Orchestra Standard - Technical Proposal

Version 1.1 RC1

November 15, 2023

V0.6

Proposal Status: Public Comment

For Global Technical Committee Governance Internal Use Only

Submission Date	Nov 16, 2023	Control Number	
Submission Status	Submitted	Ratified Date	
Primary Contact Person	Francesco Lo Conte, Esprow	Release Identifier	

DISCLAIMER

Table of Contents

Document History

Revision	Date	Author	Revision Comments
V0.1	June 14, 2022	Donald Mendelson, Silver Flash LLC	Initial draft
V0.2	June 16, 2022	Donald Mendelson, Silver Flash LLC	Edits suggested by the Orchestra working group on June 15, 2022.
V0.3	September 20, 2023	Hanno Klein, GTC	Edits to align with additions made to RC1.
V0.4	September 22, 2023	Hanno Klein, GTC	Edits after Orchestra Repository Schema WG call on Sep 21, 2023.
V0.5	October 30, 2023	Hanno Klein, GTC	Edits after Orchestra Repository Schema WG call on Oct 26, 2023.
V0.6	November 15, 2023	Hanno Klein, GTC	Edits after Orchestra Repository Schema WG call on Nov 15, 2023. • Renamed attribute "baseFieldId" to "nonEncodedFieldId" and limit its use to encoded fields. • Added discussion point to provide reason for name change

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 the FIX Protocol itself in any way, nor does it obsolete existing FIX engines or tools.

1.1 Roadmap

Orchestra version 1.0 Technical Standard is deemed usable as-is, and firms are successfully building systems around it. However, several technical improvements have been proposed to increase flexibility and fix minor defects. Therefore, version 1.1 Release Candidate 1 is proposed. As always, this release candidate will be circulated for a 90-day public review. If more changes are proposed, subsequent release candidates will be published. When the working group is satisfied with the specification, a Beta Standard, and ultimately, a final Technical Standard will be published.

Please see https://www.fixtrading.org/packages/technical-standard-proposal-process/ for more details.

1.2 Authors

Name	Affiliation	Contact	Role
Francesco Lo Conte	Esprow	francesco.loconte@esprow.com	Working group co-chair
Don Mendelson	Silver Flash LLC	donmendelson@silver-flash.net	Working group member
Jim Northey	FPL Director and ISO TC68 Chair	jim.northey@fintechstandards.us	Working group member
Hanno Klein	FIX Technical Director	hanno.klein@fixtrading.org	GTC co-chair EMEA
Emil Rakadjiev	Esprow	emil.rakadjiev@esprow.com	Working group member

2 Requirements

2.1 *Business Requirements*

This section describes enhancements or changes to the repository schema of the Orchestra standard since the v1.0 Technical Standard was published. The issues are recorded in GitHub at

- A – https://github.com/FIXTradingCommunity/fix-orchestra/issues or
- B – https://github.com/FIXTradingCommunity/fix-orchestra-spec/issues.

The GitHub issue numbers are shown in the header of the sections below with "A-" or "B-" as prefix.

2.1.1 Reduce restricted characters in names (A-#118, A-#119)

There is a business requirement (already in V1.0) to support all financial industry protocols, not just FIX.

The names of message, groups, components, fields, code sets, codes in the repository are defined as follows in V1.0, a format that limits the character set to what is required in FIX.

```
<xs:simpleType name="Name_t">
       <xs:restriction base="xs:string">
              <xs:minLength value="1"/>
              <xs:maxLength value="255"/>
              <xs:pattern value="([A-Z]|[a-z])([0-9]|[A-Z]|[a-z]|_)*"/>
       </xs:restriction>
</xs:simpleType>
```

Other naming conventions for messages and components are slightly different from FIX and may require dot ('.') in the names.

