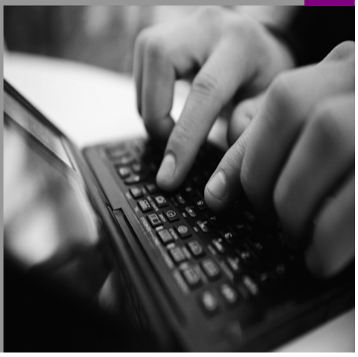
How-to Guide
SAP NetWeaver '04



How To… Configure Inbound Processing in ccBPM Part II: Queue Assignment

Version 1.00 – May 2007

Applicable Releases:
SAP NetWeaver '04

1 Scenario

Starting with SPS19 of SAP NetWeaver '04 (SPS10 of SAP NetWeaver '04s respectively) the configuration of the inbound processing of ccBPM using transaction SWF_INB_CONF has been enlarged to specify the inbound queue processing.

This How-to Guide is the second part of a series of documents providing decision criteria how to set the relevant inbound processing parameters properly based on process definitions. The first part handled the delivery mode whereas the last part will deal with transaction handling of ccBPM.

In this part of the series, the way to set the queue assignment and its runtime impact are discussed and typical queue settings are presented in the light of exemplary ccBPM fragments.

2 Introduction

Inbound processing is a runtime component of XI's ccBPM that links the Integration Engine to the Business Process Engine (BPE). It defines the way how XI messages are handed over to process instances. The messages can thereby either start a new process instance or can be bound to a running process instance using a correlation.

Within this How-to Guide we will first repeat the architectural principles of inbound processing with regard to queue assignment (Chapter 3). In a second step we will highlight the impact of the queue assignment parameter on the business process runtime based on selected process fragments (Chapter 4). Finally the necessary steps to set the queue assignment for the runtime representation of an Integration Process Service are presented (Chapter 5).

3 Architectural Principles of Inbound Queue Processing

3.1 Basics of BPE Inbound Processing

XI messages are handed over to BPE using process type specific queues. In former SPSs (i.e. SPS < SPS19 of NW04 or SPS10 of NW04s) each process instance of a given process type was provided with messages using a single process type specific queue. This queue was named according to the pattern XBQO$PE_WS<Task>, where the task name represented the runtime object of the relevant process type. Each and every message dedicated to an instance of a given process type was routed through that queue (see Figure 1).

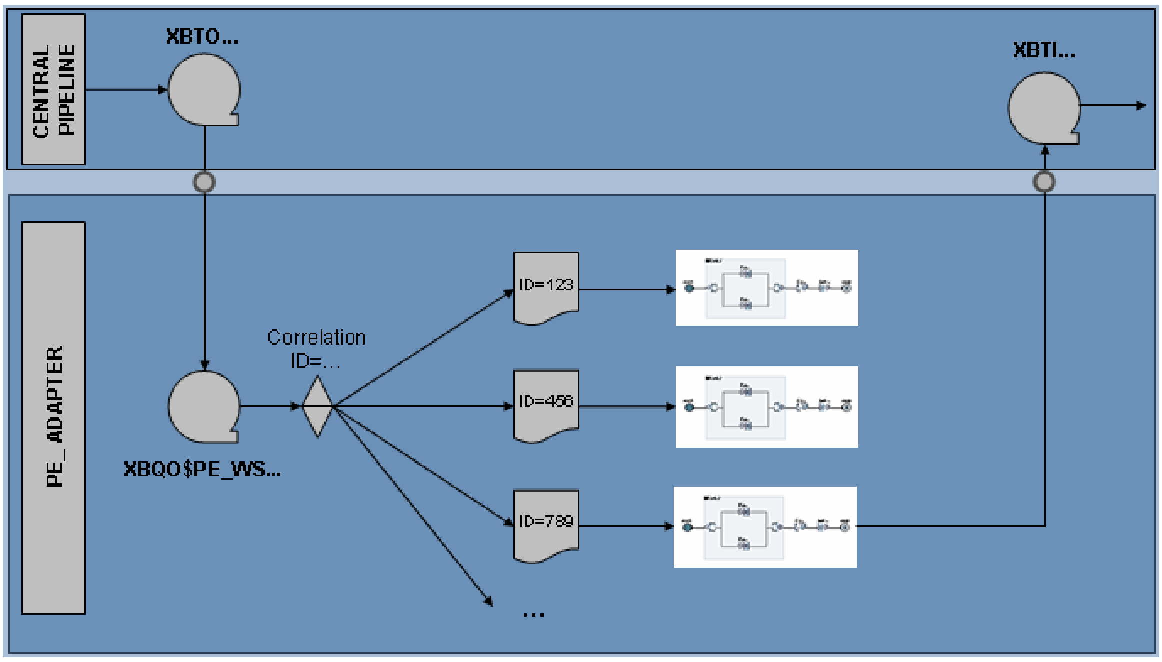


Figure 1: Distribution of messages to process instances using a correlation. The scenario uses the classical queue setting ("One Queue").

We now want to discuss the details of the inbound processing based on the classical queue setting used in former SPSs. The behavior is equivalent with the queue assignment "One Queue" which has been introduced in SPS19 of NW04 or SPS10 of NW04s.

Figure 2 displays potential interactions between this process type specific queue and the individual process instances for a simple collect scenario by means of a sequence diagram. The process is defined in a way, that each process instance collects two messages and sends out a merged document. We assume that multiple process instances with varying correlations may exist in parallel by means of correlations. The figure shows relevant steps during processing and is explained in details below. The details of correlation handling are neglected for ease of understandability.

