# Ordered Queue Processing

iWay Service Manager (iSM) provides channels that link processes within iSM to other processes in the same instance or another instance of iSM. The following channel types are available for *internal* and *ordered* queue processing:

- **Internal.** Passes messages between channels for asynchronous or synchronous execution.
- **Ordered.** Passes messages between channels for asynchronous execution, maintaining execution order and batch control.

This how-to includes the following topics:

---

## Supported Ordered Queue Processing Components

iWay Service Manager (iSM) provides the following components for Ordered queue processing:

- **Ordered Emit Service (com.ibi.agents.XDOrderedEmitAgent)**

  The Ordered Emit service sends a message to a group that is managed by an ordered queue. The document is marshaled with its context and then placed on the queue to be executed by the Ordered Queue listener and channel.

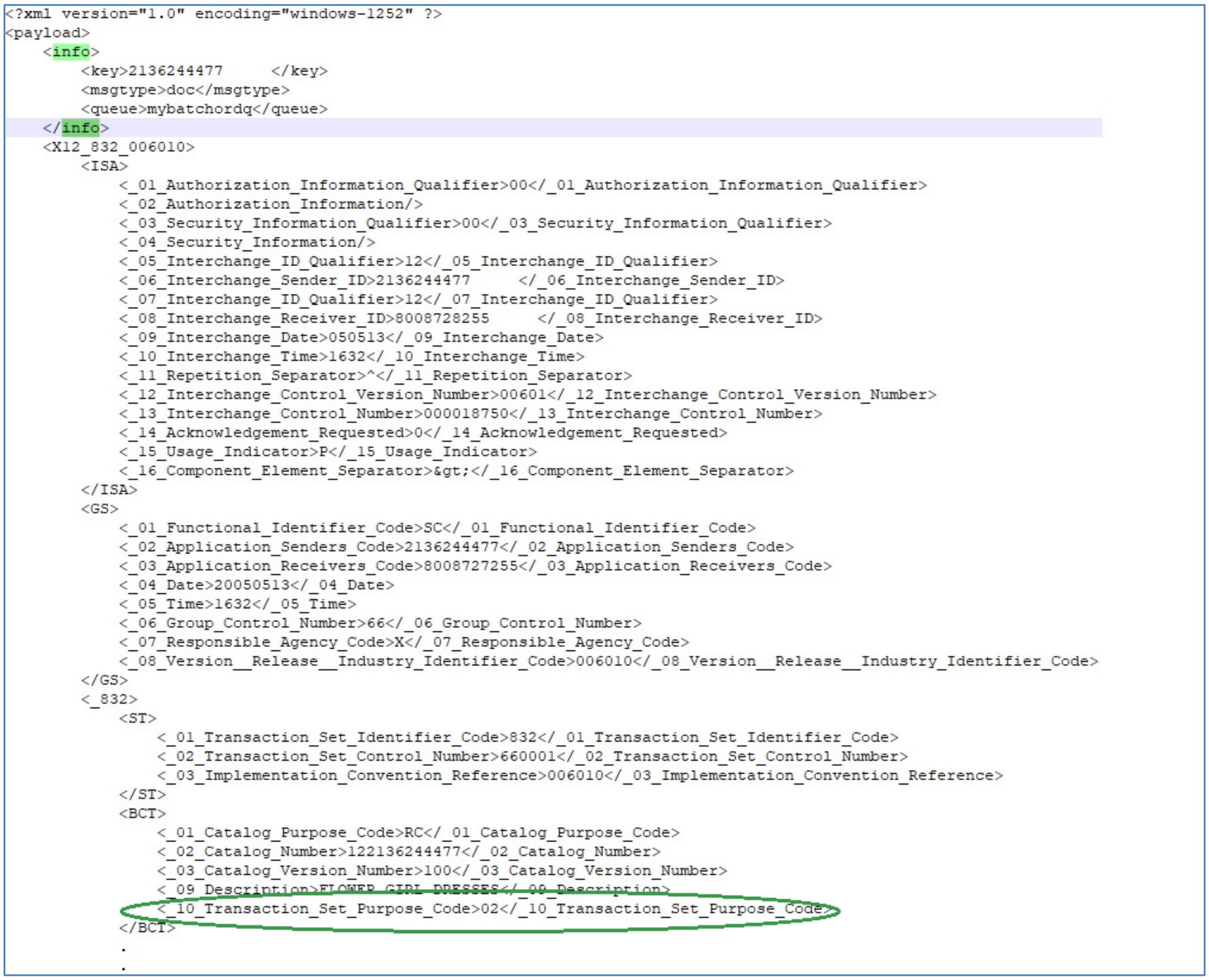
- **Ordered Queue Listener**

  The ordered queue is a collection of *queuelets*. Each queuelet collects the messages of one group in the order they were received. A new queuelet is created automatically when a message arrives with a key that has no associated queuelet. A queuelet is automatically deleted when it becomes empty, that is, when all the messages in that group have completed execution. Messages are deleted from the queuelet after their execution is completed. This ensures that messages cannot be lost when the queue is configured to be persistent, and maintains the ordering should a message arrive for the group while execution of the last message is proceeding.

## Configuring the Components for the How-To

### Step 1: Identifying the Input

The following image shows a sample XML input document that is referenced by this how-to.



```
<?xml version="1.0" encoding="windows-1252" ?>
<payload>
    <info>
        <key>2136244477     </key>
        <msgtype>doc</msgtype>
        <queue>mybatchordq</queue>
    </info>
    <X12_832_006010>
        <ISA>
            <_01_Authorization_Information_Qualifier>00</_01_Authorization_Information_Qualifier>
            <_02_Authorization_Information/>
            <_03_Security_Information_Qualifier>00</_03_Security_Information_Qualifier>
            <_04_Security_Information/>
            <_05_Interchange_ID_Qualifier>12</_05_Interchange_ID_Qualifier>
            <_06_Interchange_Sender_ID>2136244477     </_06_Interchange_Sender_ID>
            <_07_Interchange_ID_Qualifier>12</_07_Interchange_ID_Qualifier>
            <_08_Interchange_Receiver_ID>8008728255     </_08_Interchange_Receiver_ID>
