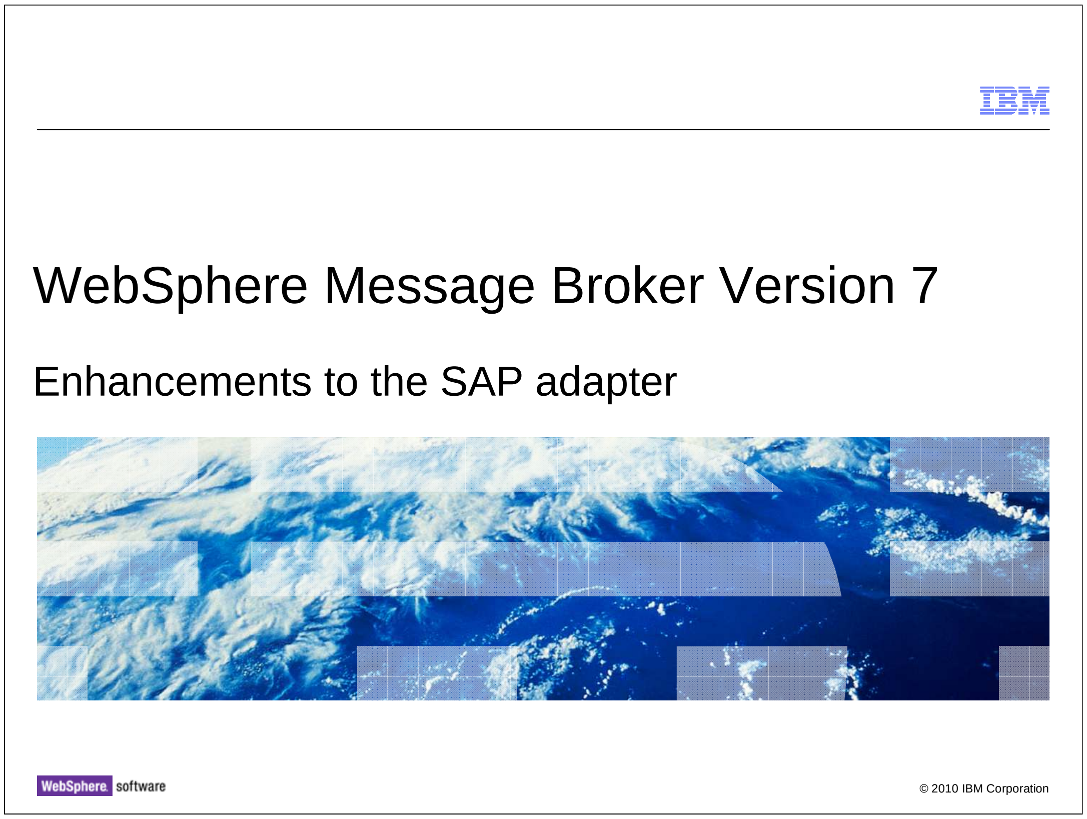


WebSphere Message Broker Version 7

Enhancements to the SAP adapter

WebSphere software

This session describes the enhancements that have been made to the SAP adapter in WebSphere® Message Broker version 7.

In addition to these specific enhancements for SAP, there are some general enhancements for all adapters, which are applicable to SAP. These are described in the next IBM Education Assistant session, covering iterative development and deployment.

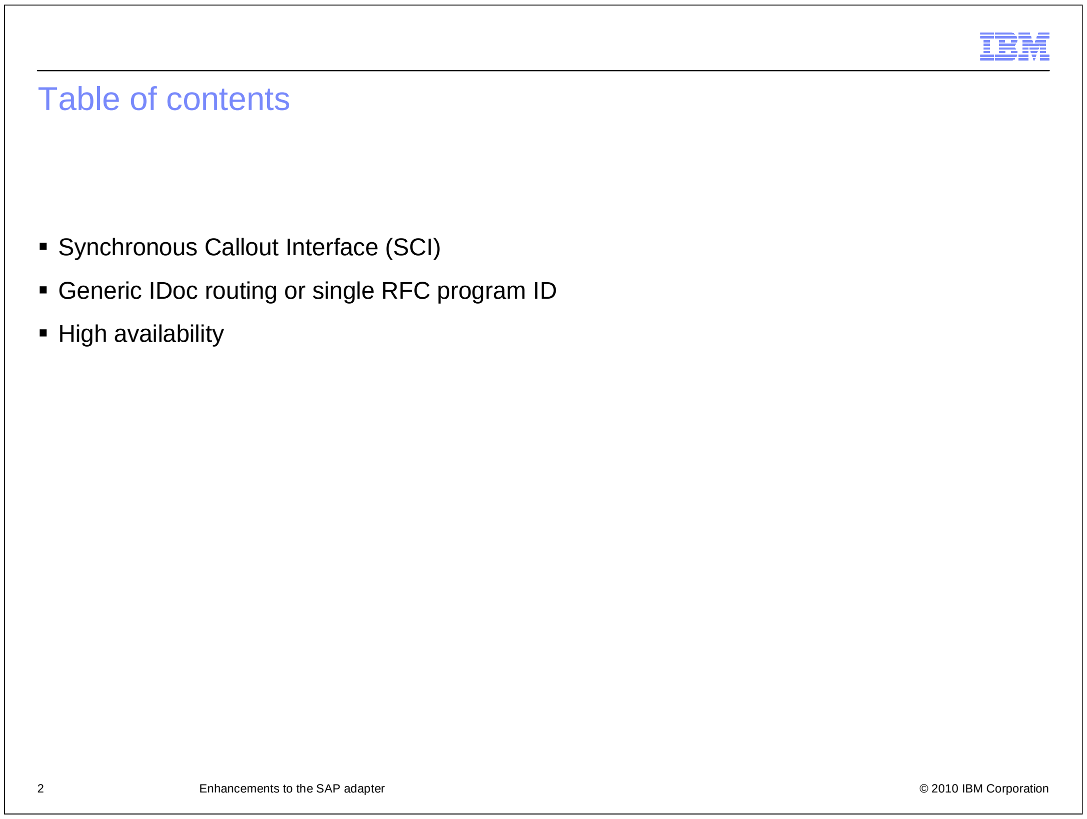


Table of contents

- Synchronous Callout Interface (SCI)
- Generic IDoc routing or single RFC program ID
- High availability