Since Orchestra supports both FIX and non-FIX protocols, naming rules are relaxed in the XML schema for V1.1. FIX and other style rules should be enforced by other means, such as a validator applications. The only restriction remaining is that names are of XML schema datatype “token”, which trims leading and trailing spaces and disallows some non-printable characters like line feeds and carriage returns. Tokens are limited to 64 characters (previously 255) to avoid long names causing issues with tools such as code generators.

```
<xs:simpleType name="Name_t">
       <xs:restriction base="xs:token">
              <xs:minLength value="1"/>
              <xs:maxLength value="64"/>
       </xs:restriction>
</xs:simpleType>
```

2.2 *Technical Requirements for Repository Schema*

This section describes enhancements or changes to the repository schema of the Orchestra standard since the v1.0 Technical Standard was published. The issues are recorded in GitHub at

- A – https://github.com/FIXTradingCommunity/fix-orchestra/issues or
- B – https://github.com/FIXTradingCommunity/fix-orchestra-spec/issues.

The GitHub issue numbers are shown in the header of the sections below with "A-" or "B-" as prefix.

The following proposed changes are not backward-compatible with v1.0.

- Change of sort attribute for "codeType" to from string to numeric (see section 2.2.3)
- Scenario reference with a numerical ID instead of by name (see section 2.2.13)
- Distinguish between datatype and code set for a field type (see section 2.2.16)

2.2.1 Add explicit link between fields for their encoded versions (A-#94)

Fields like Issuer(106) are also supported in an encoded version, in this case EncodedIssuer(349). The latter requires a second field EncodedIssuerLen(348) for tag value encoding that defines the length of the encoded field. It refers to the encoded field with the attribute "lengthId".

The link to the non-encoded field Issuer(106) is only implicit in the description of the encoded field. It is now captured by the attribute "nonEncodedFieldId" (positive integer) of a field definition or field reference, e.g. for EncodedIssuer(349) the attribute "nonEncodedFieldId" can be set to "106" to link to Issuer(106).

```
<xs:attribute name="nonEncodedFieldId" type="fixr:id_t"/>

<xs:simpleType name="id_t">
       <xs:restriction base="xs:positiveInteger"/>
</xs:simpleType>
```

2.2.2 Orchestra should support xi:include A-#112

There is a simple way in Orchestra v1.0 to support xi:include in the XML repository files, a basic requirement for Orchestra to serve very large projects is to split the repository into multiple XML files. It is proposed to enhance the documentation to clarify the usage already supported with V1.0. The spec should explicitly list all top-level elements that support XInclude: actors, concepts, sections, categories, messages, groups, components, fields, code sets, datatypes, scenarios.

For example, the `<fixr:datatypes>` element can be replaced with the path to the XML file, e.g. `<xi:include href="src/test/resources/datatypes.xml"/>`. The datatypes then need to be defined in a separate file `datatypes.xml` that needs to contain the same namespace as the root element of the repository, i.e. `<fixr:datatypes xmlns:fixr="http://fixprotocol.io/2023/orchestra/repository">`.

2.2.3 Change sort attribute of codeType to numeric (A-#113)

The sort attribute is used to enforce order on codes within a code set for documentation purposes only. (A set is by definition unordered.) It is currently of type string in the XML schema but has been changed to nonNegativeInteger as it was in the Unified Repository.

V1.0: `<xs:attribute name="sort" type="xs:string"/>`

V1.1: `<xs:attribute name="sort" type="xs:nonNegativeInteger"/>`

2.2.4 Remove latestEP attribute (A-#122)

This attribute of categories, sections, messages, groups, components, fields, is redundant to metadata for the repository and has been removed.

2.2.5 Allow a message without a structure (A-#123)

The *UnsequencedHeartbeat* message of the FIX Performance session layer (FIXP) has no body, just a header on the wire. However, Orchestra schema doesn't let you have an empty message structure. This is an unusual case, but it should be allowed by the schema.

```
<xs:complexType name="messageType">
       <xs:sequence>
              <xs:element name="structure" minOccurs="0">
              …
```

