FINANCIAL INFORMATION EXCHANGE PROTOCOL (FIX)

Version 4.4 Schema

FIXML 4.4 Schema Version Guide

This document provides an overview of the direction taken to enhance FIXML by providing a schema-based version as discussed in the original FIXML road map.

INITIAL DRAFT

October 30, 2003

DISCLAIMER

THE INFORMATION CONTAINED HEREIN AND THE FINANCIAL INFORMATION EXCHANGE PROTOCOL (COLLECTIVELY, THE "FIX PROTOCOL") ARE PROVIDED "AS IS" AND NO PERSON OR ENTITY ASSOCIATED WITH THE FIX PROTOCOL MAKES ANY REPRESENTATION OR WARRANTY, EXPRESS OR IMPLIED, AS TO THE FIX PROTOCOL (OR THE RESULTS TO BE OBTAINED BY THE USE THEREOF) OR ANY OTHER MATTER AND EACH SUCH PERSON AND ENTITY SPECIFICALLY DISCLAIMS ANY WARRANTY OF ORIGINALITY, ACCURACY, COMPLETENESS, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. SUCH PERSONS AND ENTITIES DO NOT WARRANT THAT THE FIX PROTOCOL WILL CONFORM TO ANY DESCRIPTION THEREOF OR BE FREE OF ERRORS. THE ENTIRE RISK OF ANY USE OF THE FIX PROTOCOL IS ASSUMED BY THE USER.

NO PERSON OR ENTITY ASSOCIATED WITH THE FIX PROTOCOL SHALL HAVE ANY LIABILITY FOR DAMAGES OF ANY KIND ARISING IN ANY MANNER OUT OF OR IN CONNECTION WITH ANY USER'S USE OF (OR ANY INABILITY TO USE) THE FIX PROTOCOL, WHETHER DIRECT, INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL (INCLUDING, WITHOUT LIMITATION, LOSS OF DATA, LOSS OF USE, CLAIMS OF THIRD PARTIES OR LOST PROFITS OR REVENUES OR OTHER ECONOMIC LOSS), WHETHER IN TORT (INCLUDING NEGLIGENCE AND STRICT LIABILITY), CONTRACT OR OTHERWISE, WHETHER OR NOT ANY SUCH PERSON OR ENTITY HAS BEEN ADVISED OF, OR OTHERWISE MIGHT HAVE ANTICIPATED THE POSSIBILITY OF, SUCH DAMAGES.

No proprietary or ownership interest of any kind is granted with respect to the FIX Protocol (or any rights therein).

PREFACE

This document covers the XML variant of the FIX protocol, and in particular the XML Schema for version 4.4 of the protocol. It is anticipated that the introduction of this schema will help promote the adoption of FIXML, taking the FIX standard forward into the next generation of financial messaging solutions.

The Financial Information eXchange (FIX) effort was initiated in 1992 by a group of institutions and brokers interested in streamlining their trading processes. These firms felt that they, and the industry as a whole, could benefit from efficiencies derived through the electronic communication of indications, orders and executions. The result is FIX, an open message standard controlled by no single entity, that can be structured to match the business requirements of each firm. The benefits are:

* From the business flow perspective, FIX provides institutions, brokers, and other market participants a means of reducing the clutter of unnecessary telephone calls and scraps of paper, and facilitates targeting high quality information to specific individuals.
* For technologists, FIX provides an open standard that leverages the development effort so that they can efficiently create links with a wide range of counter-parties.
* For vendors, FIX provides ready access to the industry, with the incumbent reduction in marketing effort and increase in potential client base.

Openness has been the key to FIX's success. For that reason, while encouraging vendors to participate with the standard, FIX has remained vendor neutral. Similarly, FIX avoids over-standardization. It does not demand a single type of carrier (e.g., it will work with leased lines, frame relay, Internet, etc.), nor a single security protocol. It leaves many of these decisions to the individual firms that are using it. We do expect that, over time, the rules of engagement in these non-standardized areas will converge as technologies mature.

FIX is now used by a variety of firms and vendors. It has clearly emerged as the inter-firm messaging protocol of choice. FIX has grown from its original buyside-to-sellside equity trading roots. It is now used by markets (exchanges, “ECNs”, etc) and other market participants. In addition to equities, FIX currently supports four other products: Collective Investment Vehicles (CIVs), Derivatives, Fixed Income, and Foreign Exchange. The process for modifications to the specification is very open with input and feedback encouraged from the community. Those interested in providing input to the protocol are encouraged use the FIX website Discussion section or contact the FIX Global Technical Committee Chairpersons, Scottt Atwell, American Century Investments, (US) 816-340-7053 (scott_atwell@americancentury.com) or Dean Kauffman, TradeWeb LLC, (US) 201-536-5827 (dean.kauffman@tradeweb.com).. **The FIX website at *http://www.fixprotocol.org* is the main source of information, discussion, and notification of FIX-related events.**

We look forward to your participation.

FIX Protocol Ltd October 22, 2003

Contents – FIXML Schema Version Guide

Introduction

The FIXML language is in a state of transition. It has been four years since the initial release of FIXML. XML technology has advanced considerably in those four years. FPL committed to deliver an XML Schema representation for FIXML starting with FIX 4.3. Issues confronting FIXML users in the derivatives post trade area preempted release of the FIXML Schema for FIX 4.3. Instead the effort shifted to attempts to exploit the capabilities available in XML Schema to define a version of FIXML that was optimized to reduce message size. This version of FIXML was referred to as Transport Optimized FIXML during its development. The Global Technical Committee chose to release the transport optimizations in two phases.

- The **FIX 4.4 DTD Version** released with the FIX 4.4 Errata introduced standardized abbreviations for field names and removal of container elements used to represent repeating groups and component blocks.
- The **FIX 4.4 Schema Version** exploits the enhanced capabilities of XML Schema to further optimize FIXML message size by introducing the use of attributes to represent fields. Prior to the availability of XML Schema, there was no way to define datatypes for attributes.

FIX and FIXML Version and Comparison using New Order Single Message

The following section compares the implementation of the same FIX new order single message in FIX 4.2 tag=value format, FIXML 4.2 DTD version, and FIXML Schema Version.

FIX tag=value Version

The following is a FIX 4.2 New Order Single message in classic tag-value pair format:

```
8=FIX.4.2^9=251^35=D^49=AFUNDMGR^56=ABROKER^34=2^52=20030615-
01:14:49^11=12345^1=111111^63=0^64=20030621^21=3^110=1000^111=50000^55=IBM^48=459200101^22=1^5
4=1^60=2003061501:14:49 38=5000^40=1^44=15.75^15=USD^59=0^10=127
```

NOTE: ^ represents the SOH separator.

The message is 195 bytes in length.

FIXML 4.2 DTD Version

The following is a roughly equivalent FIXML 4.2 DTD-based message:

```
<FIXML>
      <FIXMLMessage>
            <Header>
                  <PossDupFlag Value="N" />
                  <PossResend Value="N" />
                  <SendingTime>20020103-12:00:01</SendingTime>
                  <Sender>
                        <CompID>AFUNDMGR</CompID>
                  </Sender>
                  <Target>
                        <CompID>ABROKER</CompID>
                  </Target>
            </Header>
            <ApplicationMessage>
                  <Order>
                        <ClOrdID>1968</ClOrdID>
                        <Account>4130287</Account>
```

```
                        <HandlInst Value="1" />
                        <ExDestination Value="L" />
                        <Instrument>
                              <Symbol>IBM</Symbol>
                              <SecurityID>459200101</SecurityID>
                              <SecurityIDSource Value="1" />
                        </Instrument>
                        <Side Value="2" />
                        <TransactTime>20021120-12:13:12</TransactTime>
                        <OrderQtyData>
                              <OrderQty>1000</OrderQty>
                        </OrderQtyData>
                        <OrdType Value="2" />
                        <Price>93.25</Price>
                        <Currency Value="USD" />
                  </Order>
            </ApplicationMessage>
      </FIXMLMessage>
</FIXML>
```

This message is 684 bytes; over three times the message size of the raw FIX message. In practice, FIXML messages could be 3-5 times their FIX equivalents.

FIXML 4.4 Schema Version

The following is a New Order Single message based on the FIXML 4.4 Schema.

```
<FIXML>
     <NewOrdSingle ClOrdID="123456"
                   Side="2"
                   TransactTm="2001-09-11T09:30:47-05:00"
                   OrdTyp="2"
                   Px="93.25"
                   Acct="26522154">
          <Hdr Snt="2001-09-11T09:30:47-05:00"
               PosDup="N"
               PosRsnd="N"
               SeqNum="521">
              <Sndr ID="AFUNDMGR"/>
              <Tgt ID="ABROKER"/>
          </Hdr>
          <Instrmt Sym="IBM"
                   ID="459200101"
                   IDSrc="1"/>
          <OrdQty Qty="1000"/>
       </NewOrdSingle>
</FIXML>
```

NOTE: The XML attributes in the message have been placed on separate lines to aid readability

This message is 358 bytes in length; approximately 70% larger than the raw FIX message, but roughly half the size of the previous FIXML format without significant loss in readability.

Sample Message Content

The following table is included to help clarify the message content shown above

Tag/Attribute	Meaning
<FIXML>	Root element
<NewOrdSingle ClOrdID="123456" Side="2" TransactTm="2001-09-11T09:30:47-05:00" OrdTyp="2" Px="93.25" Acct="26522154">	New order Client's order ID Sell order Transaction time Limit order Limit price Customer's account
<Hdr Snt="2001-09-11T09:30:47-05:00" PossDupFlag="N" PossResend="N" MsgSeqNum="521"> <Sndr ID="AFUNDMGR"/> <Tgt ID="ABROKER"/> </Hdr>	Header element Sending time Possible duplicate flag Resend flag Message sequence number Buyside's CompID Sellside's CompID End of Header Element
<Instrmt Sym="IBM" ID="459200101" IDSrc="1"/>	Stock symbol Stock CUSIP (ID type=CUSIP)
<OrdQty Qty="1000"/>	Order quantity
</NewOrdSingle>	Close of order
</FIXML>	Close root element

FIXML Transition to Schema

FIXML was initiated at a time when the only mechanism available to define and validate an XML syntax was the Document Type Definition (DTD) originally created as part of the Standardized General Markup Language (SGML). The DTD provided only minimal ability to define XML syntax.