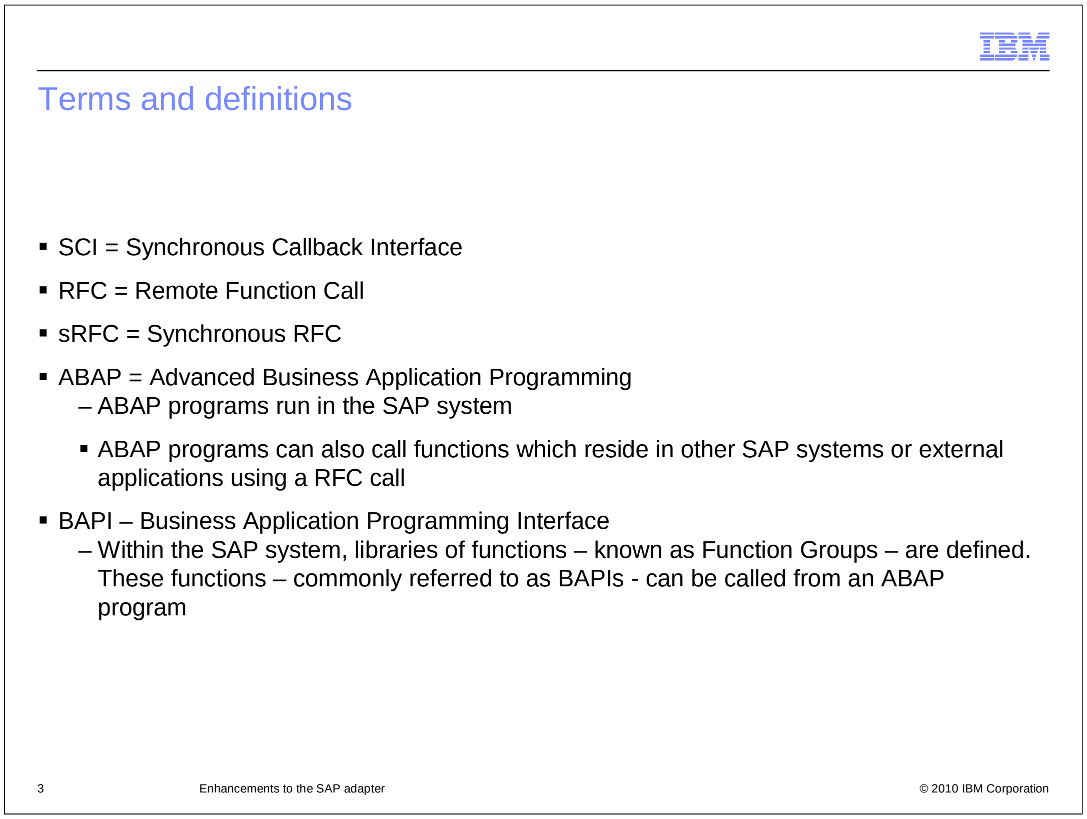
2 Enhancements to the SAP adapter © 2010 IBM Corporation

This session will cover three topics.

First, the SAP synchronous call interface, sometimes referred to as “SCI”, is now supported by WebSphere Message Broker. This is enabled with the introduction of a new node, the SAP reply node.

Secondly, many SAP installations require different outbound data formats to be managed by a single adapter connection. This is now provided in Message Broker version 7, with the single program ID function.

And finally, the SAP adapter now supports highly available scenarios.

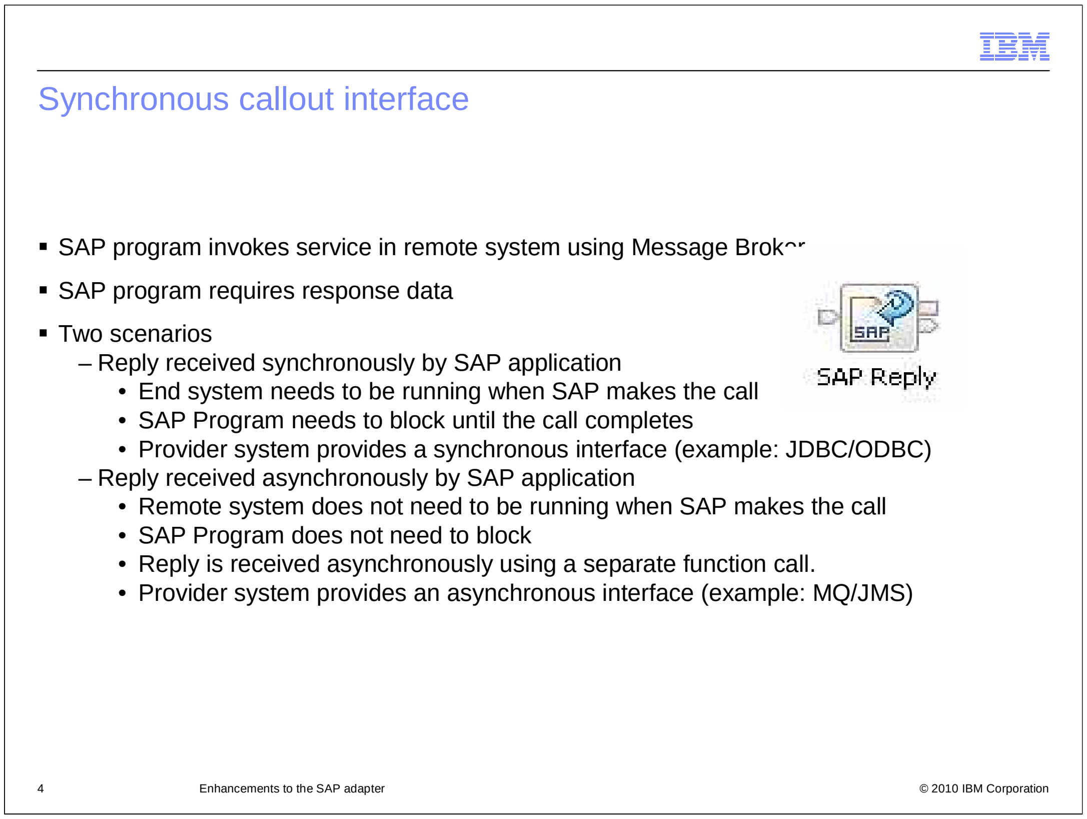


Terms and definitions

- SCI = Synchronous Callback Interface
- RFC = Remote Function Call
- sRFC = Synchronous RFC
- ABAP = Advanced Business Application Programming
  - ABAP programs run in the SAP system
    - ABAP programs can also call functions which reside in other SAP systems or external applications using a RFC call
- BAPI – Business Application Programming Interface
  - Within the SAP system, libraries of functions – known as Function Groups – are defined. These functions – commonly referred to as BAPIs - can be called from an ABAP program

3 Enhancements to the SAP adapter © 2010 IBM Corporation

This slide is provided for reference. It includes the primary SAP functions that are appropriate for this session. These terms will be familiar to an SAP administrator, and will be referenced throughout this session.