            <_09_Interchange_Date>050513</_09_Interchange_Date>
            <_10_Interchange_Time>1632</_10_Interchange_Time>
            <_11_Repetition_Separator>^</_11_Repetition_Separator>
            <_12_Interchange_Control_Version_Number>00601</_12_Interchange_Control_Version_Number>
            <_13_Interchange_Control_Number>000018750</_13_Interchange_Control_Number>
            <_14_Acknowledgement_Requested>0</_14_Acknowledgement_Requested>
            <_15_Usage_Indicator>P</_15_Usage_Indicator>
            <_16_Component_Element_Separator>&gt;</_16_Component_Element_Separator>
        </ISA>
        <GS>
            <_01_Functional_Identifier_Code>SC</_01_Functional_Identifier_Code>
            <_02_Application_Senders_Code>2136244477</_02_Application_Senders_Code>
            <_03_Application_Receivers_Code>8008727255</_03_Application_Receivers_Code>
            <_04_Date>20050513</_04_Date>
            <_05_Time>1632</_05_Time>
            <_06_Group_Control_Number>66</_06_Group_Control_Number>
            <_07_Responsible_Agency_Code>X</_07_Responsible_Agency_Code>
            <_08_Version__Release__Industry_Identifier_Code>006010</_08_Version__Release__Industry_Identifier_Code>
        </GS>
        <_832>
            <ST>
                <_01_Transaction_Set_Identifier_Code>832</_01_Transaction_Set_Identifier_Code>
                <_02_Transaction_Set_Control_Number>660001</_02_Transaction_Set_Control_Number>
                <_03_Implementation_Convention_Reference>006010</_03_Implementation_Convention_Reference>
            </ST>
            <BCT>
                <_01_Catalog_Purpose_Code>RC</_01_Catalog_Purpose_Code>
                <_02_Catalog_Number>122136244477</_02_Catalog_Number>
                <_03_Catalog_Version_Number>100</_03_Catalog_Version_Number>
                <_09_Description>FLOWER_GIRL_DRESSES</_09_Description>
                <_10_Transaction_Set_Purpose_Code>02</_10_Transaction_Set_Purpose_Code>
            </BCT>
            .
            .
```

This XML input document contains an `<info>` section that is relevant for queuing and an `<X12_832_006010>` message payload section.

The `<info>` section is described in more detail as follows:

```
<info>
        <key>2136244477</key>
        <msgtype>doc</msgtype>
        <queue>mybatchordq</queue>
</info>
```

where:

`<key>`

Identifies the group for this message.

`<msgtype>`

Is the classification of the message that is being sent to the ordered queue.

`<queue>`

Is the name of the ordered queue that is serviced by the Ordered Queue listener.