Since then, the World Wide Web Consortium (***http://www.w3c.org***) adopted XML Schema as a way of representing the format of XML messages using XML syntax. Document Type Definitions, which were originally part of XML, have limited syntax and capabilities for defining XML syntax. XML Schema was designed to address many of the deficiencies of DTDs. The FPL Global Technical Committee has received numerous requests from FIX users for an XML Schema representation of the FIX Protocol and believes that a version of FIXML defined using XML Schema will provide a more robust, optimized message format and provide a better environment for users implementing FIXML applications.

The following limitations of DTDs determined much of the FIXML implementation;

1. Meta data could not be included in the DTD - so attributes were used for meta-data.
2. Attributes could not be "typed" so this restricted datatyping to elements. Many XML syntaxes then relied heavily on elements for data, attributes for meta-data. This is the approach taken for FIXML up through the FIX 4.4 Errata 20030618 release.

Since the initial release of FIXML in 1999, XML technology has advanced. The primary advancement has been in the area of standards that are used to define XML based languages. First among these is XML Schema - which has been adopted as a standard by the W3C. XML Schema addresses many of the limitations in DTDs, including:

1. Advanced datatyping, including datatyping for attributes.
2. Ability to include user defined meta-data in addition to standardized annotation and documentation.
3. XML Schema is written in XML, permitting manipulation by XML tools, such as XSLT, Xpath, etc.

FIXML FIX 4.4 DTD Version Enhancements

Several enhancements were introduced in the DTD version of FIXML in Release 4.4 of FIX.

- Modified the DTD based version of FIXML using the following techniques:
  - Roll up - eliminated extra levels of elements for repeating groups
  - Created standard abbreviation rules
  - Expanded meta data
    - Fullname
    - Category
    - ComponentType
      - Field
      - Message
      - Block

        - BlockRepeating
        - RepeatingGroup
    - Volume
        - Volume within FIX specification

FIXML 4.4 Schema Version Enhancements

The Schema version introduces the following enhancements

- Incorporated further transport optimizations
    - Adoption of attributes
    - Contextual Abbreviations – further reducing field names
- Addressed component blocks built around limitations of FIX tag=value by using consistent field names across component blocks
    - InstrumentLeg, NestedParties, Nested2Parties,UnderlyingInstrument
- Develop XML Schema Design Approach
    - Leverage work already done by ISO/XML and FpML
    - Design to support extensibility (customization) capabilities provided by FIX tag=value syntax

Next Interim Release of FIXML 4.4 Schema Version

The next version of the FIXML 4.4 Schema will provide:

- Support FpML product definitions to provide similar functionality to FIX tag=value syntax
- Address backward compatibility via the provision of XSL transforms

FIXML 4.4 Schema Version Design Objectives

Design objectives for FIXML messages (instance documents)

These design objectives refer to the FIXML instance documents. Instance documents are the actual FIXML messages.

- FIXML implementation shall adhere to XML technology standards as specified by the W3C.
- FIXML implementation shall be suitable implementation for use in high volume transaction scenarios. Target applications:
  - Order Routing
  - Trade Reporting and Post Trade Processing
  - Distribution of product (instrument) information
  - Market making for lower volume applications
- FIXML implementation shall minimize bandwidth consumption (reduced message size). The goal is to have FIXML messages be less than 1.5 X the size of an equivalent FIX tag=value message.
- FIXML implementation shall maintain human readability of FIXML message, while still adhering to performance goals.
- FIXML implementation shall support integration of FpML product specifications within the FIXML message in an equivalent manner to FIX 4.4 tag=value. This integration should use commonly agreed upon, de facto standard XML design patterns.
- FIXML implementation shall support a ready translation to and from FIX tag=value messages.
- FIXML implementation shall provide a cross-reference to ISO 15022 repository for each message, element, and component.
- FIXML implementation shall maintain the extensibility and customization available via the FIX tag=value message format, including:
  - Ability to add custom messages,
  - Ability to add custom fields to messages, component blocks, and repeating groups.
- FIXML Implementation shall provide full transport level independence.
- FIXML Implementation shall support version identification.

Design Objectives for the Schema Document

- FIXML Schema shall be implemented using the current de facto industry best practices for XML Schema usage.
- FIXML Schema shall be implemented in such a way as to fully support the FIXML 4.4 "Schema Version" Instance Requirements defined above.
- FIXML Schema shall support version identification.
- FIXML Schema shall provide meta-data sufficient to identify the FIX field name, component type, tag number, ISO 15022 repository cross-reference.
- FIXML Schema shall be interoperable and compatible with the FpML schema.
- The FIXML Schema shall be based upon and be compatible with the current version of XML schema: ***http://www.w3.org/2001/XMLSchema***

Structure of FIXML Schema Version Messages

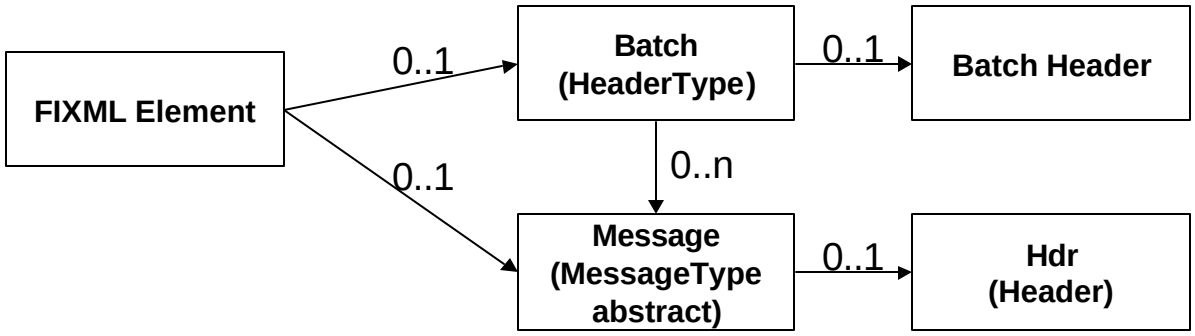
FIXML 4.4 Schema Design Guidelines

The following design guidelines were created to meet the design objectives for the FIXML Schema and the FIXML instance documents defined above.

1. Use meaningful abbreviations for element and attribute names wherever possible. Use standard abbreviations for common words (i.e., Price = Px, Currency = Ccy, etc.).
2. FIX Messages shall be implemented as XML Elements.
3. Individual, non-repeating fields shall be implemented as attributes of FIX Message elements.
4. FIX Component Blocks shall be implemented as an XML element.
5. Component blocks that were duplicated within FIX to circumvent tag=value requirements for uniqueness across fields and tag numbers, such as the Parties, NestedParties, NestedParties2 component blocks, shall use common naming in FIXML. The datatypes for each of the ComponentTypes will provide the mapping back to FIX tag=value format.
6. Non-repeating fields belonging to a FIX component block shall be implemented as attributes.
7. Repeating groups shall be implemented as XML elements.
8. Non-repeating fields belonging to a repeating group shall be implemented as attributes.
9. Identical repeating groups that occur across FIX messages will be identified as implicit components and reused across messages.
10. Field name prefixes that were used in FIX tag=value format for uniqueness shall be removed – thus creating a contextual abbreviation.
11. FIX datatypes will be mapped to the closest XML Schema datatype whenever possible, thus making FIXML more compatible with standard XML toolsets.

FIXML Schema Root Element

The FIXML Schema root element has been expanded to include the ability to include a batch of FIXML application messages. Batch capability was provided to deliver groups of messages, such as post trade confirms or position reports at the end of a trading session. Single message capability is still supported. Note that the headers are optional.



Single Message Usage

```
<FIXML>
    <OrderSingle>
       <Hdr/>
     </OrdSingle>
</FIXML>
```

Batch Message Usage

```
<FIXML>
   <Batch>
      <Hdr/>
      <OrdSingle>
        <Hdr/>
      </OrdSingle>
      <OrdSingle>
         <Hdr/>
      </OrdSingle>
    </Batch>
</FIXML>
```

An Example FIXML Single Message

The following is a New Order Single FIXML Schema message sent individually.

Notice that the header is provided at the message level.

```
<FIXML v="4.4" r="20030618" s="20030907">
    <NewOrdSingle ClOrdID="123456" Side="2" TransactTm="2001-09-11T09:30:47-05:00" OrdTyp="2" Px="93.25" Acct="26522154">
        <Hdr Snt="2001-09-11T09:30:47-05:00" PossDupFlag="N" PossResend="N" MsgSeqNum="521">
            <Sndr ID="AFUNDMGR"/>
            <Tgt ID="ABROKER"/>
        </Hdr>
        <Instrmt Sym="IBM" ID="459200101" IDSrc="1"/>
        <OrdQty Qty="1000"/>
    </NewOrdSingle>
</FIXML>
```

An Example FIXML Batch Message

The following example shows a batch of position reports.