2.2.6 Support annotations for top-level elements (A-#121, A-#127)

Each element, e.g. `<message>`, can have an annotation element, but the parent element, e.g. `<messages>`, does not have an annotation in the Orchestra repository schema to hold documentation for all messages. This has been added for all top level elements and applies to, messages, fields, components, etc. This is useful to keep common documentation as part of the Orchestra XML file. The following shows this for messages:

```
<xs:element name="messages">
      <xs:complexType>
            <xs:sequence>
                  <xs:element name="message" type="fixr:messageType" minOccurs="0"
                              maxOccurs="unbounded"/>
                  <xs:element name="annotation" type="fixr:annotation" minOccurs="0"/>
            </xs:sequence>
            …
```

2.2.7 Allow empty collection elements (A-#128)

Currently, `<codeSets>` is optional, but if supplied it must contain at least one instance of `<codeSet>`. It can now be empty.

```
<xs:element name="codeSets">
      <xs:complexType>
            <xs:sequence>
                  <xs:element name="codeSet" type="fixr:codeSetType" minOccurs="0"
                              maxOccurs="unbounded">
```

2.2.8 Change identifier attributes from string to token (A-#129)

Most identifier attributes in the repository schema are of XML schema datatype "string". The identifiers for abbreviations and scenarios have been changed to "token" (defined by simple type Name_t) which suppresses leading and trailing whitespace and line breaks. It allows internal spaces, but replaces multiple spaces with a single space. See also section 2.1.1 *Reduce restricted characters in names (A-#118, A-#*119).

```
V1.0: <xs:attribute name="abbrName" type="fixr:Abbreviation_t"/>
```

```
V1.1: <xs:attribute name="abbrName" type="fixr:Name_t"/>

V1.0: <xs:attribute name="scenario" type="fixr:Scenario_t" default="base">
V1.1: <xs:attribute name="scenario" type="fixr:Name_t" default="base">
```

2.2.9 Change version attribute to be less restrictive (A-#132)

The format of the repository version attribute should be more freeform. Orchestra v1.0 explicitly supported the FIX naming convention for versions as well as a generic conventions. The version attribute is now simply a token, which suppresses leading and trailing whitespace and line breaks. The following shows the old definition for V1.0 followed by the new definition for V1.1.

```
<xs:simpleType name="Version_t">
      <xs:restriction base="xs:string">
            <xs:pattern value="(FIX.2.7)|(FIX.3.0)|(FIX\.4\.[0-4])|((FIX.Latest|(FIX\.5\.0
(SP\d{1,2})?))(_EP((9[8-9])|([1-9][0-9][0-9])))?)|(FIXT.1.1)|([0-9]+)\.([0-9]+)|(\d{8})"/>
      </xs:restriction>
</xs:simpleType>

<xs:simpleType name="Version_t">
      <xs:restriction base="xs:token">
      </xs:restriction>
</xs:simpleType>
```

2.2.10 Allow name in element references (A-#136, A-#188, B-#27)

References to groups, components, and fields may now contain the name in addition to the identifier. The optional name is for convenience (legibility of Orchestra XML file) but will not be enforced by referential integrity. Only the name in a referred object (field, component, group, message) is authoritative. A validator may check the consistency between the name used for the reference and the name of the referred object.

The samples in the Orchestra v1.1 Technical Specification have been enhanced with the names of the referenced elements to show the effect.

```
<xs:attributeGroup name="refidGrp">
      <xs:attribute name="id" type="fixr:id_t" use="required"/>
      <xs:attribute name="name" type="fixr:Name_t"/>
      …
```

2.2.11 Add scenarios for datatypes (A-#151)

Messages, components, groups, fields, and codesets can have scenarios in Orchestra v1.0, but datatypes do not. In the Orchestra schema, a field has a type. Field type can be one of the FIX datatypes listed in that section, such as int, Price, or UTCTimestamp, or type can refer to a code set. This makes sense since a code set is semantically a specialized datatype that declares a finite set of valid values. The fact that a code set can have scenarios while a datatype cannot breaks cross-references that try to follow two different shapes. It also fails XML validation.

