FIX Orchestra Technical Standard Proposal

Release Candidate 3

March 8, 2018

V0.1

Proposal Status: Public Review

For Global Technical Committee Governance Internal Use Only

Submission Date	March 8, 2018	Control Number	
Submission Status	Public Review	Ratified Date	
Primary Contact Person	Don Mendelson	Release Identifier	

r0.3

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.

DRAFT OR NOT RATIFIED PROPOSALS (REFER TO PROPOSAL STATUS AND/OR SUBMISSION STATUS ON COVER PAGE) ARE PROVIDED "AS IS" TO INTERESTED PARTIES FOR DISCUSSION ONLY. PARTIES THAT CHOOSE TO IMPLEMENT THIS DRAFT PROPOSAL DO SO AT THEIR OWN RISK. IT IS A DRAFT DOCUMENT AND MAY BE UPDATED, REPLACED, OR MADE OBSOLETE BY OTHER DOCUMENTS AT ANY TIME. THE FIX GLOBAL TECHNICAL COMMITTEE WILL NOT ALLOW EARLY IMPLEMENTATION TO CONSTRAIN ITS ABILITY TO MAKE CHANGES TO THIS SPECIFICATION PRIOR TO FINAL RELEASE. IT IS INAPPROPRIATE TO USE FIX WORKING DRAFTS AS REFERENCE MATERIAL OR TO CITE THEM AS OTHER THAN “WORKS IN PROGRESS”. THE FIX GLOBAL TECHNICAL COMMITTEE WILL ISSUE, UPON COMPLETION OF REVIEW AND RATIFICATION, AN OFFICIAL STATUS ("APPROVED") OF/FOR THE PROPOSAL AND A RELEASE NUMBER.

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

Table of Contents

Table of Figures

Document History

Revision	Date	Author	Revision Comments
RC3	March 22, 2018	Don Mendelson Silver Flash LLC	Initial draft

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.

2.1.6 Stable semantic concepts

The encoding of some semantic concepts changed between FIX versions, especially between FIX 4.2 and later versions. Furthermore, different brokers implement the same concepts with different tags and even different message types. Orchestra should identify stable concepts across versions of the protocol to inform applications such as message translators. By matching semantic concepts in different message layours, a translation plan can be devised.

The names of the concepts should be standardized to maximize portability. We propose to publish a taxonomy of selected concepts. To be clear, this will not constitute a comprehensive business model of FIX, but rather a restricted scope of message elements that were documented to change between versions.

2.2 Technical Requirements

This section discusses enhancements to the Orchestra standard since Release Candidate 2.

2.2.1 FIX Repository XML schema

2.2.1.1 A single protocol per file

In RC2, an Orchestra file could contain multiple protocol versions. However, this led to unnecessary complexity. In RC3, the schema was changed to hold a single protocol definition. Organizations will need to issue an Orchestra file per protocol version.

2.2.1.2 Component scenarios

Like messages, components now support scenarios in RC3. The benefit is that a common block such as Instrument can have variations for different security types and the like.

2.2.1.3 Rendering hints

An attribute was added to most message elements to provide a hint to code generators and other tools about how the element should be rendered in a UI or protocol implementation.

2.2.1.4 Discriminator fields

The XML schema was enhanced in RC3 to explicitly link a field that modifies the valid values of another field. For example, the value of SecurityIDSource modifies the valid values of SecurityID. The modifying field is called a discriminator, and the whole data structure is known in programming terminology as a discriminated union. Previously, the Repository had no machine-readable syntax to represent this relationship; it was only explained in humanly readable documentation.

2.2.1.5 Semantic concepts

Features were added to the XML to link FIX semantic concepts to their representations in different versions of a protocol or between different protocols.

2.2.2 Demonstration projects

These utilities and demonstration projects have been added since RC2:

2.2.2.6 Documentation generator

This application generates a collection of HTML pages to document an Orchestra file. Some pages are supplemented by diagrams.
Documented elements include:

- Messages
- Components and repeating groups
- Fields
- Datatypes
- Actors
- State machines
- Flows

2.2.2.7 Test generator

This application generates acceptance test suites using the Cucumber framework. It generates:

- A QuickFIX data dictionary based on message structures in the Orchestra file
- A Cucumber feature file for each actor in the Orchestra file. A feature file is a text file that describes system behavior in language that is readable by non-technical users.

2.2.2.8 Message translator

A message translator has been developed that converts between two FIX formats as described by Orchestra files. In a preparation phase, the translator uses declarations of semantic concepts added in

Orchestra RC3 to develop a translation plan for a pair of message definitions. In its run-time phase, the translator executes the plan to translate an incoming message in one format to the format described by the second Orchestra file.

The translator code has not yet been published publicly. It awaits a decision about whether it should be shared with everyone as with other Orchestra demonstrations, or whether it should only be made available to FIX Trading Community members (see below).

3 Issues and Discussion Points

3.1 Questions to be decided beyond RC3

3.1.1 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 RC4.

3.1.2 Provenance and pedigree format

The Orchestra file format has a provision for metadata about the file that tells who issued it, when, and in what format. Categorically, this is known as provenance. This metadata is based on the standardized Dublin Core model.

Also, FIX Repository contains the history of each message element—when it was added, updated, and possibly deprecated. Element history is called pedigree. Orchestra has so far kept the pedigree format of Repository 2010 Edition. However, to make it easier to interact with other protocols, it would be desirable to use a standardized model. T he W3C PROV standard has been proposed for this. It is based on a comprehensive model of provenance and pedigree. PROV has multiple representations that follow the same semantics, including an XML encoding.

3.1.3 Reference implementations

FIX Orchestra is intended to be a standard for information exchange, not a software product. However, the working group has sponsored reference implementations of some aspects of the standard. This will help firms and vendors adopt the standard while adding their own special value. Up until now, all demonstration projects were made available to the public in GitHub so any interested software developer may take advantage, whether they were a member of FIX Trading Community or not.

However, it is now proposed that certain advanced applications should be considered premium and only be made available to members. Complex applications require more development time and maintenance, and therefore should be made available to firms that contribute. At the same time, those applications will serve as a value-added to membership, providing an incentive for firms to join.

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 3 deliverables

These artifacts will be delivered as Release Candidate 3:

- 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 Orchestra files
- Demonstration projects:
    - Documentation generator
    - QuickFIX data dictionary and code generators
    - XML diff/merge utility to manage Orchestra file changes
    - QuickFIX session configuration
    - Test generator

8.1.2 Roadmap

The next task of the Orchestra working group is develop a roadmap for release candidate 4 and beyond. In addition to standard development, a plan will be created to migrate the building of Extension Packs from Repository 2010 Edition to Orchestra. This plan will need to account for the tools that consume Repository, including FIXimate, FIXML schema generation, and so forth.

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. A test is provided to validate that isolated DSL expressions conform to the grammar. However, a comprehensive compliance test has not yet been developed.