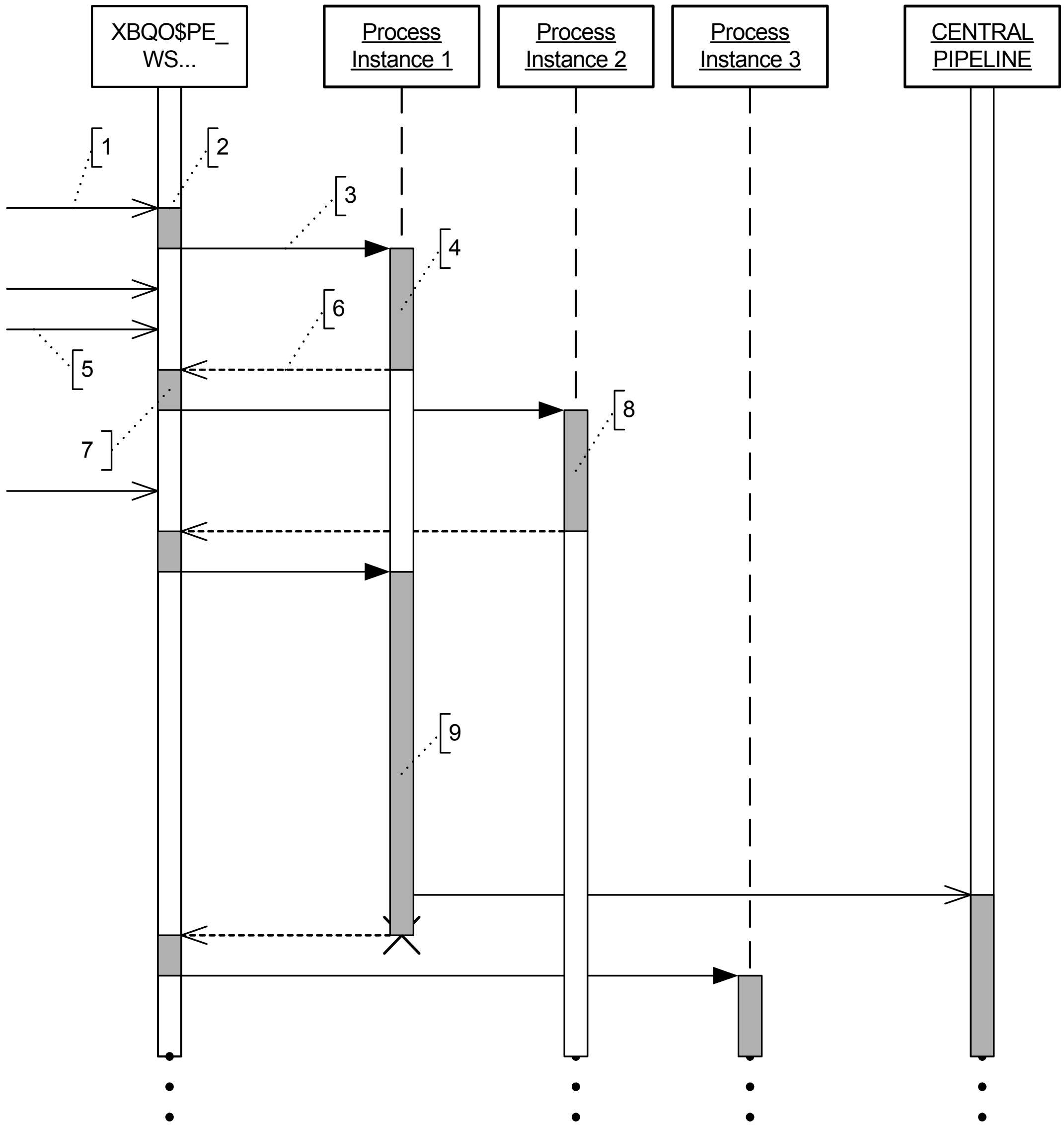


Figure 2: Sequence diagram showing interactions between the process type specific inbound queue and multiple process instances based on the classical queue setting.

The following steps are executed:

1. A message dedicated for a specific process type is handed over to the process type specific queue (XBQO$PE_WS...).
2. The queue gathers the messages and handles them sequentially
3. Correlation handling detects that a new process instance has to be created.
4. The process instance is now executed according to the process type definition.
5. The process type specific queue still receives new messages for that process type. But no message is handed over to the process instances until the synchronous processing of the active process instance is interrupted by a *receive* or *wait* step. This blocking mechanism is crucial to prevent messages from being sent to process instances without open receive steps (see How-to Guide: "How To…Configure Inbound Processing in ccBPM. Part I: Delivery Mode")
6. If a *receive* or *wait* step is encountered in the process flow, the synchronous response to step 3 releases the queue for processing of the intercepted messages.
7. The next message is now handled by the queue. The message either starts a new process instance or is distributed to a running process instance according to the correlation settings.

Steps 1 to 7 are executed sequentially for each message being sent to this process type.

8. In our example, the following message starts a new process instance according to the correlation settings. In the meantime Process Instance 1 waits for the following message to be captured (i.e. until the process type specific queue triggers further processing).
9. Finally if the last message is transferred to this instance it is processed following the process definition until its end (in this example it will merge the documents and send them to the CENTRAL PIPELINE of the Integration Engine). Process Instance 1 terminates and the queue takes over the control again.

This simple thought experiment shows that an efficient and fast queue handling is crucial for the throughput of messages intended for being sent to the Business Process Engine. Therefore the queue assignment is now configurable to prevent messages from being retarded by waiting queues. Consequently queue assignment provides a means to raise the throughput of messages sent to the BPE.

3.2 Configure Inbound Queue Handling

The inbound queue handling defines whether the delivery of messages to a process instance can be executed in parallel or if the messages are delivered sequentially. The impact on processing performance is significant since the process queue is waiting to deliver the following message until the first process step breaking synchronous processing is encountered within the process definition. Process steps interrupting synchronous process execution are *receive* steps and *wait* steps.

The runtime of the Business Process Engine supports four potential queue assignments "One Queue", "One Configurable Queue", "Multiple Queues (Random)" and "Multiple Queues (Content-Specific)". The runtime behavior and typical use cases of these settings will be presented in detail in the following sections.

3.3 Using Multiple Inbound Queues

In section 3.1 we've discussed the impact of inbound queue handling on the throughput of messages being sent to an integration process instance. Since the queue waits until the synchronous processing has ended before it handles the following message (Step 6 in Figure 2), all messages are treated sequentially if queue handling is set to "One Queue" or "One Configurable Queue". In order to raise the message throughput it is crucial to parallelize inbound processing, i.e. to create multiple inbound queues (using settings "Multiple Queues (Random)" or "Multiple Queues (Content-Specific)").

On the other hand, the usage of multiple inbound queues might have crucial impact on correlation handling. Since the evaluation of correlations is executed after the distribution of the messages to the diverse inbound queues (see Figure 3), the inbound handling has to guarantee that messages with identical behavior in terms of correlations end up in the same inbound queue. If this is not the case, messages handled by different queues might start new process instances instead of being correlated to a running instance - even if they semantically belong together according the correlation.

Therefore the setting "Multiple Queues (Content-Specific)" has been introduced to guarantee that messages will be distributed to the correct process instance provided that the process definition follows well-defined rules. These rules will be discussed in detail in section 4.4.

Note that the settings using a single queue (queue assignments "One Queue" and "One Configurable Queue") do not produce these side effects. Therefore it is highly recommended using these settings in case of any doubt!