Scenarios for datatypes not only solves the XML reference problem for types, but it is a useful feature. In FIX datatypes, there is currently only "int" for integer while in SBE, integers can be encoded as 8-, 16-, 32-, or 64-bit and signed or unsigned integers. It would be useful to create scenarios of "int" for short and long integers, or ordinals which are only positive numbers. (Example of an ordinal is to convey the level of an order book as 1st, 2nd, 3rd, etc.) Another possible application would be subclasses of Price for equities vs. FX, that have very different precisions, or non-decimal agricultural commodity prices. In short, adding scenarios to datatypes makes them both more targeted and extensible. Furthermore, mapping from datatypes to encodings such as SBE can now be made more precise.

```
<xs:element name="datatype">
      <xs:complexType>
            <xs:sequence>
                  <xs:element name="mappedDatatype" type="fixr:mappedDatatype"
                            minOccurs="0" maxOccurs="unbounded"/>
                  <xs:element name="annotation" type="fixr:annotation" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="name" type="fixr:Name_t" use="required"/>
            <xs:attribute name="scenario" type="fixr:Name_t" default="base"/>
            <xs:attribute name="scenarioId" type="fixr:id_t" default="1"/>
            <xs:attribute name="baseType" type="fixr:Name_t"/>
            <xs:attributeGroup ref="fixr:entityAttribGrp"/>
      </xs:complexType>
</xs:element>
```

2.2.12 Add size attribute to mappedDatatype (A-#152)

The mappedDatatype element specifies how a datatype maps to a message encoding. Most of the elements originated in the Unified Repository schema that were originally used to map FIX datatypes to XML Schema. Orchestra was enhanced to map to any FIX or other encoding, such as SBE or Google Protocol Buffers. There is one gap in that the size or length attribute cannot be expressed, e.g. for aggregate datatypes such as an array. SBE and other encodings support fixed-length fields. A new attribute "size" is being proposed to support this capability.

```
<xs:complexType name="mappedDatatype">
      <xs:sequence>
            <xs:element name="extension" minOccurs="0">
            …
            </xs:element>
            <xs:element name="annotation" type="fixr:annotation" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="standard" type="fixr:datatypeStandard_t" use="required"/>
      <xs:attribute name="builtin" type="xs:boolean"/>
      <xs:attribute name="base" type="xs:string"/>
      <xs:attribute name="pattern" type="xs:string"/>
      <xs:attribute name="element" type="xs:string"/>
      <xs:attribute name="size" type="xs:nonNegativeInteger"/>
      <xs:attribute name="parameter" type="xs:string"/>
      <xs:attribute name="minInclusive" type="xs:string"/>
      <xs:attribute name="maxInclusive" type="xs:string"/>
</xs:complexType>
```

This is a duplicate to B-#33 (https://github.com/FIXTradingCommunity/fix-orchestra-spec/issues/33).

2.2.13 Centralize declarations of scenarios (PR A-#157)

In the v1.0 schema, scenario names were entered independently on each message or message element without any enforced consistency. Unlike other Orchestra elements, scenarios were also only identified by name but had no numeric identifier.

It is now possible to centralize the declaration of scenarios in an Orchestra repository file to make clear what the themes are. Scenarios may be used for different use cases, including variations by asset class and the like.

```
<xs:element name="scenarios">
      <xs:annotation>
            <xs:documentation>
                  The default scenario is id='1' name='base'.
            </xs:documentation>
      </xs:annotation>
      <xs:complexType>
            <xs:sequence>
                  <xs:element name="scenario" type="fixr:scenarioType" minOccurs="0"
                              maxOccurs="unbounded"/>
                  <xs:element name="annotation" type="fixr:annotation" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute ref="xml:base"/>
      </xs:complexType>
</xs:element>
```

References to declared scenarios have been changed to numeric IDs rather than by name since numbers are more efficient to compare than strings, and don't have complications with spacing, capitalization, etc. However, the names may supplement the numeric keys to make the file more humanly readable. This is consistent with cross references for field references. See section 2.2.10 above *Allow name in element references (A-#136, A-#188,* B-#27).

```
<xs:attributeGroup name="refidGrp">
      <xs:attribute name="id" type="fixr:id_t" use="required"/>
      <xs:attribute name="name" type="fixr:Name_t"/>
      <xs:attribute name="scenario" type="fixr:Name_t" default="base"/>
      <xs:attribute name="scenarioId" type="fixr:id_t" default="1"/>
</xs:attributeGroup>
```

