

FIXatdl Version 1.2

Release Candidate 1

Technical Standard Proposal

November 10, 2020

Rev 4

Proposal Status: Draft

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.

[](https://creativecommons.org/licenses/by-nd/4.0/)  
This work is licensed under a [Creative Commons Attribution-NoDerivatives 4.0 International License](https://creativecommons.org/licenses/by-nd/4.0/).

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

Copyright 2003-2020 FIX Trading Community™, all rights reserved.

Table of Contents

Auto-generate the entire table of contents (press F9) here - or customize up to 3 levels deep.

[Document History 4](#_Toc45145217)

[1 Introduction 5](#_Toc45145218)

[1.1 Authors 5](#_Toc45145219)

[2 Requirements 5](#_Toc45145220)

[2.1 Business Requirements 5](#_Toc45145221)

[2.2 Technical Requirements 5](#_Toc45145222)

[2.2.1 New Method to Enable/Disable Clock Controls 5](#_Toc45145223)

[2.2.2 Duration as an Alternative to Expiration Time 8](#_Toc45145224)

[2.2.3 Additional Clock Control Features 10](#_Toc45145225)

[2.2.4 Grid Layout for Strategy Panels 11](#_Toc45145226)

[2.2.5 Support for Basket, List and Multileg Order Types 14](#_Toc45145227)

[2.2.6 Additional Global Definitions 20](#_Toc45145228)

[2.2.7 OMS Hooks 20](#_Toc45145229)

[2.3 Samples 23](#_Toc45145230)

[3 Issues and Discussion Points 23](#_Toc45145231)

[3.1 Questions to be decided beyond RC1 23](#_Toc45145232)

[4 References 23](#_Toc45145233)

[5 Relevant and Related Standards 24](#_Toc45145234)

[6 Intellectual Property Disclosure 24](#_Toc45145235)

[7 Definitions 24](#_Toc45145236)

[8 Project Milestone 24](#_Toc45145237)

[8.1 Release Candidate 1 Deliverables 24](#_Toc45145238)

[8.2 Roadmap 24](#_Toc45145239)

A Table of Figures is not required. If used, use styles to tag the captions and auto-generate the list here. If not used, remove this section.

# Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Date | Author(s) | Revision Comments |
| 1 | Feb 11, 2020 | Greg Malatestinic, Jordan & Jordan | Initial draft |
| 2 | Jul 8, 2020 | Greg Malatestinic, Jordan & Jordan | Addition material describing and clarifying Grid features. |
| 3 | Oct 8, 2020 | Greg Malatestinic, Jordan & Jordan | Removed option of Duration parameter defined as an int. |
| 4 | Nov 10, 2020 | Hanno Klein, GTC | Review prior to GTC submission   * Standardized field references |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

The above document history section, including date, author, and comments, is required to track editing changes to the document. List revisions in **ascending order**. Please insert additional rows in the table as needed.

Template version information:

r0.0: 2013-03-13 Initial draft

r0.1: 2013-03-27

r0.2: 2013-08-16 Clarified Section 2, provided subsections for "Business Requirements" and "Technical Requirements". Updated instructions.

# Introduction

Provide an introduction to the content, purpose, or impetus of the proposal; the business need / problem being solved; and the scope. Include and label any references, supporting documentation, and related proposals. If the proposal is based on existing implementations, describe them here in the appropriate subsections. It is recommended that a "Summary of Proposed Changes" sub-section be provided within this section.

Since the approval of FIXatdl 1.1 errata in December 2010, we have received feedback from the community regarding features that are not currently supported by the standard. In January 2011, proposals for these features were investigated, to varying degrees, to determine their feasibility and impact to existing FIXatdl OMS vendor implementations. However, nothing progressed beyond that stage.

In August 2018, a questionnaire was sent to several sell-side firms which have developed their own FIXatdl definition files and several vendors of OMS platforms that support FIXatdl, asking what new features or changes they would like to see in the standard. The features proposed during the original community feedback period and as well as those features recently proposed were collected.

The FIXatdl Working Group was established in April 2019 with a goal of adding new functionality and addressing the requests for new features and changes that were collected.

## Authors

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Affiliation** | **Contact** | **Role** |
|  |  |  |  |
|  |  |  |  |

Provide list of authors of technical standard, their company or organizational affiliation, public email and or telephone number, and role in drafting the standard.

# Requirements

## Business Requirements

This proposal is intended to introduce changes to FIXatdl 1.1, which was made available to the public in 2010, which address specific pain-points while limiting breaking-changes that would significantly affect existing OMS implementations and require new development.

## Technical Requirements

This section presents the major enhancements and new functionality which will be included in the first release candidate of FIXatdl 1.2.

### New Method to Enable/Disable Clock Controls

Clock controls are the GUI component rendered in an OMS/EMS that allows a user to enter a time of day value. For example, the most common parameters to an algorithmic order, “Start Time” and “End Time” will be rendered via a clock control. A common use case involving a clock control is one where the user enters an order without specifying a time in the control, thus keeping a value from going out over the wire in the order message. The receiving broker in this case will apply a default value or default behavior based on the non -presence of this field. To do this, two helper controls are used. They are either check boxes or radio buttons and affect the value of the Clock control by use of a StateRule.

For example:

<lay:StrategyPanel orientation="VERTICAL" title="Start Time">

<lay:StrategyPanel orientation="HORIZONTAL">

<lay:Control ID="c\_NoStartTime" xsi:type="lay:RadioButton\_t" label="Now" initValue="true" radioGroup="StartTimeRB">

</lay:Control>

</lay:StrategyPanel>

<lay:StrategyPanel orientation="HORIZONTAL”>

<lay:Control ID="c\_EnableStartTime" xsi:type="lay:RadioButton\_t" label="" radioGroup="StartTimeRB">

</lay:Control>

<lay:Control ID="StartTimeClock" xsi:type="lay:Clock\_t" label="" parameterRef="StartTime">

<flow:StateRule enabled="false" value="{NULL}">

<val:Edit field="c\_EnableStartTime" operator="EQ" value="false"/>

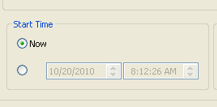
</flow:StateRule>

</lay:Control>

</lay:StrategyPanel>

</lay:StrategyPanel>

Here’s how it might be rendered:



The value of the Clock control depends on which radio button is selected. If the first is selected, then the state rule defined within the Clock control will set the value of the Clock to null. If the Control is null, then the Parameter bound to the Control is null, and the parameter / FIX Tag is not populated when the order message goes out on the wire. If the second button is selected, then the value of the parameter that goes out on the wire is taken directly from the value the user entered.

