MQ Rate Importer Configuration

Review & Approval			
	Name	**Signature**	**Date**
Author	Rob McConnell		08/02/05
Reviewer	Dick Buttler		
Approval	Joe Locke		

Issue History			
Version	**Date**	**Changes since previous version**	**Release**
1.0		First version	

Contents

Introduction

This document details what is required to configure rate import. For purposes of example the system is siena/test and the installation directory is c:\siena for windows and /home/user1/siena for UNIX. Within this document the symbol <- indicates a line wrap, so:

```
Abc=very long s
<-      tring
```

Should be read as:

```
Abc=very long string
```

The following configuration files require editing.

1. twist_TwistContainer.properties
2. MQRate.properties
3. services.properties.
4. MQ.properties
5. externalInterface.properties

All these files should be present in c:\siena\std\systems\siena\test\config

MQ.properties

There must exist at least one queue group to read rates from. If each rate type (FX, MM, CALL and SPOT FIXING) are to be on different queues then a queue group for each type must be created. The following example assumes there are three queue groups already configured (for deal export). Assuming there will be only one queue for all rate types, an entry in MQ.properties should be made as follows:

```
group.4=RateImport
# The host name on which the MQ Manager is running
server.RateImport.hostname=EBS11
# Port on which the MQ manager is listening
server.RateImport.port=1414
# The channel used for MQ Communication
channel.RateImport.name=CHANNEL1
channel.RateImport.userName=
channel.RateImport.password=
# The QueueManager name
queueManager.RateImport.name=venus.queue.manager
# Must be set although is unused
queue.RateImport.output.name=UNUSED
# The queue used for reading rate messages
queue.RateImport.input.name=RATEIMPORT.QUEUE
queue.RateImport.input.failureMode=1
queue.RateImport.input.DLQ=
queue.RateImport.utf=n
queue.RateImport.inputOpenOptions.static.0=MQOO_INPUT_AS_Q_DEF
queue.RateImport.inputOpenOptions.static.1=MQOO_BROWSE
queue.RateImport.inputOptions.static.0=MQGMO_WAIT
queue.RateImport.outputOpenOptions.static.0=MQOO_OUTPUT
```

Entries in bold are those that need to be edited.

In addition a queue group must be defined for the MQ subscription queue as follows.

```
group.5=Subscribe
# The host name on which the MQ Manager is running
server.Subscribe.hostname=EBS11
# Port on which the MQ manager is listening
server.Subscribe.port=1414
# The channel used for MQ Communication
channel.Subscribe.name=CHANNEL1
channel.Subscribe.userName=
channel.Subscribe.password=
# The QueueManager name
queueManager. Subscribe.name=venus.queue.manager
# The subscription queue name
queue.Subscribe.output.name=WLB.SUBSCRIBE.QUEUE
# Unused but must be set
queue.Subscribe.input.name=UNUSED
queue.Subscribe.input.failureMode=1
queue.Subscribe.input.DLQ=
queue.Subscribe.utf=n
queue.Subscribe.inputOpenOptions.static.0=MQOO_INPUT_AS_Q_DEF
queue.Subscribe.inputOpenOptions.static.1=MQOO_BROWSE
queue.Subscribe.inputOptions.static.0=MQGMO_WAIT
queue.Subscribe.outputOpenOptions.static.0=MQOO_OUTPUT
```

Note that for both the above two queue definitions if you wish to test them using the MQClient.bat/MQClient.sh tool, you must ensure both the input queue and output queue are both set to the same queue name. For example to write a test message to the RateImport queue group the **queue.RateImport.output.name** must be correctly set.

twist_TwistContainer.properties

The MQ rate importer requires that an external interface and Twist application be configured.