Previously, the attribute group "refidGrp" only had the attribute "scenario", i.e. reference by name. The change includes the addition of the attribute "scenarioId" to "refidGrp" and the definition of keys with the new attribute "scenarioId" for all scenario references as follows.

```
            <xs:key name="scenarioIdKey">
                  <xs:selector xpath="fixr:scenarios/fixr:scenario"/>
                  <xs:field xpath="@id"/>
            </xs:key>
            <xs:keyref name="codeSetScenarioKeyRef" refer="fixr:scenarioIdKey">
                  <xs:selector xpath="fixr:codeSet"/>
                  <xs:field xpath="@scenarioId"/>
            </xs:keyref>
            <xs:keyref name="datatypeScenarioKeyRef" refer="fixr:scenarioIdKey">
                  <xs:selector xpath="fixr:datatype"/>
                  <xs:field xpath="@scenarioId"/>
            </xs:keyref>
            <xs:keyref name="componentScenarioKeyRef" refer="fixr:scenarioIdKey">
                  <xs:selector xpath="fixr:component"/>
                  <xs:field xpath="@scenarioId"/>
            </xs:keyref>
            <xs:keyref name="groupScenarioKeyRef" refer="fixr:scenarioIdKey">
                  <xs:selector xpath="fixr:group"/>
                  <xs:field xpath="@scenarioId"/>
            </xs:keyref>
            <xs:keyref name="messageScenarioKeyRef" refer="fixr:scenarioIdKey">
            <xs:selector xpath="fixr:message"/>
            <xs:field xpath="@scenarioId"/>
        </xs:keyref>
```

2.2.14 Add unionDataType as attribute of a code set (A-#161)

The attribute unionDataType is mainly used on fields that have code sets. It is used there to constrain the range of user-defined values that extend the standard code set. To keep usage consistent in that case, unionDataType is added as an attribute of a code set element. This avoids having to add it to every field that references the code set. For example, there is a large number of different fields for party roles in FIX (PartyRole, PartyDetailRole, RootPartyRole, NestedPartyRole,…). They all use the same code set PartyRoleCodeSet. The change allows to define the uinionDataType in a single place.

The working group decided to retain unionDataType as a field attribute to support the few cases that are not related to code sets. It is recommended to move the unionDateType attribute from the field definition to the code set definition.

```
<xs:complexType name="codeSetType">
        <xs:sequence>
                <xs:element name="code" type="fixr:codeType" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="annotation" type="fixr:annotation" minOccurs="0"/>
        </xs:sequence>
        <xs:attributeGroup ref="fixr:oidGrp"/>
        <xs:attribute name="type" type="fixr:Name_t" use="required"/>
        <xs:attribute name="default" type="xs:string"/>
        <xs:attribute name="specUrl" type="xs:anyURI"/>
        <xs:attributeGroup ref="fixr:entityAttribGrp"/>
        <xs:attribute name="unionDataType"  type="fixr:UnionDataType_t"/>
</xs:complexType>
```

2.2.15 Correlate incoming tag-value message to its scenario (A-#166)

The class of a FIX tag-value message is traditionally identified by MsgType(35). However, this encoding does not by itself support scenarios. In fact, scenarios in Orchestra were conceived to create useful subsets of overloaded message types, each for a specific use case. Conversely, a message definition in the FIX standard is a union of all possible uses, across all markets, asset classes, and use cases.

The proposal is to add sufficient metadata to an Orchestra file to infer the scenario using a rule based on message contents. For example, if ExecutionReport(35=8) has a scenario for trade executions, then a rule could be expressed as (pseudocode, not actual syntax) "if MsgType(35)=8 (ExecutionReport) and 150=F (Traded) then ScenarioID=6". If a message arrived with that combination of field values, the Orchestra scenario could be selected with confidence. For performance purposes, the decision tree could be generated as a table or generated code so there wouldn't be reckoning of XML at runtime.