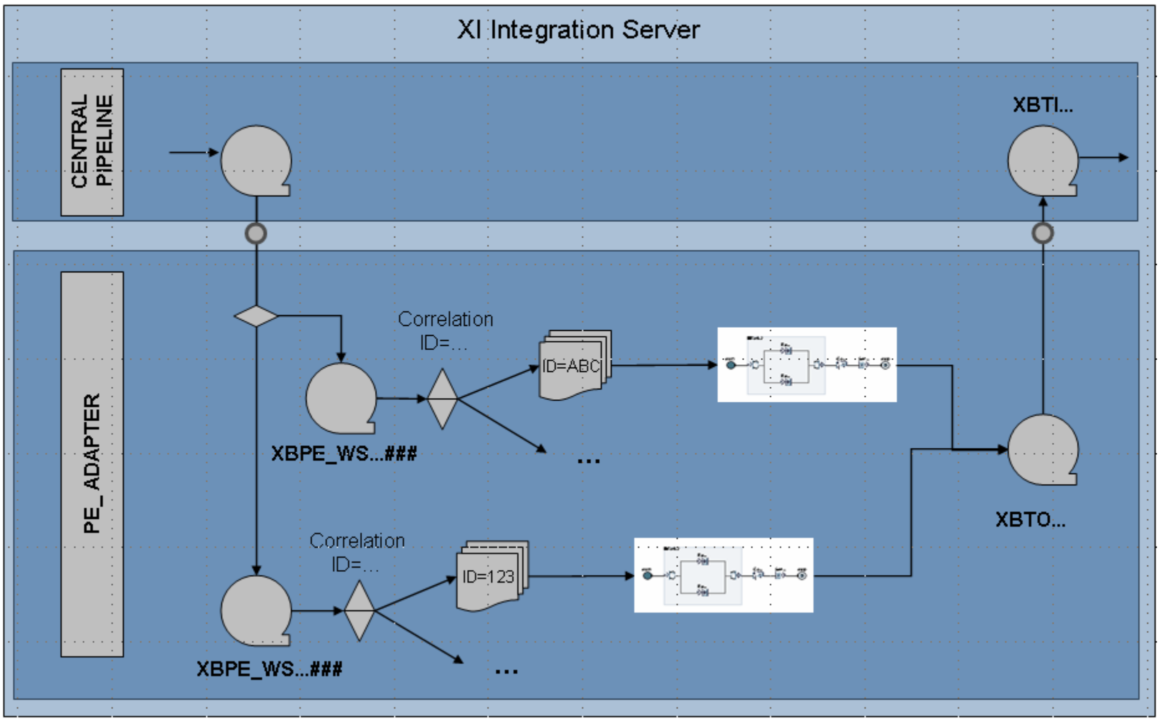


Figure 3 Distribution of messages to process instances using a correlation. The scenario uses the queue setting "Multiple Queues (Content-Specific)".

4 Impact of Queue Assignment

4.1 One Queue (Classic Setting)

4.1.1 Runtime Behavior

The runtime behavior of the setting "One Queue" has been discussed in detail in section 3.1.

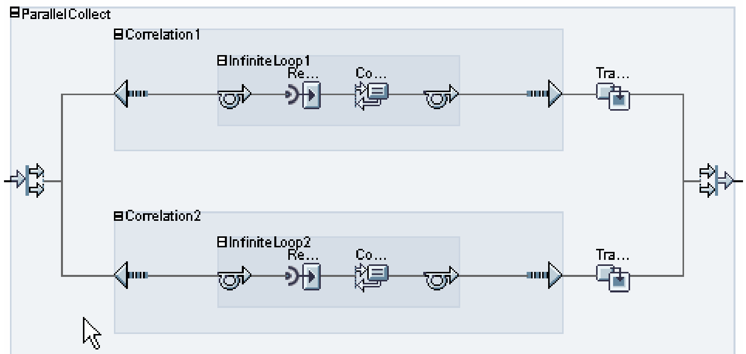
4.1.2 Use cases

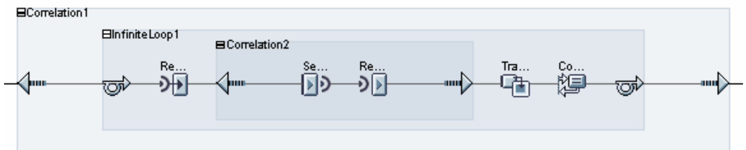
1. For ease of configuration the setting "One Queue" (default setting) can be kept for all non performance-critical processes.

Works for all process definitions.

2. All processes with more than one correlation **require** one inbound queue. Examples are:

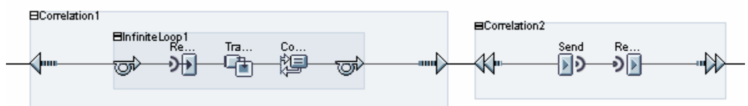
- Processes with overlapping/nested active correlation





Since the evaluation of correlations depends on the process type specific queue, this queue has to be kept unique per correlation to guarantee that the messages are associated to the correct process instance. Overlapping correlations might be activated in parallel or nested and hence the uniqueness of the inbound queue as to be set fixedly. If the messages were not handled by a single queue, messages belonging logically together could end up in distinct process instances.

- Processes with sequential correlations



Processes with overlapping or nested active correlations might be changed so the correlations are activated sequentially. In this case setting "Multiple Queues (Content-Specific)" can be chosen.

4.2 One Configurable Queue

4.2.1 Runtime Behavior

The runtime behavior of the setting “One Configurable Queue” is similar to the processing based on a single inbound queue (setting “One Queue”). The processing sequence follows the schema displayed in Figure 2. By choosing queue assignment “One Configurable Queue”, the name of the process type specific queue will be changed to XBPE_<Task>, where <Task> represents the runtime object of the Integration Process Service. This queue can then be assigned to a dedicated server providing special processing capabilities (i.e. a high-performance server, a dedicated server for batch processing or specific logon groups). See section 5.3 for details.

