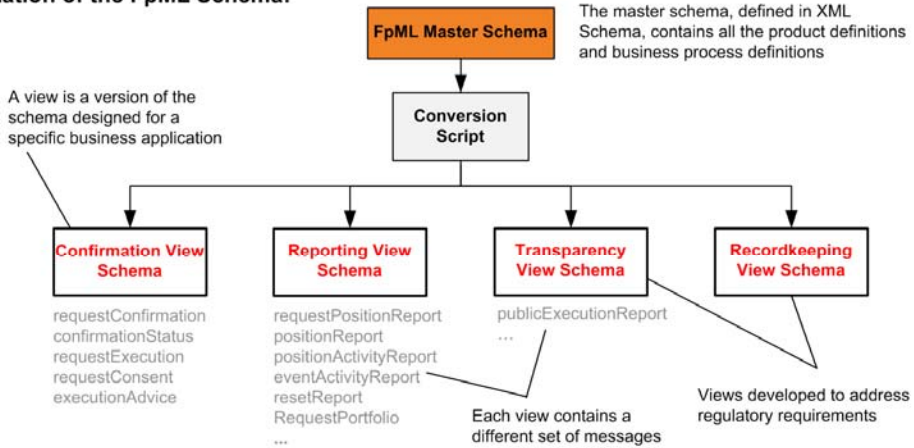
# FpML at a Glance

## Organization of the FpML Standard ("FpML Specification"):

<http://www.fpml.org/documents/FpML5-at-a-glance.pdf>



## Organization of the FpML Schema:



## Anatomy of an FpML Message:

Structure of a typical FpML message: e.g., requestExecution

