Sterling Integrator®

FIFO Message Processing

Version 5.1

Sterling Commerce
An IBM Company

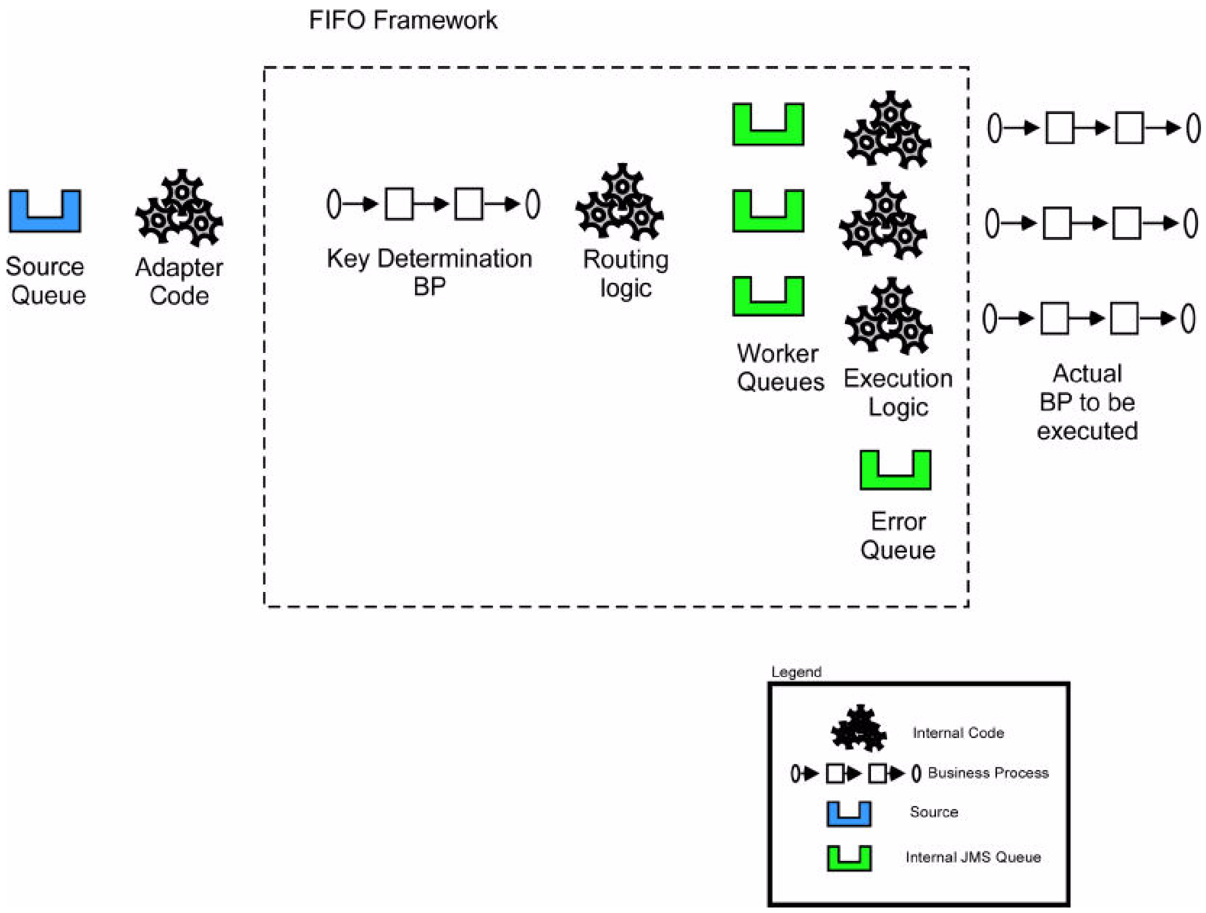
FIFO Message Processing

Sterling Integrator supports ordered processing of files and messages for the following adapters:

- JMS Queue adapter
- JMS Topic adapter
- MSMQ adapter

The ordered processing in Sterling Integrator is processed by the FIFO (first in first out) framework.

The following figure demonstrates the FIFO framework:



Sterling Integrator supports FIFO processing of messages through adapters. The messages passed to the FIFO framework are first executed through a specialized routing key initialization business process that returns a single string value known as the routing key. The routing logic is then applied, which places all the messages with equal keys on the same internal routing queue. Messages with different routing key values process in parallel. Messages with the same routing key value maintain FIFO ordering. Each queue to user specified business process processes the message and waits for the business process to end the metadata describing the errant process, then processes the next message. If an error is encountered while processing the messages, metadata describing the errant process are routed to an error queue. Thereafter, the message processing continues.