Synchronous callout interface

- SAP program invokes service in remote system using Message Brok
- SAP program requires response data
- Two scenarios
  - Reply received synchronously by SAP application
    - End system needs to be running when SAP makes the call
    - SAP Program needs to block until the call completes
    - Provider system provides a synchronous interface (example: JDBC/ODBC)
  - Reply received asynchronously by SAP application
    - Remote system does not need to be running when SAP makes the call
    - SAP Program does not need to block
    - Reply is received asynchronously using a separate function call.
    - Provider system provides an asynchronous interface (example: MQ/JMS)

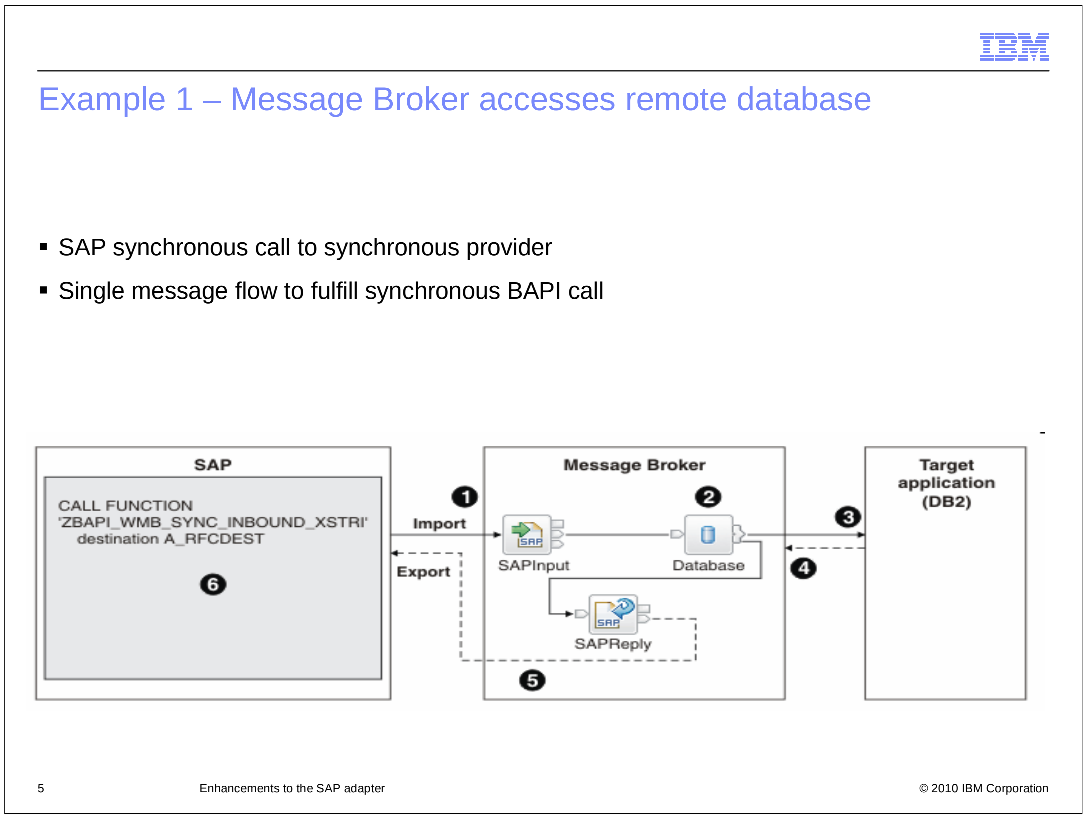
4 Enhancements to the SAP adapter © 2010 IBM Corporation

The first topic is the SAP synchronous call interface.

This function is supported in Message Broker version 7 with the new SAP reply node. This node enables the SAP system to invoke a synchronous callout to a message flow.

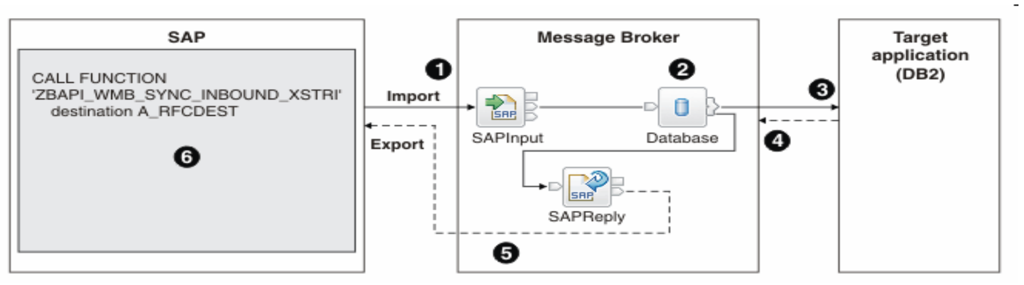
The SAP application that invokes this call will require a response to the request. This response can be received synchronously or asynchronously. If the response is required synchronously, the system which provides the response must be available at the time the initial request is made. The SAP application will wait until the response is received, and might consume resources until the response is received. The system that provides the response must be capable of being invoked so that a response is provided immediately.

If the response is required asynchronously, the providing application does not have to be running at the time the request is made. The SAP application will not wait until the response is received, so no resources will be consumed while waiting for the response. The provider application will typically have an asynchronous interface, such as an MQ or JMS interface. In this case, the response message will need to be related to the request; this is done using a reply identifier.



Example 1 – Message Broker accesses remote database

- SAP synchronous call to synchronous provider
- Single message flow to fulfill synchronous BAPI call

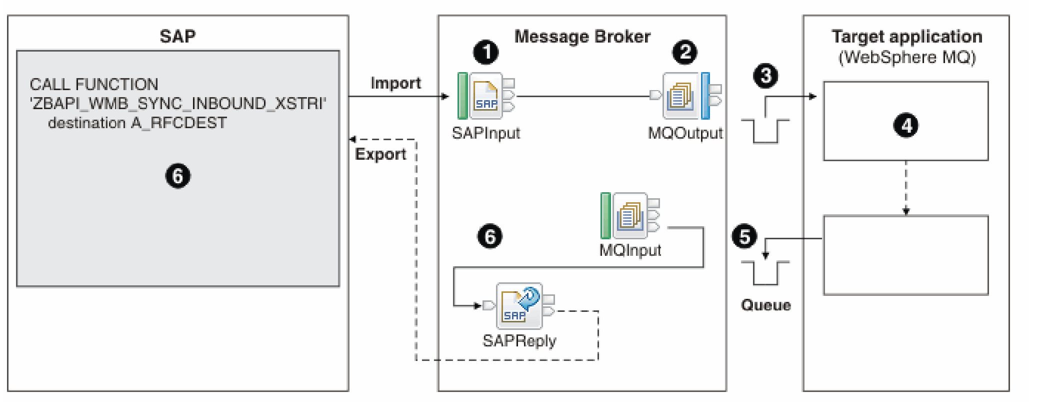


This example shows the first case, where the SAP application requires a response synchronously. The SAP application invokes an external application, and performs a blocked wait until the response is received. In the Message Broker system, this is provided by a single message flow. This message flow is initiated by a request placed on the SAP input node. In this example, the message flow communicates directly with a relational database, and returns the data directly to SAP using the SAP reply node. The target application can be any application that is invoked synchronously, such as an application in WebSphere Application Server, or a CICS® application on z/OS.

In this example, the SAP input and reply nodes are contained within a single message flow. You can define the input and reply nodes in separate message flows, although both flows must be deployed in the same execution group. You must also override the message identifier field. This is described later in this session.