The `<X12_832_006010>` message payload section contains the value that was selected to allow the numerical ordering (the *Transaction Set Purpose Code*).

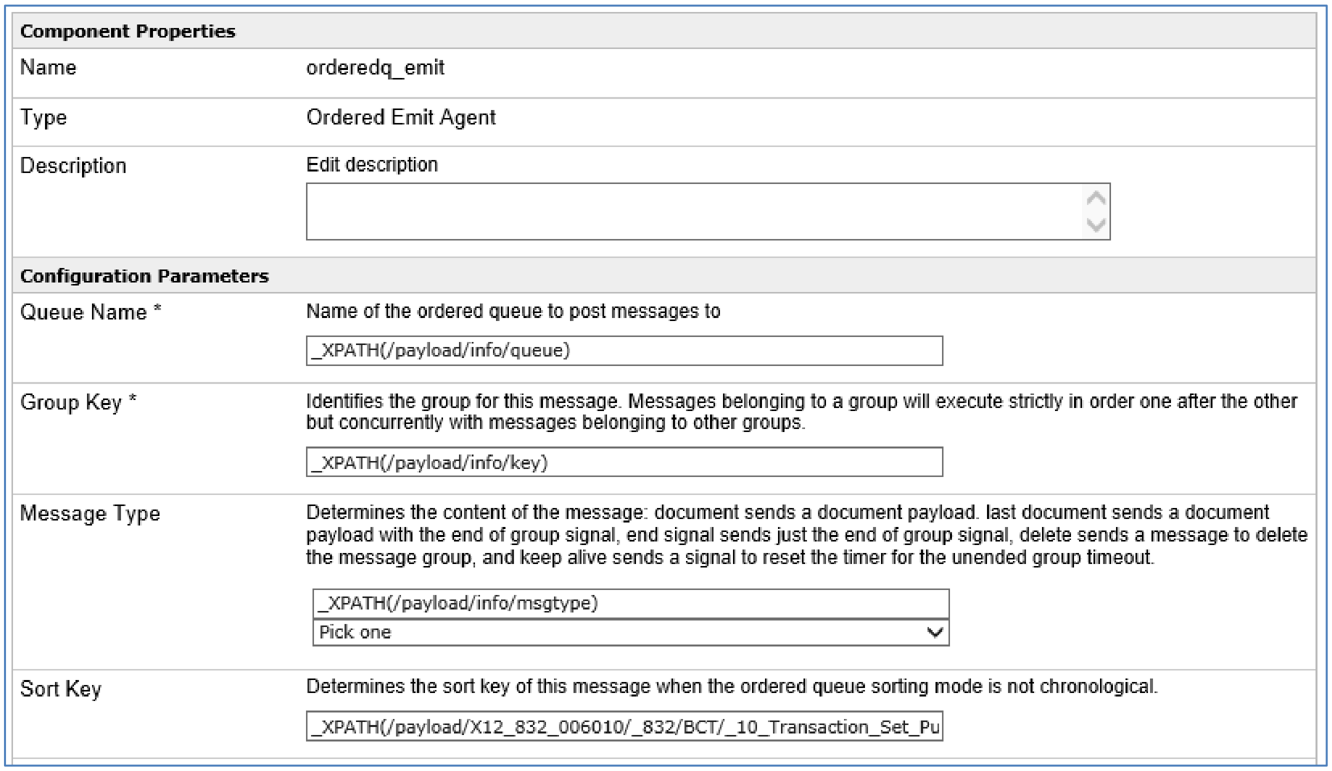
**Note:** All inputs containing a message payload can be sent simultaneously. However, the last message that is sent must be a non-payload message and contain:

`<msgtype>`**`docend`**`</msgtype>`

**Step 2: Defining a Channel to Emit Messages**

The channel that will be used to emit messages can be structured as follows:

- **Inlet.** Any supported protocol listener can be used, such as the File listener.
- **Route.** Contains the Ordered Emit service (agent), which is configured as shown in the following image.
- **Outlet.** The default outlet can be used (default.outlet).



