FIX Orchestra Technical Standard Proposal

Release Candidate 2

May 18, 2017

V0.3

Proposal Status: Public Review

For Global Technical Committee Governance Internal Use Only

Submission Date	May 18, 2017	Control Number	
Submission Status	Public Review	Ratified Date	
Primary Contact Person	Don Mendelson	Release Identifier	

DISCLAIMER

Table of Contents

Table of Figures

Document History

Revision	Date	Author	Revision Comments
RC2	5/11/2017	Don Mendelson	Initial draft
rev. 2	5/15/2017	Don Mendelson	Added ATDL integration, demo projects
rev. 3	5/19/2017	Don Mendelson	Minor corrections

1 Introduction

FIX Orchestra was conceived as **machine readable rules of engagement** between counterparties. As such, it is a standard for exchange of metadata about the behavior of FIX applications. Orchestra is intended to cut time to onboard counterparties and improve accuracy of implementations.

Orchestra does not change FIX protocol itself in any way, nor does it obsolete existing FIX engines or tools.

1.1 Authors

Name	Affiliation	Contact	Role
Don Mendelson	Silver Flash LLC	donmendelson@silverflash.net	Working group co-chair
Jim Northey	Itiviti	Jim.Northey@itiviti.com	Working group co-chair

2 Requirements

2.1 Business Requirements

2.1.1 Rules of engagement

FIX Orchestra was designed to overcome often vague, humanly readable specifications and thus cut onboarding time significantly.

The contents of Orchestra files are machine readable (that is, processed as data) may include:

- Message structure by each scenario, implemented as an extension of FIX Repository.
- Accepted values of enumerations by message scenario
- Workflow: when I send this message type under this condition, what can I expect back?
- How external states affect messages, e.g. market phases
- Express a condition such as for a conditionally required field using a Domain Specific Language (DSL)
- Document and exchange the Algorithmic Trading Definition Language (FIXatdl) files associated with a FIX service offering
- FIX session identification and transport configuration

Given a standard for information interchange, firms and vendors will be enabled to develop tools to automate configuration of FIX engines and applications, and generation of code, test cases, and documentation. The various aspects are not an all-or-nothing proposition, however. Users may implement only the features that they find most beneficial, and add features as needed.

2.1.2 Application layer behavior

The main purpose of Orchestra is to define application layer behavior such as how a market responds to orders. The properties of defined messages are intended to be semantic while independent of wire format. In other words, an Orchestra description of behavior should work whether messages are encoded in tag=value, FIXML, or a binary format such as SBE.

2.1.3 Session layer behavior

Another usage of Orchestra is to define behavior at the session layer. Session protocols such as FIXT and FIXP are typically stateful. For example, it is invalid to send an application message if the session establishment protocol has not been performed. There are other states involving sequencing of messages, heartbeats and the like.

2.1.4 Best practices

Working groups will be able to issue best-practices recommendations as Orchestra files to demonstrate baseline behavior. Firms may of course enhance the baseline as they see fit, but enhancement takes much less time than starting from scratch.

2.1.5 Integration of multiple services

Firms may offer multiple services that use FIX protocol, e.g. order routing, market data, and algorithm controls through FIXadtl. Firms have requested a single entry point to access the various service offerings, either through a bundle of local files or through internet interfaces.

2.2 Technical Requirements

This section discusses enhancements to the Orchestra standard since Release Candidate 1.

2.2.1 FIX Repository XML schema

2.2.1.1 Unique keys

Features were added to the XML schema to enforce uniqueness of keys, preventing accidental duplication. Unique keys are now enforced for abbreviations, categories, sections, code set names, actor names, component and repeating group IDs, and field IDs (tags). For a message, the combination of message name and scenario attribute must be unique. Violations of uniqueness would be reported by a schema-validating XML parser.

2.2.1.2 Valid key references

The XML schema was enhanced to guarantee that the type of field references an actual datatype or code set name. The new flow element enforces that its source and destination actor attributes are valid references. Invalid cross-references would be reported by a schema-validating XML parser.