External System

The following describes the external system definition required for the rate import process. It assumes there are already three external interfaces configured

```
twist.external.interface.4=RateImporter
# The translator from external format to internal Twist
twist.external.interface.RateImporter.translator.class=tantus.tdsnt.external
<-     .implementation.MQRates.Translator
# Errors in the Rate import are not thrown to the MQ Reader
twist.external.interface.RateImporter.throwUserError=n
# Only start the external interface when the import application is started
twist.external.interface.RateImporter.lazyLoad=y
# Don't log rate messages
twist.external.interface.RateImporter.doLog=n
# Parameters to the import process
# The MQ configuration file
twist.external.interface.RateImporter.parameter.1=system=MQ
# The queue group to use
twist.external.interface.RateImporter.parameter.2=queueGroup=RateImport
# The types of rates for this external interface to process
twist.external.interface.RateImporter.parameter.3=rateTypes=fx,ir
```

The last entry in the above example indicates what rate types are to be processed by this external system definition. In this case only FX and Interest Rates are specified. The full rate type list is as follows.

- FX Type = fx
- MM Type = ir
- Call Type = call
- Spot Fixing = sf

For the RateImporter definition above to process all rate types the last line should be amended to:

```
twist.external.interface.RateImporter.parameter.3=rateTypes=fx,ir,call,sf
```

Twist Application definition

The following describes the configuration required for the application that actually imports the MQ rates into Siena. It assumes there is a Twist application already configured.

```
twist.application.2=RateImporter
# The MQ rate import implementation
twist.application.RateImporter.class=tantus.tdsnt.twist.
<-     implementation.siena.rateimporter.TwistRateImporter
# The interface to associate with this application
twist.application.RateImporter
<-     .interface.name.1=RateImporter
```

externalInterface.properties

This file provides the mapping between external interface names and their implementations. For Rate import it should contain:

```
external.interface.name=RateImporter
# The external implementation that reads rate messages from MQ

external.interface.RateImporter=tantus.tdsnt.external.
<-     implementation.MQRates.ExternalRateImporter
```

services.properties

Make sure the following entry exists in the services.properties file. This entry accommodates both the deal export service and the rate importer.

```
# The name of the service
service13 = Twist Container
# Default host on which to run
service13.defaultLoc={hostName}
# Unique Id
service13.stream =TwistContainer
service13.args = LogFile|{logPath}/TwistContainer.log, Jvm|{javaExe}
<-     -Dsiena.working={systemPath}/data -Xms8M -Xmx16M, ProcessList|Siena
<-     Deal Export Handler=True*Rate Importer=True
# Applications contained. These are the applications available to
# start from within the Services Manager
# The Deal Exporter
service13.TwistContainer.application.0=SienaDealExportHandler
```

```
# The Rate Importer
service13.TwistContainer.application.1=RateImporter
```

MQRates.properties

This configuration file is responsible for defining the nature of an MQ rate for the four possible rate types. The following edits will be required:

```
# The following four items define the mapping between the topic and
# the siena rate type.
# They should be set to the values within the topic.
# For example for the spot topic Reuters/realtime/fx-spot/USDARS/spot
# the fx-spot part of the topic must appear in the list for
# topic.type.names.fx
topic.type.names.fx=fx-outright,fx-spot
topic.type.names.ir=interestrate
topic.type.names.call=callrate
topic.type.names.sf=spot_fixing

# The subscription type. These may be changed to snapshot if so desired
subscribe.type.fx=subscribe
subscribe.type.sf=subscribe
subscribe.type.call=subscribe
subscribe.type.ir=subscribe
```

```
# This should be set to the queue group name
# defined in MQ.properties for the MQ
# subscription queue
subscribe.queue=Suscribe

# Each rate type has a polling time defined in milliseconds
# FX
topic.polltime.fx=2000
# Spot Fixing
topic.polltime.sf=5000
# Call
topic.polltime.call=3000
# Interest rates
topic.polltime.ir=5000
```

Creating separate Rate Queues

The above examples have assumed that all rate types will be delivered on the same MQ queue. It is possible to subscribe such that certain rate types are delivered on different queues. For example call rates may be given a queue of their own. The following example shows how this is done.