The same can be done using a dropdown (combo box) with two items instead:

<lay:StrategyPanel orientation="VERTICAL" title="Start Time">

<lay:StrategyPanel orientation="HORIZONTAL">

<lay:Control ID="c\_StartTimeOption" xsi:type="lay:DropDownList\_t" label="">

<lay:ListItem enumID="e\_now" uiRep="Now"/>

<lay:ListItem enumID="e\_custom" uiRep="Custom"/>

</lay:Control>

</lay:StrategyPanel>

<lay:StrategyPanel orientation="HORIZONTAL”>

<lay:Control ID="c\_StartTimeClock" xsi:type="lay:Clock\_t" label="" parameterRef="StartTime">

<flow:StateRule enabled="false" value="{NULL}">

<val:Edit field="c\_StartTimeOption" operator="EQ" value="e\_now"/>

</flow:StateRule>

</lay:Control>

</lay:StrategyPanel>

</lay:StrategyPanel>

Rendering the following:

A screenshot of a cell phone

Description automatically generated

Here the “Custom” item has been selected from the dropdown list. If the user had selected “Now” then the time below the dropdown would be greyed out.

In FIXatdl 1.2 we define a set of new Clock control attributes which will allow this behavior to be supported without the need of helper controls or state rules. To do so, an OMS would need to implement a compound GUI control (a GUI control with at least two underlying controls: a datetime picker and a check box / radio button). To achieve this goal, we propose the following attributes for the Clock control:

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Req’d | Description |
| enablingControlType | String. Valid values are restricted to “CheckBox”, “RadioButton” or “DropDown”. | N | Description of the GUI control to be rendered which directs the OMS to enable the clock and allow it to accept user input. is enabled (greyed-out with null value).  If an OMS supports this feature and enablingControlType is provided, then the OMS may render a GUI control of this type next to the GUI control intended to accept datetime values from the user. |
| disablingControlType | String. Valid values are restricted to “CheckBox”, “RadioButton” or “DropDown”. | N | Description of the GUI control to be rendered which directs the OMS to disable the clock (greyed-out with null value) and block users from providing input. |
| disablingControlText | String | N | Text to display next to the disabling GUI control. Intended to describe the effective result of explicitly disabling the clock via its disabling control. |

Using these attributes, the previous FIXatdl sample could be written as follows:

<lay:StrategyPanel orientation="VERTICAL" title="Start Time">

<lay:Control ID="StartTimeClock" xsi:type="lay:Clock\_t" label=""

disablingControlType=”RadioButton”

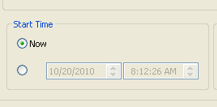
disablingControlLabel=”Now”

enablingControlType=”RadionButton” parameterRef="StartTime">

</lay:Control>

</lay:StrategyPanel>

The rendering would remain as before:



If an explicit GUI control used to disable user input is not desired, then the disabingControlType attribute can be omitted:

<lay:StrategyPanel orientation="VERTICAL" title="Start Time">

<lay:Control ID="StartTimeClock" xsi:type="lay:Clock\_t" label=""

enablingControlType=”RadionButton” parameterRef="StartTime">

</lay:Control>

</lay:StrategyPanel>

An OMS may choose to render this clock control as follows:

A screenshot of a cell phone

Description automatically generated

As with previous examples, if the radio button is selected then the value that goes on the wire is derived from what the user had entered. If the radio button is not selected, then no value goes on the wire.

### Duration as an Alternative to Expiration Time

While it is the best practice to receive the effective time of an order through use of FIX fields EffectiveTime(168) and ExpireTime(126), some algorithm providers employ a user-defined tag to receive a duration rather than an expiration time. The following example shows the layout of GUI controls that will collect a start time and end time:

<StrategyPanel orientation="HORIZONTAL">

<StrategyPanel orientation="VERTICAL" title="Start Time">

<StrategyPanel orientation="HORIZONTAL">

<Control ID="c\_NoStartTime" xsi:type="RadioButton\_t" label="Now" initValue="true" radioGroup="StartTimeRB">

</Control>

</StrategyPanel>

<StrategyPanel orientation="HORIZONTAL”>

<Control ID="c\_EnableStartTime" xsi:type="RadioButton\_t" label="" radioGroup="StartTinmeRB">

</Control>

<Control ID="StartTimeClock" xsi:type="Clock\_t" label="" parameterRef="StartTime">

<flow:StateRule enabled="false" value="{NULL}">

<val:Edit field="c\_EnableStartTime" operator="EQ" value="false"/>

</flow:StateRule>

</Control>

</StrategyPanel>

</StrategyPanel>

<StrategyPanel orientation="VERTICAL" title="End Time">

<StrategyPanel orientation="HORIZONTAL">

<Control ID="c\_NoEndTime" xsi:type="RadioButton\_t" label="Mkt Close" initValue="true" radioGroup="EndTimeRB">

</Control>

</StrategyPanel>

<StrategyPanel orientation="HORIZONTAL”>

<Control ID="c\_EnableEndTime" xsi:type="RadioButton\_t" label="" radioGroup="EndTimeRB">

</Control>

<Control ID="EndTimeClock" xsi:type="Clock\_t" label="" parameterRef="EndTime">

<flow:StateRule enabled="false" value="{NULL}">

<val:Edit field="c\_EnableEndTime" operator="EQ" value="false"/>

</flow:StateRule>

</Control>

</StrategyPanel>

</StrategyPanel>

</StrategyPanel>

Rendering:

A screenshot of a cell phone

Description automatically generated

For FIXatdl 1.2 we introduce the Duration control which represent a time span rather than a point in time. Replacing the EndTime control in the previous example with a duration control we get:

<StrategyPanel orientation="HORIZONTAL">

<StrategyPanel orientation="VERTICAL" title="Start Time">

<StrategyPanel orientation="HORIZONTAL">

<Control ID="c\_NoStartTime" xsi:type="RadioButton\_t" label="Now" initValue="true" radioGroup="StartTimeRB">

</Control>

</StrategyPanel>

<StrategyPanel orientation="HORIZONTAL”>

<Control ID="c\_EnableStartTime" xsi:type="RadioButton\_t" label="" radioGroup="StartTinmeRB">

</Control>

<Control ID="StartTimeClock" xsi:type="Clock\_t" label="" parameterRef="StartTime">

<flow:StateRule enabled="false" value="{NULL}">

<val:Edit field="c\_EnableStartTime" operator="EQ" value="false"/>

</flow:StateRule>

</Control>

</StrategyPanel>

</StrategyPanel>

<StrategyPanel orientation="VERTICAL" title="Duration">

<StrategyPanel orientation="HORIZONTAL">

<Control ID="c\_NoDuration" xsi:type="RadioButton\_t" label="Until the close" initValue="true" radioGroup="DurationRB">

</Control>

</StrategyPanel>

<StrategyPanel orientation="HORIZONTAL”>

<Control ID="c\_EnableDuration" xsi:type="RadioButton\_t" label="" radioGroup="DurationRB">

</Control>

<Control ID="c\_Duration" xsi:type="Duration\_t" label="" parameterRef="Duration">

<flow:StateRule enabled="false" value="{NULL}">

<val:Edit field="c\_EnableDuration" operator="EQ" value="false"/>

</flow:StateRule>

</Control>

</StrategyPanel>

</StrategyPanel>

Rendering:

A screenshot of a cell phone

Description automatically generated

The parameter, Duration, can be defined as a UTCTimeOnly field, as in the following statement:

<parameter name="Duration" xsi:type="UTCTimeOnly\_t" fixTag="9003" uiRep="Duration" use="optional"/>

It will be the responsibility of the OMS to correctly populate the Duration parameter on the wire (an integer or a time-related type) from the value returned by the Duration GUI control.

### Additional Clock Control Features

The following attributes are intended to give the algorithm provider additional control over what the user may see and do.

| Attribute | Type | Req’d | Dflt | Description |
| --- | --- | --- | --- | --- |
| @displayableDate | boolean | N | false | Instructs the OMS to display the date. |
| @editableDate | boolean | N | true | Instructs the OMS to allow the user to change the date. The OMS would decide how to do this. |
| @displayableTz | Boolean | N | true | Instructs the OMS to display the time zone associated with the value entered by the user. |
| @editableTz | Boolean | N | false | Instructs the OMS to allow the user to change the time zone. The OMS would decide how to do this. |

### Grid Layout for Strategy Panels

To better support the ability of FIXatdl to describe how GUI controls should be oriented when presented to an OMS user, we have added a feature that allows controls to be arranged in grid. Specifically, we are adding the value of “GRID” to the type PanelOrientation\_t. The XML schema definition is now:

<xs:simpleType name="PanelOrientation\_t">

<xs:restriction base="xs:string">

<xs:enumeration value="HORIZONTAL"/>

<xs:enumeration value="VERTICAL"/>

<xs:enumeration value=”GRID”/>

</xs:restriction>

</xs:simpleType>

As before, StrategyPanels define their orientation by setting their orientation attribute to one of these values which can now include “GRID”. For example:

<StrategyPanel orientation="GRID">

This will allow for all the elements contained in the panel, whether they are controls or other panels, to be arranged by rows and columns. Any item contained within a grid may declare a row, column, row span or column span value to explicitly guide its placement in the grid. However, explicitly declaring the placement of an item in the grid is optional. If row and column values are not provided, then the items are expected to be arranged in row-major or column-major order. A new attribute, fillOrder, will indicate which to use. If row or column span values are not provided, then the item is assumed to take up one row or column.

The following attributes may be specified in any <Control> or <StrategyPanel> elements which are child elements of a grid-oriented <StrategyPanel> element.

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Req’d | Description |
| row | Non-negative integer | N | Row in which this item is to appear in a grid-oriented panel. |
| col | Non-negative integer | N | Column in which this item is to appear in a grid-oriented panel. |
| rowSpan | Non-negative integer | N | Number of rows an item is to span in a grid-oriented panel. (Default 1.) |
| colSpan | Non-negative integer | N | Number of colums an item is to span in a grid-oriented panel. (Default 1.) |

The following attributes may be specified in any grid-oriented <StrategyPanel> element.

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Req’d | Description |
| numRows | Non-negative integer | N | Number of rows. Applicable for grid-oriented panels. |
| numCols | Non-negative integer | N | Number of columns. Applicable for grid-oriented panels. |
| fillOrder | String. Valid values are “COL-MAJOR” and “ROW-MAJOR”. | N | Describes how the grid items are to be arranged. Applicable for grid-oriented panels. (Default: COL-MAJOR.) |

In the following three code samples we create a panel with two rows and two columns. The rendering from each sample is identical. In the first, the controls, which are contained within the panel, each explicitly declare a row and column number.

<lay:StrategyPanel orientation="GRID" numRows=”2” numCols=”2”>

<lay:Control ID=”control1” label=”control1” row=”0” col=”0”/>

<lay:Control ID=”control2” label=”control2” row=”1” col=”0”/>

<lay:Control ID=”control3” label=”control3” row=”0” col=”1”/>

<lay:Control ID=”control4” label=”control4” row=”1” col=”1”/>

</lay:StrategyPanel>

Next, we leave out the <StrategyPanel> attributes, numRows and numCols, and can still determine how to render the controls based on the row and col attributes of each control.

<lay:StrategyPanel orientation="GRID">

<lay:Control ID=”control1” label=”control1” row=”0” col=”0”/>

<lay:Control ID=”control2” label=”control2” row=”1” col=”0”/>

<lay:Control ID=”control3” label=”control3” row=”0” col=”1”/>

<lay:Control ID=”control4” label=”control4” row=”1” col=”1”/>

</lay:StrategyPanel>

Finally, we allow an implicit declaration of each item’s placement by not specifying their row and column attributes. Given the number of rows and columns and the fill order, the arrangement of the controls is easily determined.

