FIX TRADING COMMUNITY™
INDUSTRY-DRIVEN • INDEPENDENT • NEUTRAL

FIX Algorithmic Trading Definition Language (FIXatdl®) Technical Specification

Version 1.2 – Release Candidate 1 – May 20, 2021

THIS DOCUMENT IS A RELEASE CANDIDATE FOR A PROPOSED FIX TECHNICAL STANDARD. A RELEASE CANDIDATE HAS BEEN APPROVED BY THE GLOBAL TECHNICAL COMMITTEE AS AN INITIAL STEP IN CREATING A NEW FIX TECHNICAL STANDARD. POTENTIAL ADOPTERS ARE STRONGLY ENCOURAGED TO BEGIN WORKING WITH THE RELEASE CANDIDATE AND TO PROVIDE FEEDBACK TO THE GLOBAL TECHNICAL COMMITTEE AND THE WORKING GROUP THAT SUBMITTED THE PROPOSAL. THE FEEDBACK TO THE RELEASE CANDIDATE WILL DETERMINE IF ANOTHER REVISION AND RELEASE CANDIDATE IS NECESSARY OR IF THE RELEASE CANDIDATE CAN BE PROMOTED TO BECOME A FIX TECHNICAL STANDARD DRAFT.

Table of Contents

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), except as expressly set out in FIX Protocol Limited's Copyright and Acceptable Use Policy.

1 Introduction

This document serves as a specification of the FIX Algorithmic Trading Definition Language (FIXatdl®), a markup language that works in conjunction with the FIX Protocol. FIXatdl® is used to define the FIX interface of algorithmic order types. An algorithmic order interface description consists of: a description of tags that are to be included in FIX NewOrderSingle(35=D), OrderCancelRequest(35=F), and OrderCancelReplaceRequest(35=G) messages that are to be sent to an order recipient; rules for validating the data entered into an order form by a user; suggestions on how to render GUI controls within an order entry form; and rules affecting the visual state of the GUI controls as information is being entered into the order form.

Rather than describing interfaces in a natural language, such as English, which can be subject to differing interpretations, FIXatdl® standardizes the way algorithmic interfaces are described thus reducing interpretation errors and allowing for the creation of documents in a machine-readable format. It is envisioned that applications supporting this standard would be able to receive an XML document conforming to FIXatdl® and, based on the information within this document, be able to:

- Dynamically display an order ticket containing algorithmic order parameters.
- Change the visual state of GUI controls based on user input.
- Validate the values entered into the ticket before an order is transmitted.
- Create and transmit a FIX order message with the appropriate standard and/or user-defined fields (UDFs) populated.

These capabilities are achievable without the need for custom software development or subsequent product deployment.

1.1 Audience

This specification is intended for those interested in either: (1) developing applications with FIX order entry capabilities supporting order type definition via FIXatdl®; or (2) algorithmic order providers who wish to describe the interface to their algorithms in FIXatdl®.

2 FIXatdl® Schema Files

A set of XML Schema files has been created to describe the structure of a FIXatdl® document instance. These files can be used with commercial XML parsing software to validate a FIXatdl® document instance. They can also be used with XML data binding utilities to generate source code which maps classes to XML representations. The files are grouped into two functional categories:

- **Data Contract** – Defines the wire-value interface of an algorithmic order. For each algorithm/strategy it defines the valid set of parameters and availability of the strategy for specific markets. For each parameter of an algorithm/strategy it defines the type; the legal range of values (including minimum and maximum values); whether it is optional or required; and value constraints based on certain conditions or the value of other parameters (validation rules).
- **GUI** – Defines the recommended GUI controls that should be rendered on the order entry screen and their location on the screen. Defines the rules that affect the state of a GUI control. Provides a mapping of the on screen controls with the parameters of the data contract.

The constructs of the schema files have been categorized this way to ensure that the data contract is de-coupled from the GUI. This provides some flexibility for E/OMS vendors in how FIXatdl® is applied. For example, data contract functions, such as parameter validation, may be performed in an application downstream from the E/OMS without the need for the XML that describes the GUI.

The FIXatdl® language definition is contained within six XML Schema files:

XML Schema file / namespace	Purpose
fixatdl-core-1-2.xsd http://www.fixprotocol.org/FIXatdl-1-2/Core	Data: Defines attributes and elements that are used to describe the data content of the algorithm and the parameters.
fixatdl-validation-1-2.xsd http://www.fixprotocol.org/FIXatdl-1-2/Validation	Data: Defines attributes and elements used to author rules that are applied to the parameter values as a validation check. These rules can be simple where boundary conditions are checked, or complex where compound boolean expressions involving several parameters are evaluated.
fixatdl-layout-1-2.xsd http://www.fixprotocol.org/FIXatdl-1-2/Layout	GUI: XML constructs to describe how a parameter should be rendered within a user interface – this includes recommendations about GUI controls and their relative location within the interface.
fixatdl-flow-1-2.xsd http://www.fixprotocol.org/FIXatdl-1-2/Flow	GUI: Provides the ability to dynamically affect the behavior of a GUI control. Rules can be created to enable or disable parameters based on values entered by the user in other parameters.
fixatdl-regions-1-2.xsd http://www.fixprotocol.org/FIXatdl-1-2/Regions	Data: Enumeration values for countries within three regions: TheAmericas, EuropeMiddleEastAfrica and AsiaPacificJapan.
fixatdl-timezones-1-2.xsd http://www.fixprotocol.org/FIXatdl-1-2/Timezones	Data: Lists enumeration values for world timezones based on zoneinfo database.

3 Key Concepts

3.1 Element Hierarchy

The FIXatdl® schema provides a set of XML elements that are used to create a conforming FIXatdl® document. These elements are described later in this section. The attributes of each of these elements are described in latter in this document.

In a FIXatdl® document an algorithm provider can define any number of algorithmic order interfaces by using multiple `Strategy` elements. Each strategy is identified by a unique name that must be provided in the XML of each of the `Strategy` elements. Instances of documents begin with the root element `Strategies` and follow the hierarchy:

```
<Strategies>
    <Strategy>
        ... strategy definition ...
    </Strategy>
    <Strategy>
        ... strategy definition ...
    </Strategy>
        ...
    <Strategy>
        ... strategy definition ...
    </Strategy>
</Strategies>
```

At the root level, the algorithm provider must specify which tag to use to identify the individual strategies. (At one time TargetStrategy(847) was intended to carry this information. However, most providers use a user-defined field for this purpose.) For example to indicate that tag 25009 will be used to identify strategies the `Strategies` element would be written as

```
<Strategies strategyIdentifierTag="25009"/>
```

Parameters for each strategy are defined via `Parameter` elements. Validation rules are defined via `StrategyEdit` elements. Each strategy can have any number of parameters or validation rules. An algorithm can have only one section where the layout of the controls is defined. A layout is defined via the `StrategyLayout` element. So when looking deeper into the strategy definition, one can see that it follows the hierarchy:

```
<Strategy>
    <Parameter>
    <Parameter>
    ...
    <Parameter>
    <StrategyEdit>
    <StrategyEdit>
    ...
    <StrategyEdit>
    <StrategyLayout>
</Strategy>
```

The following figure shows the hierarchy of elements in tree form starting from the root element, `Strategies`. The XML Schema values minOccurs and maxOccurs are given for each branch of the tree. Elements with optional or required child elements are indicated by double-line borders. Elements with no children (leaf nodes) have single-line borders. Abstract elements, ones which require the use of a substitution group, are shaded. The elements `Parameter`, `StrategyLayout`, and `StrategyEdit` are somewhat complex; the hierarchy of their children is shown in figures 2 through 4.

Note that the figures that follow are intended to give a generalized structure of the schema and do not necessarily show all the child elements. Please refer to the FIXatdl® XML Schema files for a complete list and definition of the FIXatdl® elements.

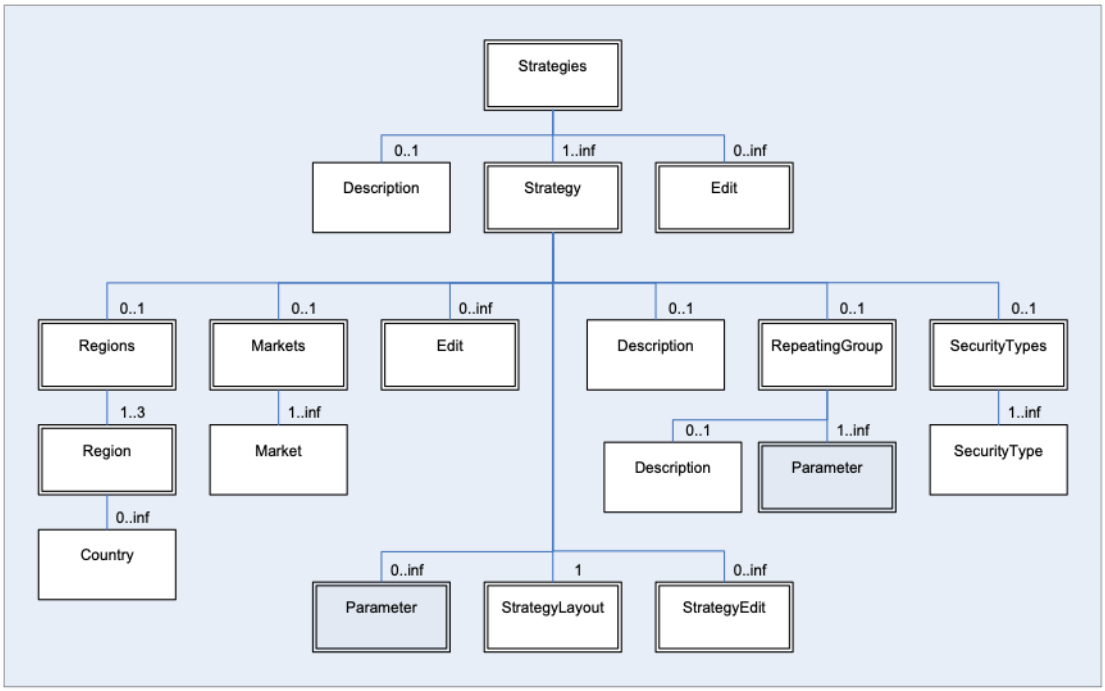

Figure 1 — Root Element Hierarchy

The following figure gives the hierarchy of elements descending from the `Parameter` element.

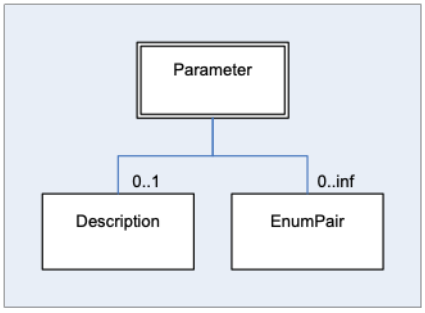

Figure 2 — Parameter Hierarchy

The following figure gives the hierarchy of elements descending from the `StrategyLayout` element. This element is responsible for binding GUI controls to parameters and describing their arrangement on the order-entry screen.

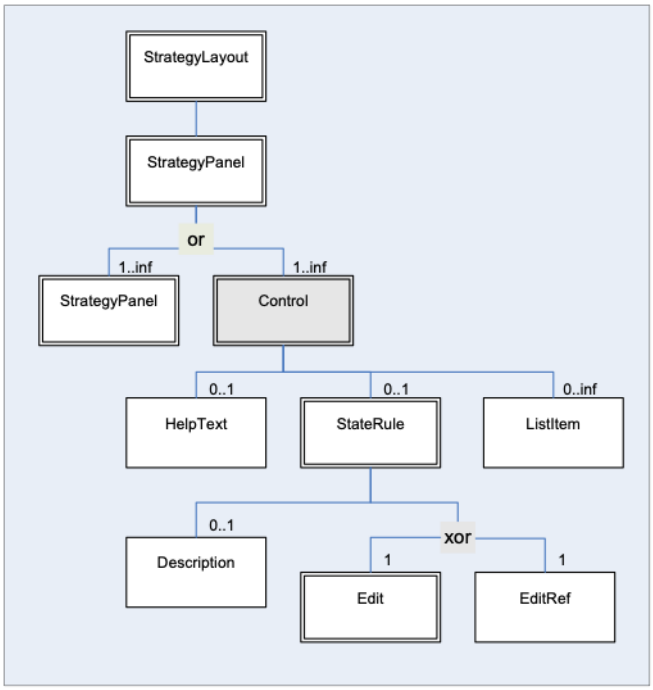

Figure 3 — StrategyLayout Hierarchy

The following figure shows the `StrategyEdit` hierarchy. This element is used to describe validation rules which are applied to the values of a strategy's parameters at order-generation time. Each `StrategyEdit` element must contain a single `Edit` element (may contain further nested `Edit` rules) which is used to describe a particular condition that must be met in order to pass validation.

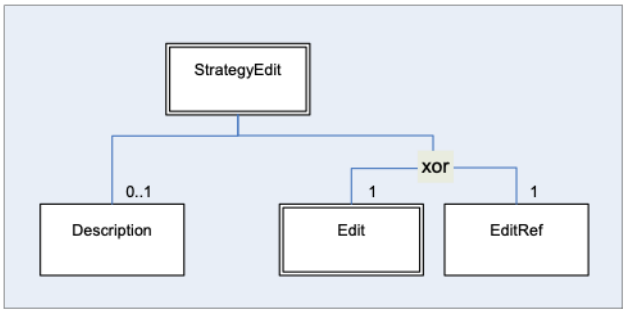

Figure 4 — StrategyEdit Hierarchy

3.2 Parameter Description

The interface of an algorithmic order type is described by a set of FIX messages, the required, optional and user-defined fields of those messages, and user-defined restrictions on the range of values for particular fields. In general, when speaking of the parameters of an algorithmic order one is, in fact, referring to the user-defined fields of a NewOrderSingle(35=D), OrderCancelRequest(35=F), and OrderCancelReplaceRequest(35=G) message. (In some cases a parameter may also refer to a standard FIX field, one with a tag number in the range 1-5000, that broker-dealers commonly included in their algorithmic interface specifications, such as EffectiveTime(168) and ExpireTime(128).)

Parameters are strictly described in FIXatdl® by the target firm who will receive them (*order recipients*), and are communicated via an XML file to various senders (*order initiators*). To describe these parameters, FIXatdl® provides the `Parameter` element. `Parameter` elements are identified by their "name" attribute. There is no limit to the number of parameters a strategy may have as long as each is uniquely identified at the strategy level. Besides a parameter's name, other parameter attributes include: its type; its maximum and minimum values (if applicable); its corresponding FIX tag number; and its usage (optional vs. required). For example, the following code snippet describes an integer type parameter:


```
<Parameter name="SampleRate" xsi:type="Int_t" fixTag="28000"
use="optional" minValue="1" maxValue="9"/>
```

This listing describes a parameter named “SampleRate” which can optionally be populated in tag 28000 of an order message. The attributes “minValue” and “maxValue” describe the minimum and maximum values that the recipient of an order message is expecting. Orders with “SampleRate” values outside that range may be rejected. The attribute “xsi:type” describes the parameter’s type which must be one of the datatypes specified by the FIX Protocol. FIXatdl® provides enumeration values for xsi:type that map directly to the FIX datatypes. (An explanation of xsi:type can be found in this document in the section Abstract Element Extensions.)