Note that the header is provided for the entire batch of messages.

```
<FIXML v="4.4" r="20030618" s="20031030">
    <Batch>
      <Hdr Snt="2001-12-17T09:30:47-05:00">
          <Sndr ID="OCC"/>
          <Tgt ID="Firm"/>
       </Hdr>
            <PosRpt RptID="541386431" Rslt="0" BizDt="2003-09-10T00:00:00" Acct="1" AcctTyp="1" SetPx="0.00" SetPxTyp="1" PriSetPx="0.00" ReqTyp="0" Ccy="USD">
        <Pty ID="OCC" Role="21"/>
            <Pty ID="99999" Role="4"/>
             <Pty ID="C" Role="38">
                <PtySub SubID="ZZZ" SubIDTyp="2"/>
            </Pty>
            <Qty Typ="SOD" Long="35" Short="0"/>
            <Qty Typ="FIN" Long="20" Short="10"/>
            <Qty Typ="IAS" Long="10"/>
            <Amt Typ="FMTM" Amt="0.00"/>
          <Instrmt Sym="AOL" ID="KW" IDSrc="J" CFI="OCASPS" MMY="20031122" Mat="2003-11-22T00:00:00" Strk="47.50" StrkCcy="USD" Mult="100"/>
      </PosRpt>
            <PosRpt RptID="541386536" Rslt="0" BizDt="2003-09-10T00:00:00" Acct="1" AcctTyp="1" SetPx="0.00" SetPxTyp="1" PriSetPx="0.00" ReqTyp="0" Ccy="USD">
        <Pty ID="OCC" Role="21"/>
            <Pty ID="99999" Role="4"/>
             <Pty ID="C" Role="38">
                <PtySub SubID="ZZZ" SubIDTyp="2"/>
            </Pty>
            <Qty Typ="SOD" Long="35" Short="0"/>
            <Qty Typ="FIN" Long="20" Short="10"/>
            <Qty Typ="IAS" Long="10"/>
            <Amt Typ="FMTM" Amt="0.00"/>
          <Instrmt Sym="AOL" ID="KW" IDSrc="J" CFI="OCASPS" MMY="20031122" Mat="2003-11-22T00:00:00" Strk="47.50" StrkCcy="USD" Mult="100"/>
      </PosRpt>
```

```
            <PosRpt RptID="541386678" Rslt="0" BizDt="2003-09-10T00:00:00" Acct="1" AcctTyp="1" SetPx="0.00" SetPxTyp="1" PriSetPx="0.00" ReqTyp="0" Ccy="USD">
        <Pty ID="OCC" Role="21"/>
          <Pty ID="99999" Role="4"/>
           <Pty ID="C" Role="38">
              <PtySub SubID="ZZZ" SubIDTyp="2"/>
          </Pty>
          <Qty Typ="SOD" Long="35" Short="0"/>
          <Qty Typ="FIN" Long="20" Short="10"/>
          <Qty Typ="IAS" Long="10"/>
          <Amt Typ="FMTM" Amt="0.00"/>
         <Instrmt Sym="AOL" ID="KW" IDSrc="J" CFI="OCASPS" MMY="20031122" Mat="2003-11-22T00:00:00" Strk="47.50" StrkCcy="USD" Mult="100"/>
      </PosRpt>
    </Batch>
</FIXML>
```

Version Identification

FIXML versions are identified explicitly in the schema file names and also with constant attribute values defined in the fixml-component-base schema file.

FIXML Schema File Versioning

FIXML Schema employed the file naming convention developed for FpML. The major and minor version numbers of the FIX version represented by the schema are appended to all FIXML schema file names. This approach was taken to explicitly force users to recognize when counterparties have changed their version of the schema.

FIXML Message Versioning

The FIXML root element <FIXML> contains three attributes that define the version of the message. The FIXML root element is defined in the **fixml-components-base** schema file.

Attribute	Description	Format	Example
v	FIX Version	N.N	4.4
r	FIX Version release date (used to designate errata releases between FIX versions)	YYYYMMDD	20030618
s	Schema Release (used to designate schema releases between errata releases)	YYYYMMDD	20031030

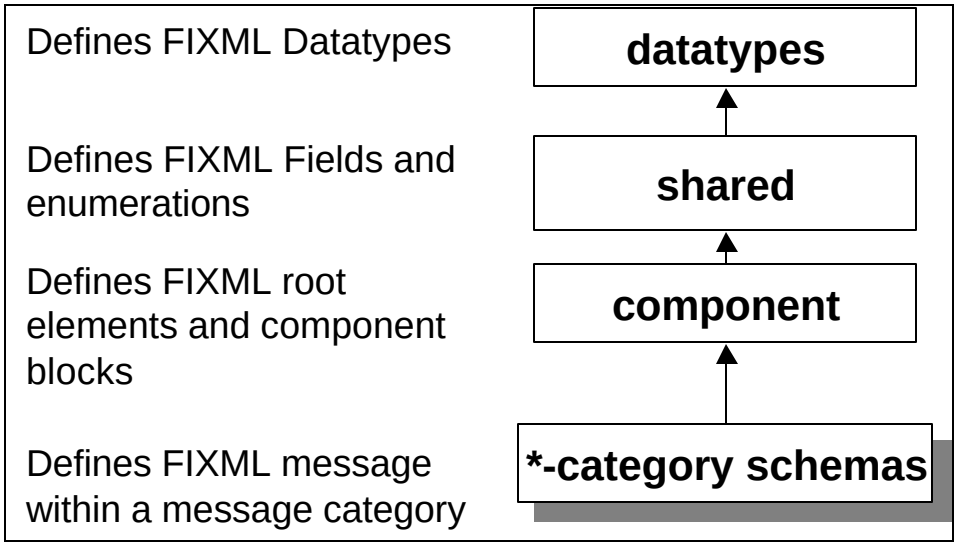
Example:

```
<FIXML v="4.4" r="20030618" s="20031030"> </FIXML>
```

FIXML Schema File Structure

Organization of files was driven largely by the requirement to support customization of the FIXML Schema per the requirements set forth by the FIXML Schema Working Group.

The basic organization of the schema has the datatypes used by the fields maintained in a separate file. FIX fields are defined in the shared file. Components and the FIXML root element are defined in the component files. FIXML messages are defined within separate category files.



Extensibility Design Pattern

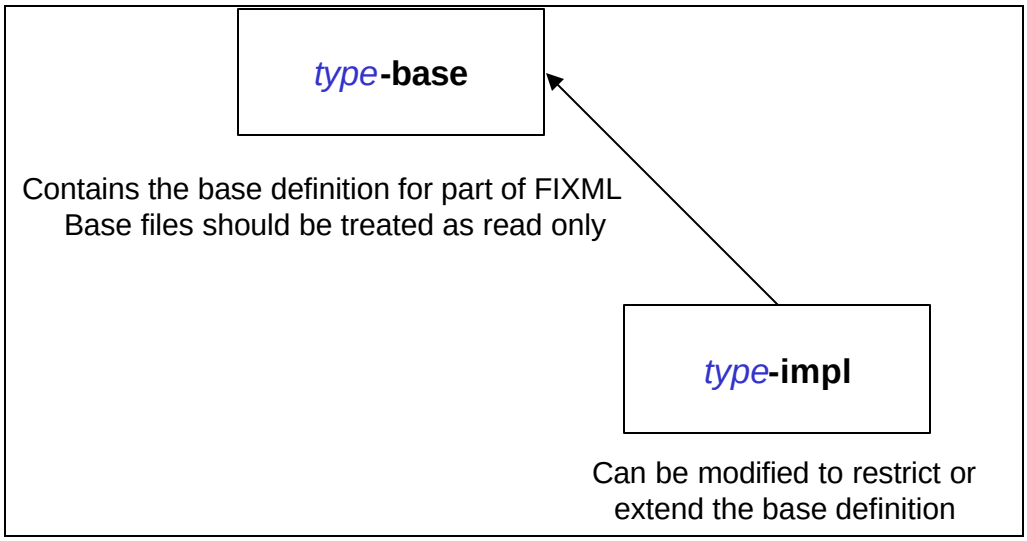
Much of the design work that went into the FIXML Schema was done to permit counterparties to further refine the FIXML language either by restriction or extension.

A possible scenario for restriction would be a market place that only supports a subset of the enumerations available for OrdType (tag=39). The exchange can override the OrdType_t FIXML datatype in the fixml-shared-impl-M-N.xsd file to restrict the set of possible values to only those supported by the market place.

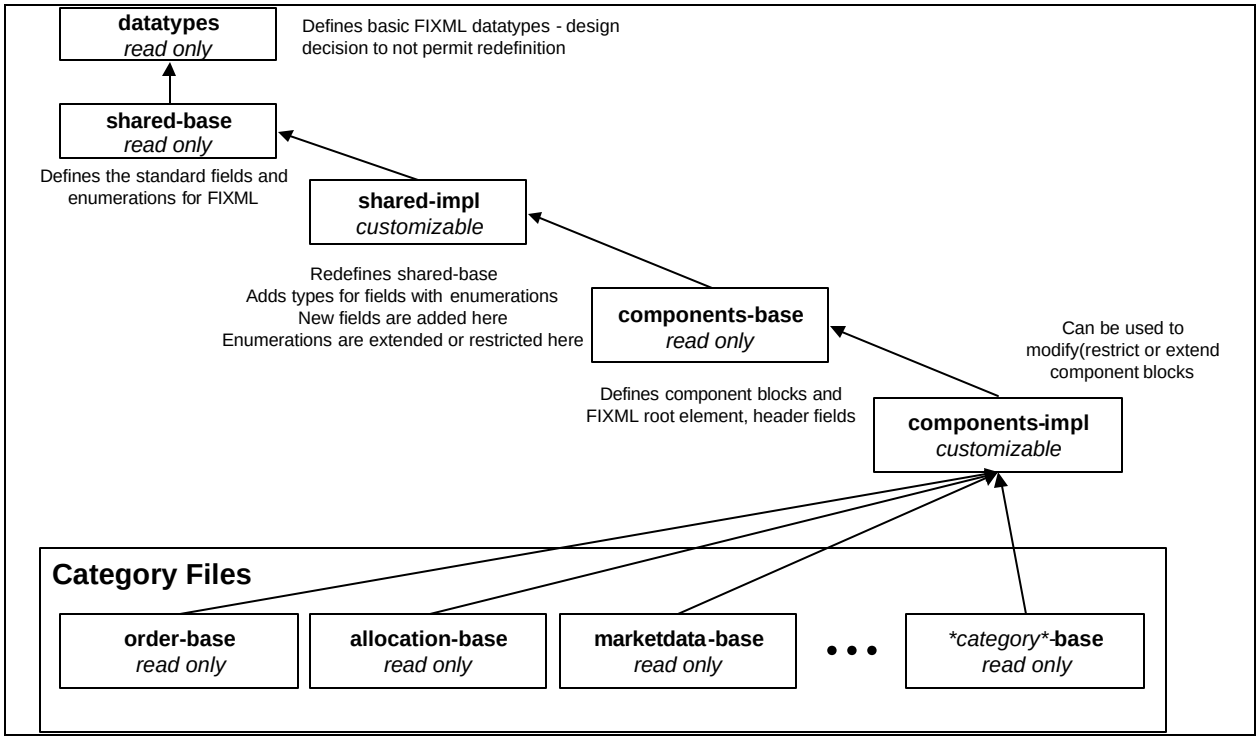
An example ofextension would be counterparties that require an additional custom field to be added to a new message.

In order to provide a uniform method in defining customizations that could be readily absorbed by counterparties an extensibility design pattern was developed that defines how the FIXML definition was partitioned and organized within separate schema files.

Each level of schema file (with the exception of datatypes) provides a base definition file that defines the standard (default) FIXML language. Redefining this base file an implementation file ("impl") is provided that by default simply references the base definition.



Subsequent levels of the schema reference the impl from the previous level – thus providing a customization entry point at the field level, component level, and message level.