<lay:StrategyPanel orientation="GRID" numRows=”2” numCols=”2” fillOrder=”COL-MAJOR”>

<lay:Control ID=”control1” label=”control1”/>

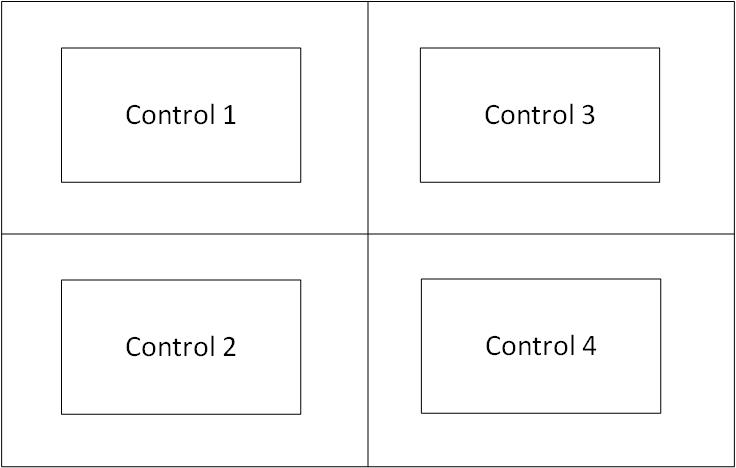
<lay:Control ID=”control2” label=”control2”/>

<lay:Control ID=”control3” label=”control3”/>

<lay:Control ID=”control4” label=”control4”/>

</lay:StrategyPanel>

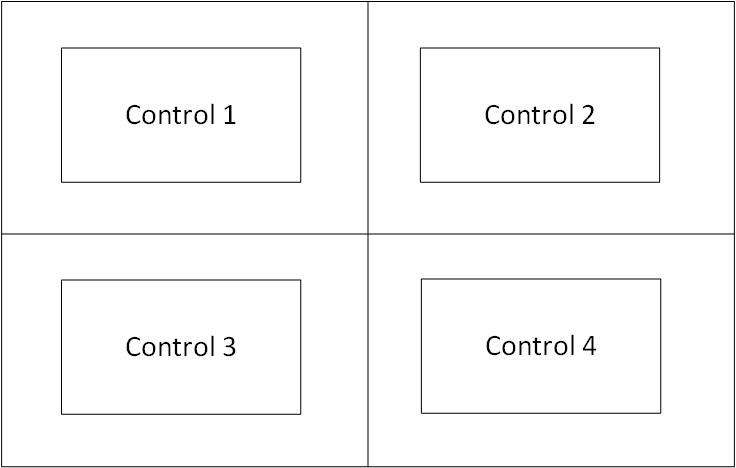
Each of the previous three samples will result in the same arrangement of the GUI controls:



Note that if we switch from column-major to row-major order, as in

<lay:StrategyPanel orientation="GRID" numRows=”2” numCols=”2” fillOrder=”ROW-MAJOR”>

the controls are rendered as follows:



When it makes sense for a control (or panel) to span multiple columns or rows, we can use the colSpan and rowSpan attributes. They provide the same functionality as “merge cell” in spreadsheet programs like Excel. The value of either attribute must be a positive integer and specifies the number of columns or rows that the control (or panel) fills. For example,

<lay:StrategyPanel orientation="GRID" numRows=”2” numCols=”2” fillOrder=”ROW-MAJOR”>

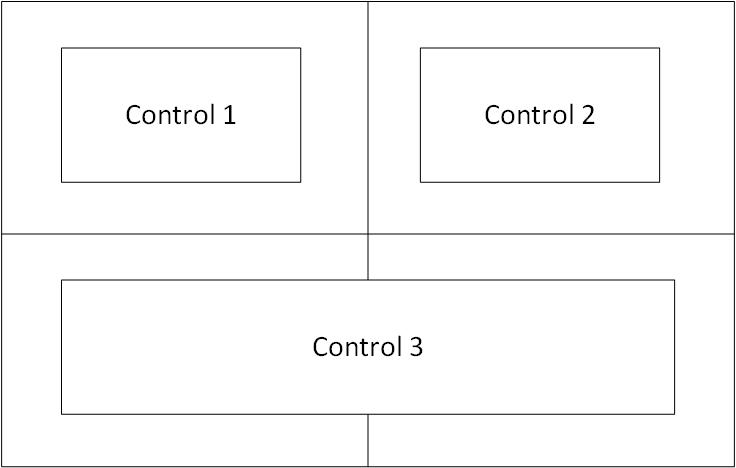
<lay:Control ID=”control1” label=”control1”/>

<lay:Control ID=”control2” label=”control2”/>

<lay:Control ID=”control3” label=”control3” colSpan=”2”/>

</lay:StrategyPanel>

will render the following:



#### Error Conditions

Since the attributes row, col, numRows, numCols, rowSpan, and colSpan are optional, their use may be prone to error. One must be rather careful not to define their values in such a way as to make their arrangement ambiguous or to be in conflict. With that in mind, guidance is provided for the following error conditions:

|  |  |  |
| --- | --- | --- |
| Error | Scenario | Resolution |
| Row/col conflicts. | Two or more items in a grid specify the same row and col values. | Ignore all row/col attributes of all grid items and render as if they had not been specified. |
| Row or col values are out-of-range. | The numRows and numCols attributes of a <StrategyPanel> are defined as N and M, but a child control’s row attribute is >= N or its col attribute is >= M. | Override the numRows or numCols attribute of the <StrategyPanel> with a value large enough to accommodate the child control’s row or col attribute. |
| Mismatch in parent-child orientation. | Grid attributes are defined on a control whose parent is not a grid. | The grid attributes of the control are ignored. |

This list is not definitive and is expected to grow as issues are raised and identified by those implementing FIXatdl.

### Support for Basket, List and Multileg Order Types

Though FIXatdl 1.1 provided support for multileg order entry, it proved to be insufficient in that some capabilities were missing and for those that were included, there was little clarity or direction in how to use them. As a result, few broker-dealers attempted to describe their multileg algorithmic order interfaces with FIXatdl.