It is therefore proposed to add an optional `<when>` element to a message definition `<messageType>` to differentiate scenarios of a message type. The new element would follow the exact same syntax as `<when>` elements in responses and rules that are used to explain conditionally required fields. The contents of `<when>` is a Score DSL expression. It is a predicate (Boolean expression) that tells if the scenario applies. That is, if the expression evaluates to true. The expression could distinguish scenarios by order state, asset class, party role, or the like.

```
<xs:element name="when" type="fixr:expressionType" minOccurs="0">
	<xs:annotation>
		<xs:documentation>
			A condition that distinguishes when a scenario of a message type
			applies. It could be used to generate a decision tree to correlate an
			incoming message to its scenario, or to decide which scenario of a request
			message to send.
		</xs:documentation>
	</xs:annotation>
</xs:element>
```

The following is an example showing the actual syntax for the pseudocode above.

```
<fixr:message msgType="8" id="9" name="ExecutionReport" scenarioId="6">
	<fixr:structure>
		<fixr:componentRef presence="required" id="1024" name="StandardHeader"/>
		...
		<fixr:componentRef presence="required" id="1025" name="StandardTrailer"/>
	</fixr:structure>
	<fixr:when>ExecType == ^Trade/>
</fixr:message>

<fixr:scenario name="Execution" id="6"/>
```

2.2.16 Distinguish between datatype and code set for a field (A-#170)

The field type attribute does not distinguish between a datatype and a code set reference. In both cases the attribute "type" is being used, e.g. "type=int" or "type=ExecTypeCodeSet". There should be a different attribute for each kind of reference. It is proposed to add a new attribute "codeSet" when the reference is made to a code set, e.g. "codeSet=ExecTypeCodeSet" and to use "type" only for references to `<datatype>` elements.

```
<xs:sequence>
	<xs:element name="field" type="fixr:fieldType" minOccurs="0" maxOccurs="unbounded">
		<xs:key name="typeKey">
			<xs:selector xpath="."/>
			<xs:field xpath="@type|@codeSet"/>
		</xs:key>
	</xs:element>
	<xs:element name="annotation" type="fixr:annotation" minOccurs="0"/>
</xs:sequence>
```

2.2.17 Removal of simple type "Edition_t" (A-#180)

The repository schema contains a simple type "Edition_t" that is not used anywhere. It is proposed to remove the following:

```
<xs:simpleType name="Edition_t">
	<xs:restriction base="xs:string">
		<xs:maxLength value="8"/>
	</xs:restriction>
</xs:simpleType>
```

2.2.18 Deprecated attribute needs definition (B-#2)

Elements in XML schema have attributes for pedigree (history of changes). The `deprecated` attribute needs a more precise definition. The following is proposed to be added:

Deprecated elements are not removed from the repository. They may still be used based upon bilateral agreement. However, the publisher of the Orchestra XML file recommends to avoid its usage and should use the `replacedByField` attribute in case of a 1:1 change. The `<annotation>` element may additionally be used to describe the alternate element(s) and new approach.

2.3 Technical Requirements for Interfaces Schema

This section describes enhancements or changes to the interfaces schema of the Orchestra standard since the v1.0 Technical Standard was published. The issues are recorded in GitHub at

- A – https://github.com/FIXTradingCommunity/fix-orchestra/issues or
- B – https://github.com/FIXTradingCommunity/fix-orchestra-spec/issues.

The GitHub issue numbers are shown in the header of the sections below with "A-" or "B-" as prefix.

2.3.1 Add status information for an interface document (A-#149)

For auditing, analysis, and other purposes, messages need to be stored for a long time, perhaps years. How can they be brought out of archives and correlated to scenarios in a Rules of Engagement document that was perhaps published years ago but changed many times since?

An effective date-time is added to a defined interface that may be used to correlate an interface version to a message that was captured on a certain date. Since session configurations already have a concept of "activationTime" and "deactivationTime", those concepts can be reused for service offerings and orchestration versioning.