4.2.2 Use cases

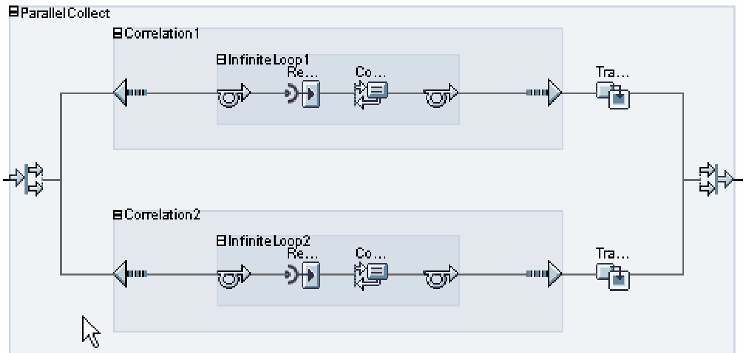
Here again, the use cases are equivalent to the ones presented for setting “One Queue”. So the process fragments are identical to the ones presented in section 4.1.2. But the setting “One Configurable Queue” is supposed to be applied for all performance-critical and workload intense processes since the processing can be either done on dedicated ABAP instances (running on a high-performance server or a dedicated server for batch processing) or can be allocated to multiple instances applying logon groups.

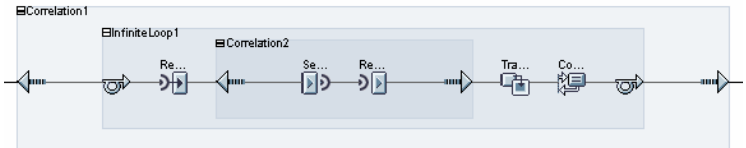
1. For ease of configuration the setting “One Configurable Queue” can be chosen for all **time-critical** processes.

Works for all process definitions.

2. Time-critical processes with more than one correlation

3. Processes with more than one correlation provoking heavy workload on the server

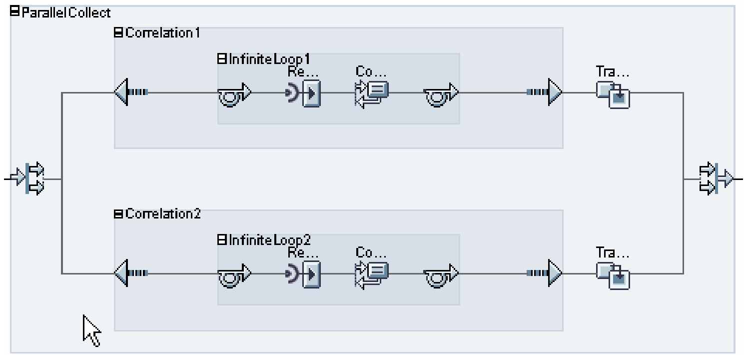


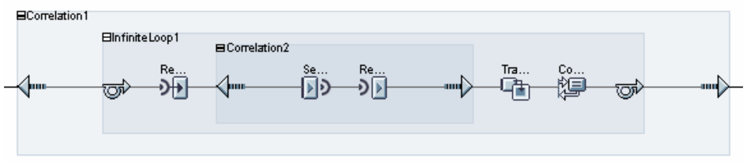


Since the evaluation of correlations depends on the process type specific queue, this queue has to be kept unique per correlation to guarantee that the messages are associated to the correct process instance. Overlapping correlations might be activated in parallel or nested and hence the uniqueness of the inbound queue as to be set fixedly. If the messages were not handled by a single queue, messages belonging logically together could end up in distinct process instances.

4. Time-critical processes with more than one correlation / Processes with more than one correlation provoking heavy workload on the server. Examples are:

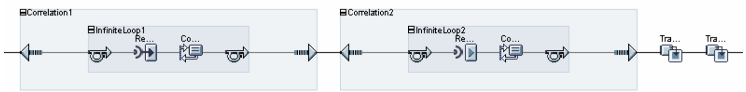
- Processes with overlapping/nested active correlation

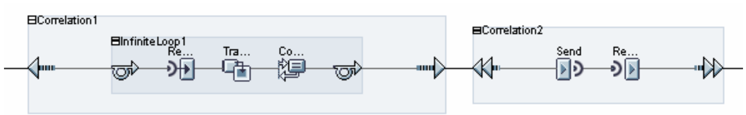




Since the evaluation of correlations depends on the process type specific queue, this queue has to be kept unique per correlation to guarantee that the messages are associated to the correct process instance. Overlapping correlations might be activated in parallel or nested and hence the uniqueness of the inbound queue as to be set fixedly. If the messages were not handled by a single queue, messages belonging logically together could end up in distinct process instances.

- Processes with sequential correlations





Processes with overlapping or nested active correlations might be changed so the correlations are activated sequentially. In this case setting "Multiple Queues (Content-Specific)" can be chosen.

4.3 Multiple Queues (Random)

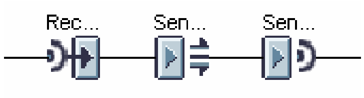
4.3.1 Runtime Behavior

When choosing "Multiple Queues (Random)" messages are allocated randomly to multiple parallel inbound queues regardless of any correlation settings. Since the evaluation of correlations is executed after the distribution of the messages to the multiple inbound queues (see Figure 3), messages handled by different queues might start new process instances instead of being correlated to a running instance. Therefore the setting "Multiple Queues (Random)" can only be used in all scenarios where no correlation is involved.

4.3.2 Use cases

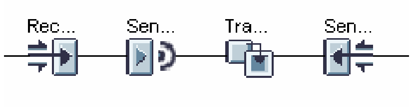
The setting "Multiple Queues (Random)" can be used to speed up time critical processes **without** correlations. Examples for these kinds of scenarios are sync-async bridges, lookup scenarios, split and multicasting scenarios not waiting for response messages.

1. Async-sync bridges / Lookup scenarios



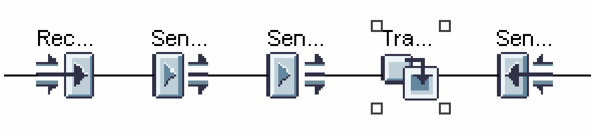
Scenarios employing an async-sync bridge do not require any correlation.

2. Sync-async bridges without response messages (using acknowledgements)



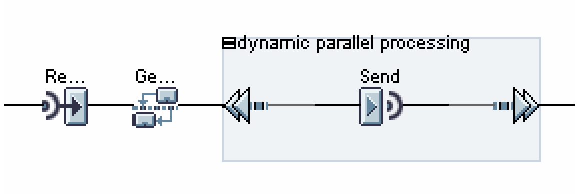
If the response to the asynchronous send step is not required to build up the synchronous response message closing the bridge, the asynchronous send step may simply demand an acknowledgement. **Note that it's crucial, that no asynchronous response and hence no correlation is involved.**

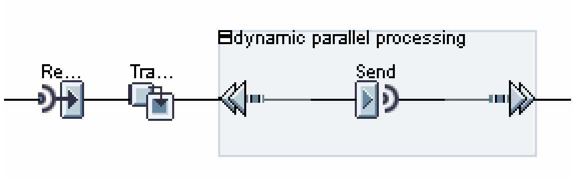
3. Sync-sync bridges / Sync-multisync bridges



Synchronous calls do not require any correlation. Therefore sync-sync and sync-multisync bridges may use setting "Multiple Queues (Random)".