Configuring FIFO Services

To configure FIFO services:

1. Login to Sterling Integrator.
2. Select **Deployment** > **Services** > **Configuration**.
3. Create new service and click **Go**.
4. In the Service Type field, enter the applicable adapter you want to use and click **Next**. You can also select it from the Tree View or List View.
5. Enter a suitable name and description in **Name** and **Description** fields.
6. Select or create a new group if required. By default, it is None.
7. Select the business process you want to execute.

   Note: This business process must be set to use at least Minimal Event Processing and cannot be set to Error Only persistence level.
8. Select **FIFO** from Processing Mode drop-down list and click **Next**.
9. Select the business process that will receive the message and returns the routing key from the **FIFO Route Lookup BP** drop-down list.

   Note: You should create a business process and import it into Sterling Integrator.
10. Review and click **Finish**. The service is saved and the system displays *The system update completed successfully* message.

The example below demonstrates routing key business process, which executes a set of XML documents in FIFO order by OrderID field:

```
<process name=”AssignQueueKey">
  <sequence>
    <assign to=”FifoRoutingKey”    from=”DocToDOM(PrimaryDocument)/Order/@OrderId” />
  </sequence>
</process>
```

The routing information is not limited to XML documents only. Translation, Document Extraction, and other data extraction services can also be employed to retrieve routing data. In addition to the routing information in the document, the routing key business process has access to all information passed from the adapter in process data. If the routing key process fails, the error information will be placed in the *Business Process Error Queue* as described below.

The routing key process must be configured with the *Enable Async Start Mode* disabled via the routing business process manager. If this is not configured, the routing key process will fail and the error information will be placed in the error queue.

Note: The FIFO Routing adapter must be enabled for message processing to occur. If this adapter is not enabled, messages will remain on the internal FIFO routing queues and no processing will occur.

Configuring FIFO Execution

You can customize the name and number of queues used in the FIFO framework. The number of task queues determines the number of concurrent processes that can execute in the system at a time. You can increase the number of queues, but it will consume more resources. The queue is defined in the *fifo.properties* property file in the properties directory. All settings in the *fifo.properties* configuration file can be overridden via *customer_overrides.properties*. Please see the *fifo.properties* file for additional information pertaining to customer overrides. The default queue configuration is as follows:

```
workflow.taskqueue.2=FIFO.GIS.QUEUE.2
workflow.taskqueue.3=FIFO.GIS.QUEUE.3
workflow.taskqueue.4=FIFO.GIS.QUEUE.4
workflow.taskqueue.5=FIFO.GIS.QUEUE.5
workflow.taskqueue.6=FIFO.GIS.QUEUE.6
workflow.taskqueue.7=FIFO.GIS.QUEUE.7
workflow.taskqueue.8=FIFO.GIS.QUEUE.8
workflow.taskqueue.9=FIFO.GIS.QUEUE.9
workflow.taskqueue.10=FIFO.GIS.QUEUE.10

fifo.workflow.errorqueue=FIFO.GIS.ERROR
```

Business Process Queues

The FIFO business process execution queues are defined by rows that are prefixed with workflow.taskqueue. A queue row consists of a unique ID with prefix workflow.taskqueue to the left and a unique name without spaces or punctuation to the right.

You can add a queue by adding an additional row to the existing property file or to *customer_overrides.properties*. The simplest way to add additional queues is to continue the existing numbering scheme. You can remove a queue by deleting a row.

Note: Queues cannot be reduced below their default set of ten queues using *customer_overrides.properties*. If this is required, the queues must be removed directly from *fifo.properties*.

FIFO processing must be complete and the queues must be empty to change the queue configuration. You must disable the inbound adapter while changing the queue configuration. If the inbound adapter is not disabled and the queues are not drained, it may result in message execution that is out or order.