FIXML Schema file naming conventions

FIXML file naming conventions are shown in the following illustration.

All filenames begin with lowercase “fixml-“

“-“ is used to separate portions of the filename

The type of the schema file is identified in the second component of the file name. The datatypes file contains the basic datatypes used within FIXML. The shared files contain the definitions for FIX fields. The components file contains definitions for FIXML components (as defined in Volume 1 of the specification, additional components identified while defining the FIXML schema, and the outer elements for FIX.

Files are either a **base** file or an implementation (**impl**). Base files define the standard FIXML language. Impl files are used to extend or restrict the base FIXML language.

fixml-*Type*-{base | impl}-*FixMajorVersion*-*FixMinorVersion*.xsd

Type is one of
datatypes
shared
components
category -where category is one of the FIX message categories, such as *confirmation*, *listorder*, *order*, *settlement*, etc.

FixMajorVersion is the FIX Major Version number, such as "4"

FixMinorVersion is the FIX Minor Version number, such as "4"

Example File Names
Shared base file for FIX Version 4.4: fixml-shared-base-4-4.xsd
Order Category base file for FIX Version 4.4: fixml-order-base-4-4.xsd
Component implementation file for FIX Version 4.4: fixml-components-impl-4-4.xsd

Refer to appendix B for a complete list of schema files used in FIXML as of FIX release 4.4.

Datatypes schema file

A decision was made to use native XML Schema datatypes wherever possible. Many of the XML Schema standards are based upon ISO standard datatypes. This means that the FIX representation of UTCTimestamp is different from the FIXML representation. The FIXML Schema working group felt it more important to be compatible with XML and as a result XML toolsets. The requirement for conversion between FIX tag=value datatypes and XML is left to implementors.

The **fixml-datatypes** schema file contains definitions for the FIXML datatypes.

Shared schema file (fixml-shared-*-M-N.xsd)

Shared base file (fixml-shared-base-M-N.xsd)

The **fixml-shared-base** file contains simple type definitions for all FIX application level fields and session level fields that are used as part of the FIXML header.

All fields are defined as simple types. The simple type name is derived from the full FIX field name appended with a "_t".

All fields with enumerations are defined as simple types. The enumeration simple type name is derived from the full FIX field name appended with a "enum_t".

Field definition examples

An example of a field definition for the AvgPx (tag=6) field:

```
<xs:simpleType name="AvgPx_t">
   <xs:annotation>
      <xs:documentation xml:lang="en">Calculated average price of all fills
on this order    For Fixed Income trades AvgPx is always expressed as
percent of par  regardless of the PriceType  423  of LastPx  3    I e
AvgPx will contain an average of percent of par values  see LastParPx  669
for issues traded in Yield  Spread or Discount
      </xs:documentation>
      <xs:appinfo xmlns:x="http://www.fixprotocol.org/fixml/metadata.xsd">
         <xs:Xref Protocol="FIX" name="AvgPx" tag="6" datatype="Price"
ComponentType="Field"/>
         <xs:Xref Protocol="ISO_15022_XML"/>
      </xs:appinfo>
   </xs:annotation>
   <xs:restriction base="Price"/>
</xs:simpleType>
```

An example of an enumerated field:

```
<xs:simpleType name="CommType_enum_t">
   <xs:annotation>
      <xs:documentation xml:lang="en">Commission type Valid values:  = per
unit  implying shares  par  currency  etc   2 = percentage 3 = absolute
total monetary amount  4 =   for CIV buy orders  percentage waived   cash
discount 5 =   for CIV buy orders  percentage waived   enhanced units 6 =
points per bond or or contract   Supply ContractMultiplier  23  in the
Instrument  component block if the object security is denominated in a size
other than the industry default   000 par for bonds
      </xs:documentation>
      <xs:appinfo xmlns:x="http://www.fixprotocol.org/fixml/metadata.xsd">
         <xs:Xref Protocol="FIX" name="CommType" tag="13" datatype="char"
ComponentType="Field"/>
         <xs:Xref Protocol="ISO_15022_XML"/>
      </xs:appinfo>
      <xs:appinfo xmlns:x="http://www.fixprotocol.org/fixml/metadata.xsd">
         <x:EnumDoc value="1" desc="PerShare"/>
         <x:EnumDoc value="2" desc="Percent"/>
         <x:EnumDoc value="3" desc="Absolute"/>
         <x:EnumDoc value="4" desc="PctWaivedCshDisc"/>
         <x:EnumDoc value="5" desc="PctWaivedEnUnits"/>
         <x:EnumDoc value="6" desc="PerBond"/>
      </xs:appinfo>
   </xs:annotation>
   <xs:restriction base="xs:string">
      <xs:enumeration value="1"/>
      <xs:enumeration value="2"/>
      <xs:enumeration value="3"/>
      <xs:enumeration value="4"/>
      <xs:enumeration value="5"/>
      <xs:enumeration value="6"/>
   </xs:restriction>
</xs:simpleType>
```

Shared implementation file (fixml-shared-impl-M-N.xsd)

One of the more convoluted constructs used was the need to place the field level definitions for enumerated types in the **fixml-shared-impl** file. As shown above, the **fixml-shared-base** file defines each enumerated field as a simple type named ***fieldname*_enum_t**. This enumerated type is then used to define a corresponding field type in the **fixml-shared-impl** schema file named ***fieldname*_t**. It is this ***fieldname*_t** type that is referenced in subsequent schema files (fixml-components and the message category schema files).

This construct was required to provide a mechanism to extend enumerations. The ***fieldname*_t** can be modified in the **fixml-shared-impl** file to include additional enumerations. The ***fieldname*_t** can be restricted by redefining the ***fieldname*_enum_t** simple type within the fixed-shared-impl file.

An example of exactly how to restrict and extend enumerations is provided in the customization section of this document.

The shared implementation file is also the location where new custom fields are introduced.

Components (fixml-components-*-M-N.xsd)

Component files are used to define the reusable components that are used across FIX messages. The FIXML root element and headers are defined in the components file, as well.

Components base file (fixml-components-base-M-N.xsd

The **fixml-components-base file** contains the definitions for all FIX component blocks defined in volume 1 of the FIX specification. The FIXML root element, FIXML headers, the batch element, and the abstract message type are also defined within this file.

Components (and messages) are defined using element groups and attribute groups. The advantage of these groups is that you can redefine the groups (using either restriction or extension) to change the overall structure of the component (or message).

There are three element groups defined for each component and message.

*componentOrMessageName*ElementsRequired	Contains a list of elements required by the component. As only components and messages are defined as elements, this would be a list of components required to make up the message or component.
*componentOrMessageName*ElementsOptional	Contains a list of optional elements that can be included in the component or message.
*componentOrMessageName*ElementsCustom	Defined as an empty group, this group is overriden in the impl version of the file to add new custom elements to a message or component.

Three attribute groups are defined for each component and message.

*componentOrMessageName*AttributesRequired	Contains a list of Attributes required by the component.
*componentOrMessageName*AttributesOptional	Contains a list of optional Attributes that can be included in the component or message.
*componentOrMessageName*AttributesCustom	Defined as an empty group, this group is overriden in the impl version of the file to add new custom attributes to a message or component.

The Parties Component block is shown below. Notice the overall definition pattern. This pattern is followed for all component blocks and message definitions.

```
        <xs:group name="PartiesElementsRequired">
                <xs:sequence/>
        </xs:group>
        <xs:group name="PartiesElementsOptional">
                <xs:sequence>
                <xs:element name="PtySub" type="PtysSubGrp_Block_t" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:group>
        <xs:group name="PartiesElementsCustom">
                <xs:sequence/>
        </xs:group>
        <xs:attributeGroup name="PartiesAttributesRequired">

        </xs:attributeGroup>
        <xs:attributeGroup name="PartiesAttributesOptional">
                <xs:attribute name="ID" type="PartyID_t" use="optional"/>
                <xs:attribute name="IDSrc" type="PartyIDSource_t" use="optional"/>
                <xs:attribute name="Role" type="PartyRole_t" use="optional"/>
        </xs:attributeGroup>
        <xs:attributeGroup name="PartiesAttributesCustom"/>

        <xs:complexType name="Parties_Block_t" final="#all">
        <xs:annotation>
                <xs:documentation xml:lang="en">**Desc**
                </xs:documentation>
                <xs:appinfo xmlns:x="http://www.fixprotocol.org/fixml/metadata.xsd">
                        <xs:Xref Protocol="FIX" name="Parties"
ComponentType="BlockRepeating"/>
                        <xs:Xref Protocol="ISO_15022_XML"/>
                </xs:appinfo>
        </xs:annotation>
                <xs:sequence>
                        <xs:group ref="PartiesElementsRequired"/>
                        <xs:group ref="PartiesElementsOptional"/>
                        <xs:group ref="PartiesElementsCustom"/>
                </xs:sequence>
                <xs:attributeGroup ref="PartiesAttributesRequired"/>
                <xs:attributeGroup ref="PartiesAttributesOptional"/>
                <xs:attributeGroup ref="PartiesAttributesCustom"/>
        </xs:complexType>
```

Components implementation file (fixml-components-impl-M-N.xsd)

The default version **fixml-components-impl** file simply redefines the components-base file. This is the file where modifications (restrictions or extensions) would be made to component blocks used in the FIX protocol.

Categories (fixml-*categoryName*-base-M-N.xsd)

Each message category defined within the FIX specification has its own schema file. This provides a granular level of usage for applications only requiring access to one message categoryThe message category schema files contain the component and message definitions that belong to a specific message category defined within the FIX Protocol. Examples of message categories include: Indications, Market Data, Positions, Allocation. . A complete list of the category files for FIXML is provided in Appendix B.