Example 2 – Message Broker connects to MQ application

- SAP synchronous call to asynchronous application provider
- Two message flows to handle SAP request and response


SAP
CALL FUNCTION 'ZBAPI_WMB_SYNC_INBOUND_XSTRI' destination A_RFCDEST
6
Import
Export
Message Broker
1
2
SAPInput
MQOutput
6
MQInput
SAPReply
3
Queue
5
Target application (WebSphere MQ)
4

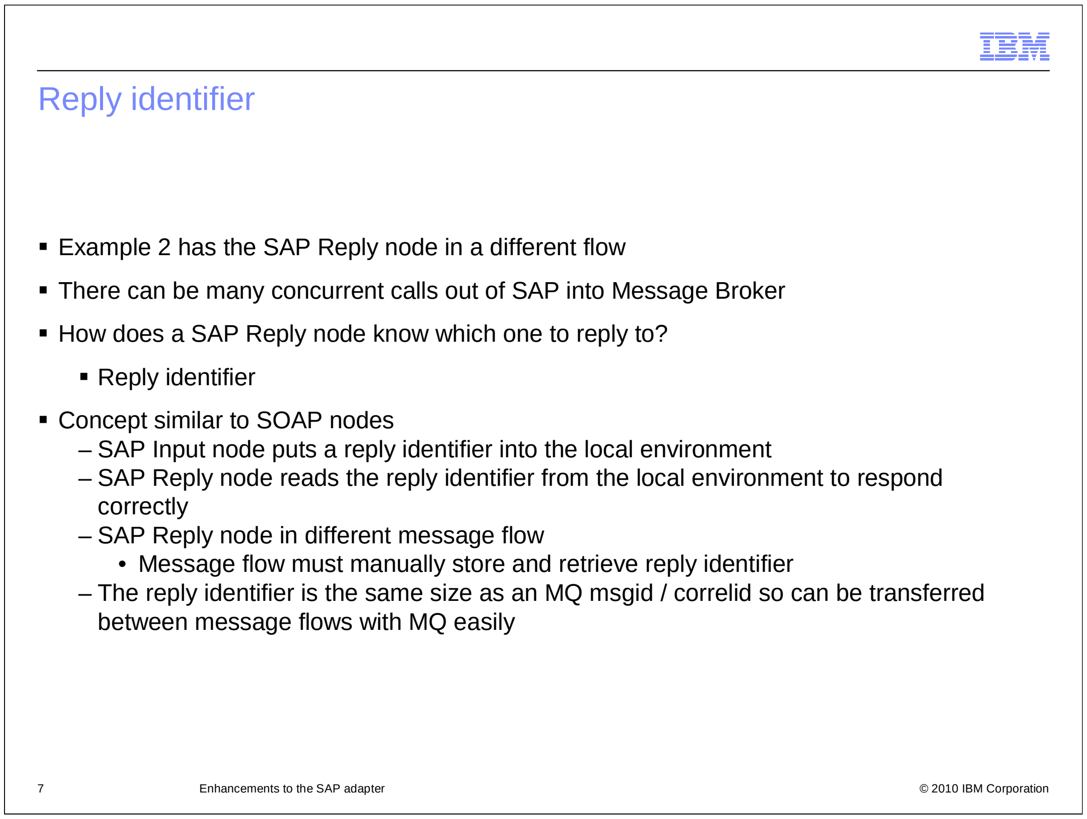
6 Enhancements to the SAP adapter © 2010 IBM Corporation

This example shows the second case, where the SAP application requires an asynchronous response. In this case, the Message Broker applications comprise two message flows. The first message flow is invoked by a request from the SAP, and is initiated with the SAP input node. This message flow extracts the data from the incoming SAP request and uses it to send to the provider application. This is done by sending an MQ message to the provider. The response from the provider is received asynchronously, and the MQ response message initiates the second message flow. This transforms that data to the format expected by the SAP application, and uses the SAP reply node to send the response.

Since the provider application is being invoked with MQ, the response message flow might be invoked some time after the request flow. Because the SAP reply node is in a separate message flow, it is necessary to provide a correlation between the SAP input node and reply node. This approach has the advantage that the two message flows are only active for a short time, and do not hold resources unnecessarily.

A variation on this example is to have the SAP input and reply nodes contained in a single message flow. The provider application can still be invoked using MQ, and an MQGet node can be used to retrieve the response from this application. In this case, the message flow instance might be active for some time; in particular, if the provider application is not active, this can result in many concurrent instances of the message flow. This might in turn result in degradation in the overall system performance.

In this asynchronous case, there are more performance considerations to consider. For example, you should consider the number of BAPI threads, and how these compare to the number of message flow instances, and the number of adapter instances.



Reply identifier

- Example 2 has the SAP Reply node in a different flow
- There can be many concurrent calls out of SAP into Message Broker
- How does a SAP Reply node know which one to reply to?
  - Reply identifier
- Concept similar to SOAP nodes
  - SAP Input node puts a reply identifier into the local environment
  - SAP Reply node reads the reply identifier from the local environment to respond correctly
  - SAP Reply node in different message flow
    - Message flow must manually store and retrieve reply identifier
  - The reply identifier is the same size as an MQ msgid / correlid so can be transferred between message flows with MQ easily

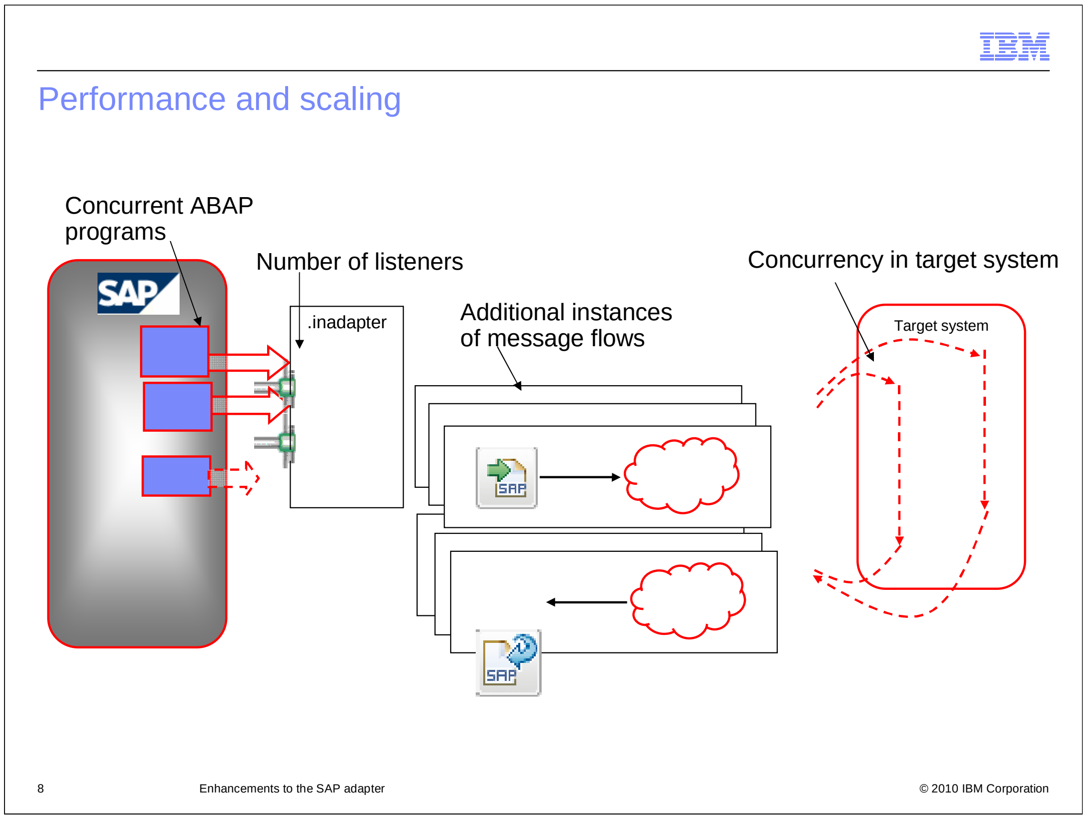
7 Enhancements to the SAP adapter © 2010 IBM Corporation