With FIXatdl 1.2 we attempt to provide more robust support for multileg order types and clarify how their interfaces are described. In doing so, the following concepts were considered.

* **Order Delivery**

Any algorithm provider must inform its clients, or its clients’ OMSs, which method(s) to use so the order can be delivered as expected. There are three acceptable methods used to delivery multileg orders to an execution venue:

* + NewOrderSingle(35=D) messages - one for each leg, where an additional identifier is used to associate the individual legs with one another.
  + A NewOrderMultiLeg(35=AB) message - information about individual legs are placed into the LegOrdGrp repeating group.
  + A NewOrderList(35=E) message - information about individual legs are placed into the ListOrdGrp repeating group. Note that it may also be possible to partition the legs into several NewOrderList(35=E) messages provided they share the same values for ListID(66) and TotNoOrders(68).

The most common method in use today is to issue multiple NewOrderSingle(35=D) messages; one for each leg or each order in a basket.

* **Leg Count**

The description of the order interface must include a number representing the required number of legs for the strategy. An OMS should use this information to render a fixed number of GUI controls where values for the leg parameters can be entered. For example, a pairs strategy would require two legs; with this information, an OMS should render two sets of GUI controls which are associated with the fields of the legs. A value of “unbounded” must be allowed in order to support the delivery of a variable number of legs such as in the case of a basket/portfolio strategy. In FIXatdl 1.2 we introduce two attributes of the <Strategy> element for this purpose.

|  |  |
| --- | --- |
| Attribute | Description |
| Strategy/@minLegs | Use to indicate the minimum number of legs an order of this type requires. A renderer would use this information to display the required number of GUI panels where parameters can be entered and to properly package a multileg order in one or more FIX messages.  Valid values:  [a non-negative integer] – indicating the minimum leg count. |
| Strategy/@maxLegs | Use to indicate the maximum number of legs an order of this type requires. A renderer would use this information to display the required number of GUI panels where parameters can be entered and to properly package a multileg order in one or more FIX messages.  Valid values:  [a non-negative integer] – indicating the maximum leg count;  “unbounded” – indicating the strategy accepts any number of legs. |

The following examples show the description of a strategy which requires exactly two legs, followed by a strategy which requires one or more legs.

<Strategy name=”two-legged-order”>

. . .

minLegs="2" maxLegs="2"

. . .

</Strategy>

<Strategy name=”one-or-more-legged-order”>

. . .

minLegs="1" maxLegs="unbounded"

. . .

</Strategy>

* **Linking and Sequencing of Single Orders**

When accepting multileg orders via a group of NewOrderSingle(35=D) messages, the orders need to be linked. So, FIXatdl must allow for the definition of a tag number in which the OMS will place a unique ID (or “Global” Order ID) in each of the messages, the definition of a tag number in which the OMS will place a leg sequence number, and the definition of a tag number in which the OMS will place the total number of legs of the order.

|  |  |
| --- | --- |
| Attribute | Description |
| Strategy/@commonIDTag | Used to denote the tag which must contain a common ID linking all legs of a multileg order together. Applicable when multileg orders are delivered via several NewOrderSingle(35=D) messages. |
| Strategy/@legSequenceTag | Used to denote the tag which will contain the sequence number of an order of a basket or leg of a multileg order. Applicable when multileg orders are delivered via several NewOrderSingle(35=D) messages.  (Formerly @orderSequenceTag in FIXatdl 1.1.) |
| Strategy/@totalLegsTag | Used to denote the tag which will contain the total number of legs of an order. Applicable when multileg orders are delivered via several NewOrderSingle(35=D) messages. (Formerly @totalOrdersTag in FIXatdl 1.1.) |

For example, the following stratgey requires that linking and sequencing data is to be populated in the user-defined tags 7066, 7067 and 7068.

<Strategy>

. . .

commonIDTag="7066"

legSequenceTag="7067"

totalLegsTag="7068”

. . .

</Strategy>

Note that the semantics of these attributes are analogous to ListID(66), ListSeqNum(67) and TotNoOrders(68) in a NewOrderList(35=E) message. However, use of these values should be avoided as the associated fields are not members of the NewOrderSingle(35) message. Instead, broker/dealers should use UDFs.

* **Parameter Scope**

Algorithmic order parameters can either apply to the entire order or to the legs of the order. In the description of a parameter, the scope must be clear; at the order level or at the leg level. To represent this, we now wrap the leg parameters in a new XML element, <Leg>.

For example:

<Parameter name="p\_OrdParamA" xsi:type="Int\_t" fixTag="5000"/>

<Parameter name="p\_OrdParamB" xsi:type="Int\_t" fixTag="5001"/>

<Leg>

<Parameter name="p\_LegParamA" xsi:type="Int\_t" fixTag="6001"/>

<Parameter name="p\_LegParamB" xsi:type="Int\_t" fixTag="6002"/>

</Leg>

Here we have two strategy parameters; both of which will be included in each leg of the order.

* **Cancel/Modify of Legs**

When the delivery option being used is the NewOrderSingle(35=D) message then the OMS must know how to handle cancellation and modification of the order. The description of the strategy must indicate whether an individual leg can be cancelled or modified and, if so, whether it necessary to re-send all the legs that were not modified.

|  |  |
| --- | --- |
| Attribute | Description |
| Strategy/@legsAreSeverable | Boolean. If true, then an individual leg may be canceled or replaced. Otherwise, every leg of the order must be canceled, or every leg must be resent when only one is replaced. Applicable when multileg orders are delivered via several NewOrderSingle(35=D) messages. |

* **Validation of Leg Parameter Values**

The validation rules supported in FIXatdl 1.1 allow references to parameter values in the evaluation of its Boolean expression. Now with the support for multileg orders, we need a way to use values of leg parameters in our validation rules. So in FIXatdl 1.2 we include an additional attribute to the Edit element to indicate a leg number.

|  |  |
| --- | --- |
| Attribute | Description |
| Edit/@legNo | Indicates the leg in which the value of the field is to be retrieved in the evaluation of the Boolean expression. |

The following example shows a rule for validating a Pairs trade, confirming, without knowing the sequence of the legs, that one leg is a Buy and the other is a Sell.