Furthermore, a "deprecated" attribute is added to an interface to inform users that a service offering may be removed or replaced in the future.

```
<xs:complexType name="protocolType">
      <xs:sequence>
            <xs:element name="annotation" type="fixi:annotation" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="name" type="fixi:protocolName_t"/>
      <xs:attribute name="version" type="xs:token"/>
      <xs:attribute name="activationTime" type="xs:dateTime"/>
      <xs:attribute name="deactivationTime" type="xs:dateTime"/>
      <xs:attribute name="deprecated" type="xs:dateTime"/>
      <xs:attribute name="layer" type="fixi:layer_t"/>
      <xs:attribute name="reliability" type="fixi:reliability_t"/>
      <xs:attribute name="orchestration" type="xs:anyURI"/>
      <xs:anyAttribute processContents="lax"/>
</xs:complexType>
```

2.3.2 Change string values to tokens (A-#150)

The Orchestra interfaces schema now defines all attributes as tokens that were previously defined as strings, e.g. the version of a protocol.

```
V1.0: <xs:attribute name="version" type="xs:string"/>
V1.1: <xs:attribute name="version" type="xs:token"/>
```

Only exception is the `<securityKeys>` element of `<sessionType>` as it is a textual encoding as specified by IETF RFC 7468.

2.3.3 Typo in complex type "userInterfaceType" (A-#180)

The name of the complex type to define a user interface is missing an "r" in two places that should be changed as follows:

```
<xs:complexType name="userInterfaceType">
      <xs:complexContent>
            <xs:extension base="fixi:protocolType"/>
      </xs:complexContent>
</xs:complexType>
<xs:element name="userInterface" type="fixi:userInterfaceType" minOccurs="0"
maxOccurs="unbounded"/>
```

2.4 Open issues

This section lists open issues that have either been deferred to the next Release Candidate or dropped from Orchestra v1.1.

2.4.1 Open issues deferred to RC2

The following open issues are from https://github.com/FIXTradingCommunity/fix-orchestra

- Syntax for header and trailer inclusion (https://github.com/FIXTradingCommunity/fix-orchestra/issues/27)
- Should Orchestra indicate which application messages are eligible for session level retransmission? (https://github.com/FIXTradingCommunity/fix-orchestra/issues/70)
- Consider adding section as an attribute to components (https://github.com/FIXTradingCommunity/fix-orchestra/issues/93)
- Semantic equivalence of fields, components, and groups (https://github.com/FIXTradingCommunity/fix-orchestra/issues/134)
- Support mixed expression/script languages (https://github.com/FIXTradingCommunity/fix-orchestra/issues/164)
- Explicit type scenarios for fields and type scenario overrides for fieldRefs (https://github.com/FIXTradingCommunity/fix-orchestra/pull/173)
- Missing AppInfo FIXML schema (https://github.com/FIXTradingCommunity/fix-orchestra/issues/179)
- Add repository attribute for application extension ID (https://github.com/FIXTradingCommunity/fix-orchestra/issues/186)

- Allow names in correlation and assignment references (https://github.com/FIXTradingCommunity/fix-orchestra/issues/187)
- Attribute group messageAttribGrp seems to be unused (https://github.com/FIXTradingCommunity/fix-orchestra/issues/189)
- Consider moving "Datatype" from repository.xsd's root level to repositoryType.xsd (https://github.com/FIXTradingCommunity/fix-orchestra/issues/190)
- Some categoryType's attributes should be required (https://github.com/FIXTradingCommunity/fix-orchestra/issues/191)
- Some protocols's message names are not supported by Name_t restrictions (https://github.com/FIXTradingCommunity/fix-orchestra/issues/193)

The following open issues are from https://github.com/FIXTradingCommunity/fix-orchestra-spec

- Datatypes in DSL (https://github.com/FIXTradingCommunity/fix-orchestra-spec/issues/3)
- Add verbiage for all attributes provided in the XSD file (https://github.com/FIXTradingCommunity/fix-orchestra-spec/issues/28)
- Use ISO/IEC 11179 terminology for Concepts (https://github.com/FIXTradingCommunity/fix-orchestra-spec/issues/32)