For certain parameters it may be appropriate to limit the legal values to a set of enumerated values. This is done by adding child `EnumPair` elements to the `Parameter` element. Each `EnumPair` represents one of the enumerated values expected to be transmitted over the wire. For example:

```
<Parameter name="Aggression" xsi:type="Char_t" fixTag="28001" use="required">
    <EnumPair enumID="low" wireValue="L"/>
    <EnumPair enumID="medium" wireValue="M"/>
    <EnumPair enumID="high" wireValue="H"/>
</Parameter>
```

This describes the “Aggression” parameter. An order recipient would expect to receive one of the values, “L”, “M” or “H” in tag 8001 of an order message. The attribute `EnumPair/@enumID` is a unique identifier of `EnumPair` elements.

If a user of an order-entry system were to submit an order with “SampleRate” set to 5 and “Aggression” set to “high”, the order recipient would expect to receive a FIX message containing a substring similar to:

```
...35=D|11=0001|55=AXP|44=77.25| ... 28000=5|28001=H ...
```

3.3 Validation Rules

Validation rules are defined by use of the `StrategyEdit` element. This XML element enables the creation of complex and conditional rules which can be applied to the orders generated by an E/OMS. The goal of a validation rule is to process the

values of the strategy parameters after they have been entered by the user. Each validation rule consists of a condition and an error message. If the condition is true then the values of the parameters are valid. If the condition is false, then the values of the parameters are invalid and the provided error message should be displayed. That is to say, validation conditions are much like assertions. When an assertion has failed an error has occurred.

The conditions described within a validation rule are defined by use of the `Edit` element. An `Edit` element defines a Boolean expression where values of parameters can be compared to one another or to constant values.

To illustrate, consider the most common parameters of all algorithms, `StartTime` and `EndTime`. Their description and a rule guaranteeing that `StartTime` precedes `EndTime` can be described by the following statements:

```
<Parameter name="StartTime" xsi:type="UTCTimestamp_t" fixTag="28005" use="required">
<Parameter name="EndTime" xsi:type="UTCTimestamp_t" fixTag="28006" use="required">
<StrategyEdit errorMessage="Start Time must precede End Time.">
    <Edit field="StartTime" operator="LT" field2="EndTime"/>
</StrategyEdit>
```

Here both `StartTime` and `EndTime` are defined as UTCTimestamp parameters. At validation time, the rule described in the `StrategyEdit` element instructs the E/OMS to perform an evaluation of the Boolean expression provided by the `Edit` element. In this case a comparison of `StartTime` and `EndTime` will be made using the “LT” (less than) operator. If `StartTime` is less than `EndTime` then the parameter values are deemed to be valid. However, if `StartTime` is greater than or equal to `EndTime` then the parameter values are invalid and the E/OMS can inform the user by displaying the error message in a dialog box.

For more complex rules, Boolean expression may be formed by multiple `Edit` elements organized in an expression tree using logical operators AND, OR, XOR and NOT. For example consider these declarations:

```
<Parameter name="ParticipationRate" xsi:type="Float_t" fixTag="28008" use="optional"/>
<StrategyEdit errorMessage="If Participation Rate is entered it must be between 1 and 50">
```



```
    <Edit logicOperator="OR">
        <Edit field="ParticipationRate" operator="NX"/>
        <Edit logicOperator="AND">
            <Edit field="ParticipationRate" operator="GE" value="1"/>
            <Edit field="ParticipationRate" operator="LE" value="50"/>
        </Edit>
    </Edit>
</StrategyEdit>
```

This is a tree of `Edit` elements. The root `Edit` element is describing a logical "OR" condition asserting that either "ParticipationRate" was not provided or its value is in the range from 1 to 50. Note how in the "AND" expression a parameter value is compared not to another parameter but to a constant value.

Also note that the logical operators, AND and OR, can have more than two operands. Furthermore, they both perform short-circuit evaluation of their operands. (I.e. their operands are evaluated from left to right. As soon as the value is known, evaluation of the expression stops and the value is returned. Consequently, not all operands need to be evaluated. For example, consider the previous example in which "ParticipationRate" is an optional parameter. It is quite possible that the user does not provide a value for "ParticipationRate". If that is the case then evaluation of the "OR" statement will terminate after it is established that its first operand, `<Edit field="ParticipationRate" operator="NX"/>`, is true. The "AND" statement that follows is never evaluated – which is a good result since, if one attempts to evaluate it, it is quite possible that a "Null Reference" error would occur.) That being the case, it is important that XML parsing or binding libraries maintain the order of the elements as they appear; otherwise unexpected results may occur.

The logical operator XOR can also have more than two operands. As a convention, XOR is defined as "one and only one", which means it evaluates to "true" when one and only one of its operands is true. If none or more than one of its operands is true then XOR is false. Short-circuit evaluation cannot be applied to XOR.

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>
```

This rule incorporates the value of TimeInForce(59) which is a standard tag found in most order messages. The values associated with standard tags are those that are sent over the wire. For example, TimeInForce(59) is an enumeration of char values ranging from "0" to "9" and "A" to "C" (FIX Latest as of EP266). So care must be taken to assure the corresponding operand, "value", is of a similar type. Support for these types of expressions is highly dependent on a vendor's implementation of FIXatdl®. Not all standard tags may be available.

In cases where the field attribute is not recognized or not supported, the rule containing the offending `Edit` element should be skipped over by a vendor's application and should not cause a validation error. The end-result will be the same as if the condition of the rule were true.

3.4 GUI Layout Description

In order to render a parameter within an order entry screen, an OMS must be able to pick an appropriate GUI control to display. For instance, a parameter representing a price would best be rendered as a number spinner control while a parameter representing a choice between limited numbers of values, such as "High", "Medium" and "Low", would best be rendered as a combo box.

Once the GUI controls have been selected, the OMS must appropriately arrange them on the screen. By using the elements and attributes of the Layout Schema, an algorithm provider can describe the GUI controls to use and describe how they should be arranged on the screen.

FIXatdl® does not attempt to dictate user-interface style or look-and-feel. It is designed to be platform neutral. The components that are provided are those typically found in .Net, Java and Web environments.

The layout schema allows GUI controls to be arranged by adding them to a container define by the `StrategyPanel` element. Controls within a panel may be arranged either vertically or horizontally. Panels themselves may be nested and arranged either vertically or horizontally as well. The attributes of the `StrategyPanel` element include

- **Title** – a string representing the panel title which may or may not be displayed
- **Collapsible** – a Boolean value indicating whether the panel can be collapsed.
- **Collapsed** – a Boolean value indicating the panel's initial state.
- **Orientation** – defines whether the panel's components should be vertically, horizontally or grid aligned.

An important aspect of the GUI description is that it is platform neutral. The algorithm provider describes GUI controls without knowing how an E/OMS has been implemented or knowledge of the widget toolkit that it uses. The controls provided by FIXatdl® are those typically found in .Net, Java or Web environments. (The initial intention was to adopt a standard such as XAML or XUL. However, it was believed that this would put an excessive constraint on the E/OMS vendors. So a conscious decision was made not to adopt any one of these languages. Instead FIXatdl® presents its own with the understanding that a vendor may extend or transform it to be aligned with their architecture and internal data structures.)

Most Controls are associated with a particular Parameter. This is done via the Control attribute, parameterRef. However some controls may not have an associated Parameter. These controls are typically defined in order to affect the state of other controls via the use of a `StateRule` element.

The following listing describes four parameters and the layout of their four associated controls. If we examine the code we'll notice that the controls are enclosed in two `StrategyPanel` elements, one entitled "Time Parameters" and the other entitled "Advanced". These two panels are nested horizontally into the top-level `StrategyPanel` element of the `StrategyLayout` element.

```
<Parameter name="StartTime" xsi:type="UTCTimestamp_t" fixTag="28005" use="required"/>
<Parameter name="EndTime" xsi:type="UTCTimestamp_t" fixTag="28006" use="required"/>
<Parameter name="ParticipationRate" xsi:type="Float_t" fixTag="28007" use="optional"/>
<Parameter name="Aggression" xsi:type="Char_t" fixTag="28001" use="required">
    <EnumPair enumID="e_low" wireValue="L"/>
    <EnumPair enumID="e_med" wireValue="M"/>
    <EnumPair enumID="e_high" wireValue="H"/>
</Parameter>
<StrategyLayout>
    <StrategyPanel orientation="HORIZONTAL">
        <StrategyPanel title="Time Parameters" orientation="VERTICAL">
            <Control ID="c_ST" xsi:type="Clock_t" label="Start Time" parameterRef="StartTime"/>
            <Control ID="c_ET" xsi:type="Clock_t" label="End Time" parameterRef="EndTime"/>
        </StrategyPanel>
        <StrategyPanel title="Advanced" orientation="VERTICAL">
            <Control ID="c_PR" xsi:type="SingleSpinner" label="Partic. Rate"
             parameterRef="ParticipationRate"/>
            <Control ID="c_A" xsi:type="DropDownList_t" label="Aggression"
             parameterRef="Aggression">
                <ListItem enumID="e_low" uiRep="Low"/>
                <ListItem enumID="e_med" uiRep="Medium"/>
                <ListItem enumID="e_high" uiRep="High"/>
            </Control>
        </StrategyPanel>
    </StrategyPanel>
</StrategyLayout>
```

Notice how the `Parameter/@name` attributes match with the `Control/@parameterRef` attributes. This creates the binding between parameters and controls. Also note how the `EnumPair/@EnumID` attributes match with the

`ListItem/@EnumID` attributes. This creates the binding between the enumeration values of the parameter and the items of a drop-down list.

If an application were to render this information on an order ticket it would have to decide which GUI controls to instantiate and find a way to insert them into panels and lay the panels out according to the instructions of the XML. Different platforms will have different controls and panels available for this purpose and the application built on these platforms will have different appearances. So, a rendering of the controls described in the previous listing may look similar to the following image:

Figure 5 — GUI Layout Example

3.4.1 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` element.

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 is how it might be rendered:

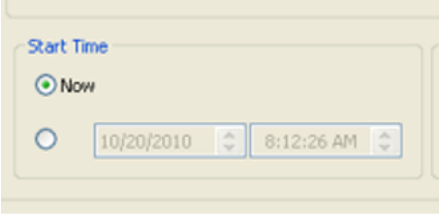

Figure 6 — Clock Control Example 1

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:

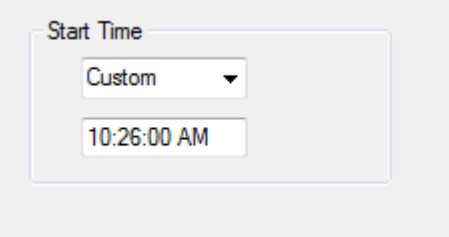

Figure 7 — Clock Control Example 2

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.

The following set of Clock control attributes allows 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, the attributes “enablingControlType”, “disablingControlType”, and “disablingControlText” are available for the Clock 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:

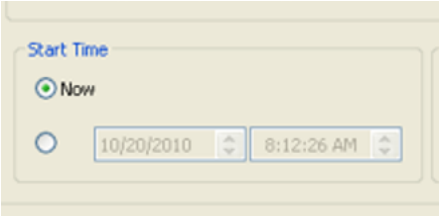

Figure 8 — Clock Control Example 1

If an explicit GUI control used to disable user input is not desired, then the "disablingControlType" 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:

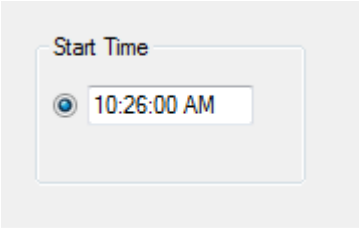

Figure 9 — Clock Control Example 3

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.

3.4.2 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:

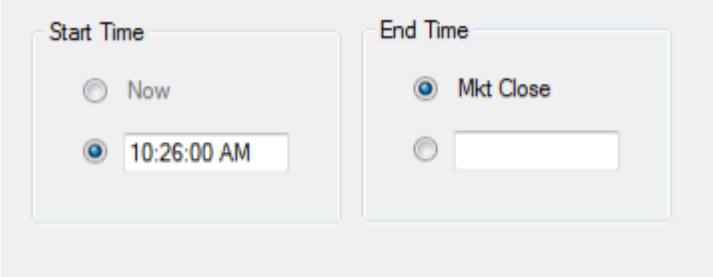

Figure 10 — Duration Example 1

A Duration control represents a time span rather than a point in time. Replacing the EndTime control in the previous example with a Duration control results in:

```
<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>
</StrategyPanel>
```

Rendering:

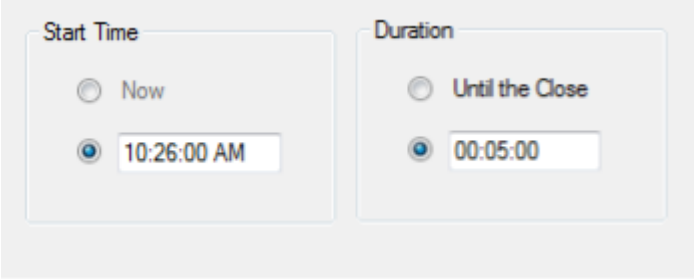

Figure 11 — Duration Example 2

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

```
<parameter name="Duration" xsi:type="UTCTimeOnly_t" fixTag="29003" 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.

3.4.3 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, a feature is available that allows controls to be arranged in grid. Specifically, the value of “GRID” is available for 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, `StrategyPanel` elements define their orientation by setting their orientation attribute to one of these values which can now include “GRID”. For example:

```
<StrategyPanel orientation="GRID">
```

This allows 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. The attribute “fillOrder” indicates 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 attributes “row”, “col”, rowSpan“, and”colSpan" may be specified in any `Control` or `StrategyPanel` elements which are child elements of a grid-oriented `StrategyPanel` element. The attributes “numRows”, “numCols”, and “fillOrder” may be specified in any grid-oriented `StrategyPanel` element.