Category messages and components are defined following the same pattern defined above for components. The following defines the New Order Single message from the fixml-categoryOrder-4-4.xsd:

```
  <xs:group name="NewOrderSingleElementsRequired">
        <xs:sequence>
        <xs:element name="Instrmt" type="Instrument_Block_t" minOccurs="1"
maxOccurs="1"/>
        <xs:element name="OrdQty" type="OrderQtyData_Block_t" minOccurs="1"
maxOccurs="1"/>
        </xs:sequence>
  </xs:group>
  <xs:group name="NewOrderSingleElementsOptional">
        <xs:sequence>
        <xs:element name="Pty" type="Parties_Block_t" minOccurs="0"
maxOccurs="unbounded"/>
        <xs:element name="Stip" type="Stipulations_Block_t" minOccurs="0"
maxOccurs="unbounded"/>
        <xs:element name="FinDetls" type="FinancingDetails_Block_t" minOccurs="0"
maxOccurs="1"/>
        <xs:element name="SprdBnchmkCurve" type="SpreadOrBenchmarkCurveData_Block_t"
minOccurs="0" maxOccurs="1"/>
        <xs:element name="Yield" type="YieldData_Block_t" minOccurs="0"
maxOccurs="1"/>
        <xs:element name="Comm" type="CommissionData_Block_t" minOccurs="0"
maxOccurs="1"/>
        <xs:element name="PegInstr" type="PegInstructions_Block_t" minOccurs="0"
maxOccurs="1"/>
        <xs:element name="DiscInstr" type="DiscretionInstructions_Block_t"
minOccurs="0" maxOccurs="1"/>
        <xs:element name="PreAll" type="PreAllocGrp_Block_t" minOccurs="0"
maxOccurs="unbounded"/>
        <xs:element name="TrdSes" type="TrdgSesGrp_Block_t" minOccurs="0"
maxOccurs="unbounded"/>
        <xs:element name="Undl" type="UndInstrmtGrp_Block_t" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
  </xs:group>
  <xs:group name="NewOrderSingleElementsCustom">
        <xs:sequence/>
  </xs:group>
  <xs:attributeGroup name="NewOrderSingleAttributesRequired">
        <xs:attribute name="ClOrdID" type="ClOrdID_t" use="required"/>
        <xs:attribute name="Side" type="Side_t" use="required"/>
        <xs:attribute name="TransactTm" type="TransactTime_t" use="required"/>
        <xs:attribute name="OrdTyp" type="OrdType_t" use="required"/>

  </xs:attributeGroup>
  <xs:attributeGroup name="NewOrderSingleAttributesOptional">
        <xs:attribute name="ScndClOrdID" type="SecondaryClOrdID_t" use="optional"/>
        <xs:attribute name="ClOrdLinkID" type="ClOrdLinkID_t" use="optional"/>
        <xs:attribute name="TrdOrigntnDt" type="TradeOriginationDate_t"
use="optional"/>
        <xs:attribute name="TrdDt" type="TradeDate_t" use="optional"/>
        <xs:attribute name="Acct" type="Account_t" use="optional"/>
        <xs:attribute name="AcctIDSrc" type="AcctIDSource_t" use="optional"/>
        <xs:attribute name="AcctTyp" type="AccountType_t" use="optional"/>
        <xs:attribute name="DayBkngInst" type="DayBookingInst_t" use="optional"/>
        <xs:attribute name="BkngUnit" type="BookingUnit_t" use="optional"/>
        <xs:attribute name="PreallocMethod" type="PreallocMethod_t" use="optional"/>
        <xs:attribute name="AllocID" type="AllocID_t" use="optional"/>
        <xs:attribute name="SettlTyp" type="SettlType_t" use="optional"/>
        <xs:attribute name="SettlDt" type="SettlDate_t" use="optional"/>
        <xs:attribute name="CshMgn" type="CashMargin_t" use="optional"/>
```

```
        <xs:attribute name="ClrngFeeInd" type="ClearingFeeIndicator_t"
use="optional"/>
        <xs:attribute name="HandlInst" type="HandlInst_t" use="optional"/>
        <xs:attribute name="ExecInst" type="ExecInst_t" use="optional"/>
        <xs:attribute name="MinQty" type="MinQty_t" use="optional"/>
        <xs:attribute name="MaxFloor" type="MaxFloor_t" use="optional"/>
        <xs:attribute name="ExDest" type="ExDestination_t" use="optional"/>
        <xs:attribute name="ProcCode" type="ProcessCode_t" use="optional"/>
        <xs:attribute name="PrevClsPx" type="PrevClosePx_t" use="optional"/>
        <xs:attribute name="LocReqd" type="LocateReqd_t" use="optional"/>
        <xs:attribute name="QtyTyp" type="QtyType_t" use="optional"/>
        <xs:attribute name="PxTyp" type="PriceType_t" use="optional"/>
        <xs:attribute name="Px" type="Price_t" use="optional"/>
        <xs:attribute name="StopPx" type="StopPx_t" use="optional"/>
        <xs:attribute name="Ccy" type="Currency_t" use="optional"/>
        <xs:attribute name="ComplianceID" type="ComplianceID_t" use="optional"/>
        <xs:attribute name="SolFlag" type="SolicitedFlag_t" use="optional"/>
        <xs:attribute name="IOIID" type="IOIID_t" use="optional"/>
        <xs:attribute name="QID" type="QuoteID_t" use="optional"/>
        <xs:attribute name="TmInForce" type="TimeInForce_t" use="optional"/>
        <xs:attribute name="EfctvTm" type="EffectiveTime_t" use="optional"/>
        <xs:attribute name="ExpireDt" type="ExpireDate_t" use="optional"/>
        <xs:attribute name="ExpireTm" type="ExpireTime_t" use="optional"/>
        <xs:attribute name="GTBkngInst" type="GTBookingInst_t" use="optional"/>
        <xs:attribute name="Cpcty" type="OrderCapacity_t" use="optional"/>
        <xs:attribute name="Rstctions" type="OrderRestrictions_t" use="optional"/>
        <xs:attribute name="CustOrdCpcty" type="CustOrderCapacity_t" use="optional"/>
        <xs:attribute name="ForexReq" type="ForexReq_t" use="optional"/>
        <xs:attribute name="SettlCcy" type="SettlCurrency_t" use="optional"/>
        <xs:attribute name="BkngTyp" type="BookingType_t" use="optional"/>
        <xs:attribute name="Txt" type="Text_t" use="optional"/>
        <xs:attribute name="EncTxtLen" type="EncodedTextLen_t" use="optional"/>
        <xs:attribute name="EncTxt" type="EncodedText_t" use="optional"/>
        <xs:attribute name="SettlDt2" type="SettlDate2_t" use="optional"/>
        <xs:attribute name="Qty2" type="OrderQty2_t" use="optional"/>
        <xs:attribute name="Px2" type="Price2_t" use="optional"/>
        <xs:attribute name="PosEfct" type="PositionEffect_t" use="optional"/>
        <xs:attribute name="CoveredOrUncovered" type="CoveredOrUncovered_t"
use="optional"/>
        <xs:attribute name="MaxShow" type="MaxShow_t" use="optional"/>
        <xs:attribute name="TgtStrategy" type="TargetStrategy_t" use="optional"/>
        <xs:attribute name="TgtStrategyParameters" type="TargetStrategyParameters_t"
use="optional"/>
        <xs:attribute name="ParticipationRt" type="ParticipationRate_t"
use="optional"/>
        <xs:attribute name="CxllationRights" type="CancellationRights_t"
use="optional"/>
        <xs:attribute name="MnyLaunderingStat" type="MoneyLaunderingStatus_t"
use="optional"/>
        <xs:attribute name="RegistID" type="RegistID_t" use="optional"/>
        <xs:attribute name="Designation" type="Designation_t" use="optional"/>

</xs:attributeGroup>
  <xs:attributeGroup name="NewOrderSingleAttributesCustom"/>

  <xs:complexType name="NewOrderSingle_message_t" final="#all">
        <xs:complexContent>
          <xs:extension base="Abstract_message_t">
             <xs:sequence>
                <xs:group ref="NewOrderSingleElementsRequired"/>
                <xs:group ref="NewOrderSingleElementsOptional"/>
```

```
                <xs:group ref="NewOrderSingleElementsCustom"/>
            </xs:sequence>
            <xs:attributeGroup ref="NewOrderSingleAttributesRequired"/>
            <xs:attributeGroup ref="NewOrderSingleAttributesOptional"/>
            <xs:attributeGroup ref="NewOrderSingleAttributesCustom"/>

        </xs:extension>
      </xs:complexContent>
  </xs:complexType>

<xs:element name="NewOrdSingle" type="NewOrderSingle_message_t"
substitutionGroup="Message" final="#all"/>
```

Categories (fixml-*categoryName*-impl-M-N.xsd)

Each message category defined within the FIX specification has its own schema file. This provides a granular level of usage for applications only requiring access to one message category. A complete list of the category files for FIXML is provided in Appendix B.

Trading Life Cycle files

Convenience files are provided with the FIXML schema version that includes the message categories for each of the trade life cycles (pre-trade, trade, post-trade) used by FIX. These files are provided to make it easier for applications that require access to multiple message categories within one of the trading life cycles.

Pretrade file (fixml-pretrade-M-N.xsd)

Includes the pre-trade message category implementation files.

Provided to make it eas

Trade file (fixml-trade-M-N.xsd)

Includes the trade message category implementation files.

Post trade file (fixml-trade-M-N.xsd)

Includes the post trade message category implementation files.

Main (fixml-main-M-N.xsd)

A main schema file is included that pulls in the pretrade, trade, and post trade schema files. This is provided for applications that require access to the full suite of FIX messages.

Customization

The FIXML Schema files have been organized to permit extensibility. Implementation versions of each schema file (with the exception of the datatypes file) are provided to permit users to redefine the base FIXML Schema version, as defined in the base files. This section provides guidelines for customizing the FIXML syntax. Even though a considerable amount of work has gone into making FIXML extensible, users are strongly encouraged to minimize modifications, in order to promote more consistent usage of the FIXML syntax within the industry. Obviously, the less customization, the easier it is to connect to counterparties. If customization is required, you are encouraged to communicate your requirements that are not being met by FIX to the FPL Global Technical Committee. There you may find out that there is a technique to meet your business requirement. Or, you may find that the Technical Committee has already addressed the issue for a planned future release. At a minimum you will receive coaching and assistance in how to extend FIXML in such a way as to make the new feature a part of a future version of FIX.

Defining a custom field

New fields are defined as an XML SimpleType in the fixml-shared-impl-N-N.xsd file. You are recommended to add the file to the end of the schema document. You also are strongly encouraged to include XML comments to define the reason for the field.

The field should then be added to the component or message where it will be used, once the field is defined in the fixml-shared-impl schema file.

If the field will be added to a component contained in fixml-components-base-N-N.xsd, you must now redefine that component in the fixml-components-impl-N-N.xsd file.

Adding a field to a component or message contained in one of the message categories is done in the same way you modify the components schema file. You need to redefine the portion of the message in the implementation version of the file.