4. Multicasting / Split scenarios without response messages (using acknowledgements)





If the asynchronous send step can be set up **using acknowledgements only,** multicasting as well as split scenarios can make use of randomly created multiple queues.

4.4 Multiple Queues (Content-Specific)

4.4.1 Runtime Behavior

In section 3.3 we've seen, that the usage of parallel inbound queues enhances throughput of messages being sent to ccBPM. The evaluation of correlations is executed after the message has been assigned to an inbound queue. Therefore messages that should end up in the same process instance according to their correlation settings might be distributed to distinct instances if they were processed by different queues. It is hence crucial to link inbound queue handling with correlation handling if multiple queues are to be used in all scenarios requiring correlations.

Applying the queue assignment "Multiple Queues (Content-Specific)", the system determines the name of the queue from the message fields in the correlation container. This guarantees that messages are distributed to the correct process instance using correlations even though parallel queues are used for inbound processing.

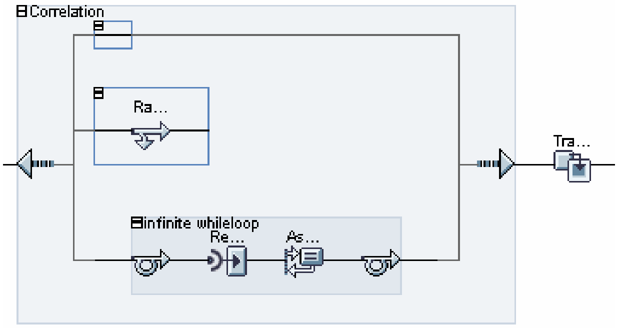
There are some prerequisites that your process definition has to fulfill if you want to make use of multiple queues being determined based on the correlation settings:

- Each process instance must contain only one correlation (see use cases for queue assignment "One Queue", or "One Configurable Queue").
- The content of a correlation must not change during the life time of a process instance.
- The correlation should be selective (i.e. it should instantiate multiple integration processes in parallel). Otherwise all messages will be processes by a single queue. Hence there won't be any performance gain compared to the "One Queue" behavior.

4.4.2 Use cases

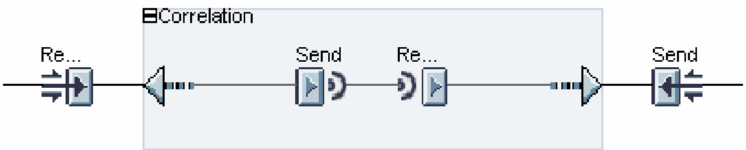
The setting "Multiple Queues (Content-Specific)" can be used to speed up time critical processes with non-overlapping correlations.

1. Collect scenarios



Typical use cases for multiple queues (calculated based on correlations) are simple collect scenarios (fulfilling the conditions mentioned in section 4.4.1). The time based collection depicted here just serves as example for other collect scenarios (e.g. content based or message based collections).

2. Single request-response scenarios



If the response to the asynchronous send step is required to build up the synchronous response message, the asynchronous response step has to be bound to the process instance using a correlation. Since this correlation is unique and stable, multiple content-specific queues might be used.

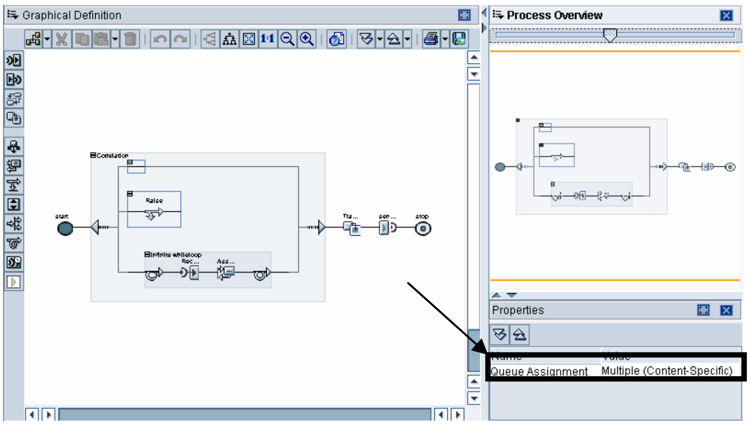
5 Assigning the Inbound Queue Settings

5.1 Preselection of the Queue Assignment

The developer can preselect the relevant queue assignment when setting up the process definition in the Integration Repository. This is normally done at the design phase of the Integration Process.

Please note that this preselection only defines the scope for the real queue assignment described in section 5.2. It has no direct impact on the runtime representation of your Integration Process but defines permitted values in transaction swf_inb_conf. The default value for the queue assignment preselection is “One Queue”. The actual queue assignment carried out using transaction swf_inb_conf (see below) has either to follow the preselection or may be set the queue assignment to “One Queue” or “One Configurable Queue”.

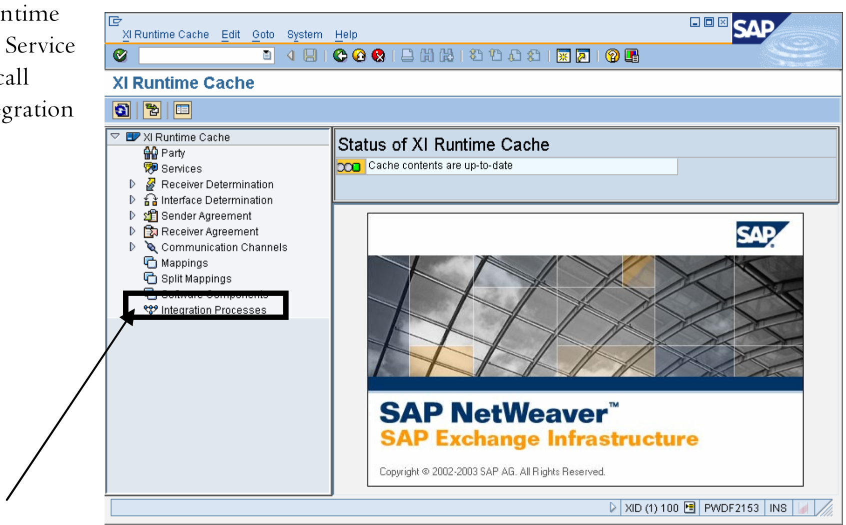
1. Open the relevant Integration Process in the Integration Repository. In the property area for the entire Integration Process you can set the parameter Queue Assignment to the values “One Queue”, “Multiple (Content-Specific)”, or “Multiple (Random)”.