Business Process Error Queue

The business process error queue is defined within the *fifo.properties* file. The error queue configuration defines the destination of errors within the FIFO framework. The error queue name should not contain spaces or punctuation. The default business process error queue is shown below:

```
fifo.workflow.errorqueue=FIFO.GIS.ERROR
```

Recovering Errant Data

The messages in the error queue are written in XML format. The XML format provides information to determine the nature and source of the document containing the error. The error message contains information that enables the retrieval of document data; however, contents of the document are not stored in the message. The error message format is as below:

```
<?xml version="1.0" encoding="UTF-8"?>
<FifoError ErrorMessage="" ErrorType="" TaskId="" TaskQueueId="" TaskQueueKey=""
Type="">
  <WorkFlowError PrimaryDocumentId="" WorkFlowContextId="" WorkFlowId=""
WorkFlowInitiator="">
    <FifoErrorNode/>
   <FifoInitializationBpReport AdvancedStatus="" BasicStatus="" PrimaryDocumentId=""
ServiceName="" WfdName="" WfdVersion=""
      WorkFlowContextId="" WorkFlowId="">
        <StatusReport></StatusReport>
        <ProcessData>
          <PrimaryDocument SCIObjectID=""/>
        </ProcessData>
    </FifoInitializationBpReport>
  </WorkFlowError>
</FifoError>
```

FIFO Error Elements

FifoError Element

The *FifoError Type* indicates the type of FIFO task that is being executed. At present, Async WorkFlow is the only type supported.

The table below lists the other FifoError elements:

Type	Description
TaskId	A unique ID given to each FIFO task executed by the FIFO framework.
TaskQueueId	The queue where the FIFO task was executed.
TaskQueueKey	The key that was returned through the FIFO routing key business process execution.
ErrorMessage	This element contains the information that assists in determining the cause of the failure.

WorkFlow Error Element

The table below lists the WorkFlow Error elements:

Type	Description
WorkFlowId	This element contains the workflow id that was executed.
WorkFlowContextId	This element contains the workflow context id for the first step of the business process. This information is used to retrieve the workflow and extract additional data in advanced scenarios.

Type	Description
WorkFlowInitiator	This element contains the name of the workflow initiator. In most cases, name of the adapter that started the process will be the workflow initiator name.
PrimaryDocumentId	This element contains the ID for the primary document of the business process.

FifoInitializationBPReport

This element contains metadata that describes the execution of the routing key initialization business process.

This is an optional node. It will be included both in process data of the executed business process and in the error queue XML. It is automatically included in the XML data if an error occurs during task initialization. To force the inclusion of this data, both in the error report and process data of the executed business process, ForceFifoInitializationDump to "true" in the routing key business process

The table below lists the initialization BP report elements:

Type	Description
AdvancedStatus	This element contains the advanced status for the final step of this business process.
BasicStatus	This element contains the basic status for the final step of this business process.
PrimaryDocumentId	This element contains the primary document id at the last step of this business process.
ServiceName	This element contains the service name for the last step of this business process.
wfdName	This element contains the workflow definition name for this business process.
wfdVersion	This element contains the workflow definition version for this business process.
WorkFlowContextId	This element contains the workflow context id for this business process.
WorkFlowID	This element contains the workflow id for this business process.
StatusReport	This element contains the status report, if any, at the last step of this business process.
ProcessData	This element contains the process data at the last step of the business process.

FifoErrorNode Element

When the routing key business process is executed, the business process author can optionally write additional metadata to the FifoErrorNode element in the process data. This element and all the child nodes will be included in the FifoError document as part of this element.

The routing key business process has access to all process data information passed onto it through the adapter. See the example below for additional information about generating an error node.

```
<process name=”AssignQueueKey">
  <sequence>
    <assign to=”FifoRoutingKey”    from=”DocToDOM(PrimaryDocument)/Order/@OrderId” />

   <assign to=”FifoErrorNode/MSMQ/@QueueName” from=”string(MSMQ/@QueueName)”
append=”true”/>
  </sequence>
</process>
```

The additional information from the adapter can be included in the element to preserve the context of the error information in an easily identifiable manner.

FIFO Error Queue Listener

An out of the box adapter is configured on each node to listen to the error queue. This adapter is named "FIFO Error Queue Listener {nodename}". The adapter will bootstrap a business process named FifoError. This process is configured to retrieve the data from the errant process, including the original document and to integrate it into this process. This allows you to automate the re-processing of the data and other activities.

The FifoError process is defined as follows:

```
<process name="FifoError">
  <sequence>
    <operation>
      <participant name="FIFORouting" />
        <output message="Xout">
          <assign to="." from="*"></assign>
          <assign to="FifoTask">FifoErrorRecord</assign>
        </output>
        <input message="Xin">
          <assign to="." from="*"></assign>
        </input>
    </operation>
  </sequence>
</process>
```

The FifoError process provides a basic implementation for error handing. A user-specified business process may be configured to allow for customized error handling. A user-specified business process must contain the FIFORouting service as configured in the default FifoError process. Details surrounding FIFORouting service are described below.