You are encouraged to follow the same procedure for procuring new custom field names as is done for the FIX tag=value version of FIX. The FIX website provides a web page of custom fields and a form to submit requests for additional custom fields.

Restricting enumeration values for a FIX field

Restricting enumeration values is done by modifying the type definition in the fixml-shared-impl schema file.

{Example to be provided in next draft}

Extending enumeration values for a FIX field

Extending enumeration values is done by creating a union of the original enumeration type definition with new enumeration values.

{Example to be provided in next draft}

Making an optional field required

Making an optional field required is done by redefining the optional attribute group, modifying the usage of the field from “optional” to “required”. This redefinition is done within the implementation file for either the components or a particular message category.

{Example to be provided in next draft}

Making a required field optional

{to be provided in next draft}

It is not possible to make a required field optional without modifying the original required element or attribute group. Making required fields optional does go against the standard base definition of FIX and should be avoided.

Adding a custom message

Custom messages are added by creating a the message structure within the category to which the custom message belongs. Required and optional element and attribute groups should be created for the custom message.

{Example to be provided in next draft}

FIXML Schema Metadata

{to be provided in next draft}

Appendix A – FIXML Schema Version Datatypes

FIX Datatype	FIX Datatype Description	XML Schema Type	XML Schema Example Format
int	Negative or Nonnegative	int	
Length	int field (see definition of "int" above) representing the length in bytes. Value must be positive.	<xs:simpleType name="Length"> <xs:annotation> <xs:documentation> int representing the length in bytes. Value must be positive. </xs:documentation> </xs:annotation> <xs:restriction base="xs:nonNegativeInteger"/> </xs:simpleType>	
NumInGroup	int field (see definition of "int" above) representing the number of entries in a repeating group. Value must be positive.	Not used in FIXML Schema	

FIX Datatype	FIX Datatype Description	XML Schema Type	XML Schema Example Format
SeqNum	int field (see definition of "int" above) representing a message sequence number. Value must be positive.	<xs:simpleType name="RefSeqNum_t"> <xs:annotation> <xs:appinfo xmlns:x="http://www.fixprotocol.org/fixml/metadata.xsd"> <xs:Xref Protocol="FIX" name="RefSeqNum" tag="45" ***datatype="SeqNum"*** ComponentType="Field"/> <xs:Xref Protocol="ISO_15022_XML"/> </xs:appinfo> </xs:annotation> <xs:restriction base="SeqNum"/> </xs:simpleType>	
TagNum	int field (see definition of "int" above) representing a field's tag number when using FIX "Tag=Value" syntax. Value must be positive and may **not** contain leading zeros. 0-4999 standard 5000+ custom	Not used in FIXML Schema	
DayOfMonth	int field (see definition of "int" above) representing a day during a particular month (values 1-31).	gDay	

FIX Datatype	FIX Datatype Description	XML Schema Type	XML Schema Example Format
float	Sequence of digits with optional decimal point and sign character (ASCII characters "**-**", '**0**' - '**9**' and "**.**"); the absence of the decimal point within the string will be interpreted as the float representation of an integer value. All float fields must accommodate up to fifteen significant digits. The number of decimal places used should be a factor of business/market needs and mutual agreement between counterparties. Note that float values may contain leading zeros (e.g. “00023.23” = “23.23”) and may contain or omit trailing zeros after the decimal point (e.g. “23.0” = “23.0000” = “23” = "23."). Note that fields which are derived from float may contain negative values unless explicitly specified otherwise.	decimal	
	XML Schema datatype supports 18 significant digits		

FIX Datatype	FIX Datatype Description	XML Schema Type	XML Schema Example Format
Qty	float field (see definition of "float" above) capable of storing either a whole number (no decimal places) of "shares" (securities denominated in whole units) or a decimal value containing decimal places for non-share quantity asset classes (securities denominated in fractional units).	<xs:simpleType name="Qty"> <xs:annotation> <xs:documentation> decimal capable of storing either a whole number (no decimal places) of "shares" (securities denominated in whole units) or a decimal value containing decimal places for non-share quantity asset classes (securities denominated in fractional units). </xs:documentation> </xs:annotation> <xs:restriction base="xs:decimal"/> </xs:simpleType>	
Price	float field (see definition of "float" above) representing a price. Note the number of decimal places may vary. For certain asset classes prices may be negative values. For example, prices for options strategies can be negative under certain market conditions. Refer to ***Volume 7: FIX Usage by Product*** for asset classes that support negative price values.	<xs:simpleType name="Price"> <xs:annotation> <xs:documentation> decimal representing a price. Note the number of decimal places may vary. For certain asset classes prices may be negative values. For example, prices for options strategies can be negative under certain market conditions. Refer to Volume 7: FIX Usage by Product for asset classes that support negative price values. </xs:documentation> </xs:annotation> <xs:restriction base="xs:decimal"/> </xs:simpleType>	Strk="47.50"

FIX Datatype	FIX Datatype Description	XML Schema Type	XML Schema Example Format
PriceOffset	float field (see definition of “float” above) representing a price offset, which can be mathematically added to a "Price". Note the number of decimal places may vary and some fields such as LastForwardPoints may be negative.	<xs:simpleType name="PriceOffset"> <xs:documentation> decimal representing a price offset, which can be mathematically added to a "Price". Note the number of decimal places may vary and some fields such as LastForwardPoints may be negative. </xs:documentation> </xs:annotation> <xs:restriction base="xs:decimal"/> </xs:simpleType>	
Amt	float field (see definition of “float” above) typically representing a Price times a Qty.	<xs:simpleType name="Amt"> <xs:annotation> <xs:documentation> decimal typically representing a Price times a Qty </xs:documentation> </xs:annotation> <xs:restriction base="xs:decimal"/> </xs:simpleType>	Amt="6847.00"

FIX Datatype	FIX Datatype Description	XML Schema Type	XML Schema Example Format
Percentage	float field (see definition of "float" above) representing a percentage (e.g. .05 represents 5% and .9525 represents 95.25%). Note the number of decimal places may vary.	<xs:simpleType name="Percentage"> <xs:annotation> <xs:documentation> decimal representing a percentage (e.g. .05 represents 5% and .9525 represents 95.25%). Note the number of decimal places may vary. </xs:documentation> </xs:annotation> <xs:restriction base="xs:decimal"/> </xs:simpleType>	
char	Single character value, can include any alphanumeric character or punctuation except the delimiter. All char fields are case sensitive (i.e. **m** ≠ **M**).	<xs:simpleType name="char"> <xs:restriction base="xs:string"> <xs:pattern value=".{1}"/> </xs:restriction> </xs:simpleType>	
	Does it makes sense to make this a string that is limited to a length of 1 character?		
Boolean	a char field (see definition of "char" above) containing one of two values: 'Y' = True/Yes 'N' = False/No	boolean	

FIX Datatype	FIX Datatype Description	XML Schema Type	XML Schema Example Format
String	Alpha-numeric free format strings, can include any character or punctuation except the delimiter. All char fields are case sensitive (i.e. **morstatt ≠ Morstatt**).	xs:string	
MultipleValueString	String field (see definition of "String" above) containing one or more space delimited values.	NMTOKENS	
Country	String field (see definition of "String" above) representing a country using ISO 3166 Country code (2 character) values.	<xs:simpleType name="Country"> <xs:annotation> <xs:documentation> string representing a country using ISO 3166 Country code (2 character) values. </xs:documentation> </xs:annotation> <xs:restriction base="xs:string"/> </xs:simpleType>	

FIX Datatype	FIX Datatype Description	XML Schema Type	XML Schema Example Format
Currency	String field (see definition of "String" above) representing a currency type using ISO 4217 Currency code (3 character) values.	<xs:simpleType name="Currency"> <xs:annotation> <xs:documentation> string representing a currency type using ISO 4217 Currency code (3 character) values. </xs:documentation> </xs:annotation> <xs:restriction base="xs:string"/> </xs:simpleType>	StrkCcy="USD"
Exchange	String field (see definition of "String" above) representing a market or exchange.- ISO 10383 Market Identifier Code (MIC)	<xs:simpleType name="Exchange"> <xs:restriction base="MIC"/> </xs:simpleType>	

FIX Datatype	FIX Datatype Description	XML Schema Type	XML Schema Example Format
month-year	String field representing month of a year. An optional day of the month can be appended or an optional week code. ***Valid formats:*** YYYYMM YYYYMMDD YYYYMMWW YYYY = 0000-9999, MM = 01-12, DD = 01-31, WW = w1, w2, w3, w4, w5.	<xs:simpleType name="MonthYear"> <xs:annotation> <xs:documentation> String field representing month of a year. An optional day of the month can be appended or an optional week code. Valid formats: YYYYMM YYYYMMDD YYYYMMWW YYYY = 0000-9999, MM = 01-12, DD = 01-31, WW = w1, w2, w3, w4, w5. </xs:documentation> </xs:annotation> <xs:restriction base="xs:string"> <xs:pattern value="\d{4}(0|1)\d([0-3wW]\d)?"/> </xs:restriction> </xs:simpleType>	MonthYear=”200303” MonthYear =”20030320” MonthYear =”200303w2”

FIX Datatype	FIX Datatype Description	XML Schema Type	XML Schema Example Format
UTCTimestamp	Time/date combination represented in UTC (Universal Time Coordinated, also known as "GMT") in **either** YYYYMMDD-HH:MM:SS (whole seconds) or YYYYMMDD-HH:MM:SS.sss (milliseconds) format, colons, dash, and period required. Valid values: YYYY = 0000-9999, MM = 01-12, DD = 01-31, HH = 00-23, MM = 00-59, SS = 00-60 (60 only if UTC leap second) (without milliseconds). YYYY = 0000-9999, MM = 01-12, DD = 01-31, HH = 00-23, MM = 00-59, SS = 00-60 (60 only if UTC leap second), sss=000-999 (indicating milliseconds). **Leap Seconds:** Note that UTC includes corrections for leap seconds, which are inserted to account for slowing of the rotation of the earth. Leap second insertion is declared by the International Earth Rotation Service (IERS) and has, since 1972, only occurred on the night of Dec. 31 or Jun 30. The IERS considers March 31 and September 30 as secondary dates for leap second insertion, but has	<xs:simpleType name="UTCTimestamp"> <xs:restriction base="xs:dateTime"/> </xs:simpleType>	TransactTm=" 2001-12-17T09:30:47-05:00"