Example 2 has the SAP input and reply nodes contained in separate message flows. Since SAP can invoke many concurrent calls, you must provide a mechanism to correlate the response to SAP with the original request. This is done with the reply identifier.

The SAP Input node puts a reply identifier into the local environment which maps to a specific invocation of the message flow.

The SAP Reply node reads the reply identifier from the local environment to work out which invocation to reply to. If the reply node is in the same flow, and there is nothing in the local environment, then this is done automatically.

If the reply node is in a separate message flow, the reply identifier from the SAP input node must be stored and made available to the SAP reply node. This can be done in many ways. One way is to store the reply identifier in the MQMD correlation ID field in the MQ message. The reply identifier is the same size as an MQ message ID and correlation ID so can be transferred between message flows with MQ. This is the technique that is used in the SAP asynchronous sample in the Message Broker samples gallery.



Performance and scaling

The number of Message Broker components and MQ components can be significant in this SAP scenario.

The “Number Of Listeners” property on the “in adapter” defines the number of physical connections to SAP and therefore the number of concurrent callouts from SAP which can be handled at any one time.

The “number of additional instances” property on the message flow defines how many of those concurrent calls can be active within the message flow at any one time.

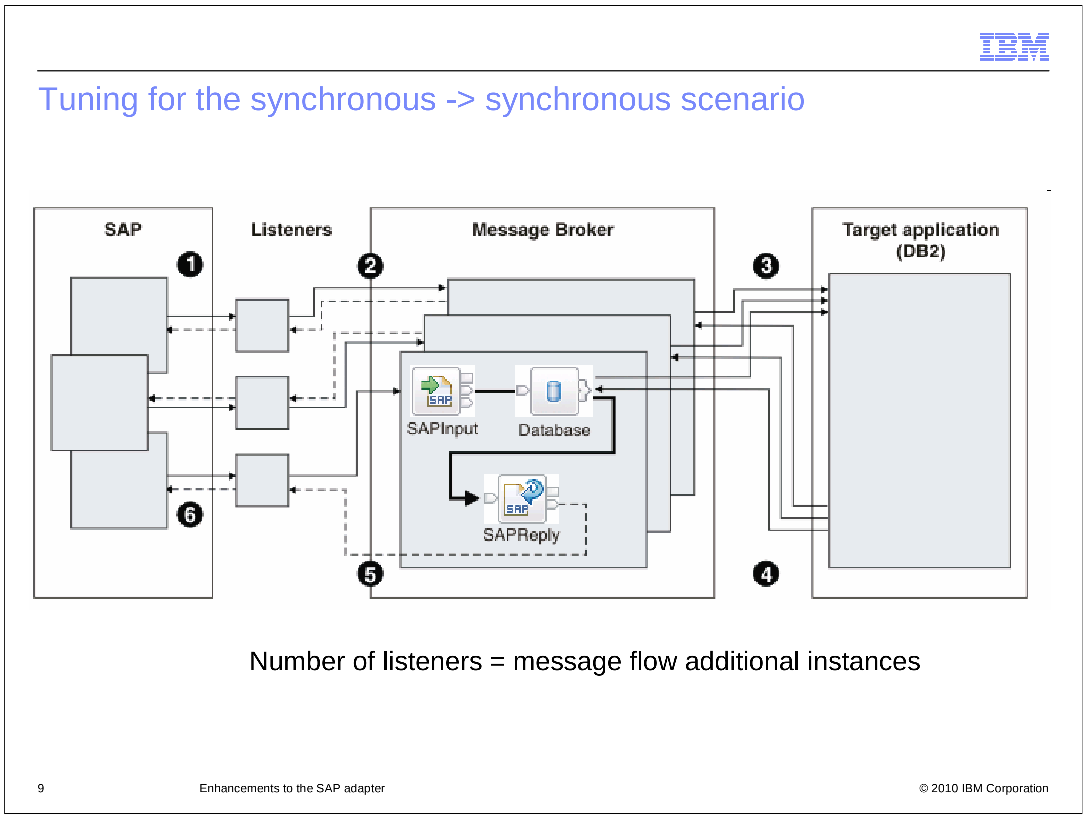
The best practice for balancing these properties depends on the scenario being implemented.

There are several places where you can configure for scalability. First, the number of listeners; this is the number of concurrent callouts from SAP which can be active at any one time. The resource associated with this number are the extra SAP connections. This needs to be configured in accordance with the SAP configuration.

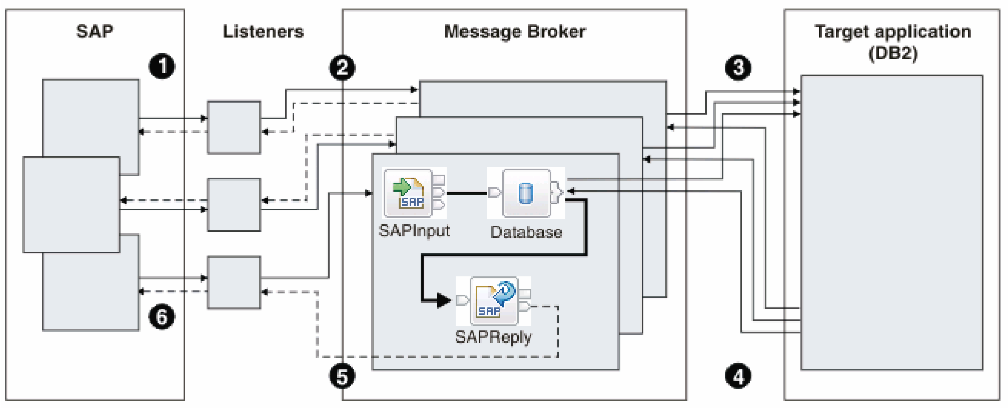
Second, the number of additional instances on the SAP Input Node; this is the number of message flow threads available to process the input message for the BAPI request. The resources associated with this number depend on the other nodes in the flow, for example the number of MQ connection handles and ODBC connections.