In the following three code samples a panel is created 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, the `StrategyPanel` attributes `numRows` and `numCols` are left out. One can still determine how to render the controls based on the attributes “row” and “col” of each `Control` element.

```
<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, an implicit declaration of each item’s placement is supported 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:

Figure 12 — GUI Controls Example 1

Note that when switching 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:

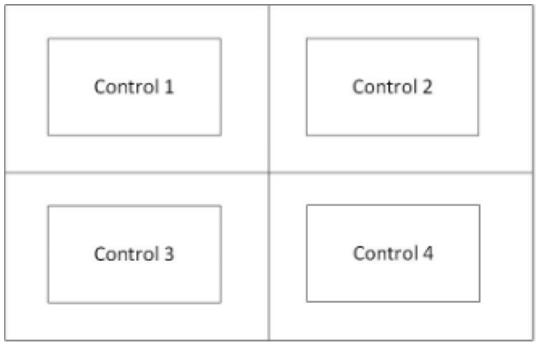

Figure 13 — GUI Controls Example 2

When it makes sense for a control (or panel) to span multiple columns or rows, the “colSpan” and “rowSpan” attributes can be used. 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:

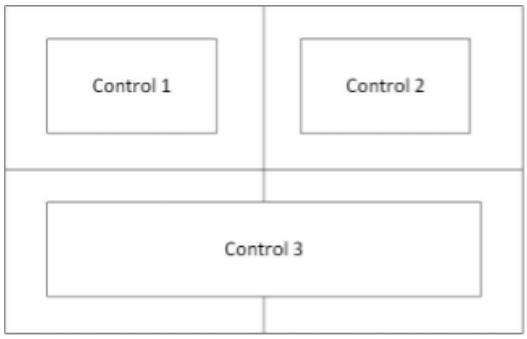

Figure 14 — GUI Controls Example 3

3.4.3.1 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/column conflicts.	Two or more items in a grid specify the same row and column values.	Ignore all “row”/“col” attributes of all grid items and render as if they had not been specified.
Row or column values are out-of-range.	The “numRows” and “numCols” attributes of a `StrategyPanel` element are defined as N and M, but a child control’s “row” attribute b is >= N or its “col” attribute is >= M.	Override the “numRows” or “numCols” attribute of the `StrategyPanel` element 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®.

3.5 Flow Control Rules

Interdependencies among standard FIX tags affecting their applicability are quite common. For example, Price(44) is not applicable when OrdType(40) is set to “Market”. The same can be said for algorithmic order types and their parameters. Many algorithmic order types will have parameters whose applicability is dependent on the value of one or more other parameters. These rules are often listed in algorithmic order specifications in the comments column of tables that describe the parameters of the algorithm.

In order to standardize the way these rules are described, a sub-schema is provided, which contains elements and attributes used to define rules that can be applied to the visual state of GUI Controls. This capability is a means to direct the user’s workflow and this is why it has been called “flow control”. When creating flow-control rules the expectations are that they are evaluated every time a Control’s value has changed. Based on the outcome of the

evaluations, certain GUI controls may become grayed-out or hidden as the user enters values into text fields or selects items from drop-down lists.

Flow-control rules can be described via the `StateRule` element. A `StateRule` element will consist of a Boolean expression and an action to take when the Boolean expression is true. There are three actions that are supported: (1) change the "enabled" state of a control to either "true" or "false"; (2) change the "visible" state of a control to either "true" or "false"; and (3) change the current value of the control to a supplied value. (Supplied values may be a constant string value, an enumID, or the special token {NULL}.)

As with validation rules, flow-control rules employ the `Edit` element to describe the condition (or Boolean expression). However, when an `Edit` element is used in a Flow-control rule, it will not make comparisons of parameter values; rather it will compare the values returned by the controls. For example, the attributes `Edit/@field` and `Edit/@field2` will refer to either control values or constant values.

Another difference between validation rules and flow-control rules is that the action of a flow-control rule is performed when the condition it describes is true. This differs from validation rules, where the action of "raising an error" occurs when the condition is false.

To illustrate the description of a Flow-control rule consider the following code snippet. (Note how the `Control/@ID` attribute value "c_AlphaMode" matches the `Edit/@field` attribute value "c_AlphaMode" and how the "enumID" attribute value "e_Custom" matches the "value" attribute value "e_Custom"):

```
<Parameter name="AlphaMode" xsi:type="Int_t" fixTag="28300" use="required">
    <EnumPair enumID="e_Annual" wireValue="1"/>
    <EnumPair enumID="e_Daily" wireValue="2"/>
    <EnumPair enumID="e_Custom" wireValue="3"/>
</Parameter>
<Parameter name="CustomValue" xsi:type="Float_t" fixTag="28301" use="optional"/>
<StrategyLayout>
    <StrategyPanel orientation="HORIZONTAL">
        <Control ID="c_AlphaMode" xsi:type="DropDownList" label="Alpha Benchmark"
                 parameterRef="AlphaMode">
            <ListItem enumID="e_Annual" uiRep="Annual"/>
            <ListItem enumID="e_Daily" uiRep="Daily"/>
            <ListItem enumID="e_Custom" uiRep="Custom"/>
        </Control>
        <Control ID="c_CustomValue" xsi:type="SingleSpinner_t" label="Custom Alpha"
                 parameterRef="CustomValue">
            <StateRule enabled="true">
                <Edit field="c_AlphaMode" operator="EQ" value="e_Custom"/>
            </StateRule>
            <StateRule value="{NULL}">
                <Edit field="c_AlphaMode" operator="NE" value="e_Custom"/>
            </StateRule>
        </Control>
    </StrategyPanel>
</StrategyLayout>
```

Two parameters are defined in this listing, "AlphaMode" and "CustomValue". Also two controls corresponding to the parameters are defined. A rule has been supplied to the control identified by "c_CustomValue" governing its visual behavior. The rule should be interpreted as: "The control c_CustomValue is enabled only when the value of control c_AlphaMode has been set to"Custom". So a user who selects"Annual" or "Daily" would not able to enter a custom Alpha value. Only when "Custom" is selected from the dropdown list would the custom Alpha control be able to accept values entered by the user.

While StateRules are explicit in defining the changes to a control when the condition, described by its `Edit` element, makes the transition from being false to being true, it is not clear what changes to make when the condition becomes false again (or is initially false). So, to clarify the behavior of the controls, the following conventions are applied:

1. A `StateRule` element that changes the "enabled" property of a control to X when its condition becomes true, will implicitly cause the "enabled" property of the control to change to NOT(X) when its condition becomes false, where X is Boolean. (The "enabled" property simply controls whether or not the value

within the control can be changed (is read-only) and is not a determining factor in whether or not the control's value is to be included in the message transmitted over the wire.)

2. A `StateRule` element that changes the "visible" property of a control to X when its condition becomes true, will implicitly cause the "visible" property of the control to change to NOT(X) when its condition becomes false, where X is Boolean.
3. A `StateRule` element that changes the value of a control when its condition becomes true will cause no action to take place when its condition becomes false. Provided the vale expressed in the `StateRule` element is not the special token "{NULL}".
4. A `StateRule` element that changes the value of a control to "{NULL}" when its condition becomes true will cause the control's value to revert back to its previous non-{NULL} value or its initial value.

Note that due to point 4 above, when a `StateRule` element condition becomes false it may cause the control to become un-initialized. When this occurs the control will have no value. Should a NewOrderSingle(35=D), OrderCancelRequest(35=F) or OrderCancelReplaceRequest(35=G) message be generated while the control is in this condition, the associated parameter will not be included in that message.

Also note that the state of a control's enabled property or visible property does not influence whether the control's associated parameter is sent on the wire or not. This behavior is governed entirely by the control's value. To clarify this, one must adhere to another convention:

5. To the extent that a control's value determines the "wire-value" of a particular parameter, if the control is un-initialized or has been set to the value of "{NULL}" then the associated parameter will not have a "wire-value" and will not have its tag-value pair included in a NewOrderSingle(35=D), OrderCancelRequest(35=F) or OrderCancelReplaceRequest(35=G) message.

In other words, if a user enters a value into a control and subsequently the control becomes disabled then the value that was entered would cause a tag to be populated in the generated FIX message and the value would go out over the wire. This is why, in the previous listing, a second `StateRule` element was required:

```
<StateRule value="{NULL}">
    <Edit field="c_AlphaMode" operator="NE" value="e_Custom"/>
</StateRule>
```

If this rule had not been provided, a "CustomValue" parameter (tag 28301) would be transmitted on the wire if the user had entered a value into the spinner and then selected "Daily" or "Annual" from the drop-down list.

3.6 Parameter-to-Control Bindings

In order for an E/OMS to generate an order message it must iterate through all the parameters, find the associated controls, retrieve the control values and determine appropriate values with which to populate the custom FIX tags of the order message. In order for this to be accomplished FIXatdl® provides a means for relating controls to parameters, mainly, the `parameterRef` attribute of the `Control` element. This attribute is set to the value of a `Parameter` element's name attribute, thus providing a binding between the two.

Bindings of controls to parameters may be either one-to-one, where one control is bound to one parameter, or many-to-one, where multiple controls are bound to one parameter. (The only cases of many-to-one bindings involve groups of radio buttons. All other bindings are one-to-one.)

When a binding of a control to a parameter is declared it must be possible for the control's value to be converted to a legal wire-value of the control. For example, it makes little sense for a checkbox control to be bound to a floating point parameter. Rather, a checkbox is more logically fit to be bound to a Boolean parameter.

Not all parameters need an associated control. Some parameters are intended to act as constants and have no GUI control representation. The FIX tags of the parameters are expected to be populated with the same value in every order message regardless of the values of other parameters. When this is the case, an attribute of the `Parameter` element, `constValue`, is used to indicate that the parameter is a constant and provides the value, as in the following listing.

```
<Parameter name="ExecService" xsi:type="Char_t" fixTag="29050" constValue="A"/>
```


Based on this description of "ExecService" the order recipient would expect to receive a FIX message containing the substring "29050=A".

Conversely, it is also the case that not all control need to be bound to a parameter. Controls with no declared `parameterRef` attribute are considered helper controls. They are used to manage the state of other controls via the use of flow-control rules. For example, the following listing describes two controls – a helper control and a control bound to some integer parameter named "CrossQty".

```
<Control ID="EnableCross" xsi:type="CheckBox_t" label="Enable Cross" initValue="false">
<Control ID="CrossQty" xsi:type="SingleSpinner_t" label="Cross Qty" parameterRef="CrossQty">
    <StateRule enable="true">
        <Edit field="EnableCross" operator="EQ" value="true"/>
    </StateRule>
</Control>
```

For a strategy rendered from this description, the user would not be able to enter a value into the "CrossQty" spinner control unless the "EnableCross" checkbox is checked.

3.7 Transport of Strategy Parameters

The FIX Protocol allows algorithmic order parameters to be transported between parties either by use of the StrategyParametersGrp repeating group or by use of user-defined tags mutually agreed upon by the order originator and order recipient. FIXatdl® provides a means for the order recipient to inform the order originator which of these methods to use.

An algorithmic order provider indicates that it can receive parameters through the StrategyParametersGrp component (tags 957-960) by setting the attribute of the `Strategies` element, `tag957Support`, to true. The recipient can also indicate that it is able to receive parameters via user-defined tags by proving values for the `fixTag` attributes of each `Parameter` element. An algorithmic order provider may support both transport methods.

To illustrate, consider the following listing:

```
<Strategies strategyIdentifierTag="27000" versionIdentifierTag="27001" tag957Support="true">
    <Strategy name="POV" uiRep="POV" wireValue="v" version="1" fixMsgType="D">
        <Parameter name="PctVol" xsi:type="Percentage_t" fixTag="27002" use="required"/>
        <Parameter name="FC" xsi:type="Boolean_t" fixTag="27003" use="required"/>
        <StrategyLayout>
            <StrategyPanel>
                <Control ID="c_PctVol" xsi:type="SingleSpinner_t" label="Pct of Volume"
                        parameterRef="PctVol"/>
                <Control ID="c_FC" xsi:type="CheckBox_t" label="Force Completion"
                        parameterRef="FC"/>
            </StrategyPanel>
        </StrategyLayout>
    </Strategy>
</Strategies>
```

This document instance describes an algorithm with two parameters, PctVol and ForceCompletion. The algorithm provider has also indicated that it supports receipt of these parameters via StrategyParametersGrp and via the UDFs 27002 and 27003. So an E/OMS would be free to choose between the two methods when it transmits the parameters. If this were to be rendered by an E/OMS and a user was to enter a "PctVol" value of 0.15 and check the Force Completion checkbox, then the order generated may contain a substring similar to:

```
... 35=D|11=1234|55=AXP|...
|27000=v|27001=1|957=2|958=PctVol|959=11|960=0.15|958=FC|959=13|960=Y
```

In this case the E/OMS has decided to use the StrategyParametersGrp repeating group. If the `tag957Support` attribute were set to false then the E/OMS would be forced to use the UDFs, 27002 and 27003, as in:

```
... 35=D|11=1234|55=AXP|... |27000=v|27001=1|27002=0.15|27003=Y
```

The general rule for determining which method to use is as follows.

tag957Support	fixTag attributes provided	Method for transmitting parameters
true	no	StrategyParametersGrp
true	yes	StrategyParametersGrp or UDFs (but never both)
false	yes	UDFs
false	no	(Not allowed – at least one method must be specified)

3.8 Support for Basket, List and Multileg Order Types

FIXatdl® provides robust support for multileg order types and clarify how their interfaces are described. In doing so, the following concepts were considered.

3.8.1 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 deliver multileg orders to an execution venue:

- Multiple NewOrderSingle(35=D) messages – one for each leg, where an additional identifier is used to associate the individual legs with one another.
- A single NewOrderMultiLeg(35=AB) message – information about individual legs are placed into the repeating group LegOrdGrp.
- A single NewOrderList(35=E) message – information about individual legs are placed into the repeating group ListOrdGrp. 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.

3.8.2 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. The attributes "minLegs" and "maxLegs" are provided for this purpose.

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>
```

3.8.3 Linking and Sequencing of Single Orders

When accepting multileg orders via a group of NewOrderSingle(35=D) messages, the orders need to be linked. FIXatdl® supports the definition of a FIX 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. The attributes "commonIDTag", "legSequenceTag", and "totalLegsTag" are provided for this purpose.

For example, the following strategy requires that linking and sequencing data is to be populated in the user-defined tags 27066, 27067 and 27068.

```
<Strategy
     ...
    commonIDTag="27066"
    legSequenceTag="27067"
    totalLegsTag="27068">
     ...
</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.

3.8.4 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, the leg parameters are wrapped in a the `Leg` element.

For example:

```
<Parameter name="p_OrdParamA" xsi:type="Int_t" fixTag="25000"/>
<Parameter name="p_OrdParamB" xsi:type="Int_t" fixTag="25001"/>
<Leg>
  <Parameter name="p_LegParamA" xsi:type="Int_t" fixTag="26001"/>
  <Parameter name="p_LegParamB" xsi:type="Int_t" fixTag="26002"/>
</Leg>
```

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

3.8.5 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 is necessary to re-send all the legs that were not modified. The attribute Strategy/@legsAreSeverable is provided for this purpose.

3.8.6 Validation of Leg Parameter Values

The validation rules allow references to parameter values in the evaluation of its Boolean expression. For multileg orders, the values of leg parameters need to be supported in the validation rules. This is provided with an additional attribute Edit/@legNo to indicate a leg number.

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>
```

3.8.7 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 it can be assumed that 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 a 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.