The steps are as follows:

1. Create a new queue group in MQ.properties
2. Create a new External Interface definition in twist_TwistContainer.properties associated with the new queue group

3. Associate the new external interface with the Twist Application rate import definition.

MQ.properties

The example below shows a queue definition for a Call rates queue

```
group.6=CallRates
# The host name on which the MQ Manager is running
server.CallRates.hostname=EBS11
# Port on which the MQ manager is listening
server.CallRates.port=1414
# The channel used for MQ Communication
channel.CallRates.name=CHANNEL1
channel.CallRates.userName=
channel.CallRates.password=
# The QueueManager name
queueManager.CallRates.name=venus.queue.manager
# Must be set although is unused
queue.CallRates.output.name=UNUSED
# The queue used for reading rate messages
queue.CallRates.input.name=WLB.CALLRATES.QUEUE
queue.CallRates.input.failureMode=1
queue.CallRates.input.DLQ=
queue.CallRates.utf=n
queue.CallRates.inputOpenOptions.static.0=MQOO_INPUT_AS_Q_DEF
queue.CallRates.inputOpenOptions.static.1=MQOO_BROWSE
queue.CallRates.inputOptions.static.0=MQGMO_WAIT
queue.CallRates.outputOpenOptions.static.0=MQOO_OUTPUT
```

externalInterface.properties

A new entry should be added to the externalInterface.properties file as below.

```
external.interface.name=CallRateImporter
external.interface.RateImporter=tantus.tdsnt.external.implementation.MQRates
<-     .ExternalRateImporter

This defines the new external interface and specifies its implementation.
```

twist_TwistContainer.properties

An entry must be made in the twist_TwistContainer.properties file to create an instance of the external interface defined above:

```
# The name for this definition
twist.external.interface.5=CallRateImporter
# The Translator to use on the XML
twist.external.interface.CallRateImporter.translator.class=tantus.tdsnt.exte
<-     rnal.implementation.MQRates.Translator
# Don't throw errors generated within the import process to
# the external interface
twist.external.interface.CallRateImporter.throwUserError=n
# Start importing as soon as the Twist Manager is loaded
twist.external.interface.CallRateImporter.lazyLoad=n
# Don't log rate messages
twist.external.interface.CallRateImporter.doLog=n
# The properties file for MQ configuration
```

```
twist.external.interface.CallRateImporter.parameter.1=system=MQ
# The queue group configured
twist.external.interface.CallRateImporter.parameter.2=queueGroup=CallRates
# The list of rate types to process. In this case just 'call'
twist.external.interface.CallRateImporter.parameter.3=rateTypes=call
```