Third, the number of additional instances on the SAP Reply Node; this is the number of message flow threads available to process the reply message. The reply node might be on the same thread as the input node.

These parameters should be configured based on a range of usage values. This will include the expected frequency of call outs from SAP, the expected latency in target system, the complexity of the message flow, and the resources needed for each message flow instance. The SAP configuration itself should also be taken into account.



Tuning for the synchronous -> synchronous scenario



Number of listeners = message flow additional instances

In the synchronous reply scenario, the number of listeners is equal to the number of additional instances of the message flow. In this case, the scenario has been configured with three listeners and three message flow instances.

SAP makes three calls, each of which is received by a listener.

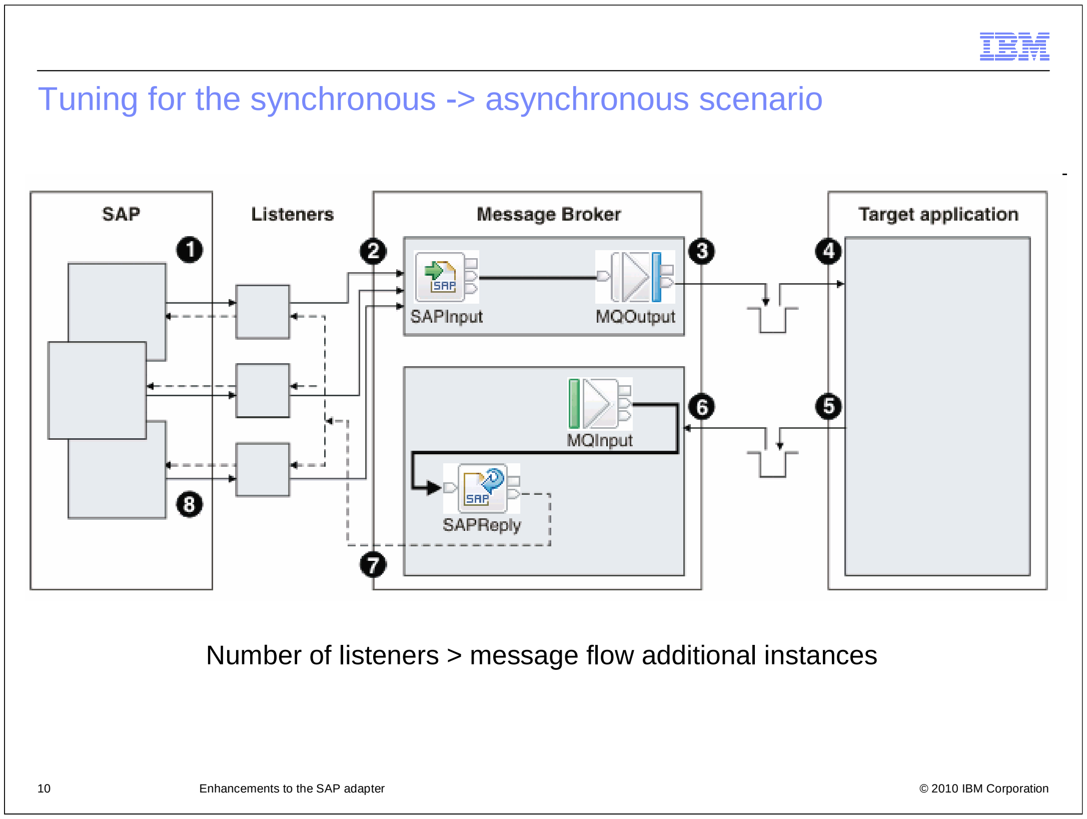
Each of the listeners sends the input parameters of each call to the SAP Input node in one of three instances of the message flow.

Each message flow instance sends its message to the target application.

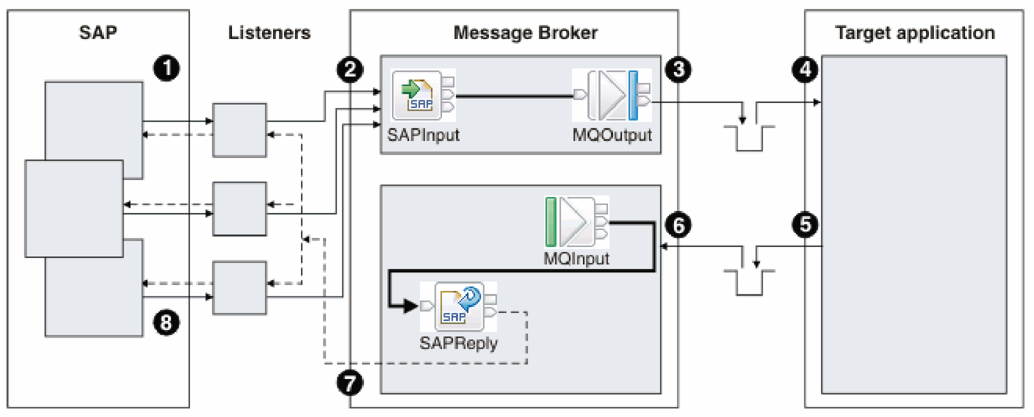
The target application processes the messages and returns replies to the message flow instances.

The SAP Reply node in each message flow instance sends the reply message to one of the three listeners.

Each listener returns the reply message to the appropriate SAP program.



Tuning for the synchronous -> asynchronous scenario



Number of listeners > message flow additional instances

10 Enhancements to the SAP adapter © 2010 IBM Corporation

In the asynchronous reply scenario, the message flow has low latency compared to the time taken by the entire scenario. Therefore, you can limit the resources that are used by the message flow containing the SAP Input node by configuring fewer additional instances for this message flow.

One instance of the message flow can service many listeners because the message flow propagates the import parameters quickly for processing.

In this example, SAP makes three calls, each of which is received by a listener.

Each of the listeners sends the input parameters of each call to a message flow that contains an SAP Input node.