2.2.1.3 Field assignment

The XML schema was updated to assign the value of a field in an outgoing message by a rule. The rule is expressed in the Score DSL and it can give a constant value, a reference to a field in an incoming message or variable, or it can specify a calculation based on one or more fields. Furthermore, new syntax can be used to assign different values in entries of a repeating group. For example, it can specify different values for the first, second and third entries of Parties group.

2.2.1.4 Rules for unique field values

Syntax was added to the schema to specify the scope of uniqueness expected of a field such as ClOrdId. A field may be required to be globally unique, or the scope of uniqueness may be specified in terms of a key field or combination of fields. For example, a rule may specify that ClOrdId must be unique by firm and trade date.

2.2.1.5 Flow element

A new XML element was added to the schema to describe flows of information between actors. A flow is an application layer view of message exchange. It may correspond to a session or transport, but it is not required to be a 1:1 relationship. Multiple logical flows may be multiplexed in a session. In the current schema, a message/scenario belongs to a flow.

2.2.1.6 Object ID

Experts in modeling have suggested that every element in our Repository should have a unique and persistent identifier across all versions and representations. A FIX field should have the same identifier whether it is represented in XML or Web Ontology Language (OWL), for example. The motivation is to prevent collisions of identifiers between many issuers of Orchestra files.

2.2.2 XML schema for interfaces and sessions

A new XML schema was introduced in RC2 to expose interfaces, service offerings, and session configurations. The schema contains elements for these concepts:

- **Protocol**—a standard for communications. A protocol stack can be enumerated with appropriate configurations at each layer.
- **Service**—a service offering by a counterparty. A service is an application layer protocol. In the schema, a service element has an attribute for a link to an Orchestra file in the Repository schema as mentioned above. The link may be either to a local file or a web accessible interface.
- **Interface**—a collection of protocols and services exposed by a counterparty. A counterparty may offer more than one interface for different purposes.
- **Session**—a specific usage of an interface. A session has one or more identifiers. It inherits services and protocols from its parent interface, but it may have further refinement or overrides of protocol settings, such as a transport address.

Integration of FIXatdl service offerings has been identified by firms as an unmet need. An Orchestra interface file provides a place to discover such offerings and access their files.

2.2.3 Score DSL

RC2 delivers a grammar for expressions in Orchestra. The XML has hooks for expressions for these purposes:

- Telling when a conditionally required field is required or when some other attribute of the field is overridden.
- Telling when a response to a message is triggered.

- Provides a value to assign to a field.

A working implementation of the grammar is provided in GitHub as open source code. In addition to implementing the grammar, the code provides mechanisms for symbol tables for variables and an abstraction to read or write message values.

2.2.4 Demonstration projects

To help users adopt Orchestra, some demonstration projects have been published as open source. They integrate with the QuickFIX API since that is a widely used, open source project. One project generates a QuickFIX data dictionary and code directly from an Orchestra file. Another demonstrates message validation and field population code generated from Orchestra.

3 Issues and Discussion Points

3.1 Questions to be decided in RC3

3.1.1 Relationship between message type and flow

There is still an open question about whether the same message type should be allowed to belong to multiple flows. This question should be answered in RC3.

3.1.2 Format of object identifier

The OID standard, created and used by ITU and ISO, was suggested as a standard to use for a unique and persistent identifier for every element in a Repository. OID provides for a registrar of high level namespaces. Unlike URI, OID is not based on network addressability, and therefore can be expected to be more stable. Alternatively, Legal Entity Identifier (LEI) was suggested as a high-level qualifier. For now, an attribute to hold an identifier has been added to the XML schema, but its format should be decided in RC3.

3.2 Cross-standard collaboration

Financial industry organizations often must deal with multiple standards aside from FIX, and sometimes data must be translated from one standard representation to another. Therefore, it is everyone's interest to standardize transformations or mappings as much as possible.

3.2.1 Code sets

One area of collaboration is standardization of code sets across message standards. This has already been accomplished with standards for currency, language, market and country codes. Others are likely to follow through collaborations with ISO, OMG and other standards bodies. To recognize its importance, the concept of a code set has been promoted to a first-class object in the Orchestra schema, and it contains provisions for external standard references.