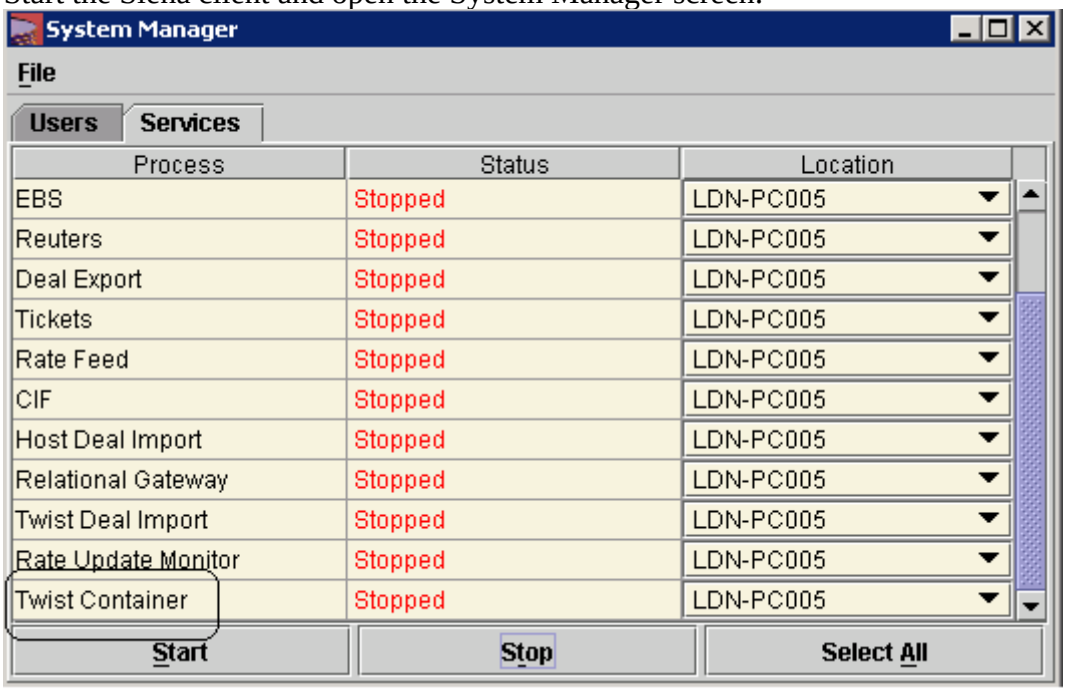
In order for Siena to import rates from this new interface the following change must be made to the Twist Application **RateImporter**. It should be amended to the following:

```
twist.application.2=SienaRateImporter
# The MQ rate import implementation
twist.application. SienaRateImporter.class=tantus.tdsnt.twist.
<-     implementation.siena.rateimporter.TwistRateImporter
# The interface to associate with this application
twist.application.SienaRateImporter
<-     .interface.name.1=RateImporter
# The new interface
twist.application.SienaRateImporter
<-     .interface.name.2=CallRateImporter
```

Note how the last line associates the new external interface with the Siena rate importer.

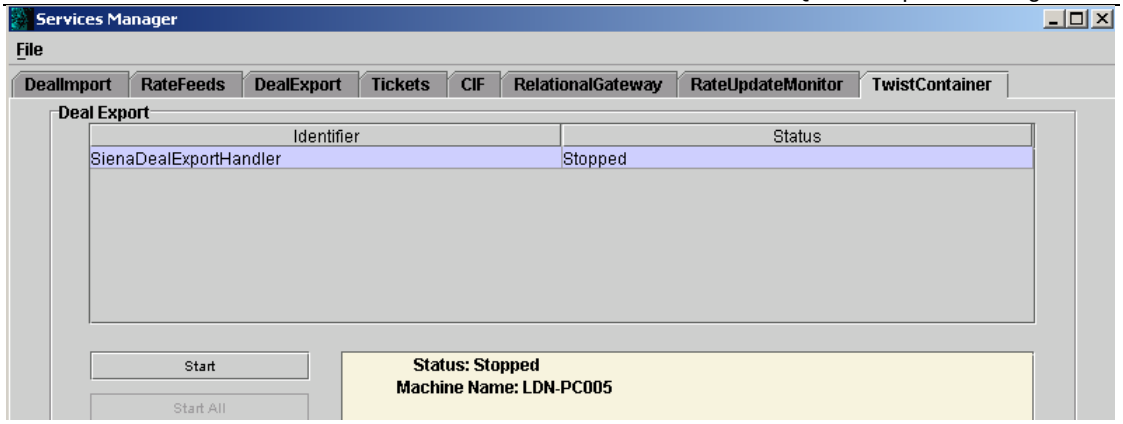
Starting the Rate Importer

Start the Siena client and open the System Manager screen.



Select the Twist container row and click on start. The Twist Manager will begin reading the MQ queues immediately but rates will not be imported into Siena until the rate importer has been started within the Services Manager screen.

To start importing rates, open the Services Manager screen and click on the Twist Container tab.



A row should be visible representing the Deal Export service called SienaDealExportHandler. Select it and click on the start button. Deals should now be available for export to MQ.