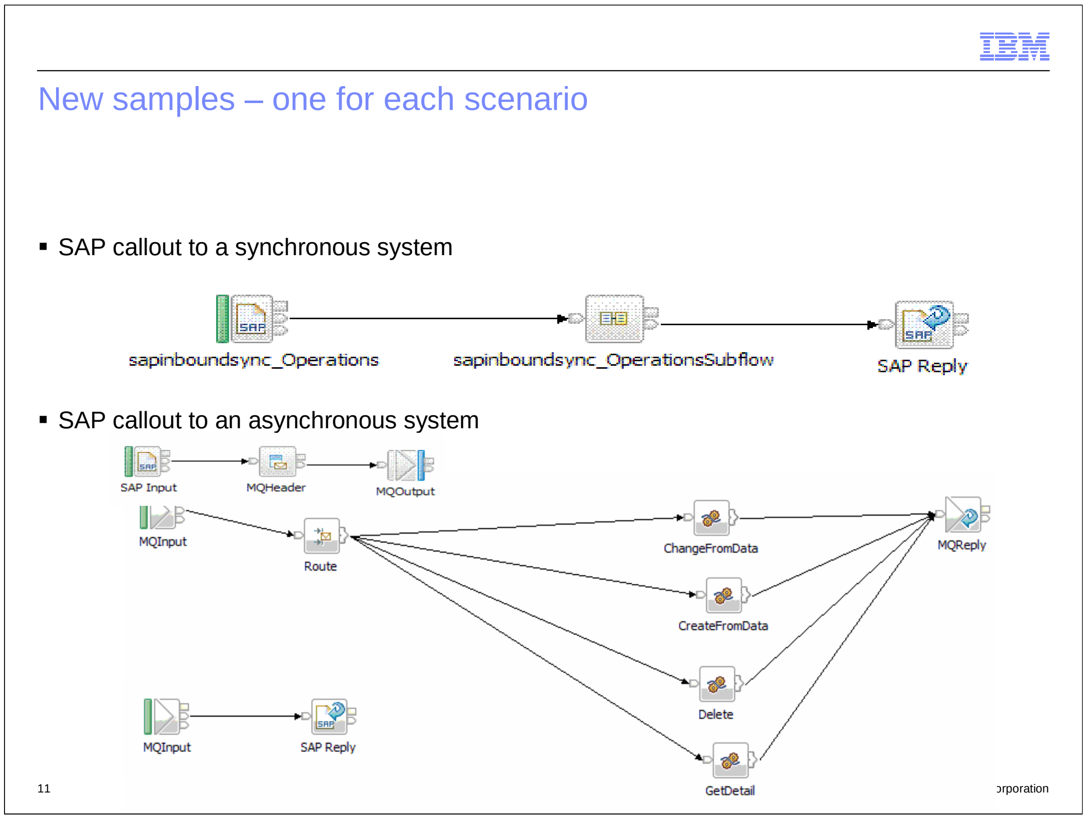
The message flow puts the message onto a queue for the target application. This message flow instance is now free to service another listener while the first listener waits for the reply to come back from the target system.

The target application gets the messages from the queue, processes them, and puts the reply messages onto a queue.

A second message flow that contains an SAP Reply node gets the messages from the queue and processes them.

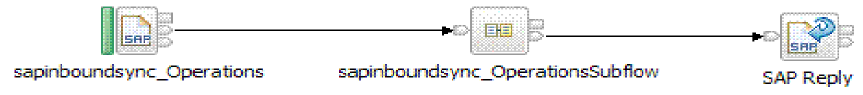
The SAP Reply node sends the reply messages to the appropriate listeners.

Each listener returns a reply message to the appropriate SAP program.

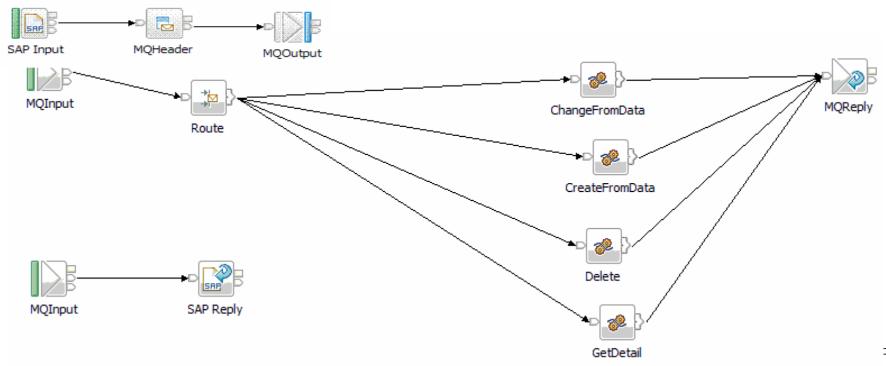


New samples – one for each scenario

- SAP callout to a synchronous system



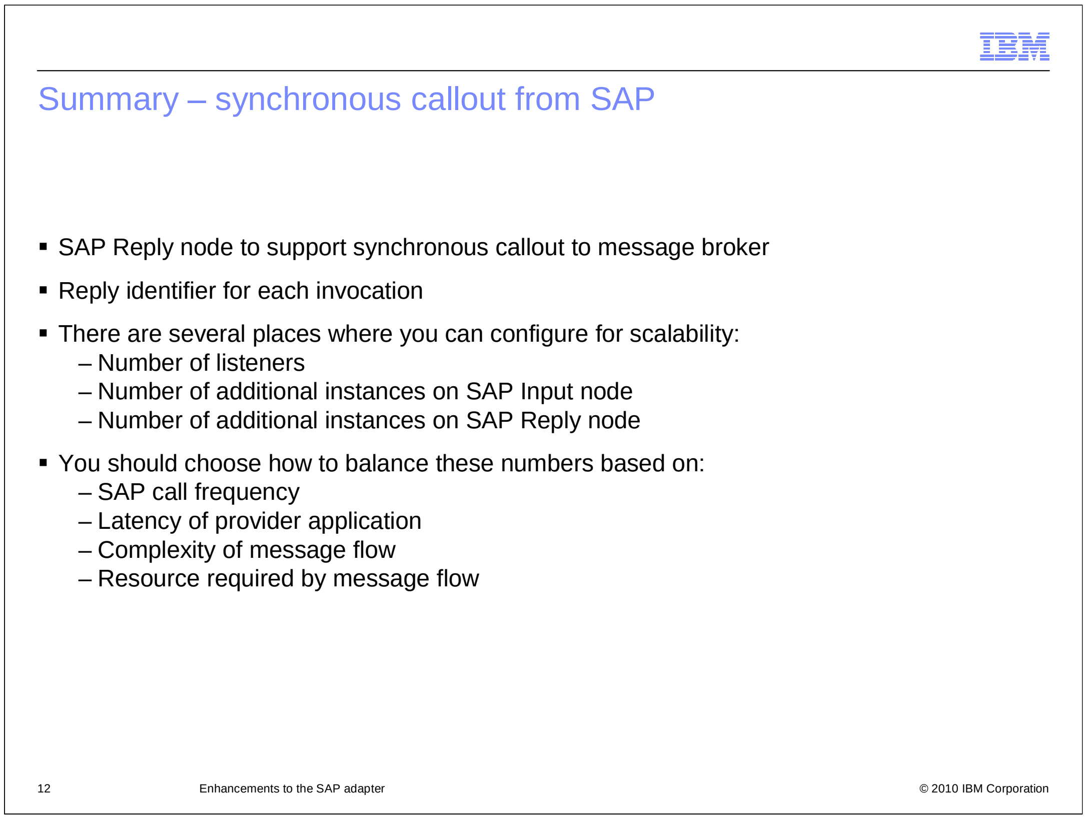
- SAP callout to an asynchronous system



11 orporation

Two new samples have been provided in the samples gallery for the synchronous call interface. These samples show the basic structure of the SAP input and reply nodes, and show how the reply identifier is used in the case of the asynchronous response.