FIXatdl® supports a panel to hold all leg-level controls. It needs to be declared just once with the expectation that it will be repeated as many times as necessary according to the value of the attribute “requiredNumberOfLegs”. The `LegPanel` element is provided 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"/>
    </LegPanel>
</StrategyLayout>
```

3.8.8 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 a 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.

3.8.9 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. (As of now the only configuration level identified is whether an OMS allows leg parameters. Some do not.)

The following two examples show 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="25000"/>
    <Parameter name="p_OrdParamB" xsi:type="Int_t" fixTag="25001"/>
    <Leg>
        <Parameter name="p_LegParamA" xsi:type="Int_t" fixTag="26001"/>
        <Parameter name="p_LegParamB" xsi:type="Int_t" fixTag="26002"/>
    </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="25000"/>
    <Parameter name="p_Ord_ParamB" xsi:type="Int_t" fixTag="25001"/>
    <Leg>
      <Parameter name="p_Buy_Leg_ParamA" xsi:type="Int_t" fixTag="26001"/>
      <Parameter name="p_Buy_Leg_ParamB" xsi:type="Int_t" fixTag="26002"/>
      <Parameter name="p_Sell_Leg_ParamA" xsi:type="Int_t" fixTag="26003"/>
      <Parameter name="p_Sell_Leg_ParamB" xsi:type="Int_t" fixTag="26004"/>
    </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 there is no support for delivery by the NewOrderMultiLeg(35=AB) message. The 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.

3.9 Additional Global Definitions

FIXatdl® 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.

3.10 OMS Hooks

One of the key features of FIXatdl® 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.

3.10.1 Validation Rules with References to Standard FIX Fields

In the FIXatdl® specification, several references are made to standard FIX fields. For example, the specification of the `Edit` element (see section Validation Rules), which is used to build validation or flow rules, states 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, the section Dependencies and Structural Constraints beyond XML Schema states 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 normative 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.) FIXatdl® allows 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® seeks to provide clarity concerning the use of standard fields and the fields of an order that an OMS would be expected to make available. The following table lists the fields of an order which may be referenced in a validation rule.

Field Name	Tag Number	Comments
Symbol	55	-
Side	54	-
OrderQty	38	-
OrdType	40	-
Price	44	-
StopPx	99	-
TimeInForce	59	-
HandlInst	21	-
ExecInst	18	This is a MultipleCharValue type. Values are space delimited and arranged in alpha-numeric order.
SecurityType	167	-
TargetSubID	57	-

3.10.2 Filtering according to OMS Environment Values

FIXatdl® defines filtering of strategies based on regions, asset classes and markets. For example, the specification of the `Regions` element (see section Element Definitions) 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. The attribute Region/@inclusion determines whether the region is included or excluded from the set.

*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>
```

The filtering capability can also be applied to parameters and controls 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 example 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="27602"/>
    <Parameter name="EndTime" xsi:type="UTCTimestamp_t" fixTag="27603"/>
    <Parameter name="Text" xsi:type="String_t" fixTag="29999" use="optional"/>
    <Parameter name="Variance" xsi:type="Float_t" fixTag="27641" filter="US-filter"/>
    <Parameter name="Benchmark" xsi:type="String_t" fixTag="27666">
        <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>
        </StrategyPanel>
    ...
    </StrategyLayout>
</Strategy>
```

The filtering criteria includes types of a 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>
```


4 Element Definitions

A high-level description of the elements is provided in the following table.

Element Name	Parent Element(s)	Description
ClientGroup	ClientGroups	Used to define a client classification. For example, large traders may be identified as `<Client ID="LT"/>`. Out-of-band coordination may be required between an algo provider and an E/OMS vendor to define which client fall under the specified categories.
ClientGroups	Filter	An element used to build a list of ClientGroup elements. Applicable when filtering a Strategy based on the firms using the E/OMS.
Control	Strategy, StrategyPanel	Base element used to define GUI controls. Specific instances of controls are defined using the XML Schema xsi:type. For example `<Control xsi:type="lay:TextField_t"/>`, where TextField_t is a complex type defined in the FIXatdl® Layout XML Schema.
ControlRef	StrategyPanel	A reference to a Control element.
Country	Region	An element used to build a list of countries that may be included or excluded from a region. Its attribute, CountryCode, contains an ISO 3166-1 alpha-2 code.
DeliveryMethods	Strategy	Indicates which FIX messages may be used to deliver orders from the order originator to the order recipient.
Description	Parameter, EnumPair, RepeatingGroup, Strategies, Strategy	Text providing a description of its parent element.
Edit	StrategyEdit, StateRule, Strategies, Strategy	Boolean expression evaluated in validation and flow control rules. An Edit element will describe a condition that is either true or false. An Edit element is most commonly used within StrategyEdit and StateRule elements where its scope is limited to its parent element. However, when an Edit is child of a Strategy element, its scope extends the entire Strategy and can be reference by the child StateRule and StrategyEdit elements of the Strategy element. When an Edit is a child of the Strategies element, its scope extends the entire XML instance and may be referenced by any StateRule or StrategyEdit.
EditRef	StrategyEdit, StateRule	Child of a StrategyEdit element used to refer to an Edit element which was declared as a child of a Strategy or as a child of Strategies.
EnumPair	Parameter	Defines a legal value of a parameter in the form of a wire value. A Parameter element will have an EnumPair element for each enumerated value which the parameter can take.
Filter	Strategies, Strategy	Defines a filtering criterion to be used in conjunction with the filter attribute of Parameter, EnumPair, Control and ListItem elements. Filter elements can contain any number of Regions, Markets, SecurityTypes or ClientGroups elements.
FixMsg	DeliveryMethods	Defines a type of FIX message that can be used when delivering orders.
HelpText	Control	Text describing the use of a particular GUI control. This element is used when information about a control is lengthy and would only be appropriate to display in a dialog box – not as a tooltip.

Element Name	Parent Element(s)	Description
LegPanel	StrategyLayout, StrategyPanel	Panel used to display leg-level controls.
ListItem	Control	Used for controls that let the user choose from a list of items. When a Control element is mapped top a Parameter element, via means of the Control element's "parameterRef" attribute, each ListItem will contain a reference to an EnumPair defined within the Parameter element.
Market	Markets	Used as a child element of the Markets element. Defines a particular market using a market identifier code (MIC). An attribute, inclusion, determines whether the market should be included or excluded from the list of markets created by the patterned element, Markets.
Markets	Strategy	This element defines the markets/exchanges (by ISO 10383 MIC Code) of which the strategy is applicable. If no Markets element is defined then the strategy is applicable for *ALL* markets. If a market is defined and has its 'inclusion' attribute set to "Include", then it is implied that the strategy is applicable for *ONLY* that market. If a region is defined and is set to "Exclude", then it is implied that the strategy is applicable for all markets *EXCEPT* that market. Include takes precedence over Exclude - for example, if XNAS is defined and set to "Include" and XLON is defined and set to "Exclude" then all other markets will also be excluded since the "Include" on XNAS takes precedence over the "Exclude" on XLON. In this example, the definition of XLON as "Exclude" is unnecessary. Markets are used in conjunction with regions and countries to define the scope of the strategy. Markets take precedence over regions and countries. For example, if AsiaPacificJapan is defined as "Exclude" but the Fukuoka Stock Exchange (XFKA) is defined as an included market, the strategy will be applicable for all markets in The Americas and EMEA, as well as only the Fukuoka Stock Exchange in the APAC region.
Parameter	Strategy, RepeatingGroup	Element to define the characteristics of an algo parameter with respect to the data interface with the algo provider.
Region	Regions	An individual region used as a child element of the Regions element.
Regions	Strategy	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. If a region is defined and has its 'inclusion' attribute set to "Include", then it is implied that the strategy is applicable for *ONLY* that region. If a region is defined and is set to "Exclude", then it is implied that the strategy is applicable for all regions *EXCEPT* that region. "Include" takes precedence over "Exclude" - for example, if "TheAmericas" is defined and set to "Include" and "EuropeMiddleEastAfrica" is defined and set to "Exclude" then "AsiaPacificJapan" will also be excluded since the "Include" on "TheAmericas" takes precedence over the "Exclude" on "EuropeMiddleEastAfrica". In this example, the definition of "EuropeMiddleEastAfrica" as "Exclude" is unnecessary.

Element Name	Parent Element(s)	Description
		Regions also contain a child element called "Country" that allows the algo author to further specify the geographic scope of the strategy. Countries can be included and excluded in the same manner as regions and the same rules of precedence apply. Please see fixatdl-regions-1-2.xsd for the list of ISO 3166 Country Code to region mappings.
RepeatingGroup	Strategy	Container of a group of Parameter elements that are intended for use with multileg or basket strategies. Parameters contained within a RepeatingGroup element are intended to have their tag=value pairs populated in either the ListOrdGrp repeating group of a NewOrderList(35=E) message or the LegOrdGrp repeating group of a NewOrderMultileg(35=AB) message. Parameters not contained within a RepeatingGroup element have their values populated in the main body of a message.
SecurityType	SecurityTypes	An element used to describe a security type that may be included or excluded from the list built by the parent element, SecurityTypes. Its attribute, "name", contains a FIX SecurityType(167) value.
SecurityTypes	Strategy	The list of security types (by SecurityType(167)) for which the given strategy is valid. The absence of any security types implies that the strategy is valid for all security types.
StateRule	Control	Defines workflow rule for a Control. Defines a workflow rule for a GUI control. Using StateRule as a child element of a Control element, rules can be defined which affect the "enabled" and "hidden" properties of the underlying Java/.Net/Web/etc. rendered on the screen. A StateRule element must contain a child Edit element. The action defined by the StateRule is in-effect when the condition described by its child Edit element is true. The action is not in-effect when the condition described by its child Edit element is false.
Strategies	[n/a]	Container for all strategy elements. It is the root element of all FIXatdl® conforming documents.
Strategy	Strategies	Root level of a strategy definition.
StrategyEdit	Strategy	Definition of a validation rule. A StrategyEdit element must contain an Edit element as a child. The boolean expression described by the Edit element is an assertion, i.e., validation succeeds if the condition described by the Edit is true and fails when the condition described by the Edit element is false. In the case where validation fails, the error message, supplied by the errorMessage attribute of StrategyEdit, may be displayed to an OMS user or logged.
StrategyLayout	Strategy	Container for StrategyPanel elements. If declared, a StrategyLayout element must contain at least one StrategyPanel as a child element.
StrategyPanel	StrategyLayout, StrategyPanel	Container for either groups of parameters or StrategyPanel elements, but not both. I.e., a StrategyPanel element will contain either all Control elements or all StrategyPanel elements.
StrategyPanelRef	StrategyLayout, StrategyPanel	A reference to a StrategyPanel element.
VendorConfig	Strategy	Element used to describe the requirements necessary for the E/OMS to support for the parent strategy to be applicable. Attributes are

Element Name	Parent Element(s)	Description
		"legParameters" (indicating whether parameters can be included in the legs of multileg orders) and "tag66support" (requiring the E/OMS to populate ListID(66) to link together components of a multileg order).

5 Attribute Definitions of Elements

The following tables describe the attributes of all the FIXatdl® XML elements. The format of the attribute name is

`<element name>/@<attribute>` where the element is one of the XML elements defined by FIXatdl®.

Since some of the attributes are overloaded due to the way the `Parameter` and `Control` elements can be extended, types of certain attributes will depend on the type of the element. For these attributes, the conditions determining their type will be listed in their description.

Attributes that are applicable to extensions of the `Parameter` and `Control` elements have not been included in the tables. These are defined in section Abstract Element Extensions.

5.1 Client Element

Attribute	Type	Req'd	Description
Client/@ID	string	Y	Used to define a client classification. For example, large traders may be identified as `<Client ID="LT"/>`.

5.2 Control Element

Attribute	Type	Req'd	Description
Control/@checkedEnumRef	StringID	N	Refers to an enumID defined in the definition of the Parameter referred by Control/@parameterRef. This enumID is the output from this control if it is checked/selected. **(See the section A Sample FIXatdl® Document in this document for an example. Examine the Parameter "AllowDarkPoolExec" and Control "DPOption" for details.)** Applicable when xsi:type is CheckBox_t or RadioButton_t.
Control/@col	non-negative int	N	Column in which this item is to appear in a grid-oriented panel. Applicable when encompassing StrategyPanel orientation is GRID.
Control/@colSpan	non-negative int	N	Number of colums an item is to span in a grid-oriented panel. (Default: 1) Applicable when encompassing StrategyPanel orientation is GRID.
Control/@disableForTemplate	boolean	N	For implementing systems that support saving order templates or pre-populated orders for basket trading/list trading this attribute specifies that the control should be disabled when the order screen is going to be saved as a template and not actually used to place an order.
Control/@disablingControlType	string	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.

Attribute	Type	Req'd	Description
			Valid values: • CheckBox • RadioButton • DropDown
Control/@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.
Control/@displayableDate	boolean	N	Instructs the OMS to display the date. Applicable when xsi:type is Clock_t (default: false)
Control/@displayableTz	boolean	N	Instructs the OMS to display the time zone associated with the value entered by the user. Applicable when xsi:type is Clock_t (default: true)
Control/@editableDate	boolean	N	Instructs the OMS to allow the user to change the date. The OMS would decide how to do this. Applicable when xsi:type is Clock_t (default: true)
Control/@editableTz	boolean	N	Instructs the OMS to allow the user to change the time zone. The OMS would decide how to do this. Applicable when xsi:type is Clock_t (default: false)
Control/@enablingControlType	string	N	Description of the GUI control to be rendered which directs the OMS to enable the clock and allow it to accept user input. 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. Valid values: • CheckBox • RadioButton • DropDown
Control/@ID	StringID	Y	Unique identifier of this control. No two controls of the same strategy can have the same ID.
Control/@increment	decimal	N	Limits the granularity of a spinner control. Useful in spinner objects to enforce odd-lot and sub-penny restrictions. Applicable when xsi:type is SingleSpinner_t or Slider_t. (In this case a Slider_t must be used to select a value within a continuous range, say a decimal value between a minimum and maximum value. As opposed to the case where the Slider_t is used to select from a set of values not unlike a DropDownList_t.)
Control/@incrementPolicy	string	N	For single spinner control, defines how to determine the

Attribute	Type	Req'd	Description
			increment. Valid values: • Static – use value from increment attribute • LotSize – use the round lot size of symbol. (If this value is not available, use the value from the increment attribute.) • Tick – use symbol minimum tick size. (If this value is not available, use the value from the increment attribute.) Applicable when xsi:type is SingleSpinner_t. If no value is supplied then use value from increment attribute. **Please note: The schema file, fixatdl-layout-1-2.xsd, does not include the "Static" enumeration value. If "Static" behavior is desired then do not populate this attribute.**
Control/@initFixField	positiveInteger	N	Indicates the initialization value is to be taken from this standard FIX field. Format: "FIX_" + FIXFieldName. E.g. "FIX_OrderQty". Required when initPolicy="UseFixField".
Control/@initPolicy	string	N	Describes how to initialize the control. If the value of this attribute is undefined or equal to "UseValue" and initValue is defined then initialize with initValue. If the value is equal to "UseFixField" then attempt to initialize with the value of the tag specified in initFixField. If the value is equal to "UseFixField" and it is not possible to access the value of the specified FIX tag then revert to using initValue. If the value is equal to "UseFixField", the field is not accessible, and initValue is not defined, then do not initialize. Valid values: • UseValue • UseFixField
Control/@initValue	(Depends on value of xsi:type)	N	The value used to pre-populate the GUI component when the order entry screen is initially rendered. The type of initValue is dependent on the value of Control/@xsi:type. The following list gives the type of this attribute based on the value of xsi:type. **xsi:type:** initValue *type* **Clock_t:** time **TextField_t:** string **SingleSelectList_t:** string **MultiSelectList_t:** MultipleStringValue