<val:StrategyEdit errorMessage="One leg must be a BUY, the other a SELL">

<val:Edit logicOperator="OR">

<val:Edit logicOperator="AND">

<val:Edit field="FIX\_Side" legNo="1" operator="EQ" field2="1"/>

<val:Edit logicOperator="OR">

<val:Edit field="FIX\_Side" legNo="2" operator="EQ" field2="2"/>

<val:Edit field="FIX\_Side" legNo="2" operator="EQ" field2="5"/>

</val:Edit>

</val:Edit>

<val:Edit logicOperator="AND">

<val:Edit field="FIX\_Side" legNo="2" operator="EQ" field2="1"/>

<val:Edit logicOperator="OR">

<val:Edit field="FIX\_Side" legNo="1" operator="EQ" field2="2"/>

<val:Edit field="FIX\_Side" legNo="1" operator="EQ" field2="5"/>

</val:Edit>

</val:Edit>

</val:Edit>

</val:StrategyEdit>

* **Display/Layout of Leg Parameters**

For single-leg order definitions there are certain standard fields that should not be included in the Parameter or Controls declarations. These include: Symbol(55), Side(54), OrderQty(38), OrdType(40), Price(44) and StopPrice(99). OMSs tend to handle these separately from strategy parameters and display them regardless of whether they are declared in the FIXatdl code, or rather, if they are declared in FIXatdl, they are somehow ignored, or some special processing is involved.

Extending this model to multileg orders, there are certain standard fields that should not be included in the leg definitions, yet we can assume they will be presented to the user. If an order requires N legs, then these standard fields will be presented in all N legs. So, in effect, if a single order entry screen is segregated into a standard section and an custom parameter section, then a multileg order entry screen is segregated into N+1 sections: a global custom parameter section and N leg sections where each leg section contains a standard section and a custom parameter section.

We now want to declare a Panel to hold all our leg-level controls. We want to declare it just once with the expectation that it will be repeated as many times as necessary according to the value of <Strategy>@requiredNumberOfLegs. We’ll introduce a new element, <LegPanel>, for this purpose.

Example:

<StrategyLayout>

<StrategyPanel collapsible="false" orientation="VERTICAL">

<Control ID="c\_OrdParam1" label="Ord Param A" parameterRef="p\_OrdParamA" xsi:type="SingleSpinner\_t"/>

<Control ID="c\_OrdParam2" label="Ord Param B" parameterRef="p\_OrdParamB" xsi:type="SingleSpinner\_t"/>

</StrategyPanel>

**<LegPanel** collapsible="false" orientation="VERTICAL">

<Control ID="c\_LegParamA" label="Leg Param A" parameterRef="p\_LegParamA" xsi:type="SingleSpinner\_t"/>

<Control ID="c\_LegParamB" label="Leg Param B" parameterRef="p\_LegParamB" xsi:type="SingleSpinner\_t">

</Control>

**</LegPanel>**

</StrategyLayout>

* **GUI State Rule for Leg Panel Controls**

GUI Controls contained within a leg panel can have their states and values change just like Controls found in regular strategy panel. However, controls that are referenced by a state rule of another control are assumed to be in the same scope as the referring control. For example, in the following listing the second GUI control is disabled and given a null value if the first control (checkbox) is checked.

<lay:LegPanel collapsible="false" orientation="VERTICAL">

<lay:Control ID="c\_LegParamA" label="Leg Param A" parameterRef="p\_LegParamA" xsi:type="lay:Checkbox\_t"/>

<lay:Control ID="c\_LegParamB" label="Leg Param B" parameterRef="p\_LegParamB" xsi:type="lay:SingleSpinner\_t">

<StateRule enabled="false" value="{NULL}">

<Edit field="c\_LegParamA" operator="EQ" value="True"/>

</StateRule>

</lay:Control>

</lay:LegPanel>

If there are several legs then this state rule will be enforced in each leg panel. The behavior of the state rules in each leg panel is independent of the others.

* **Vendor Configurations**

Different GUI layouts can be defined based on set types of vendor configurations or service levels. For an implementation of this feature, an XML element defined at the Strategy level is used. It allows filtering of strategies to be performed much in the same way as the Regions element does in FIXatdl 1.1. (As of now the only configuration level we have identified is whether an OMS allows leg parameters. Some do not.)

The following two examples shows the description of a two-legged order strategy. In the first, it is expected that the vendor’s system supports leg parameters, i.e. leg parameters may take different values from leg to leg. In the second, it is expected that the vendor’s system does not support leg parameters having different values from leg to leg; effectively disallowing their use. Populating the message with the necessary information requires that all leg parameters are repeated with the same value in each leg.

<Strategy

...

minLegs="2"

maxLegs="2"

...

>

<VendorConfig **legParameters="true"**/>

<DeliveryMethods>

<FixMsg msgType="NewOrderSingle"/>

<FixMsg msgType="NewOrderMultiLeg"/>

</DeliveryMethods>

<Parameter name="p\_OrdParamA" xsi:type="Int\_t" fixTag="5000"/>

<Parameter name="p\_OrdParamB" xsi:type="Int\_t" fixTag="5001"/>

<Leg>

<Parameter name="p\_LegParamA" xsi:type="Int\_t" fixTag="6001"/>

<Parameter name="p\_LegParamB" xsi:type="Int\_t" fixTag="6002"/>

</Leg>

<lay:StrategyLayout>

<lay:StrategyPanel collapsible="false" orientation="VERTICAL">

<lay:Control ID="c\_OrdParam1" label="Ord Param A" parameterRef="p\_OrdParamA" xsi:type="lay:SingleSpinner\_t"/>

<lay:Control ID="c\_OrdParam2" label="Ord Param B" parameterRef="p\_OrdParamB" xsi:type="lay:SingleSpinner\_t"/>

</lay:StrategyPanel>

<lay:LegPanel collapsible="false" orientation="VERTICAL">

<lay:Control ID="c\_LegParamA" label="Leg Param A" parameterRef="p\_LegParamA" xsi:type="lay:SingleSpinner\_t"/>

<lay:Control ID="c\_LegParamB" label="Leg Param B" parameterRef="p\_LegParamB" xsi:type="lay:SingleSpinner\_t"/>

</lay:LegPanel>

</lay:StrategyLayout>

</Strategy>

<Strategy>

...

minLegs="2"

maxLegs="2"

...