FIX Datatype	FIX Datatype Description	XML Schema Type	XML Schema Example Format

UTCTimeOnly	Time-only represented in UTC (Universal Time Coordinated, also known as "GMT") in **either** HH:MM:SS (whole seconds) or HH:MM:SS.sss (milliseconds) format, colons, and period required. This special-purpose field is paired with UTCDateOnly to form a proper UTCTimestamp for bandwidth-sensitive messages.
	Valid values:
	HH = 00-23, MM = 00-60), SS = 00-59. (60 only if UTC leap second (without milliseconds)
	HH = 00-23, MM = 00-59, SS = 00-60 (60 only if UTC leap second), sss=000-999 (indicating milliseconds).
		`<xs:simpleType name="UTCTimeOnly">`
		`<xs:restriction base="xs:time"/>`
		`</xs:simpleType>`
		
		`(note: UTCTimeStamp used in most cases)`
			MDEntryTime=" 13:20:00.000-05:00"

FIX Datatype	FIX Datatype Description	XML Schema Type	XML Schema Example Format

UTCDateOnly	Date represented in UTC (Universal Time Coordinated, also known as "GMT") in YYYYMMDD format. This special-purpose field is paired with UTCTimeOnly to form a proper UTCTimestamp for bandwidth-sensitive messages.
	Valid values:
	YYYY = 0000-9999, MM = 01-12, DD = 01-31.
		`<xs:simpleType name="UTCDateOnly">`
		`  <xs:restriction base="xs:date"/>`
		`</xs:simpleType>`
		
		`(note: LocalMktDate used in most cases)`
			MDEntryDate=" 2003-09-10"

LocalMktDate	Date of Local Market (vs. UTC) in YYYYMMDD format. This is the "normal" date field used by the FIX protocol.
	Valid values:
	YYYY = 0000-9999, MM = 01-12, DD = 01-31.
		`<xs:simpleType name="LocalMktDate">`
		`  <xs:restriction base="xs:date"/>`
		`</xs:simpleType>`
			BizDate=" 2003-09-10"

FIX Datatype	FIX Datatype Description	XML Schema Type	XML Schema Example Format

data	Raw data with no format or content restrictions. Data fields are always immediately preceded by a length field. The length field should specify the number of bytes of the value of the *data* field (up to but not including the terminating SOH). *Caution: the value of one of these fields may contain the delimiter (SOH) character. Note that the value specified for this field should be followed by the delimiter (SOH) character as all fields are terminated with an "SOH"*	`</xs:simpleType>`
		`<xs:simpleType name="data">`
		`<xs:restriction base="xs:string"/>`
		`</xs:simpleType>`	

Appendix B – FIXML Schema File Summary

File Name	Description
fixml-datatypes-4-4.xsd	Defines the base data types that are to be used in other fixml schema files. These fixml base data types are based on simple types built into XML Schema.
fixml-shared-base-4-4.xsd	Includes fixml-datatypes-4-4.xsd. Defines simple/complex types for each attribute value based on fixml-datatypes-4-4.xsd
fixml-shared-impl-4-4.xsd	Redefines fixml-shared-base-4-4.xsd. Derived types can be added here in the redefine.
fixml-components-base-4-4.xsd	Redefines fixml-shared-impl-4-4.xsd. Defines groups and attributeGroups that make up the complexTypes used by message schema to define that message.
fixml-components-impl-4-4.xsd	Redefines fixml-components-base-4-4.xsd. Derived types can be added here in the redefine.
fixml-allocation-base-4-4.xsd	Redefines fixml-components-impl-4-4.xsd . Defines allocation messages: AllocationInstruction AllocationInstructionAck AllocationReport AllocationReportAck
fixml-allocation-impl-4.4.xsd	Redefines fixml-allocation-base-4-4.xsd Used to customize allocation message category
fixml-collateral-base-4-4.xsd	Redefines fixml-components-impl-4-4.xsd . Defines collateral messages: CollateralRequest CollateralAssignment CollateralResponse CollateralReport CollateralInquiry CollateralInquiryAck
fixml-collateral-impl-4.4.xsd	Redefines fixml-collateral-base-4-4.xsd Used to customize collateral message category
fixml-confirmation-base-4-4.xsd	Redefines fixml-components-impl-4-4.xsd . Defines confirmation messages: Confirmation Confirmation_Ack ConfirmationRequest
fixml-confirmation-impl-4.4.xsd	Redefines fixml-confirmation-base-4-4.xsd Used to customize confirmation message category

fixml-crossorders-base-4-4.xsd	Redefines fixml-components-impl-4-4.xsd . Defines cross orders messages: NewOrderCross CrossOrderCancelReplaceRequest CrossOrderCancelRequest
fixml-crossorders-impl-4.4.xsd	Redefines fixml-crossorders-base-4-4.xsd Used to customize crossorders message category
fixml-indications-base-4-4.xsd	Redefines fixml-components-impl-4-4.xsd . Defines indications messages: IOI Advertisement
fixml-indications-impl-4.4.xsd	Redefines fixml-indications-base-4-4.xsd Used to customize indication message category
fixml-listorders-base-4-4.xsd	Redefines fixml-components-impl-4-4.xsd . Defines list orders messages: NewOrderList ListCancelRequest ListExecute ListStatusRequest ListStatus BidRequest BidResponse ListStrikePrice
fixml-listorders-impl-4.4.xsd	Redefines fixml-listorders-base-4-4.xsd Used to customize list orders message category
fixml-multilegorders-base-4-4.xsd	Redefines fixml-components-impl-4-4.xsd . Defines multiple go orders messages: NewOrderMultileg MultilegOrderCancelReplace NewOrderMultileg MultilegOrderCancelReplace NewOrderMultileg MultilegOrderCancelReplace
fixml-multilegorders-impl-4.4.xsd	Redefines fixml-multilegorders-base-4-4.xsd Used to customize multileg orders message category

fixml-marketdata-base-4-4.xsd	Redefines fixml-components-impl-4-4.xsd . Defines trade market data messages: MarketDataRequest MarketDataSnapshotFullRequest MarketDataIncrementalRequest MarketDataRequestReject
fixml-marketdata-impl-4.4.xsd	Redefines fixml-marketdata-base-4-4.xsd Used to customize marketdata message category
fixml-order-base-4-4.xsd	Redefines fixml-components-impl-4-4.xsd . Defines order messages: ExecutionReport OrderCancelReject NewOrderSingle OrderCancelRequest OrderCancelReplaceRequest OrderStatusRequest DontKnowTradeDK OrderMassCancelRequest OrderMassCancelReport OrderMassStatusRequest
fixml-orders-impl-4.4.xsd	Redefines fixml-orders-base-4-4.xsd Used to customize orders message category
fixml-positions-base-4-4.xsd	Redefines fixml-components-impl-4-4.xsd . Defines positions messages: PositionMaintenanceRequest PositionMaintenanceReport RequestForPositions RequestForPositionsAck PositionReport AssignmentReport
fixml-positions-impl-4.4.xsd	Redefines fixml-positions-base-4-4.xsd Used to customize positions message category

fixml-quotation-base-4-4.xsd	Redefines fixml-components-impl-4-4.xsd . Defines quotation messages: QuoteRequest Quote QuoteCancel QuoteStatusRequest MassQuoteAcknowledgement MassQuote QuoteRequestReject RFQRequest QuoteStatusReport QuoteResponse
fixml-quotation-impl-4.4.xsd	Redefines fixml-quotation-base-4-4.xsd Used to customize quotations message category
fixml-registration-base-4-4.xsd	Redefines fixml-components-impl-4-4.xsd . Defines registration messages: RegistrationInstructions RegistrationInstructionsResponse
fixml-registration-impl-4.4.xsd	Redefines fixml-registration-base-4-4.xsd Used to customize registration message category
fixml-securitystatus-base-4-4.xsd	Redefines fixml-components-impl-4-4.xsd . Defines security status messages: SecurityDefinitionRequest SecurityDefinition SecurityStatusRequest SecurityStatus TradingSessionStatusRequest TradingSessionStatus SecurityTypeRequest SecurityTypes SecurityListRequest SecurityList DerivativeSecurityListRequest DerivativeSecurityList
fixml-securitystatus-impl-4.4.xsd	Redefines fixml-securitystatus-base-4-4.xsd Used to customize security status message category

fixml-settlement-base-4-4.xsd	Redefines fixml-components-impl-4-4.xsd . Defines settlement messages: SettlementInstructions SettlementInstructionsRequest
fixml-settlement-impl-4.4.xsd	Redefines fixml-settlement-base-4-4.xsd Used to customize settlement message category
fixml-tradecapture-base-4-4.xsd	Redefines fixml-components-impl-4-4.xsd . Defines trade capture messages: TradeCaptureReportRequest TradeCaptureReport TradeCaptureReportRequestAck TradeCaptureReportAck
fixml-tradecapture-impl-4.4.xsd	Redefines fixml-tradecapture-base-4-4.xsd Used to customize trade capture message category
fixml-pretrade-4-4.xsd	Includes all pretrade message categories
fixml-trade-4-4.xsd	Includes all trade message categories
fixml-posttrade-4-4.xsd	Includes all post trade message categories
fixml-main-4-4.xsd	Includes pretrade, trade, and posttrade schema files

Appendix C – Files used to generate the FIXML Schema

The FIXML Schema, as with the previous FIXML DTD, was generated using the FIX repository software created by Kevin Houstoun.

{To be completed next draft}

Header files

Inventory header files here

Templates

Inventory templates here

Appendix D – FIXML FAQ from the FIX Protocol Website

The following FAQ was created to help provide a consistent message to press and industry participants. It serves as useful background regarding the direction of FIXML.

1. What is the FIXML Schema Working Group?

The FIXML Schema Working Group is a group of FPL members working together to help define the schema representation of the ***FIX protocol***. Much of the groundwork for the FIXML Schema representation, conducted by the FPL Global Derivatives Committee in conjunction with the Futures Industry Association, was done to address needs in the listed derivatives post trade space (clearing). The ***FPL Global Technical Committee*** authorized the FIXML Schema Working Group in June 2003. The group will consist of a diverse set of participants and the group's initial meeting will be held July 9th, 2003.