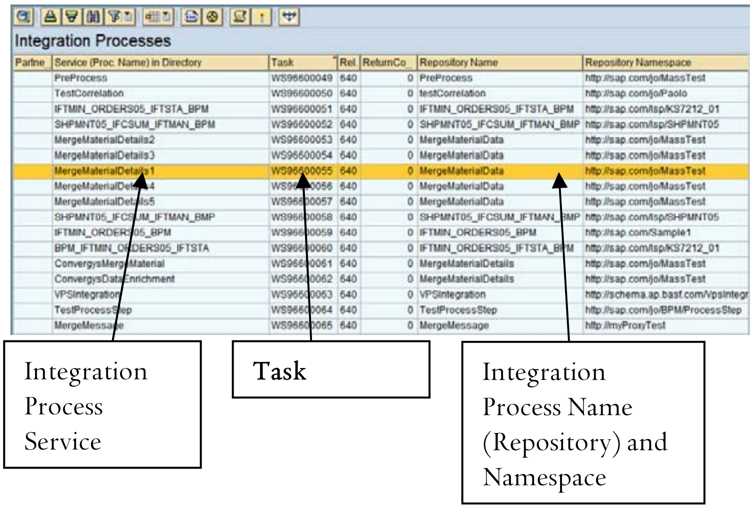
5.2 Setting the Queue Assignment

The actual queue assignment is done by the system administrator after the Integration Process Service has been activated in the Integration Directory. The values have to follow the preselection described in section 5.1.

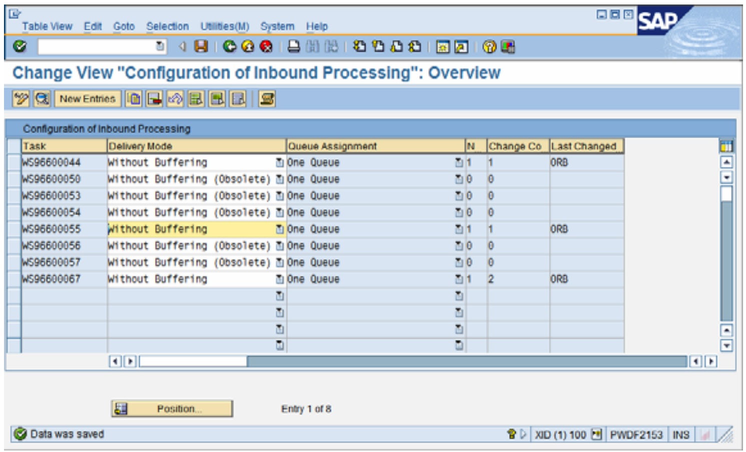
1. The queue assignment is based on the runtime representation of the Integration Process Service (Task). To retrieve the name of the task call transaction SXI_CACHE and select “Integration Processes”.



2. Search for the name of your Integration Process set in the Integration Repository or the name of the service, set in the Integration Directory to retrieve the name of the task.

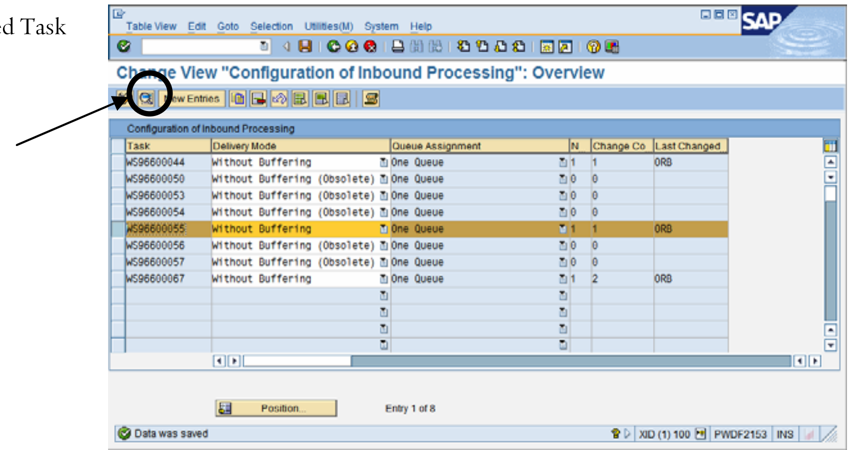


3. The system administrator sets up the actual queue assignment for the relevant Task using transaction SWF_INB_CONF. Switch to change mode.

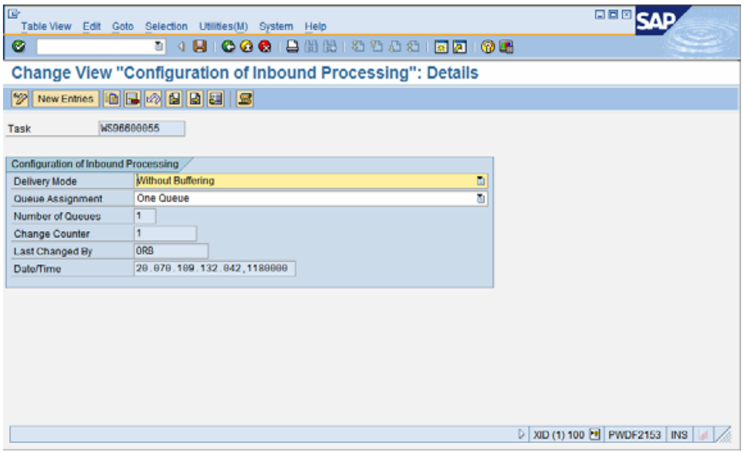


Transaction SWF_INB_CONF provides an overview over the Delivery Mode (for details see How-to Guide: “How To…Configure Inbound Processing in ccBPM. Part I: Delivery Mode”), the Queue Assignment, the Number of Queues and a counter for the number of executed changes.