| **Component Properties** | |
|---|---|
| Name | orderedq_emit |
| Type | Ordered Emit Agent |
| Description | Edit description |
| **Configuration Parameters** | |
| Queue Name * | Name of the ordered queue to post messages to<br>_XPATH(/payload/info/queue) |
| Group Key * | Identifies the group for this message. Messages belonging to a group will execute strictly in order one after the other but concurrently with messages belonging to other groups.<br>_XPATH(/payload/info/key) |
| Message Type | Determines the content of the message: document sends a document payload. last document sends a document payload with the end of group signal, end signal sends just the end of group signal, delete sends a message to delete the message group, and keep alive sends a signal to reset the timer for the unended group timeout.<br>_XPATH(/payload/info/msgtype)<br>Pick one |
| Sort Key | Determines the sort key of this message when the ordered queue sorting mode is not chronological.<br>_XPATH(/payload/X12_832_006010/_832/BCT/_10_Transaction_Set_Pu |

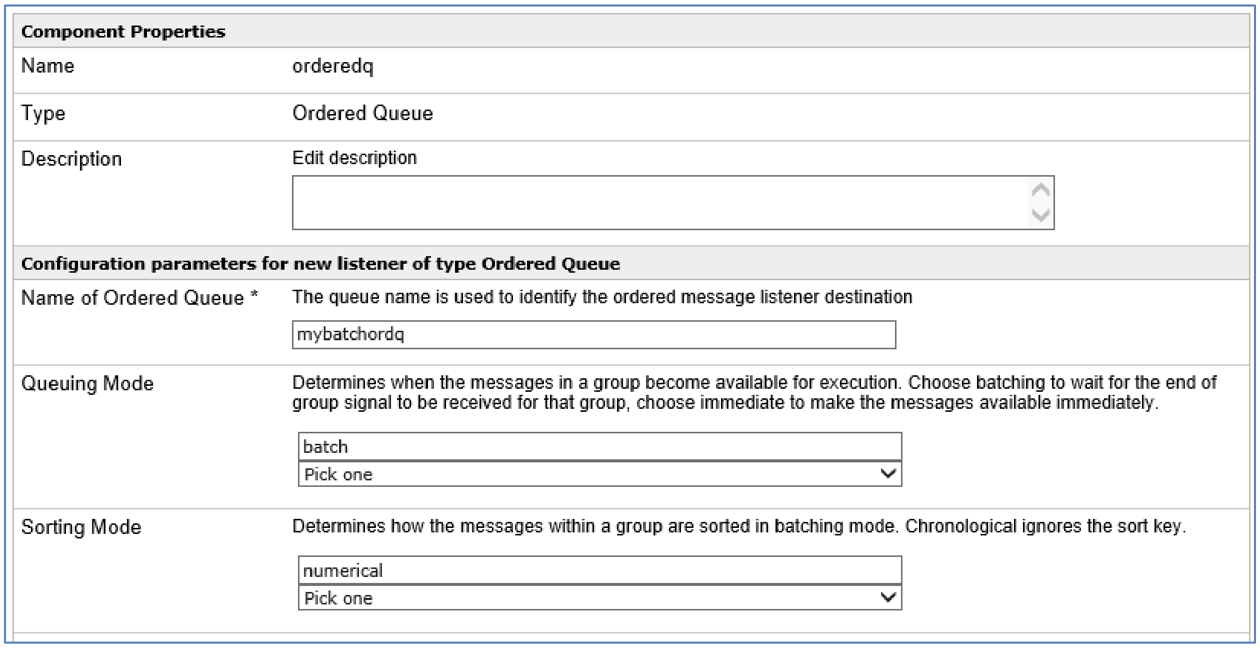
The following table lists and describes the parameters for the Ordered Emit Service (com.ibi.agents.XDOrderedEmitAgent).

| Parameter | Description |
|---|---|
| Queue Name (required) | The name of the ordered queue that is serviced by an Ordered Queue listener. The queue is created when the channel is started and exists as long as the server is running. |
| Group Key (required) | Identifies the group for this message. Groups are created in the ordered queue when a new key is presented, and are deleted when the last message for that group has completed execution. |
| Message Type | The classification of the message that is being sent to the ordered queue. Select one of the following message types from the drop-down list:<br><br>• **document.** The current document message is to be enqueued for its group key.<br><br>• **delete.** The batch queue is to be deleted. This has no effect for an immediate queue.<br><br>• **end.** The final message has been enqueued for this group and the batch queue is to be made available for dispatching. This has no effect for an immediate queue.<br><br>• **keep alive.** Resets the timeout period for a batch queue. This has no effect for an immediate queue.<br><br>• **last document.** Affects a document and end operation. Signals that this message is the final message and the queue is to be made available for dispatching. |
| Sort Key | The sort key is applied only for non-chronological batching queues. Passes the key to be used for the intra-batch sorting. Usually this will be an iFL statement extracting some value from the message itself, such as a sequence number. |