>

<VendorConfig **legParameters="false"**/>

<DeliveryMethods>

<FixMsg msgType="NewOrderSingle"/>

</DeliveryMethods>

<Parameter name="p\_Ord\_ParamA" xsi:type="Int\_t" fixTag="5000"/>

<Parameter name="p\_Ord\_ParamB" xsi:type="Int\_t" fixTag="5001"/>

<Leg>

<Parameter name="p\_Buy\_Leg\_ParamA" xsi:type="Int\_t" fixTag="6001"/>

<Parameter name="p\_Buy\_Leg\_ParamB" xsi:type="Int\_t" fixTag="6002"/>

<Parameter name="p\_Sell\_Leg\_ParamA" xsi:type="Int\_t" fixTag="6003"/>

<Parameter name="p\_Sell\_Leg\_ParamB" xsi:type="Int\_t" fixTag="6004"/>

</Leg>

<lay:StrategyLayout>

<lay:StrategyPanel collapsible="false" orientation="VERTICAL">

<lay:Control ID="c\_OrdParamA" label="Ord Param A" parameterRef="p\_OrdParamA" xsi:type="lay:SingleSpinner\_t"/>

<lay:Control ID="c\_OrdParamB" label="Ord Param B" parameterRef="p\_OrdParamB" xsi:type="lay:SingleSpinner\_t"/>

<lay:StrategyPanel orientation="HORIZONTAL">

<lay:StrategyPanel orientation="VERTICAL">

<lay:Control ID="c\_BuyLegParamA" label="Buy Leg Param A" parameterRef="p\_Buy\_Leg\_ParamA" xsi:type="lay:SingleSpinner\_t"/>

<lay:Control ID="c\_BuyLegParamB" label="Buy Leg Param B" parameterRef="p\_Buy\_Leg\_ParamB" xsi:type="lay:SingleSpinner\_t"/>

</lay:StrategyPanel>

<lay:StrategyPanel orientation="VERTICAL">

<lay:Control ID="c\_SellLegParamA" label="Sell Leg Param A" parameterRef="p\_Sell\_Leg\_ParamA" xsi:type="lay:SingleSpinner\_t"/>

<lay:Control ID="c\_SellLegParamB" label="Sell Leg Param B" parameterRef="p\_Sell\_Leg\_ParamB" xsi:type="lay:SingleSpinner\_t"/>

</lay:StrategyPanel>

</lay:StrategyPanel>

</lay:StrategyPanel>

</lay:StrategyLayout>

</Strategy>

Note that in the latter example we drop support for delivery by the NewOrderMultiLeg(35=AB) message. Our assumption being that the OMS ‘s lack of support for leg parameters is due to the way it handles its collection of NewOrderSingle(35=D) messages. It is reasonable to expect that an OMS which supports delivery by NewOrderMultiLeg(35=AB) message will also be able to support leg parameters.

### Additional Global Definitions

FIXatdl 1.2 supports the global definition of <Parameter>, <Control>, <StrategyPanel>, <Edit>, <StateRule> and <Filter> elements. This allows them to be defined once and referenced within multiple <Strategy> elements, thus making the XML less verbose and more readable.

### OMS Hooks

One of the key features requested for FIXatdl 1.2 is the ability to refer to OMS variables in the description of order interfaces. In effect a FIXatdl instance would have a “hook” into the OMS and have access to certain environment variables or order parameters not defined in the XML for use in validation rules or filtering.

**Validation Rules with References to Standard FIX Fields**

In the FIXatdl 1.1 specification, several references are made to “Standard” FIX fields. For example, in the specification of the <Edit> element, which is used to build validation or flow rules, we have the following:

|  |
| --- |
| The “field” attribute of an Edit element is not restricted to strategy parameters. Standard order tags (those not described in a FIXatdl instance but nevertheless are required tags of order, cancel and cancel/replace messages) may also be used to create Boolean expressions. For example:  <StrategyEdit errorMessage="For IOC orders Participation Rate must be between 1 and 25">  <Edit logicOperator=”OR”>  <Edit field=”FIX\_TimeInForce” operator=”NX”/>  <Edit field="FIX\_TimeInForce" operator="NE" value="3"/>  <Edit logicOperator=”AND”>  <Edit field="ParticipationRate" operator="GE" value="1"/>  <Edit field="ParticipationRate" operator="LE" value="25"/>  </Edit>  </Edit>  </StrategyEdit> |

Also, in the section of the specification entitled “Dependencies and Structural Constraints beyond XML Schema” we have the following:

|  |
| --- |
| Within an edit element the attributes field and field2 must refer to either a pre-declared parameter name or a standard FIX tag name (taken from the FIX specification) pre-pended with the string ”FIX\_”. |

The intention was that the OMS would make the standard FIX tags accessible and allow them to be referenced by the validation rules. For example, in single-leg order definitions it is generally understood that Standard fields (Symbol(55), Side(54), OrderQty(38), OrdType(40), Price(44), etc.) should not be included in the Parameter or Controls declarations. OMS platforms tend to handle these separately from strategy parameters and display them regardless of whether they are declared in the FIXatdl code (or if they are declared in FIXatdl, they are somehow ignored, or some special processing is involved.) So, as a convention, FIXtdl 1.1 allowed these fields to be referred to from within a validation rule.

This can work for a small number of fields; and in fact, some OMS platforms support this. But not all do, and the fields they make accessible are not consistent.