FIFORouting Service

The FIFORouting service provides a control and reporting mechanism for interaction between business processes and the FIFO subsystem.

The FifoTask parameter specifies the task that this service should execute. Currently, there are two operational tasks this service provides: FifoResponse and FifoErrorRecord.

The FifoErrorRecord parameter specifies that the FIFORouting service should parse an error record from the error queue, retrieve the errant business process data, and report on it, as described above. This parameter should be used in conjunction with a retrieval of an error record from the error queue. The primary document in this mode of operation must be an FifoError XML record.

When executed in the FifoErrorRecord mode, the FIFORouting service will retrieve data pertaining to the errant business process and include it in ProcessData for the current business process. All data, including documents, may then be used directly within the current business process. The service will generate data of the following format:

```
<ProcessData>
  ...

 <PrimaryDocument SCIObjectID=""/>

...

 <FifoProcess ErrorType="" WorkFlowContextId="" WorkFlowId=""
      WorkFlowInitiator="">
    <ProcessData>
      <FifoDetails>
        <FifoInitializationBpReport AdvancedStatus="" BasicStatus=""
            PrimaryDocumentId="" ServiceName="" WfdName="" WfdVersion=""
            WorkFlowContextId="" WorkFlowId="">
          <StatusReport>
          </StatusReport>
          <ProcessData>
            <PrimaryDocument SCIObjectID="" />
          </ProcessData>
        </FifoInitializationBpReport>
      </FifoDetails>
    </ProcessData>
  </FifoProcess>

</ProcessData>
```

Note: The first instance of ProcessData is that of the current error handler business process. The FifoProcess element contains the data from the errant business process. The ProcessData element within this element contains the data from the original errant business process. All data and documents within this ProcessData element may be used directly within this business process for error handing purposes.

The FifoReponse parameter specifies that the FIFORouting service should return a positive or negative success response to the FIFO subsystem. An optional parameter, FifoStatus, may also be specified. This status indicates whether or not the business process was a success and if it is an error, designations the FIFO subsystem to report an error. The FifoStatus parameter considers ERROR to be a failure and any other string data to be success.

The FifoResponse parameter is used to provide early response at to the success or failure of a FIFO business process. For example, assume business process A is the process that must be executed in FIFO. Business process A contains 10 steps. The first 5 steps must be executed in order; however, the last 5 steps provide data execution functionality where order is not important. In this example, optimal performance will be achieved by utilizing the FIFORouting service in FifoResponse mode to return the response at step 6. This will allow the next message to be processed immediately following the execution of this service and allow steps 7 through 11 to execute fully parallel.

Cluster Configuration

The FIFO messaging system requires an external clustered JMS provider to allow proper execution and failover in a clustered configuration. An out of the box configuration for ActiveMQ 5.2 is provided to streamline this deployment.

Configuring FIFO messaging in a cluster for ActiveMQ:

1. Download ActiveMQ 5.2 from http://activemq.apache.org/activemq-520-release.html for the appropriate OS.
2. Deploy an instance of ActiveMQ 5.2 on each node of the cluster.
3. An activemq.xml file is included the properties/fifo directory of the Sterling Integrator deployment of each node. For each node, take this file and copy it to the ActiveMQ deployment on that node within the "conf" directory. This file will configure ActiveMQ to use failover clustering utilizing the Sterling Integrator database for storage and configure its port usage. By default, ActiveMQ will be configured to listen at the Sterling Integrator base port + 65 and the ActiveMQ interface will be at base port + 66 (http://server:base port + 66/admin).
4. On each Sterling Integrator node, the queue configuration must be re-directed to utilize the ActiveMQ cluster. In each node, add the following to *customer_overrides.properties*:

   ```
   fifo.broker.username=
   fifo.broker.password=
   fifo.broker.url=failover:(tcp://node1_hostname:node1_base_port + 65,tcp://node2_hostname:node_2_base_port + 65, ..., tcp://noden_hostname:node_n_base_port + 65 )
   ```
5. Start the ActiveMQ instances on each node. See http://activemq.org for additional information about running an ActiveMQ instance.
6. Restart Sterling Integrator.