Attribute	Type	Req'd	Description
			Slider_t: double **CheckBox_t:** boolean ("true"/"false") **CheckBoxList_t:** MultipleStringValue **SingleSpinner_t:** double **DoubleSpinner_t** double **DropDownList_t** string **EditableDropDownList_t:** string **RadioButton_t:** boolean ("true"/"false") **RadioButtonList_t:** string **Label_t:** string **HiddenField_t:** string The use of initValue also depends on the value of xsi:type. **xsi:type:** initValue *use* **Clock_t:** time (expressed in Control/@localMktTz) **TextField_t:** string **SingleSelectList_t:** enumID of a child ListItem **MultiSelectList_t:** enumIDs of child ListItems **Slider_t:** valid value returned by the slider **CheckBox_t:** "true" (checked) or "false" (unchecked) **CheckBoxList_t:** enumIDs of ListItems to be checked (separated by single spaces) **SingleSpinner_t:** double **DoubleSpinner_t:** double **DropDownList_t:** enumID of a child ListItem **EditableDropDownList_t:** enumID of a child ListItem **RadioButton_t:** "true" (selected) or "false" (unselected) **RadioButtonList_t:** enumID of ListItem to be pushed **Label_t:** string to render **HiddenField_t:** non-displayed string Required when initPolicy="UseValue".
Control/@initValueMode	int	N	Defines the treatment of initValue time. 0: use initValue; 1: use current time if initValue time has passed. The default value is 0. Applicable only when Control/@xsi:type is Clock_t.
Control/@innerIncrement	decimal	N	Limits the granularity of the inner spinner of a double spinner control. Useful in spinner objects to enforce odd-lot and sub-penny restrictions. Applicable when xsi:type is DoubleSpinner_t.
Control/@innerIncrementPolicy	string	N	For double spinner control, defines how to determine the increment for the inner set of spinners. Valid values: • Static – use value from innerIncrement attribute • LotSize – use the round lot size of symbol

Attribute	Type	Req'd	Description
			• Tick – use symbol minimum tick size Applicable when xsi:type is DoubleSpinner_t. If no value is supplied then use value from innerIncrement attribute. **Please note: The schema file, fixatdl-layout-1-2.xsd, does not include the "Static" enumeration value. If "Static" behavior is desired then do not populate this attribute.**
Control/@label	string	N	A title for this control which may be displayed. If the control is a Label_t then Control/@label or Control/@initValue must be used to define the string which is to be rendered. If both attributes are provided then Control/@initValue takes precedence.
Control/@localMktTz	LocalMktTz	N	The timezone in which initValue is represented in. Required when initValue is supplied. Applicable when xsi:type is Clock_t.
Control/@orientation	Orientation	Y	Declares the orientation of the radio buttons within a RadioButtonList or the checkboxes within a CheckBoxList. Valid values: • HORIZONTAL • VERTICAL Applicable when xsi:type is RadioButtonList_t or CheckBoxList_t.
Control/@outerIncrement	decimal	N	Limits the granularity an outer spinner of a double spinner control. Useful in spinner objects to enforce odd-lot and sub-penny restrictions. Applicable when xsi:type is DoubleSpinner_t.
Control/@outerIncrementPolicy	string	N	For double spinner control, defines how to determine the increment for the outer set of spinners. Valid values: • Static – use value from outerIncrement attribute • LotSize – use the round lot size of symbol • Tick – use symbol minimum tick size Applicable when xsi:type is DoubleSpinner_t. If no value is supplied then use value from outerIncrement attribute. **Please note: The schema file, fixatdl-layout-1-2.xsd, does not include the "Static" enumeration value. If "Static" behavior is desired then do not populate this attribute.**

Attribute	Type	Req'd	Description
Control/@parameterRef	StringID	N	The name of the parameter for which this control gives the visual representation. A parameter with this name must be defined within the same strategy as this control.
Control/@radioGroup	String	N	Identifies a common group name used by a set of RadioButton_t among which only one radio button may be selected at a time. Applicable when xsi:type is RadioButton_t.
Control/@row	non-negative int	N	Row in which this item is to appear in a grid-oriented panel. Applicable when encompassing StrategyPanel orientation is GRID.
Control/@rowSpan	non-negative int	N	Number of rows an item is to span in a grid-oriented panel. (default: 1) Applicable when encompassing StrategyPanel orientation is GRID.
Control/@tooltip	string	N	Tool tip text for rendered GUI objects rendered for the parameter.
Control/@uncheckedEnumRef	StringID	N	Refers to an enumID defined in the definition of the parameter referred by Control/@parameterRef. This enumID is the output from this control if it is unchecked/unselected. **(See the section A Sample FIXatdl® Document in this document for an example. Examine the parameter "AllowDarkPoolExec" and Control "DPOption" for details.)** Applicable when xsi:type is CheckBox_t or RadioButton_t.
Control/@xsi:type	string	Y	Indicates the type of GUI control that should be rendered on the screen. Valid values: • CheckBox_t • CheckBoxList_t • Clock_t • DoubleSpinner_t • DropDownList_t • EditableDropDownList_t • HiddenField_t • Label_t • MultiSelectList_t • RadioButton_t • RadioButtonList_t • SingleSelectList_t

Attribute	Type	Req'd	Description
			• SingleSpinner_t • Slider_t • TextField_t

5.3 Country Element

Attribute	Type	Req'd	Description
Country/@CountryCode	String restricted to "[A-Z0-9]{2}"	Y	ISO 3166-1 alpha-2 code for the countries to include or exclude in a given region.
Country/@inclusion	string	Y	Indicates whether this country should be included or excluded from encompassing list. Valid values: • Include • Exclude

5.4 Edit Element

Attribute	Type	Req'd	Description
Edit/@field	string	N	Field name for comparison. When the edit is used within a StateRule, this field must refer to the ID of a Control. When the edit is used within a StrategyEdit, this field must refer to either the name of a parameter or a standard FIX field name. When referring to a standard FIX tag then the name must be pre-pended with the string "FIX_", e.g. "FIX_OrderQty". Required when: Edit/@operator is defined.
Edit/@field2	string	N	Value used as the second operand. Used in conjunction with Edit/@field and Edit/@operator. Similar definition to Edit/@field except that it is mutually exclusive with Edit/@value. Required when: Edit/@operator is in {GE,GT, LE, LT, EQ, NE} and Edit/@value is not specified.
Edit/@id	string	N	Optional identifier. Allows for re-use of this edit within StateRule or EditRef elements. This attribute is required if the Edit element is a direct child of either the Strategies or Strategy elements.
Edit/@legNo	non-negative int	N	Used in conjunction with Edit/@field, declares the leg number of which the field is to be retrieved when the field is evaluated.
Edit/@logicOperator	LogicalOperator	N	Operator where operands are one or more Edit elements. Short-circuit evaluation is assumed in all edit statements. Valid values: • AND • OR • XOR • NOT Required when operator is not present. An edit element must

Attribute	Type	Req'd	Description
			contain either a logicOperator attribute or an operator attribute, but never both. By convention, XOR returns true when **one and only one** of its operands is true.
Edit/@operator	Operator	N	One of the following enumerated types: • EX (Exists. I.e. the user has entered a value) • NX (Not exists. I.e. the user has not entered a value) • EQ (Equal) • LT (Less than) • GT (Greater than) • NE (Not equal) • LE (Less than equal) • GE (Greater than equal) Required when logicOperator is not present. An Edit element must contain either a logicOperator attribute or an operator attribute, but never both.
Edit/@value	string	N	Value used as the second operand. Used in conjunction with Edit/@field and Edit/@operator. Represents a string literal value and not a reference. When Edit is a descendant of a StateRule element, Edit/@value refers to the value of the control referred by Edit/@field. If the control referred by Edit/@field has enumerated values then Edit/@value refers to the enumID of one of the control's ListItem elements. When Edit is a descendant of a StrategyEdit element, Edit/@value refers to the wireValue of the parameter referred by Edit/@field. Required when: Edit/@operator is in {GE, GT, LE, LT, EQ, NE} and Edit/@field2 is not specified.
EditRef/@id	string	Y	Refers to an ID of a previously defined Edit element. The Editelement may be defined at the strategy level or at the strategies level.

5.5 EnumPair Element

Attribute	Type	Req'd	Description
EnumPair/@enumID	StringID	Y	A unique identifier of an enumPair element per parameter.
EnumPair/@index	integer	N	**Deprecated.** Previously defined an ordering of the enumerated values. If defined it should be ignored.
EnumPair/@wireValue	string	Y	The corresponding value that is used to populate the FIX message.

5.6 Filter Element

Attribute	Type	Req'd	Description
Filter/@id	StringID	Y	An identifier for a global filter definition. Elements which declare a filter attribute

Attribute	Type	Req'd	Description
			must have it refer to a global filter ID.

5.7 FixMsg Element

Attribute	Type	Req'd	Description
FixMsg/@msgType	string	Y	Method in which the algo provider can accept this type of order. Valid values: • NewOrderSingle • NewOrderMultileg • NewOrderList

5.8 ListItem Element

Attribute	Type	Req'd	Description
ListItem/@enumID	StringID	N	A reference to the enumPair specified in the parameter definition specified by the parent Control's parameterRef attribute. Use is optional when the parent Control element does not refer to a parameter. Required when: the parent Control element has a defined parameterRef attribute.
ListItem/@uiRep	string	Y	The value shown in the list. These are the values that go into Java, .Net or Web list controls.

5.9 Market Element

Attribute	Type	Req'd	Description
Market/@inclusion	string	Y	Indicates whether this market should be included or excluded from encompassing list. Valid values: • Include • Exclude
Market/@MICCode	string	Y	String representing a market or exchange – ISO 10383 Market Identifier Code (MIC).

5.10 Parameter Element

Attribute	Type	Req'd	Description
Parameter/@constValue	(Depends on value of xsi:type)	N	The value of a parameter that is constant and is not referred by a Control element. This value must be sent on the wire by the order generating application. The following list gives the type of this attribute based on the value of xsi:type. **xsi:type:** constValue type **Int_t:** int

Attribute	Type	Req'd	Description
			Length_t: positiveInteger **NumInGroup_t:** positiveInteger **SeqNum_t:** positiveInteger **TagNum_t:** positiveInteger **Float_t:** decimal **Qty_t:** Qty **Price_t:** Price **PriceOffset_t:** PriceOffset **Amt_t:** Amt **Percentage_t:** Percentage **Char_t:** char **Boolean_t:** Boolean ('Y'/'N') **String_t:** string **MultipleCharValue_t:** MultipleCharValue **Currency_t:** Currency **Exchange_t:** Exchange **MonthYear_t:** MonthYear **UTCTimestamp_t:** time **UTCTimeOnly_t:** time **LocalMktDate_t:** date **UTCDateOnly_t:** UTCDateOnly **Data_t:** Data **MultipleStringValue_t:** MultipleStringValue **Country_t:** Country **Language_t:** language **TZTimestamp_t:** time **TZTimeOnly_t:** TZTimeOnly **Tenor_t:** Tenor When defined in UTCTimestamp_t elements the following apply: • Contains only time information – not day, month or year. • Used in conjunction with Parameter/@localMktTz, this value must be used for the time portion of a UTCTimestamp that is sent on the wire by the order generating application. For example, if constValue="08:30:00" and localMktTz="America/Chicago", daylight savings time is in effect in Chicago and the date is July 1, 2010, then the value "20100701-13:30:00" would be sent on the wire.
Parameter/@definedByFIX	boolean	N	Indicates whether the parameter is a redefinition of a standard FIX tag. The default value is False. For example, if the algorithm redefines the OrderQty(38) then the parameter declaration may be similar to: `<Parameter name="OrderQty" xsi:type="Qty_t" fixTag="38" definedByFIX="true" use="required"/>`

Attribute	Type	Req'd	Description
Parameter/@falseWireValue	string	N	Applicable only when xsi:type is Boolean_t. **This attribute is targeted for deprecation.** To achieve the same functionality, it is recommended that a Char_t or String_t type parameter be used instead of a Boolean_t. The parameter should have two EnumPairs defined with one defining the false wire-value and the other defining the true wire-value. The parameter should be bound to a CheckBox control. The CheckBox control should define the parameters checkedEnumRef and uncheckedEnumRef to refer to the enumIDs of the parameter. (See the section A Sample FIXatdl® Document in this document for an example. Examine the Parameter "AllowDarkPoolExec" and Control "DPOption" for details.) The deprecated use is described as follows: Defines the value with which to populate the FIX message when the boolean parameter is False. Overrides the standard FIX boolean value of "N". I.e. if this attribute is not provided then the order-sending application must use "N". If it is desired that the FIX message is not to be populated with this tag when the value of the parameter is false, then falseWireValue should be defined as "{NULL}".
Parameter/@filter	string	N	Refers to the ID of a globally defined filter. Affects whether the parameter is applicable in the strategy.
Parameter/@fixTag	positiveInteger	N	The tag that will hold the value of the parameter. Required when: parameter value is intended to be transported over the wire. If fixTag is not provided then the Strategies-level attribute, tag957Support, must be set to true, indicating that the order recipient expects to receive algo parameters in the StrategyParameterGrp repeating group beginning at tag 957.
Parameter/@invertOnWire	boolean	N	Applicable when: xsi:type is MultipleStringValue_t or MultipleCharValue_t. Instructs the OMS whether to perform a bitwise "not" operation on each element of these lists.
Parameter/@localMktTz	string	N	Describes the time zone without indicating whether daylight savings is in effect. Valid values are taken from names in the Olson time zone database. All are of the form Area/Location, where Area is the name of a continent or ocean, and Location is the name of a specific location within that region. E.g. America/Chicago. Applicable when xsi:type is UTCTimestamp_t.
Parameter/@maxLength	non-negative	N	Applicable when xsi:type is String_t, MultipleCharValue_t