4. Mark the line representing the concerned Task and select “Details”.

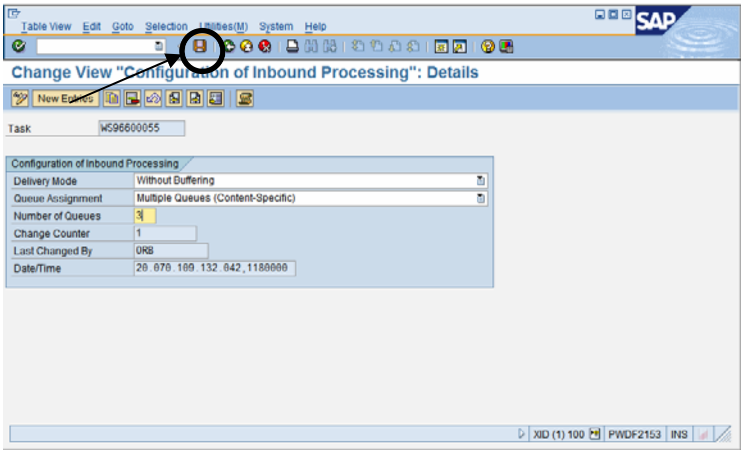


5. You get a detailed overview over the configuration of the inbound processing of that specific Task containing the delivery mode, the queue assignment, the number of queues, a counter tracing the number of changes and a date/time stamp.

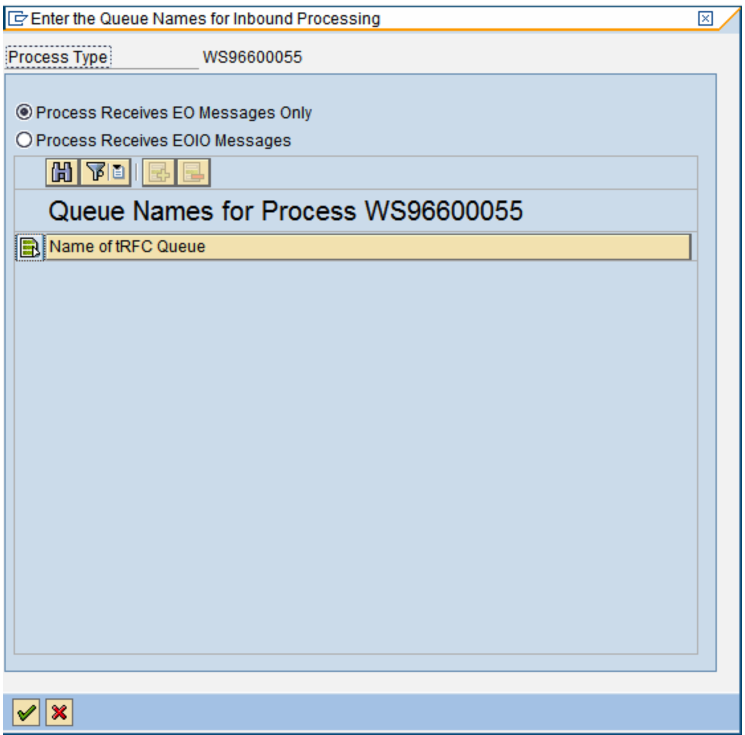


6. Set the queue assignment and the number of queues according to your needs following the preselection presented in section 5.1 as well as the recommendations presented in section 4.

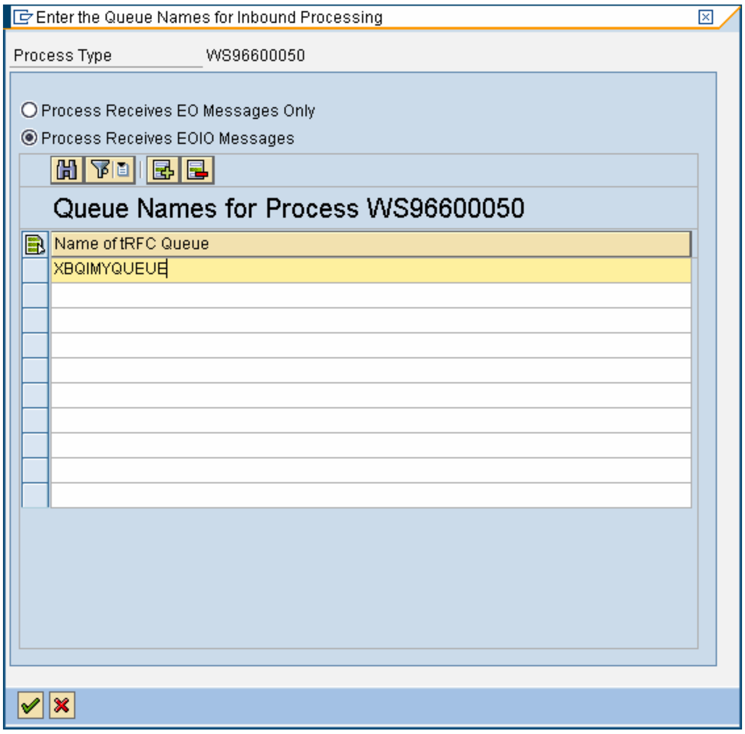
7. Save your changes. If the queue assignment does not match with the presetting in the Integration Repository, you get an error message.



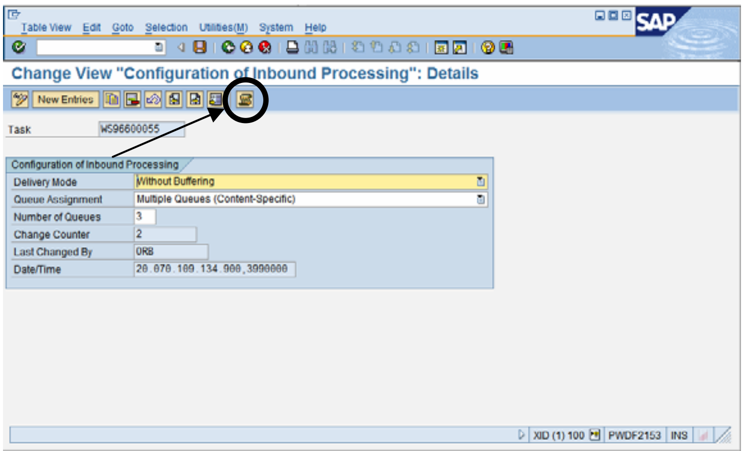
8. Since a switch of the queue assignment crucially affects the message processing in a running system, the change procedure has to guarantee that the messages already handed over to BPE are worked off before the new settings become effective. Therefore entries are written to the queues of the Integration Engine to flag whether a message has entered the queue before or after the switch occurred.