2.4.2 Open issues dropped from V1.1

The following open issues are from https://github.com/FIXTradingCommunity/fix-orchestra

- Section for default values (https://github.com/FIXTradingCommunity/fix-orchestra/issues/35)

3 Issues and Discussion Points

3.1 Name of the Standard

This standard has been called “FIX Orchestra”, but that name obscures its design goal is to describe both FIX and non-FIX protocols. Although it remains a standard published by the FIX Trading Community, it can be referred to as “Orchestra, a technical standard developed by FIX”, "Orchestra Standard" or simply “Orchestra”.

Resolution: The name “Orchestra Standard” will be used for the technical specification as of V1.1.

3.2 Comparison with Unified Repository

The Orchestra v1.0 specification contains an appendix that describes the differences to the Unified Repository (a.k.a. Repository 2010 Edition). This information is useful for the transition from Unified to Orchestra. However, it is better suited for a separate guideline than for a normative specification.

Resolution: The Appendix 7.1 is removed from the Orchestra V1.1 specification.

3.3 Attribute "baseFieldID" to reference related fields

A new attribute "baseFieldId" is introduced to relate encoded fields to their non-encoded version. It is also intended to capture relationships between fields such as party identifier and security types in the FIX Protocol where there are multiple fields that have different identifiers and names only because of the FIX tag=value encoding to enable message parsing.

However, fields like SecurityDesc(207) and EncodedLegSecurityDesc(622) have two relationships that cannot be distinguished with a single attribute "baseFieldId".

Resolution: RC1 of v1.1 will only add an attribute "nonEncodedFieldId" that is specific to the relationship of fields with their encoded versions. Additional attributes may be introduced in future release candidates to convey other relationships.

4 References

Authors should list references used in creating the technical standard proposal

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

6 Intellectual Property Disclosure

Authors should provide a list of any intellectual property

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 may be delivered in several release candidates rather than attempting a big-bang approach.

8.1.1 Version 1.1 RC1 deliverables

These artifacts will be delivered as v1.1 RC1:

- The technical specification is a separate document "Orchestra Standard – Technical Specification". The document will be displayed in the download page for Orchestra on the FIX Trading Community website (https://www.fixtrading.org/standards/fix-orchestra-standard/). The document source in markdown in available in the GitHub project FIXTradingCommunity/fix-orchestra-spec.

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

- XML schema (XSD) for repository of message definitions and workflow plus documentation of the schema
- XML schema (XSD) for interfaces and session configuration plus documentation of the schema
- Repository Validator to check whether an Orchestra file conforms to the XML schema.
- Links to other GitHub projects providing Orchestra tutorials and experimental projects.

8.2 Plan for EP production

In addition to standard development, a plan will be created to migrate the building of Extension Packs from Basic/Unified Repository to Orchestra. In addition to modernizing build facilities, this plan will need to account for the tools that consume Repository, including FIXimate, FIXML schema generation, and so forth. The tools to generate FIXimate, FIXML schema and Unified Repository have been adapted to use an Orchestra XML v1.0 file as source. The plan is to deprecate the Basic repository when migrating to Orchestra as the master source for FIX Latest.

8.3 Tool Migration

Tools that have been coded to Orchestra v1.0 will need to be updated to v1.1.

- FIXimate4 generator
- Log2orchestra
- Playlist
- Tablature
- FIXML schema generator
- SBE schema generator
- Unified Repository

Appendix A - Usage Examples

This is a required section where the sub-committee or working group can provide whole or fragments of example FIX messages with actual or dummy data. These examples are useful for illustrating usage or rules specific to the business domain covered in the proposal.

These example Orchestra files are posted in GitHub but are so far only applicable to Orchestra v1.0.

Example order entry file developed by MilleniumIT

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

Sample interface file

https://github.com/FIXTradingCommunity/fix-orchestra/blob/master/interfaces/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 technical standard must include some plan for measuring compliance with the standard. This will either be test suites, a validation tool (such as an XML Schema document as an example).

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 is not being developed by FIX.