Attribute	Type	Req'd	Description
	int		or MultipleStringValue_t. The maximum allowable length of the parameter.
Parameter/@maxValue	(Depends on value of xsi:type)	N	Maximum value of the parameter accepted by the algorithm provider. The following list gives the type of this attribute based on the value of xsi:type. **xsi:type:** initValue type **Int_t:** int **Float_t:** decimal **Qty_t:** Qty **Price_t:** Price **PriceOffset_t:** PriceOffset **Amt_t:** Amt **Percentage_t:** Percentage **MonthYear_t:** MonthYear **UTCTimestamp_t:** time **UTCTimeOnly_t:** time **LocalMktDate_t:** date **UTCDateOnly_t:** UTCDateOnly **TZTimestamp_t:** time **TZTimeOnly_t:** TZTimeOnly **Tenor_t:** Tenor This attribute is applicable only for the xsi:type values listed above. maxValue has no default value. When defined in UTCTimestamp_t elements the following applies: • Maximum local market time. Represents an instance of time that recurs every day. Contains only time information – not day, month or year. • Used in conjunction with Parameter/@localMktTz, this value represents the maximum time of day allowed for the parameter.
Parameter/@minLength	non-negative int	N	Applicable when xsi:type is String_t. The minimum allowable length of the parameter.
Parameter/@minValue	(Depends on value of xsi:type)	N	Minimum value of the parameter accepted by the algorithm provider. The following list gives the type of this attribute based on the value of xsi:type. Default values, where applicable, are provided, otherwise minValue has no default value. **xsi:type:** initValue type (default) **Int_t:** int **Float_t:** decimal **Qty_t:** Qty (0)

Attribute	Type	Req'd	Description
			Price_t: Price (0) **PriceOffset_t:** PriceOffset (0) **Amt_t:** Amt (0) **Percentage_t:** Percentage (0) **MonthYear_t:** MonthYear **UTCTimestamp_t:** UTCTimestamp **UTCTimeOnly_t:** time **LocalMktDate_t:** date **UTCDateOnly_t:** UTCDateOnly **TZTimestamp_t:** time **TZTimeOnly_t:** TZTimeOnly **Tenor_t:** Tenor This attribute is applicable only for the xsi:type values listed above. When defined in UTCTimestamp_t the following applies: • Minimum local market time. Represents an instance of time that recurs every day. Contains only time information – not day, month or year. • Used in conjunction with Parameter/@localMktTz, this value represents the minimum time of day allowed for the parameter.
Parameter/@multiplyBy100	boolean	N	Applicable for xsi:type of Percentage_t. If true then percent values must be multiplied by 100 before being sent on the wire. For example, if multiplyBy100 were false then the percentage, 75%, would be sent as 0.75 on the wire. However, if multiplyBy100 were true then 75 would be sent on the wire. If not provided it should be interpreted as false. **Use of this attribute is not recommended. The motivation for this attribute is to maximize compatibility with algorithmic interfaces that are non-compliant with FIX in regard to their handling of percentages. In these cases an integer parameter should be used instead of a percentage.**
Parameter/@mutableOnCxlRpl	boolean	N	Indication of whether the parameter's value can be modified by an OrderCancelReplaceRequest(35=G) message. Default value: true
Parameter/@name	string restricted to "[A-Za-z] [A-Za-z0-9_] {0,255}"	Y	The name of the parameter. No two parameters of any strategy may have the same name. The name may be used as a unique key when referenced from the other sub-schemas. Names must begin with an alpha character followed only by alpha-numeric characters and must not contain whitespace characters.
Parameter/@precision	non-negative int	N	The number of digits to the right of the decimal point in which to round when populating the FIX message. Lack of

Attribute	Type	Req'd	Description
			this attribute indicates that the value entered by the user should be taken as-is without rounding. **Applicable when** xsi:type is Float_t, Price_t, PriceOffset_t or Qty_t.
Parameter/@revertOnCxlRpl	boolean	N	Indicates how to interpret those tags that were populated in an original order but are not populated in a subsequent cancel/replace of the order message. If this value is true then revert to the value of the original order, otherwise a null value or the parameter's default value (Control/@initValue) is to be used or if none is specified, the parameter is to be omitted. Default value: false **NOTE:** Although revertOnCxlRpl and mutableOnCxlRpl might appear to be mutually exclusive, this is not strictly the the case, and as the default value for mutableOnCxlRpl is 'true', it is recommended practice to explicitly include **mutableOnCxlRpl="false"** if the option revertOnCxlRpl="true" is set for a given parameter (assuming of course this is the intended behaviour).
Parameter/@scope	string	N	The scope of the parameter. Applicable to a multileg order type. Valid values: • LEG (indicating the parameter appears in the legs of the order) • ORDER (indicating it appears in the main body of the order)
Parameter/@trueWireValue	string	N	Applicable only when xsi:type is Boolean_t. **This attribute is targeted for deprecation.** To achieve the same functionality, it is recommended that a Char_t or String_t type parameter be used instead of a Boolean_t. The parameter should have two EnumPairs defined with one defining the false wire-value and the other defining the true wire-value. The parameter should be bound to a CheckBox control. The CheckBox control should define the parameters checkedEnumRef and uncheckedEnumRef to refer to the enumIDs of the parameter. (See the section A Sample FIXatdl® Document in this document for an example. Examine the Parameter "AllowDarkPoolExec" and Control "DPOption" for details.) The deprecated use is described as follows: Defines the value with which to populate the FIX message when the boolean parameter is True. Overrides the standard FIX boolean value of "Y". I.e. if this attribute is not provided then the order-sending application must use "Y". If it is desired that the FIX message is not to be populated

Attribute	Type	Req'd	Description
			with this tag when the value of the parameter is false, then falseWireValue should be defined as "{NULL}".
Parameter/@use	Use_t	N	Indicates whether a parameter is optional or required. Valid values: • optional (default) • required
Parameter/@xsi:type	string	Y	Indicates the type of the parameter. The type of the parameter determines which of the extended attributes are applicable. The namespace, xsi, must be declared within the Strategies element with the statement: xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance. Valid values: • Amt_t • Boolean_t • Char_t • Country_t • Currency_t • Data_t • Exchange_t • Float_t • Int_t • Language_t • Length_t • LocalMktDate_t • MonthYear_t • MultipleCharValue_t • MultipleStringValue_t • NumInGroup_t • Percentage_t • Price_t • PriceOffset_t • Qty_t • SeqNum_t • String_t • TagNum_t • Tenor_t • TZTimeOnly_t • TZTimestamp_t • UTCDateOnly_t • UTCTimeOnly_t • UTCTimestamp_t

5.11 Region Element

Attribute	Type	Req'd	Description
Region/@inclusion	string	Y	Indicates whether this region should be included or excluded when declared within a list of regions. Valid values: • Include • Exclude
Region/@name	String	Y	The name of the region. Valid values: • TheAmericas • EuropeMiddleEastAfrica • AsiaPacificJapan

5.12 RepeatingGroup Element

Attribute	Type	Req'd	Description
RepeatingGroup/@fixTag	int	N	The FIX tag corresponding to a NoXXX tag. Indicates that the Parameter elements defined within the RepeatingGroup element are repeating group tags when sent over the wire. Valid values: • 555 (NoLegs) • 68 (TotNoOrders) In the case where fixTag=68, either multiple NewOrderList(35=E) messages may be sent where the total number of orders over the entire list must be equal to Strategy/@totalOrders, or multiple NewOrderSingle(35=D) messages may be sent where total number of orders must be equal to Strategy/@totalOrders.
RepeatingGroup/@maxSize	int	N	The maximum number of legs or list orders.
RepeatingGroup/@minSize	int	Y	The minimum number of legs or list orders.
RepeatingGroup/@name	string	N	FIX Field name of the repeating group. Must refer to a FIX field of NumInGroup type. Valid values: • TotNoOrders (when NewOrderList(35=E) messages are expected) • NoLegs (when NewOrderMultileg(35=AB) messages are expected) This field should be omitted when NewOrderSingle(35=D) messages are expected.

5.13 SecurityType Element

Attribute	Type	Req'd	Description
SecurityType/@inclusion	string	Y	Indicates whether this security type should be included or excluded from encompassing list. Valid values: • Include • Exclude
SecurityType/@name	string	Y	Indicates type of security. Valid values equivalent to FIX SecurityType(167) values.

5.14 StateRule Element

Attribute	Type	Req'd	Description
StateRule/@enabled	boolean	N	Indicates whether or not to enable the control when the edit expression of the StrategyEdit element evaluates to True. The desired behavior is as follows: • when the StateRule element's edit condition is true and enabled=true then enable the control; • when the edit condition is true and enabled=false then disable the control; • when the edit condition is false and enable=true then disable the control; • when the edit condition is false and enabled=false then enable the control. The value of a control's "enabled" property does not play a role in determining whether a value is populated when a FIX order message is generated. I.e. A control's "enabled" property does not influence what goes out on the wire.
StateRule/@value	string	N	GUI control's displayed value should be set to this value when edit condition is true. Although the type of this attribute has been listed as string, ultimately the type of this attribute must be compatible with the type of the control. For example, if the control is numeric, such as a SingleSpinner, then a string containing a numeric value would an appropriate value (e.g. "15"). If the control contains ListItem elements then allowable values of StateRule/@value are restricted to the enumIDs of the ListItem elements. A special token, "{NULL}", may be used for the value of this attribute to indicate that the control should be set to an uninitialized state. Controls that are un-initialized should have no value. The effect of an un-initialized control is as follows: When an order is to be generated, the controls which are linked to parameters will have their values retrieved. If there is no retrieved value because the control was un-initialized then the parameter should have no value and its associated FIX tag should be excluded from the message. This is relevant only for controls that can be in an un-initialized state such as spinners and text fields. Controls such as check boxes and radio buttons are always initialized. (They are either checked or unchecked.)

Attribute	Type	Req'd	Description
StateRule/@visible	boolean	N	Indicates whether or not to show the control when the boolean expression, defined by the Edit element, evaluates to True. The desired behavior is as follows: when the StateRule element's edit condition is true and visible=true then display the control; when the edit condition is true and visible=false then hide the control; when the edit condition is false and visible=true then hide the control; when the edit condition is false and enabled=false then display the control.

5.15 Strategies Element

Attribute	Type	Req'd	Description
Strategies/@changeStrategyOnCxlRpl	boolean	N	Indicates whether a new strategy can be chosen during a Cancel/Replace.
Strategies/@draftFlagIdentifierTag	positiveInteger	N	The tag within the FIX order message to be populated with a boolean ('Y'/'N') indicating whether the order is a draft.
Strategies/@imageLocation	string	N	Filepath or URL of an image file or logo of the algo providing firm.
Strategies/@strategyIdentifierTag	positiveInteger	Y	The tag within the FIX order message to be populated with a value identifying the chosen strategy. E.g. if strategyIdentifierTag is 25001 and the chosen strategy is identified by the value 'VWAP' then the FIX order message would contain the tag-value pair 25001=VWAP.
Strategies/@versionIdentifierTag	positiveInteger	N	The tag within the FIX order message to be populated with a value identifying the version of a chosen strategy. For example, if versionIdentifierTag is 25002 and the version of the chosen strategy is '2.01' then the FIX order message would contain the tag-value pair 25001=2.01
Strategies/@tag957Support	boolean	N	Indicates whether the order recipient can receive algorithmic parameters in the StrategyParametersGrp repeating group starting with NoStrategyParameters(957). If this mode of parameter transport is not supported then the fixTag attribute of all Parameter elements is required. Default value: false.

5.16 Strategy Element

Attribute	Type	Req'd	Description
Strategy/@commonIDTag	non-negative int	N	Used to denote where to place a common basket ID when linking together *single* orders. 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

Attribute	Type	Req'd	Description
			several NewOrderSingle(35=D) messages.
Strategy/@disclosureDoc	anyURI	N	URL of a disclosure document supplied by the algorithm provider.
Strategy/@filter	string	N	A reference to the ID of a Filter element defined in the scope of the Strategies element.
Strategy/@fixMsgType	string	N	Indicates the FIX message to use when transmitting the order. Values taken from FIX field MsgType(35). Valid values: • D (NewOrderSingle) • E (NewOrderList) • AB (NewOrderMultiLeg) • s (NewOrderCross)
Strategy/@imageLocation	string	N	File path or URL of an image file or logo of this particular strategy.
Strategy/@legsAreSeverable	boolean	N	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.
Strategy/@legSequenceTag	non-negative int	N	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. Used in conjunction with the totalLegsTag attribute.
Strategy/@maxLegs	non-negative int string constant "unbounded" (strategy accepts any number of legs)	N	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.
Strategy/@minLegs	non-negative int	N	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.
Strategy/@name	StringID	Y	Unique identifier of a strategy. Strategy names must be unique per provider.
Strategy/@objective	string	N	An optional classification of a multi-leg order.

Attribute	Type	Req'd	Description
			Valid values: • PAIRS • BUTTERFLY • BUY-WRITE • CALENDAR-SPREAD • PRICE-SPREAD • DIAGONAL-SPREAD • SPREAD • PORTFOLIO
Strategy/@orderSequenceTag	non-negative int	N	Used to denote the tag which will contain the sequence number of a particular order of a basket.
Strategy/@providerID	string	N	Identifies the firm providing the algorithm.
Strategy/@providerSubID	string	N	A further level of firm identification.
Strategy/@requiredNumberOfLegs	non-negative int	N	Used to denote number of repeating orders in a NewOrderList(35=E) message or a basket of NewOrderSingle(35=D) messages that the algo provider expects to receive.
Strategy/@sentOrderLink	anyURI	N	Prefix portion of a URL to access the order or draft at the target, e.g., https://xyz.com/algo/dashboard?SenderCompID=OMS. Append to this the specific SenderCompID string, an ampersand, "ClOrdID=", and the specific ClOrdID-string. Trader hits this full URL to communicate regarding the order or draft. See additional documentation.
Strategy/@totalLegs	non-negative int	N	Used when msgType(35)=AB and denotes the number of repeating legs.
Strategy/@totalLegsTag	non-negative int	N	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. Used in conjunction with the legSequenceTag attribute.
Strategy/@totalOrders	non-negative int	N	Used to denote number of repeating orders in a NewOrderList(35=E) message or a basket of NewOrderSingle(35=D) messages.
Strategy/@totalOrdersTag	non-negative int	N	In basket trading, used to denote where to place the total number of orders of a basket.
Strategy/@uiRep	string	N	The name of the strategy as rendered in the UI. If not provided then the "name" attribute should be used. (This is the value rendered on the UI when the user is presented with a choice of algorithms.)
Strategy/@version	string	Y	Information to facilitate version control.
Strategy/@wireValue	string	Y	The value used to identify the algorithm. The tag referred to by Strategy/@strategyIdentifierTag will be set to this value.

5.17 StrategyEdit Element

Attribute	Type	Req'd	Description
StrategyEdit/@errorMessage	string	Y	The error message to display when the boolean expression defined by the Edit element of a StrategyEdit element evaluates to False.