**Step 3: Defining a Channel to Accept Messages**

The channel that will be used to accept messages can be structured as follows:

- **Inlet.** Contains the Ordered Queue listener, which is configured as shown in the following image.
- **Route.** Contains the necessary business logic (for example, a process flow).
- **Outlet.** The default outlet can be used (default.outlet).



| | |
|---|---|
| **Component Properties** | |
| Name | orderedq |
| Type | Ordered Queue |
| Description | Edit description |
| **Configuration parameters for new listener of type Ordered Queue** | |
| Name of Ordered Queue * | The queue name is used to identify the ordered message listener destination<br>mybatchordq |
| Queuing Mode | Determines when the messages in a group become available for execution. Choose batching to wait for the end of group signal to be received for that group, choose immediate to make the messages available immediately.<br>batch<br>Pick one |
| Sorting Mode | Determines how the messages within a group are sorted in batching mode. Chronological ignores the sort key.<br>numerical<br>Pick one |

The following table lists and describes the parameters for the Ordered Queue listener.

| Parameter | Description |
|---|---|
| Name of Ordered Queue (required) | The name of the ordered queue that is used to identify the ordered message listener destination. |
| Queuing Mode | Determines how received messages are handled. Available modes include:<br>• **immediate.** As messages for a group are received, they become available for dispatching.<br>• **batch.** A message becomes available for dispatching only when the group is closed.<br>By default, the immediate queuing mode is selected. |
| Sorting Mode | Applies to batch queues only, and only within the batch itself. Available options include:<br>• **Chronological** (default). No intra-batch sorting is performed, and the order of presentation to the application is arrival order.<br>• **Lexical.** The subsort key is sorted in lexical order. For strings, this is generally considered alphabetic order.<br>• **Numeric.** The subsort key is sorted numerically by value rather than as a string. |

## Understanding the Business Scenario

This how-to describes an EDI X12 business scenario using ordered queue processing based on a numeric value. The EDI 832 message is used for furnishing or requesting the price of goods or services in the form of a catalog. Element BCT10 contains a Transaction Set Purpose Code, which is the numerical value used for ordering the incoming messages. Three EDI X12 832 messages are read by the listener.

The ordered queue processing involves two channels. The first channel contains the Ordered Emit service, and the second channel contains the Ordered Queue listener.

The _XPATH() values for the Ordered Emit service configuration parameters point at particular elements in the input XML document to determine the following:

- **Queue Name: _XPATH(/payload/info/queue)**

  The Queue Name is the name of the ordered queue where the Ordered Emit service places the messages and where the Ordered Queue listener picks up the messages.

- **Group Key: _XPATH(/payload/info/key)**

  A Group Key is a unique identifier that groups all messages together. The value originates from the EDI X12 input message, the Interchange_Sender_ID. The XPATH could also have pointed to the following:

  ```
  _XPATH(/payload/X12_832_006010/ISA/_06_Interchange_Sender_ID)
  ```

- **Message Type: _XPATH(/payload/info/msgtype)**

  The X12 message payload is identified as `doc`. If the service is configured in iWay Integration Tools (iIT), then the following choices apply for different scenarios:

  ```
  Delete {del}
  Document {doc}
  End Signal {end}
  Keep Alive {keepalive}
  Last document {docend}
  ```

  **Note:** As mentioned earlier, all inputs containing a message payload can be sent simultaneously. However, the last message that is sent must be a non-payload message and contain:

  ```
  <msgtype>docend</msgtype>
  ```

- **Sort Key: _XPATH(/payload/X12_832_006010/_832/BCT/_10_Transaction_Set_Purpose_Code)**

  Take from the input, this field is mandatory for lexical ordering and numeric ordering. In this how-to, the following codes are used:

- 02 (add)
- 03 (delete)
- 04 (change)

Besides receiving them in a numerical ascending order, they must be executed in the process flow in the following order: 03, 02, and 04.

Once the Ordered Queue listener receives the End message, the listener will send the messages in an ascending numerical order to the process flow in the Route.

This business case requires that the received messages are not handled in ascending sort order, as received from the listener, but rather starting with Transaction Set Purpose Code 03 (delete), then 02 (add), and finally 04 (change). One possible strategy is to first write the incoming ordered messages into a database table using the SQL object in the process flow. Then, another SQL object can be used to retrieve the messages based on the required Transaction Set Purpose Code order.