FIXatdl 1.2 seeks to provide greater clarity concerning the use of standard fields and the fields of an order that we would expect an OMS to make available. The following table lists the fields of an order which may be referenced in a validation rule.

|  |  |  |
| --- | --- | --- |
| Field | Tag | Comments |
| Symbol | 55 |  |
| Side | 54 |  |
| OrderQty | 38 |  |
| OrdType | 40 |  |
| Price | 44 |  |
| StopPx | 99 |  |
| TimeInForce | 59 |  |
| HandlInst | 21 |  |
| ExecInst | 18 | This is a MultipleStringValue type, so we may want to introduce a “contains” operator in <Edit>. |
| SecurityType | 167 |  |
| TargetSubID | 57 |  |

**Filtering according to OMS Environment Values**

The FIXatdl 1.1 specification also speaks of filtering of strategies based on regions, asset classes and markets. For example, in the specification the description of the <Regions> element includes the following:

|  |
| --- |
| This element defines the globally based regions to which the strategy is applicable. It serves as a container of Region elements. To define a set of regions for a strategy use one or more Region elements. Region elements contain the attribute “inclusion” that determines whether the region is included from the set or excluded.  If no Regions element is defined, then the strategy is applicable for \*ALL\* regions. |

Filtering, however, is restricted to strategies. Other elements of a strategy, such as parameters or controls, cannot be filtered. For example, the following strategy applies only to a specific combination of region, market, client and security type.

<Strategy name="Tazer1" uiRep="Tazer" wireValue="Tazer" providerID="ABC">

<!-- US only -->

<Regions>

<Region name="TheAmericas" inclusion="Include">

<Country CountryCode="US" inclusion="Include"/>

</Region>

</Regions>

<!-- Nasdaq only -->

<Markets>

<Market MICCode="XNAS" inclusion="Include"/>

</Markets>

<!-- Equities ("CS") only -->

<SecurityTypes>

<SecurityType name="CS" inclusion="Include"/>

</SecurityTypes>

...

</Strategy>

In FIXatdl 1.2 we propose to expand the filtering capability so we can apply it to parameters and controls. Our motivation is to allow strategies with a set of parameters that are applicable only to certain regions, markets or clients, to be defined in one <Strategy> element rather than several.

The following FIXatdl describes a strategy with a parameter that is filtered by a region and an enumerated value of another parameter filtered by the same region. Note that it makes use of a global filter definition.

<Filter **id=**"**US-filter**">

<Regions>

<Region name="TheAmericas" inclusion="Include">

<Country CountryCode="US" inclusion="Include"/>

</Region>

</Regions>

</Filter>

<Strategy>

...

<Parameter name="StartTime" xsi:type="UTCTimestamp\_t" fixTag="7602"/>

<Parameter name="EndTime" xsi:type="UTCTimestamp\_t" fixTag="7603"/>

<Parameter name="Text" xsi:type="String\_t" fixTag="9999" use="optional"/>

<Parameter name="Variance" xsi:type="Float\_t" fixTag="7641" **filter="US-filter"**/>

<Parameter name="Benchmark” xsi:type="String\_t" fixTag="7666">

<EnumPair wireValue="Arrival" enumID="e\_Arrival"/>

<EnumPair wireValue="Close" enumID="e\_Close"/>

<EnumPair wireValue="Open" enumID="e\_Open"/>

<EnumPair wireValue="SectorETF" enumID="e\_SectorETF" **filter=”US-filter”**/>

<StrategyLayout>

...

<StrategyPanel orientation="HORIZONTAL">

<Control xsi:type="TextField\_t" ID="Variance" parameterRef="Variance" **filter="US-filter"**>

</Control>

<Control xsi:type="DropDownList\_t" parameterRef="Benchmark">

<ListItem uiRep="Arrival" enumID="e\_Arrival"/>

<ListItem uiRep="Close" enumID="e\_Close"/>

<ListItem uiRep="Open" enumID="e\_Open"/>

<ListItem uiRep="SectorETF" enumID="e\_SectorETF" **filter=”US-filter”**/>

</Control>

</lay:StrategyPanel>

...

</Strategy>

FIXatdl 1.2 also expands the filtering criteria to include types of client. More specifically, a list of client types can be defined in a <ClientGroups> element. For example, a broker-dealer may classify a set of its clients as high frequency traders. A client-group filter could be declared as follows:

<Filter id="US-HFT">

<ClientGroups>

<ClientGroup ID="HFT"/>

</ClientGroups>

</Filter>

## Samples

Several samples of strategies described in FIXatdl 1.2 will be included in the release distribution.

# Issues and Discussion Points

## Questions to be decided beyond RC1

# References

Authors should list references used in created the technical standard proposal

* Reference – reference used to create the standard or related to the proposed technical standard.
* Version – version of reference
* Relevance – Relevance of specification to standard.
* Relationship – relationship of the related standard to the technical standard being proposed. Can be: **Extends** the related standard, **Overlaps** with related standard, **Incorporates** related standard, **Inspiration** fromrelated standard, **Uses** related standard, **Replaces** related standard.
* Normative – Yes – this reference contains provisions incorporated into this specification.

|  |  |  |  |
| --- | --- | --- | --- |
| Reference | Version | Relevance | Normative |
| None |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Relevant and Related Standards

|  |  |  |
| --- | --- | --- |
| Standard | Version | Reference Location |
| W3C XML Schema Definition Language (XSD) | 1.1 | <https://www.w3.org/TR/xmlschema11-1/> |

# Intellectual Property Disclosure

Authors should provide a list of any intellectual property

* Related Standard – name of related standard (can be an acronym if widely known).
* Version – version of related standard being referenced
* Reference location – URL or document publication information
* Relationship – relationship of the related standard to the technical standard being proposed. Can be: **Extends** the related standard, **Overlaps** with related standard, **Incorporates** related standard, **Inspiration** fromrelated standard, **Uses** related standard, **Replaces** related standard.
* Normative – Yes – this reference contains provisions incorporated into this specification.

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

# Definitions

|  |  |
| --- | --- |
| Term | Definition |
|  |  |
|  |  |
|  |  |
|  |  |

# Project Milestone

## Release Candidate 1 Deliverables

These artifacts will be delivered as Release Candidate 1 and will be displayed in the Tech/Specs section of the FIX Trading Community website (https://www.fixtrading.org/standards/fixatdl/) as well as in the GitHub project [FIXTradingCommunity/fix-](https://github.com/FIXTradingCommunity/fix-orchestra-spec)atdl:

* The complete set of FIXatdl 1.2 XML Schema files
* A set of sample files each containing a FIXatdl-coded strategy which highlights a specific feature

## Roadmap