Business rules configuration

This section describes Groovy message routing and event processing rules.

Overview

FIX Edge Java provides the `RoutingRule` unit as an abstraction for an internal message routing element. FEJ supports pure Java and Groovy implementations for routing rules.

The Groovy routing rules are a flexible mechanism that allows implementing custom logic for messages transformation and handling events.

The rules can be embedded in the **rules.groovy** file:

rules.groovy

```
[
         messageRule(
                // rule description
                "some Rule",
                //source filter - ignore for this rule
                null,
                // context filter - apply this rule for New Order - Single (D) messages
                { ctx -> ctx.getMessage().getTagValueAsString(35) == "D" } as RuleCondition,
                // action for rule - resend message to all session within same group
                // and stop message processing
                { ctx ->
                    routingContext.getDestinationsByGroup(ctx.sourceParams.groups).each { adapter ->
                        adapter.send(ctx.message)
                        ctx.exit()
                    }
                } as RuleAction),
                eventRule("Catching session events", FIXSessionStateEvent.class, {
                appEvent ->
                    return true //do nothing but there can be additional logic
                    },
                    {
                        sessionStateEvent ->
                            def sessionName = sessionStateEvent.getSessionId()
                            logger.info("Session '{}' has been started.", sessionName)
                    }
        )
]
```

The example represents two different types of rules, namely, *MessageRoutingRule* and *EventRoutingRule*.

"Helpers" methods for routing rules

There is a static class with methods that are named "helpers". These methods help an end-user make logic shorter by encapsulating details inside the methods.

Name	Definition	Description
generateUniqueId	String generateUniqueId()	Returns UUID that is generated using a cryptographically strong pseudo-random number generator. Example: `def id = generateUniqueId()`

stringValue	String stringValue(FIXFieldList msg, int tag)	Returns a string representation of the value provided by the tag. Example: `def destination = stringValue(msg, Header.DeliverToCompID)`
messageRule	RoutingRule messageRule(String description, SourceCondition sourceCondition, RuleCondition condition, RuleAction action)	Short version for creating MessageRoutingRule.

messageRule	RoutingRuleBuilder messageRule(String description)	Rule builder for creating MessageRoutingRule.
		
		`messageRule(description)`
		`.sourceCondition(params -> {....})`
		`.condition(msgCtx -> {....})`
		`.action(msgCtx -> {....})`
		`.build()`

eventRule	EventRoutingRuleBuilder eventRule(String description)	Rule builder for creating EventRoutingRule.
		
		`eventRule(description)`
		`.eventType(eventType)`
		`.condition(event -> {....})`
		`.action(event -> {....})`
		`.build()`

eventRule	RoutingRule eventRule(String description, Class<AppEvent> eventType, Predicate<AppEvent> ruleCondition, Consumer<AppEvent> ruleAction)	Short version for creating EventRoutingRule.
removeRepGroup	void removeRepGroup(FIXFieldList msg, int leadingTag, int groupTags[])	Removes the list of tags that corresponds to the repeating group. Example: `removeRepGroup(msg, 453, [452, 448, 447])`
send	void send(RoutingContext rc, RuleContext ruleContext, String targetCompId)	Sends a message to the session by the provided target CompId. Example: `send(routingContext, msgCtx, destination)`
send	void send(RoutingContext rc, RuleContext ruleContext, String targetCompId, String qualifier)	Sends a message to the session by the provided target CompId and sessionQualifier. Example: `send(routingContext, ctx, destination, qualifier)`

Import of the classes

It is not necessary to import classes directly in the **groovy.rules** file. Imported classes are described in **sysconf/fej-routing.xml**. The values below are predefined and can be extended by custom ones not to specify imports in rules.

sysconf/fej-routing.xml

```
        <util:list id="imports" value-type="java.lang.String">
        <value>com.epam.fej.context.Storage</value>
        <value>com.epam.fej.context.StorageManager</value>
        <value>com.epam.fej.routing.RoutingContext</value>
        <value>com.epam.fej.routing.MessageEventPool</value>
        <value>com.epam.fej.routing.rules.RoutingRule</value>
        <value>com.epam.fej.routing.rules.RuleAction</value>
        <value>com.epam.fej.routing.rules.RuleCondition</value>
        <value>com.epam.fej.routing.rules.SourceCondition</value>
        <value>com.epam.fej.server.fix.event.FIXSessionStateEvent</value>
        <value>com.epam.fej.server.fix.event.NewSessionEvent</value>
        <value>com.epam.fej.routing.event.RuleErrorEvent</value>
        <value>com.epam.fej.scheduling.event.SchedulerEvent</value>
        <value>com.epam.fej.server.fix.event.ServerStateEvent</value>
        <value>com.epam.fej.routing.endpoint.snf.events.SnFEvent</value>
        <value>com.epam.fej.routing.event.UnprocessedMessageEvent</value>
    </util:list>
```

Injected properties

These beans are available for use in the Groovy rule.

sysconf/fej-routing.xml

```
<property name="additionalProperties">
        <map key-type="java.lang.String">
            <entry key="routingContext" value-ref="routingContext"/>
        <entry key="storageManager" value-ref="storageManager"/>
        <entry key="storageManager" value-ref="messageEventPool"/>
        </map>
</property>
```

Additional info

- Message routing
    - In-memory data context
- Event processing