2. How will the FIXML Schema representation impact the current FIXML DTD representation?

The XML representation defined by the FIXML DTD will not be upwardly compatible with the FIXML Schema representation. A *Backward Compatible Schema* will be provided that matches the DTD version that can be used to help migrate to the new FIXML 4.4 Schema representation. The Backward Compatible schema will be made available at the same time as the new FIXML 4.4 Schema. XSLT translators will also be provided to convert between FIX tag=value syntax, FIXML DTD representation, and the new FIXML 4.4 Schema representation.

3. How does the FIXML Schema representation relate to the work being done by the ISO/XML working group as part of the ***ISO 15022 WG10***?

FPL remains committed to working with the ISO 15022 working groups to further provide compatibility in terms of business process, data content, and data element mapping. At this point in time the XML standards space is moving forward. The FIXML Schema effort is mainly designed to improve the usability of the ***FIX Protocol***'s FIXML syntax in order to support higher performance applications by reducing overall message size. FPL will continue to work with the ISO 15022 working group toward common standards, we see the ***FIXML Schema Working Group*** as being the next step in this evolutionary process.

4. When will the FIXML Schema be made available?

The goal is to release the schema on or about August 2003.

NOTE: The actual release of the FIXML Schema is the end of October 2003.

5. How does the work on the FIXML Schema relate to working being done by the FPL/FpML Collaboration Working Group?

FPL and ***ISDA/FpML*** held a meeting in June 2003 to discuss interoperability and cooperation. As a result of that meeting an inter-organization working group was formed to explore the possibilities for cooperation between FPL and ISDA/FpML. This working group is referred to as the **FPL/FpML Collaboration Working Group**. Jim Northey is the co-chair representing FPL in this working group. Robert Stowsky (Brook Path Partners, Inc.) is the co-chair representing the ***ISDA/FpML*** organization.

There will definitely be some coordination of activities between the ***FIXML Schema Working Group*** and the ***FPL/FpML Collaboration Working Group***. FIX is now able to support references to FpML product definitions using the tag=value format. The goal is to extend the

capability to reference an FpML product definition in FIXML using the FIXML Schema representation.

6. Why are we moving to FIXML Schema?

The World Wide Web Consortium (***http://www.w3c.org***) adopted ***XML Schema*** as a way of representing the format of XML messages using XML syntax. ***Document Type Definitions***, which were originally part of XML, have limited syntax and capabilities for defining XML syntax. ***XML Schema*** was designed to address many of the deficiencies of ***DTDs***. The ***FPL Global Technical Committee*** has received numerous requests from FIX users for an ***XML Schema*** representation of the ***FIX Protocol*** and believes that a version of FIXML defined using ***XML Schema*** will provide a more robust, optimized message format and provide a better environment for users implementing FIXML applications.

7. Will the current FIXML DTD version still be usable after the release of the FIXML Schema version?

The current FIXML DTD version will still be usable after the release of the FIXML Schema. However, the FIXML DTD version will be deprecated, being replaced by the FIXML Schema representation as the official version of FIXML 4.4. FIX users will bilaterally agree upon the version of FIXML to use in the same fashion as agreeing upon the version of FIX tag=value syntax to use.

8. Will there be a DTD available to match the planned FIXML Schema version?

This is still an open issue. Ideally there would be a DTD published to match the FIXML Schema representation. Whether this is possible is another question. The FIXML Schema Working Group will work first to develop an optimal representation of FIXML in ***XML Schema***. If this can be easily converted to a DTD then this may be done.

9. Will there be a Schema to match the existing DTD?

There will be a *Backward Compatible Schema* produced that will provide an identical representation of the current FIXML 4.4 DTD. This is being provided to help users transition forward to the new FIXML Schema representation.

10. Will FPL continue to support the FIXML DTD version for future versions of FIX? Or will FPL only support FIXML Schema for future versions of FIX going forward?

Going forward, the FIXML Schema version will be the standard XML representation of the ***FIX Protocol***. Previous versions of FIXML will of course continue to be part of the standard, just as with previous versions of the FIX tag=value format.

11. What versions of FIX will the FIXML Schema support?

FIXML Schema will support FIX 4.4 and later versions of the ***FIX Protocol***.

12. How does the schema fit in with other Schemas, e.g. Biztalk etc?

There has been no analysis done to determine how FIXML will fit with ***Biztalk*** or ***ebXML***. FPL is working closely with participants from ***ISDA/FpML*** and the ISO working groups to work towards convergence and interoperability.

Appendix E – FIXML Schema Version Timeline and History

The following bullet points highlight the timeline and major events that occurred in creating the FIXML Schema version.

- The Chicago Mercantile Exchange (CME) approached FPL regarding using FIX for post trades (6/2002)
- FIA formed the standards working group to drive the effort (7/2002)
- FIXML was selected because (7/2002)
    - There was not an existing install base of FIX tag=value applications in the listed futures and options post trade space.
    - Futures markets participants were already adopting XML based messaging solutions (including New York Mercantile Exchange (NYMEX), CME, The Clearing Corporation (formerly the Board of Trade Clearing Corporation), and Chicago Board of Trade (CBOT)).
    - There was strong push back from firms to continue using the MQ Series transport instead of using the FIX Session layer – making adoption of FIXML easier as FIX Session Layer is not required.
    - Post trade messages (allocation, trade capture, positions) have multiple levels of nesting which is difficult to implement using FIX tag=value format and ideal for XML
- CME started their pilot project (7/2002)
- CME quickly ran into problem that has plagued other FIXML initiatives - that being message size being too large for bandwidth and data storage requirements (11/2002)
    - Trade Capture in FIXML 3200 bytes for example
- FPL representatives and the CME developed a transport optimized XML representation proposal (11/2002)
    - Alternatives considered
        - FIX tag=value
        - Convert from long descriptive element names to tag numbers
        - Convert to attributes
        - Convert to attributes and use abbreviations
            - Convert from elements to attributes where possible
            - Use abbreviations
            - Based upon experience of other organizations that have successfully deployed XML in production messaging applications
    - Initially referred to as “Transport Optimized FIXML” TO-FIXML
    - Goal was not to necessarily have “human readable” XML
    - Surprise: TO-FIXML is much more readable in the attribute form
    - Trade Capture in ClearML 850 bytes
    - Message File Size was reduced from 25 MB to 9MB as a result of the optimization for a firm’s daily trade file
- FIA Standards Working Group approached the Global Technical Committee regarding the transport optimized version of FIXML (12/2002, 1/2003)
- The Global Technical Committee held discussions on the FIXML (2/2003)

    - Goal would be to have one version of FIXML
    - If message size is precluding usage we should consider converting to new version
    - Informally queried about FIXML usage - mostly internal applications
    - Agreed any approach must provide some form of backward compatibility (via XSLT for instance)
- Agreed to address transport optimization as a part of the FIXML Schema initiative that will follow release of FIX 4.4 (2/2003)
- FIX 4.4 FIXML DTD Released as part of FIX 4.4 Errata Release (20030618)
- FIXML Schema Working Group formed (6/2003)
- FIXML Draft 1 (10/2003)

Appendix F Compatibility

The FIXML Schema was tested with the following software products.

Tool	Version	Results
Xerces-C	2.3	Validates
Xerces-J 2.4 (Schema Quality Checker Version 2.2)	2.2	Error in XercesJ 2.4 prevents extension prevents custom attribute groups from being extended. This is a serious customization limitation. You are recommended to use the 2.5 version of XercesJ
Xerces-J 2.5	2.5	Validates
Microsoft .NET	2003	Validates
XMLSpy		Validates XMLSpy is extremely tolerant of incorrect Schema, so this is not saying much.
TurboXML		Validates TurboXML allows some schema errors through, but is a bit better than Spy at validating.
DataStage		Loads schema
Mercator		

Credits

The following people made specific contributions to the FIXML Schema Working Group and provided content for this document. Names are arranged alphabetically.

Scott Atwell	American Century	Co-chair of Global Technical Committee – assisted in running meetings – represented GTC in meetings – provided review and suggestions on document content.
Todd Borro	The Chicago Mercantile Exchange	Pioneered the FIXML optimization techniques, authored the FIXML design rules for the transport optimization version.
Barry Galster	The Options Clearing Corporation (Deloitte Consulting)	Led the validation of the FIXML Schema using a variety of software products. Authored examples for positions messages. One of the leads in validating the design patterns used to implement the FIXML Schema.
John Goeller	Lehman	FIXML Schema working group co-leader. One of the original innovators and champions of the usage of XML for financial markets messaging. Developed and supported original versions of FIXML and led FIX participation in ISO 15022 XML development.
Kevin Houstoun	FPL	Developer of the FIX Repository – developed software to generate the DTD and Schema versions of FIX 4.4., Co-leader, FIXML Schema Working Group, Generated several example FIXML Schema messages.
Kris Ketels	SWIFT	Provided background on ISO 15022 XML. Provided recommendations on optimization techniques that were employed as part of the ISO 15022 XML design rules.
Kevin Kobets	The Options Clearing Corporation (Deloit Consulting)	Created the FIXML datatypes table, participated in development of XSLT translator to convert from FIXML to FIX tag=value format.
Jim Northey	FPL Global Derivatives Committee co-chair	Primary author of the design guide and overview presentation. FIXML Schema working group co-leader
Alex Olaru	The Options Clearing Corporation	Helped coordinate and organize resources for the document.
Theresa Simon	The Options Clearing Corporation	Organized and formatted the document during the initial draft.
Matt Simpson	The Chicago Mercantile Exchange	Pioneered the FIXML optimization techniques, authored the FIXML design rules for the transport optimization version, provided example messages.
Robert Stowsky	BrookPath Partners and FpML Organization	Provided expertise and reviews of FIXML based upon his experience using FpML. Co-leader of the FIX-FpML Collaboration Working Group.
Lisa Taikitsadaporn	BrookPath Partners	Quality control and review of deliverables.
Steve Wilkinson	Solution Forge	Provided examples for order messages, authored sections showing differences between previous version of FIXML, tag=value FIX, and the FIXML Schema version.

FPL would also thank all the participants of the FIXML Schema Working Group.

Isabelle Cools	SWIFT
Mark Cox	CME
Sam Johnson	Transact Tools
Dean Kauffman	Tradeweb
Makoto Koizumi	Fujitsu
Peter Millington	OM Technology
Satoru Mizukami	Nikko Citigroup, Ltd.
John Munro	Rolfe & Nolan
Michael Timpone	Bloomberg
Frank Vandamme	SWIFT
Robert Woodmansey	BtoBits
Zach Zimmerer	Transact Tools