5.18 StrategyPanel Element

Attribute	Type	Req'd	Description
StrategyPanel/@border	Border	N	Recommended border for the panel. Valid values: • None • Line
StrategyPanel/@collapsed	boolean	N	Initial visual state of a panel. Indicates whether a panel is initially drawn in a collapsed state. Default value: false.
StrategyPanel/@collapsible	boolean	N	Indicates whether panel can be collapsed. Default value: false. (Note that the default value may conflict with the default value defined in the schema file, fixatdl-layout-1-2.xsd. To avoid conflict it is recommended that this attribute be treated as a required attribute.)
StrategyPanel/@color	string	N	The background color of a panel. The value should appear as the RBG combination separated by commas. **It is recommended that vendors ignore this attribute and rely on their own color scheme.**
StrategyPanel/@fillOrder	string	N	Describes how the grid items are to be arranged. Valid values: • COL-MAJOR (default) • ROW-MAJOR Applicable when encompassing StrategyPanel orientation is GRID.
StrategyPanel/@orientation	Orientation	Y	Declares the orientation of the components (parameters or nested StrategyPanel elements) within a StrategyPanel. Valid values: • HORIZONTAL • VERTICAL • GRID
StrategyPanel/@numRows	non-negative int	N	Number of rows. Applicable when encompassing StrategyPanel orientation is GRID.

Attribute	Type	Req'd	Description
StrategyPanel/@numCols	non-negative int	N	Number of columns. Applicable when encompassing StrategyPanel orientation is GRID.
StrategyPanel/@title	string	N	Title that appears in the panel border.

5.19 VendorConfig Element

Attribute	Type	Req'd	Description
VendorConfig/@legParameters	boolean	N	If true, indicates that this strategy definition is tailored for vendor E/OMSs that support leg-level parameters. If false, indicates that this strategy is tailored for E/OMSs that do not support leg-level parameters.
VendorConfig/@tag66Support	boolean	N	If true, indicates that this strategy definition is tailored for vendor E/OMSs that will deliver an ID linking components of a multi-leg order (the common ID) in ListID(66). If false, the algo provider will not expect a common ID in ListID(66).

6 Type Definitions

The types of the attribute listed in the previous table are defined here. Many of these datatypes have been leveraged from the FIXML schema file fixml-datatypes-5-0-SP2.xsd. Some come from the XML Schema namespace http://www.w3.org/2001/XMLSchema. All others have been defined explicitly within the FIXatdl® schema files.

Type Name	Source	Description
Amt	FIXML	Float value typically representing a Price times a Qty.
anyURI	XML Schema	This datatype represents a URI, which includes web page addresses (commonly called URLs).
boolean	XML Schema	Valid values are “true” and “false”.
Boolean	FIXML	Character field containing one of two values: ‘Y’ (for True/Yes), ‘N’ (for False/No).
Border	FIXatdl®	Enumerated type describing the border of a panel. Valid values are: • None • Line
char	XML Schema	Char value, can include any alphanumeric character or punctuation except the delimiter. All char fields are case sensitive (i.e. m != M). Restricted to the pattern “.{1}”.
Country	FIXML	String representing a country using ISO 3166 Country code (2 characters) values.
Currency	FIXML	String representing a currency type using ISO 4217 Currency code (3 characters) values.
Data	FIXML	String containing raw data with no format or content restrictions. Data fields are always immediately preceded by a length field. The length field should specify the number of bytes of the value of the data field (up to but not including the terminating SOH). Caution: the value of one of these fields may contain the delimiter (SOH) character. Note that the value specified for this field should be followed by the delimiter (SOH) character as all fields are terminated with an SOH. Not applicable to FIXatdl®.
decimal	XML Schema	The XML Schema built-in datatype representing arbitrary precision decimal numbers.
double	XML Schema	The XML Schema built-in datatype, double.
Exchange	FIXML	String representing a market or exchange - ISO 10383 Market Identifier Code (MIC).
int	XML Schema	An integer. May be negative.
language	XML Schema	String identifier for a national language - uses ISO 639-1 standard. Examples: • en (English) • fr (French)
Length	FIXML	Int representing a length in bytes. Value must be positive.
LocalMktTz	FIXatdl®	An enumeration type consisting of the timezone database (or Olson database) codes for various timezones around the world. For example: “Europe/Zurich”. Note

Type Name	Source	Description
		that these codes do not provide GMT offset or daylight savings information.
MonthYear	FIXML	String field representing month of a year. An optional day of the month can be appended or an optional week code. Valid formats: YYYYMM YYYYMMDD YYYYMMWW YYYY = 0000-9999, MM = 01-12, DD = 01-31, WW = w1, w2, w3, w4, w5.
MultipleCharValue	FIXML	String field containing one or more space delimited char values.
MultipleStringValue	FIXML	String field containing one or more space delimited string values.
Orientation	FIXatdl®	Enumerated type describing the orientation of a group of GUI components or controls. Valid values: “HORIZONTAL”, “VERTICAL”, “GRID”.
Percentage	FIXML	Float value representing a percentage (e.g. .05 represents 5% and .9525 represents 95.25%). Note the number of decimal places may vary.
positiveInteger (posint)	XML Schema	An integer greater than or equal to 0.
Price	FIXML	Float value representing a price. Note the number of decimal places may vary. For certain asset classes, prices may be negative values. For example, prices for options strategies can be negative under certain market conditions.
PriceOffset	FIXML	Float value representing a price offset, which can be mathematically added to a “Price”. Note the number of decimal places may vary and some fields such as LastForwardPoints(195) may be negative.
Qty	FIXML	float value capable of storing either a whole number (no decimal places) of “shares” (securities denominated in whole units) or a decimal value containing decimal places for non-share quantity asset classes (securities denominated in fractional units).
SeqNum	FIXML	Int representing a message sequence number. Value must be positive.
string	XML Schema	The string datatype represents character strings in XML.
StringID	FIXatdl®	String with pattern restriction “[A-Za-z][A-Za-z0-9_]{0,255}”.
time	XML Schema	Time specified in the format “hh:mm[:ss]” or “hh:mm[:ss]{+,-}hh:mm”. In the latter format the offset from UTC is provided.
date	XML Schema	The date data type is used to specify a date. The date is specified in the following form “YYYY-MM-DD” where: • YYYY indicates the year • MM indicates the month • DD indicates the day
TZTimestamp	FIXML	String field representing a time/date combination representing local time with an offset to UTC to allow identification of local time and timezone offset of that time. The representation is based on ISO 8601. Format is YYYYMMDD-HH:MM:SS[Z | [+ | - hh[:mm]]] where YYYY = 0000 to 9999, MM = 01-12, DD = 01-31 HH = 00-23 hours, MM = 00-59 minutes, SS = 00-59 seconds, hh = 01-12 offset hours, mm = 00-59 offset minutes Example: 20060901-07:39Z is 07:39 UTC on 1st of September 2006 Example: 20060901-02:39-05 is five hours behind UTC, thus Eastern Time on 1st of September 2006 Example: 20060901-15:39+08 is eight hours ahead of UTC, Hong Kong/Singapore

Type Name	Source	Description
		time on 1st of September 2006 Example: 20060901-13:09+05:30 is 5.5 hours ahead of UTC, India time on 1st of September 2006.
TZTimeOnly	FIXML	String field representing the time represented based on ISO 8601. This is the time with a UTC offset to allow identification of local time and timezone of that time. Format is HH:MM[:SS][Z | [+ | - hh[:mm]]] where HH = 00-23 hours, MM = 00-59 minutes, SS = 00-59 seconds, hh = 01-12 offset hours, mm = 00-59 offset minutes. Example: 07:39Z is 07:39 UTC Example: 02:39-05 is five hours behind UTC, thus Eastern Time Example: 15:39+08 is eight hours ahead of UTC, Hong Kong/Singapore time Example: 13:09+05:30 is 5.5 hours ahead of UTC, India time.
Tenor	FIXML	Pattern used to allow the expression of FX standard tenors in addition to the base valid enumerations defined for the field that uses this pattern data type. This pattern data type is defined as follows: Dx = tenor expression for “days”, e.g. “D5”, where “x” is any integer > 0 Mx = tenor expression for “months”, e.g. “M3”, where “x” is any integer > 0 Wx = tenor expression for “weeks”, e.g. “W13”, where “x” is any integer > 0 Yx = tenor expression for “years”, e.g. “Y1”, where “x” is any integer > 0
UTCDateOnly	FIXML	String Date represented in UTC (Universal Time Coordinated, also known as “GMT”) in YYYYMMDD format. This special-purpose field is paired with UTCTimeOnly to form a proper UTCTimestamp for bandwidth-sensitive messages. Valid values: YYYY = 0000-9999, MM = 01-12, DD = 01-31.
UTCTimeOnly	FIXML	String Time-only represented in UTC (Universal Time Coordinated, also known as “GMT”) in either HH:MM:SS (whole seconds) or HH:MM:SS.sss (milliseconds) format, colons, and period required. This special-purpose field is paired with UTCDateOnly to form a proper UTCTimestamp for bandwidth-sensitive messages. Valid values: HH = 00-23, MM = 00-60 (60 only if UTC leap second), SS = 00-59. (without milliseconds) HH = 00-23, MM = 00-59, SS = 00-60 (60 only if UTC leap second), sss=000-999 (indicating milliseconds).
UTCTimestamp	FIXML	String representing Time/date combination represented in UTC (Universal Time Coordinated, also known as “GMT”) in either YYYYMMDD-HH:MM:SS (whole seconds) or YYYYMMDD-HH:MM:SS.sss (milliseconds) format, colons, dash, and period required. Valid values: YYYY = 0000-9999, MM = 01-12, DD = 01-31, HH = 00-23, MM = 00-59, SS = 00-60 (60 only if UTC leap second) (without milliseconds). YYYY = 0000-9999, MM = 01-12, DD = 01-31, HH = 00-23, MM = 00-59, SS = 00-60 (60 only if UTC leap second), sss=000-999 (indicating milliseconds).

Type Name	Source	Description
		Leap Seconds: Note that UTC includes corrections for leap seconds, which are inserted to account for slowing of the rotation of the earth. Leap second insertion is declared by the International Earth Rotation Service (IERS) and has, since 1972, only occurred on the night of Dec. 31 or Jun 30. The IERS considers March 31 and September 30 as secondary dates for leap second insertion, but has never utilized these dates. During a leap second insertion, a UTCTimestamp field may read “19981231-23:59:59”, “19981231-23:59:60”, “19990101-00:00:00” (see http://tycho.usno.navy.mil/leapsec.html.)

7 Abstract Element Extensions

There are two elements in the schema that are defined as abstract. For example, they cannot be included in a FIXatdl® document without being extended by another element via the XML Schema extension element. All instances of these elements must indicate a derived type that is not abstract via use of the attribute `xsi:type` defined in the namespace http://www.w3.org/2001/XMLSchema-instance.

7.1 Parameter Element Extension

Custom parameters received by an algorithmic order recipient must be of a type known to the recipient. For example, if the recipient is expecting a floating point number in a particular tag then the order sender must make certain that an actual floating point number goes in that tag. FIXatdl® requires that any custom parameter to an algorithm must be of a type defined by the FIX Protocol. So the schema provides a set of complex types that are used to extend the `Parameter` element. These complex types directly correspond to the enumeration type description of StrategyParameterType(959) in the FIX Latest specification.

It is required that each `Parameter` element be extended by setting the attribute `xsi:type` equal to the name of one of the FIXatdl® parameter extension types. An abstract `Parameter` element has the following attributes (which are described in the section *Attribute Definitions*):

- `name`
- `fixTag`
- `use`
- `mutableOnCxlRpl`
- `revertOnCxlRpl`
- `definedByFIX`
- `filter`
- `scope`
- `xsi:type`

When the `Parameter` element is extended it gains several more attributes depending on the element to which it is extended.

The types of these attributes are also dependent on the extended element and may vary from one `Parameter` element to another.

The following table presents the `xsi:type` names, the expected data type of the wire-value and the extended attributes that apply only to the specific parameter extension type.

Parameter xsi:type	Corresponding FIX Types	Attribute Name[1]	Attribute Type[2]
Amt_t	Amt	minValue maxValue constValue	decimal decimal decimal
Boolean_t	Boolean	trueWireValue[3] falseWireValue[4] constValue	string string boolean
Char_t	char	constValue	char
Country_t	Country	constValue	Country

[1] Extended attributes specific to xsi:type
[2] Extended attributes specific to xsi:type
[3] Deprecated
[4] Deprecated

Parameter xsi:type	Corresponding FIX Types	Attribute Name[1]	Attribute Type[2]
Currency_t	Currency	constValue	string
Data_t	data	minLength maxLength constValue	Length Length Data
Exchange_t	Exchange	constValue	Exchange
Float_t	float	minValue maxValue constValue precision	decimal decimal decimal non-negative int
Int_t	int	minValue maxValue constValue	int int int
Language_t	Language	constValue	language
Length_t	Length	constValue	positiveInteger
LocalMktDate_t	LocalMktDate	minValue maxValue constValue	LocalMktDate LocalMktDate LocalMktDate
MonthYear_t	month-year	minValue maxValue constValue	MonthYear MonthYear MonthYear
MultipleCharValue_t	MultipleCharValue	minLength maxLength constValue invertOnWire	Length Length MultipleCharValue boolean
MultipleStringValue_t	MultipleStringValue	minLength maxLength constValue invertOnWire	Length Length MultipleStringValue boolean
NumInGroup_t	NumInGroup	constValue	positiveInteger
Percentage_t	Percentage	minValue maxValue constValue multiplyBy100	Percentage Percentage Percentage boolean
Price_t	Price	minValue maxValue constValue precision	Price Price Price non-negative int
PriceOffset_t	PriceOffset	minValue maxValue constValue precision	PriceOffset PriceOffset PriceOffset non-negative int
Qty_t	Qty	minValue maxValue constValue precision	Qty Qty Qty non-negative int
SeqNum_t	SeqNum	constValue	positiveInteger

Parameter xsi:type	Corresponding FIX Types	Attribute Name[1]	Attribute Type[2]
String_t	string	minLength maxLength constValue	Length Length string
TagNum_t	int	constValue	positiveInteger
Tenor_t	Tenor	constValue	Tenor
UTCDateOnly_t	UTCDateOnly	minValue maxValue constValue	UTCDateOnly UTCDateOnly UTCDateOnly
UTCTimeOnly_t	UTCTimeOnly	minValue maxValue constValue	time time time
UTCTimestamp_t	UTCTimestamp	minValue maxValue constValue localMktTz	time time time LocalMktTz
TZTimestamp_t	TZTimestamp	minValue maxValue constValue	time time time
TZTimeOnly_t	TZTimeOnly	minValue maxValue constValue	TZTimeOnly TZTimeOnly TZTimeOnly

For example, in the following code snippet an algorithmic parameter, “MktOnCloseFlag”, is defined as being a “Boolean_t” type.

```
<Parameter name="MktOnCloseFlag" xsi:type="Boolean_t" fixTag="28001" use="required"
trueWireValue="T" falseWireValue="F"/>
```

Notice that by setting `xsi:type` of this parameter to “Boolean_t”, the attributes `trueWireValue` and `falseWireValue`, which are members of the derived element and accept standard XML string values, can now be used.