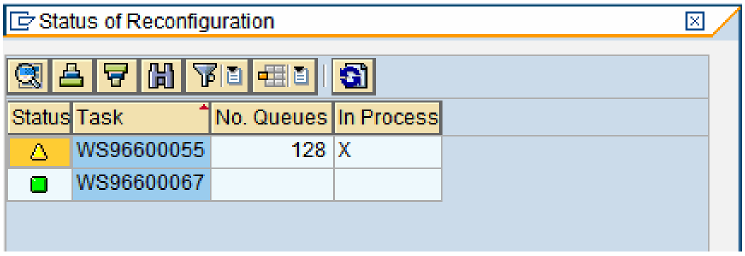
9. If the Integration Process captures **EOIO** messages, the names of the queues are defined by the sending application. Therefore these names cannot be detected automatically by the switch procedure. Hence the administrator has to add the queues with prefix XBQI*, XBQ1*, and XBQ9* to the list manually.



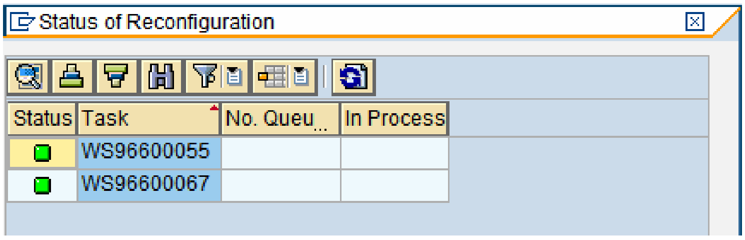
9. You can follow the state of the switch process selecting the "Status of Reconfiguration" button.



9. You'll get some information on the number of queues still to be checked and if the switch is still in process.



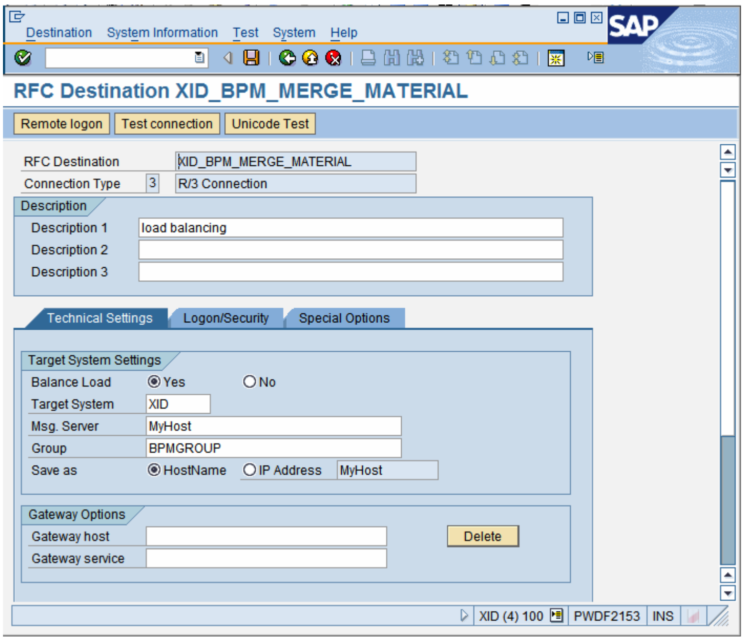
9. Once the status of your reconfiguration is equal to "Reconfiguration Complete" your new queue assignment becomes effective. A queue with the name XBPE_<task> is registered where <task> is the ID of the relevant Workflow pattern. Per default this queue makes use of RFC destination WORKFLOW_LOCAL_<CLNT>.



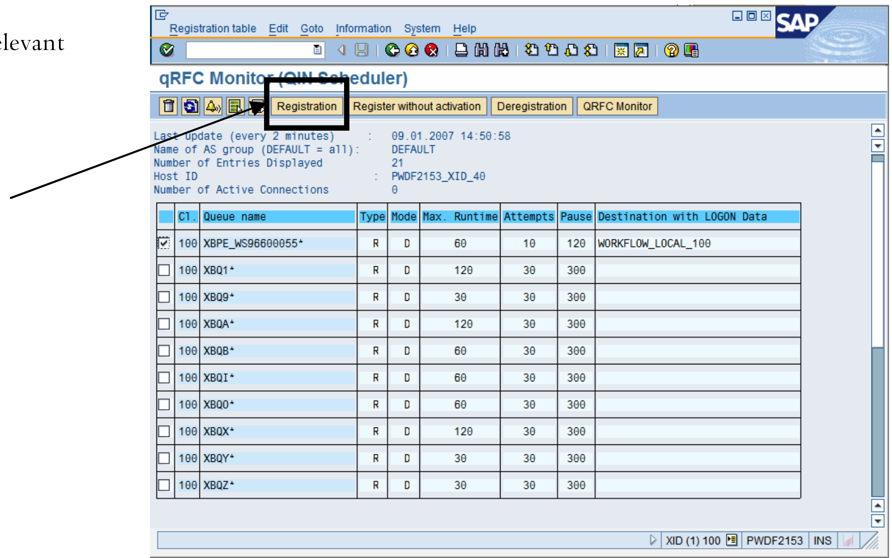
5.3 Configure Load Balancing

If you plan to run your processes on a dedicated application server for performance reasons, the queue(s) configured in section 5.2 can be assigned to a specific RFC destination. This option is available for the queue assignments “One Configurable Queue”, “Multiple Queues (Random)” or “Multiple Queues (Content-Specific)”.

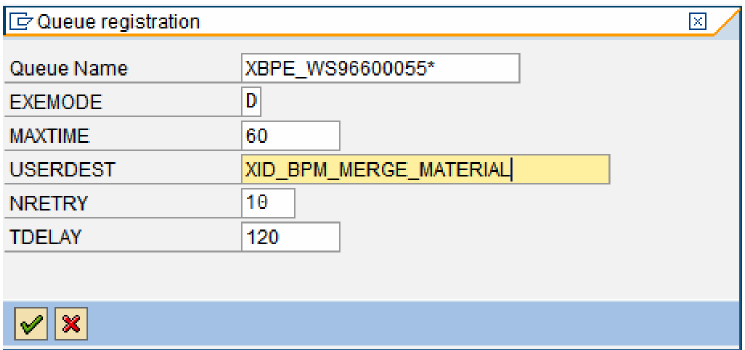
1. Set up an RFC destination (transaction SM59, connection type “3”) and specify the Technical Settings according to your load balancing requirements (e.g. using a logon group or specifying a dedicated application server). Use WF-BATCH as the logon user, which is set in transaction SXMB_ADM_BPE → Automatic BPM Customizing.



2. Call transaction SMQR, select the relevant queue and choose *Registration*



3. Enter the destination that you defined in step 1 in the USERDEST field. Choose *Continue (Enter)* to confirm your changes.



https://www.sdn.sap.com/irj/sdn/howtoguides