3.2.2 Datatype standardization

As discussed above, the schema was enhanced to support mappings to different encodings. By the same token, mapping between FIX and other standards can be defined. One approach is to map each standard once to the General Purpose Datatypes standard to discover the commonalities rather than creating many instances of direct FIX to standard X mappings.

3.2.3 Semantic web technologies

An advantage of semantic web technologies is that they make it possible to discover relationships remotely through a web API. There has been growing interest in the financial industry and others to distribute taxonomies and vocabularies using semantic web technologies such as Web Ontology Language (OWL) and Simple Knowledge Organization System (SKOS). Notably, EDM Council has developed Financial Industry Business Ontology (FIBO). Also, ISO TC68 working group 5 is developing an ontology based on ISO 20022 (SWIFT) business model. Orchestra can be similarly converted to an ontology. We have already performed experiments that show this is possible.

3.3 Reference implementations

FIX Orchestra is intended to be a standard for information exchange, not a software product. However, the working group may sponsor reference implementations of some aspects of the standard. This will help firms and vendors adopt the standard while adding their own special value. The Orchestra standard and reference implementations will be made available to the public in GitHub so any interested software developer may take advantage, whether they are a member of FPL or not.

4 References

Reference	Version	Relevance	Normative
None			

5 Relevant and Related Standards

Related Standard	Version	Reference location	Relationship	Normative
Dublin Core XML Schemas	2008-02-11	http://dublincore.org/schemas/xmls/	Dependency	Yes
XML Schema for FIX	2016		Technical guide	Yes
XML Schema	2012	https://www.w3.org/TR/xmlschema11-1/	Dependency	Yes
Namespaces	2006	https://www.w3.org/TR/xml-names11/	Dependency	Yes
Object Identifier (OID) X.660	07/11	https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.660-201107-I!!PDF-E&type=items	TBD	

6 Intellectual Property Disclosure

Related Intellection Property	Type of IP (copyright, patent)	IP Owner	Relationship to proposed standard

7 Definitions

Term	Definition
Pedigree	The recorded history of an artifact
Provenance	A record of ownership of an artifact

8 FIX Orchestra

8.1 *Project milestones*

Since Orchestra has many facets, features will be delivered in several release candidates rather than attempting a big-bang approach.

8.1.1 Release candidate 2 deliverables

These artifacts will be delivered as Release Candidate 2:

- The technical specification is a separate document "FIX Orchestra Technical Specification". The document will be displayed in the Tech/Specs section of FIX Trading Community website as well as in GitHub project FIXTradingCommunity/fix-orchestra-spec.

These resources have been published in GitHub project FIXTradingCommunity/fix-orchestra:

- XML schema (XSD) for Repository 2016 Edition and Orchestra plus documentation of the schema
- XML schema (XSD) for interfaces and session configuration plus documentation of the schema
- Grammar and implementation of the Score DSL
- A script to populate Repository 2016 from 2010 Edition (message structure only)
- Example code to process Orchestra files

8.1.2 Release candidate 3 plan

The third release will add address message validation and reference data, such as tick size tables. The objective is to the reduce the number of external dependencies to rules of engagement.

Also, all of the previously released features will be refined as needed. A working group will be engaged to distribute best practices recommendations as an Orchestra file to prove the concepts.

FIX Repository and Orchestra may also be delivered in semantic form using technologies such Web Ontology Language (OWL).

Appendix A - Usage Examples

These example Orchestra files are posted in GitHub.

Example order entry file developed by MilleniumIT

fix-orchestra/repository2016/src/test/resources/examples/

Sample interface file

https://github.com/FIXTradingCommunity/fix-orchestra/blob/master/interfaces2016/src/test/resources/SampleInterfaces.xml

A non-FIX exchange API interpreted in Orchestra

https://github.com/FIXTradingCommunity/orchestrations/tree/master/NYSE%20Pillar

Appendix B – Compliance Strategy

The first level of compliance will be provided by existing XML tools that verify conformity of a file to its schema. When the DSL is released in the second phase, a means to validate expressions against the grammar will be provided.