[Please note that the previous example shows two attributes which have been deprecated, `Parameter/@trueWireValue` and `Parameter/@falseWireValue`. The intention was to illustrate how extended elements are used.]

In this next snippet a quantity parameter is defined.

```
<Parameter name="CrossQty" xsi:type="Qty_t" fixTag="28002" use="required" minValue="100"/>
```

By setting `xsi:type` to “Qty_t”, a value for `minValue` can be provided.

7.2 Control Element Extension

As with extensions to the `Parameter` element, FIXatdl® provides a set of elements that are derived from the `Control` element. Each of these elements inherits the attributes of the `Control` element. They also have their own distinct attributes.

An abstract `Control` element has the following attributes (which are described in the section *Attribute Definitions*):

- `ID`
- `parameterRef`
- `label`

- `initFixField`
- `initPolicy`
- `tooltip`
- `disableForTemplate`
- `filter`
- `xsi:type`

When the `Control` element is extended it gains several more attributes depending on the element to which it is extended.

The types of these attributes are also dependent on the extended element and may vary from one `Control` element to another.

The following types are used to extend the `Control` element:

Control xsi:type	Description of desired control	Attribute Name[5]	Attribute Type[6]
Clock_t	Clock with hours, minutes, seconds and AM/PM setting. Depending on the parameter type, this control may optionally also display a date selector.	initValue initValueMode localMktTz enablingControlType disablingControlType disablingControlLabel displayableDate displayableTimeZone editableTimeZone	time int localMktTz_t string string string string boolean boolean
TextField_t	Standard text field.	initValue	string
SingleSelectList_t	Affords the user the ability to select one item from a list.	initValue	string
MultiSelectList_t	Affords the user the ability to select many items from a list. Values extracted from this type of control are expected to be transmitted using a MultipleStringValue or MultipleCharValue FIX type.	initValue	MultipleStringValue
Slider_t	Draggable slider with labels that map to values.	initValue increment incrementPolicy	string double string
CheckBox_t	Standard check box – initialized to checked or unchecked.	initValue checkedEnumRef uncheckedEnumRef	boolean string string
CheckBoxList_t	A list of check boxes where multiple selections can be made. Values extracted from this type of control are expected to be	initValue	MultipleStringValue

[5] Attributes specific to `xsi:type`.
[6] Attributes specific to `xsi:type`.

Control xsi:type	Description of desired control	Attribute Name[5]	Attribute Type[6]
	transmitted using a MultipleStringValue or MultipleCharValue FIX type.	orientation	Orientation
SingleSpinner_t	A numeric field that has arrows to increment and decrement	initValue increment incrementPolicy	double double string
DoubleSpinner_t	A numeric field that has two sets of arrows to increment and decrement by different values (say for pennies and dollars). When pressed, the right-most pair of arrows will increment (or decrement) the value of the control by the value of outerIncrement. Pressing the other pair of arrows will cause the value to be incremented (or decremented) by the value of innerIncrement.	initValue innerIncrement innerIncrementPolicy outerIncrement outerIncrementPolicy	double double string double string
DropDownList_t	More specific derivation of a SingleSelectList. E.g., a combo box.	initValue	string
EditableDropDownList_t	More specific derivation of a SingleSelectList. E.g., an editable combo box.	initValue	string
RadioButton_t	Standard radio button, but with no associated group.	initValue radioGroup checkedEnumRef uncheckedEnumRef	boolean string string string
RadioButtonList_t	More specific derivation of a SingleSelectList. Several items are presented with an associated radio button where the user can select only one of them.	initValue orientation	string Orientation
Label_t	Plain text. Note that the label control's text may be updated through the execution of a StateRule.	initValue	string

For example, in the following code snippet a control, "StartTimeCntl", is defined as being a "Clock_t". An initial value of "09:30" has been specified.

```
<Control ID="StartTimeCntl" xsi:type="lay:Clock_t" label="Start Time" initValue="09:30"
localMktTz="America/New_York" parameterRef="StartTime"/>
```


8 Dependencies and Structural Constraints beyond XML Schema

While W3C XML Schema is useful for describing the structure of an XML-based language, it still has its limitations. For example, it only allows the specification of whether attributes are required or optional. Furthermore, there is no way to specify more complex constraints between attributes or between attributes or elements.

With this in mind the following table presents further constraints to which XML document instances must conform if they are to be FIXatdl® compliant.

ID	Affected Elements	Affected Attributes	Description
1	Edit	logicOperator, operator	Within an Edit element, the attributes operator and logicOperator are mutually exclusive.
2	Edit	field2, value	Within an Edit element, the attributes field2 and value are mutually exclusive.
3	StrategyPanel		A StrategyPanel element cannot have as child elements both Control elements and StrategyPanel elements.
4	Edit	field, field2	Within an Edit element the attributes field and field2 must refer to either a pre-declared parameter name or a standard FIX field name (taken from the FIX specification) pre-pended with the string “FIX_”.
5	Edit	value	Within an Edit element, the type of the value attribute must safely match with the type of parameter specified by the field attribute.
6	Edit	logicOperator, operator	If an Edit element is a child of another Edit element then the parent Edit element must have its logicOperator attribute defined and its operator attribute undefined.
7	Edit	field1, field2	When a comparison is made between two operands, the values of the operands must either be of the same type or be able to be converted in such a way so that the resulting converted types are the same.
8	Control	parameterRef	If Control/@parameterRef is defined it must be equal to the name attribute of one of the defined Parameter elements.
9	EnumPair ListItem	enumID	If a Control is linked to a Parameter via use of Control/@parameterRef, and the Control contains ListItem elements, then the Parameter must contain EnumPair elements. Furthermore, each of the Control’s ListItem/@enumID values must match one and only one of the Parameter element’s EnumPair/@enumID values.
10	Control	checkedEnumRef uncheckedEnumRef	If values for Control/@checkedEnumRef or Control/@uncheckedEnumRef are provided then Control/@parameterRef must also be provided. Furthermore, the values of Control/@checkedEnumRef and Control/@uncheckedEnumRef each must be equal to one of the EnumPair/@enumID values of the Parameter element referred to by Control/@parameterRef.

9 A Sample FIXatdl® Document

The following listing shows a FIXatdl® instance document describing one strategy with six parameters. The associated controls to be rendered are aligned horizontally within two panels which are, in turn, are vertically aligned. Three validation rules are provided.

```
<Strategies
    xmlns="http://www.fixprotocol.org/FIXatdl-1-2/Core"
    xmlns:val="http://www.fixprotocol.org/FIXatdl-1-2/Validation"
    xmlns:lay="http://www.fixprotocol.org/FIXatdl-1-2/Layout"
    xmlns:flow="http://www.fixprotocol.org/FIXatdl-1-2/Flow"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.fixprotocol.org/FIXatdl-1-2/Core fixatdl-core-1-2.xsd"
    strategyIdentifierTag="27620"
    versionIdentifierTag="27621">

    <Strategy name="Tazer1" uiRep="Tazer" wireValue="Tazer" version="1" fixMsgType="D"
      providerID="ABC">

        <!--
            Declare the algorithm to be applicable in The U.S., Canada and the UK.
        -->
        <Regions>
            <Region name="TheAmericas" inclusion="Include">
                <Country CountryCode="US" inclusion="Include"/>
                <Country CountryCode="CA" inclusion="Include"/>
            </Region>
            <Region name="EuropeMiddleEastAfrica" inclusion="Include">
                <Country CountryCode="UK" inclusion="Include"/>
            </Region>
        </Regions>

        <!--
            Declare the markets where order may be executed.
        -->
        <Markets>
            <Market MICCode="BATS" inclusion="Include"/>
            <Market MICCode="NYSE" inclusion="Include"/>
            <Market MICCode="XTSE" inclusion="Include"/>
            <Market MICCode="LSE" inclusion="Include"/>
        </Markets>

        <!--
        This algorithm will be applied to equity common stock.
        -->
        <SecurityTypes>
            <SecurityType name="CS" inclusion="Include"/>
        </SecurityTypes>

        <!--
            Parameter declarations

            Five parameters are declared here. The order recipient may reject
            orders with: EndTime(7603) values greater than 4pm New York time;
            SweepDistribution(7640) values other than "U" or "G"; Variance(7641)
            values outside the range [0.01, 0.50]; and DisplayQty(7645)
            values less than 0.
        -->
        <Parameter name="StartTime" xsi:type="UTCTimestamp_t" fixTag="27602" use="required"/>
        <Parameter name="EndTime" xsi:type="UTCTimestamp_t" fixTag="27603" use="required"
         maxValue="16:00:00" localMktTz="America/New_York "/>
        <Parameter name="DisplayQty" xsi:type="Int_t" fixTag="27645" use="optional"
         minValue="0"/>
        <Parameter name="SweepDistribution" xsi:type="Char_t" fixTag="27640" use="required">
            <EnumPair enumID="e_Uniform" wireValue="U"/>
```



```
        <EnumPair enumID="e_Gaussian" wireValue="G"/>
    </Parameter>
    <Parameter name="Variance" xsi:type="Float_t" fixTag="27641" use="optional"
     minValue="0.01" maxValue="0.50"/>
    <Parameter name="AllowDarkPoolExec" xsi:type="Char_t" fixTag="27642" use="required">
        <EnumPair enumID="e_True" wireValue="T"/>
        <EnumPair enumID="e_False" wireValue="F"/>
    </Parameter>

    <!--
        Description and Layout of GUI controls
    -->
    <lay:StrategyLayout>
        <lay:StrategyPanel orientation="VERTICAL">
            <lay:StrategyPanel orientation="HORIZONTAL">
                <!--
                    The StartTimeClock control will be initialized to 9:30am (New
                    York time). If it is past 9:30am when the control is rendered,
                    then it will be initialized with the current time.

                    Note that the user will see the 9:30am New York time rendered
                    according to his/her environment's local timezone setup.
                -->
                <lay:Control xsi:type="lay:Clock_t" ID="StartTimeClock" label="Start Time"
                parameterRef="StartTime" initValue="09:30:00" localMktTz="America/New_York"
                initValueMode="1"/>

                <!--
                  The EndTimeClock control is not initialized.
                -->

                <lay:Control xsi:type="lay:Clock_t" ID="EndTimeClock" label="End Time"
                parameterRef="EndTime"/>

                <!--
                    The next control is not bound to any parameter. It is intended to
                    direct the behavior of the DisplayQty control. It presents 3
                    options in a drop-down list.
                -->

                <lay:Control ID="DQHandling" xsi:type="lay:DropDownList_t"
                label="Display Handling">
                    <lay:ListItem enumID="choice1" uiRep="Send nothing"/>
                    <lay:ListItem enumID="choice2" uiRep="Send 0"/>
                    <lay:ListItem enumID="choice3" uiRep="Send what user enters"/>
                </lay:Control>

                <!--
                    The DisplayQty control is bound to the DisplayQty parameter. The
                    control is un-initialized when it is first rendered. Its
                    subsequent behavior is directed by DQHandling control. When
                    DQHandling's choice1 is selected DisplayQty will revert to an
                    un-initialized state and become disabled. When DQHandling's
                    choice2 is selected, DisplayQty's value will be set to 0 and
                    it will become disabled. When DQHandling's choice3 is selected,
                    DisplayQty will be enabled and will accept user input.
                -->

                <lay:Control xsi:type="lay:TextField_t" ID="DisplayQty" label="Display Qty"
                parameterRef="DisplayQty">
                    <flow:StateRule enabled="true">
                      <val:Edit field="DQHandling" operator="EQ" value="choice3"/>
                    </flow:StateRule>
                    <flow:StateRule value="{NULL}">
                        <val:Edit field="DQHandling" operator="EQ" value="choice1"/>
                    </flow:StateRule>
                    <flow:StateRule value="0">
```



```
                        <val:Edit field="DQHandling" operator="EQ" value="choice2"/>
                    </flow:StateRule>
                </lay:Control>
            </lay:StrategyPanel>
            <lay:StrategyPanel orientation="HORIZONTAL">
                <!--
                    The SweepDist control will present the 2 options corresponding to
                    the enumPairs of the SweepDistribution parameter.
                -->
                <lay:Control ID="SweepDist" xsi:type="lay:DropDownList_t"
                label="Sweep Distribution"
                parameterRef="SweepDistribution" initValue="Uniform">
                    <lay:ListItem enumID="e_Uniform" uiRep="Uniform"/>
                    <lay:ListItem enumID="e_Gaussian" uiRep="Gaussian"/>
                </lay:Control>
                <!--
                    The Variance control is enabled only when SweepDist's e_Gaussian
                    item is selected.
                -->
                <lay:Control xsi:type="lay:SingleSpinner_t" ID="Variance" label="Variance"
                parameterRef="Variance">
                    <flow:StateRule enabled="true">
                        <val:Edit field="SweepDist" operator="EQ" value="e_Gaussian"/>
                    </flow:StateRule>
                </lay:Control>
            </lay:StrategyPanel>
            <lay:StrategyPanel orientation="HORIZONTAL">
                <lay:Control xsi:type="lay:CheckBox_t" ID="DPOption"
                label="Allow Dark Pool Execution" parameterRef="AllowDarkPoolExec"
                checkedEnumRef="e_True" uncheckedEnumRef="e_False">
                </lay:Control>
            </lay:StrategyPanel>
        </lay:StrategyPanel>
    </lay:StrategyLayout>

    <!--
        Validation Section

        Note that the attribute, field, always refers to a Parameter name and
        not a Control ID. Also note that short-circuit evaluation is fully
        exploited.
    -->
    <val:StrategyEdit errorMessage="End Time should be later than Start Time">
        <val:Edit field="EndTime" operator="GT" field2="StartTime"/>
    </val:StrategyEdit>

    <val:StrategyEdit errorMessage="Variance is required when Sweep Distribution is
     Gaussian.">
        <val:Edit logicOperator="OR">
            <val:Edit field="SweepDistribution" operator="NE" value="G"/>
            <val:Edit logicOperator="AND">
                <val:Edit field="SweepDistribution" operator="EQ" value="G"/>
                <val:Edit field="Variance" operator="EX"/>
            </val:Edit>
        </val:Edit>
    </val:StrategyEdit>

    <val:StrategyEdit errorMessage="Variance must be between 0 and 2.0">
        <val:Edit logicOperator="OR">
            <val:Edit field="SweepDistribution" operator="NE" value="G"/>
            <val:Edit logicOperator="AND">
                <val:Edit field="SweepDistribution" operator="EQ" value="G"/>
                <val:Edit field="Variance" operator="EX"/>
                <val:Edit field="Variance" operator="GT" value="0.0"/>
                <val:Edit field="Variance" operator="LT" value="2.0"/>
            </val:Edit>
```



```
            </val:Edit>
        </val:StrategyEdit>
    </Strategy>
</Strategies>
```


Appendix 1 - LocalMktTz Type

The valid values of attributes of the type LocalMktTz can be found at IANA. In the FIXatdl® schema a simple type, "LocalMktTz_t", has been defined as a string which is restricted to the zone names of the TZ environment variable.