As shown on this screen capture, the application provider in the case of the asynchronous response is provided by a third message flow, which receives and responds with an MQ message. This MQ message carries the SAP reply identifier in the MQMD.



Summary – synchronous callout from SAP

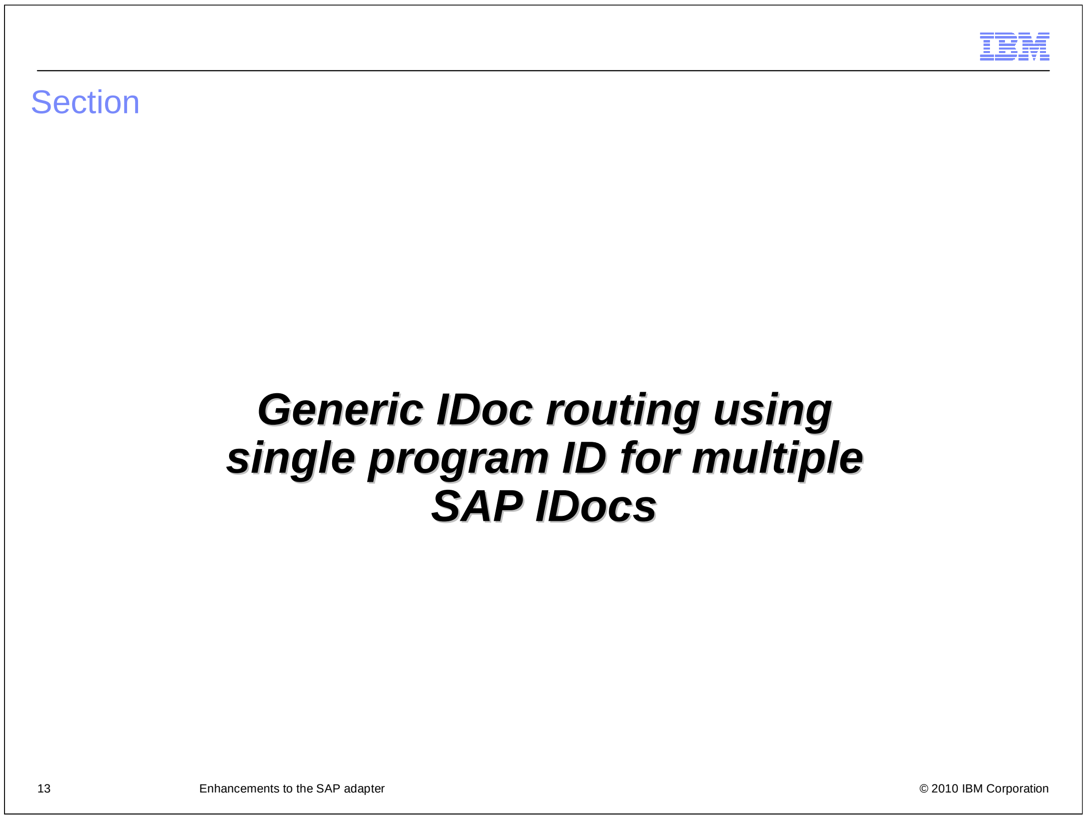
- SAP Reply node to support synchronous callout to message broker
- Reply identifier for each invocation
- There are several places where you can configure for scalability:
  - Number of listeners
  - Number of additional instances on SAP Input node
  - Number of additional instances on SAP Reply node
- You should choose how to balance these numbers based on:
  - SAP call frequency
  - Latency of provider application
  - Complexity of message flow
  - Resource required by message flow

12 Enhancements to the SAP adapter © 2010 IBM Corporation

In summary, the new SAP Reply node provides support for the SAP synchronous callout to message broker.

A reply identifier for each invocation; if the request and reply are in different message flows; you must specify the value of the reply identifier.

In this type of connection, performance tuning is important. Message Broker provides several places where tuning can be performed. These values will be dependent on your precise system requirements and configuration.

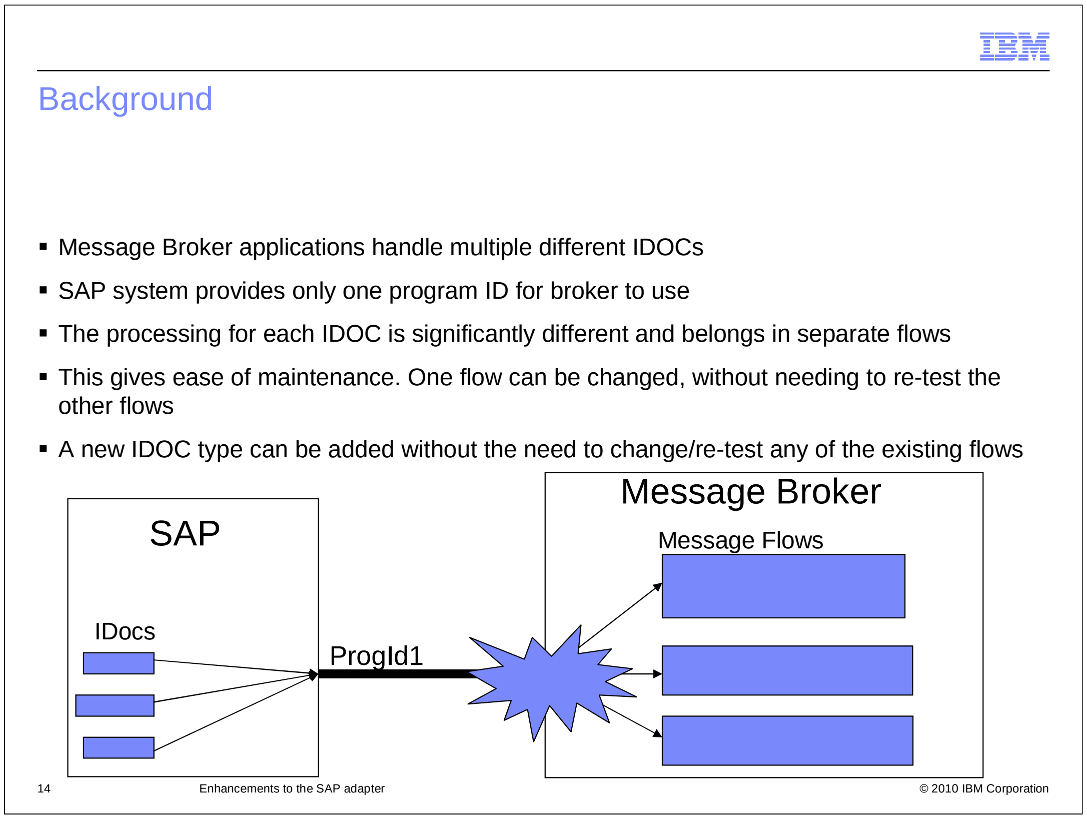


Section

Generic IDoc routing using single program ID for multiple SAP IDocs

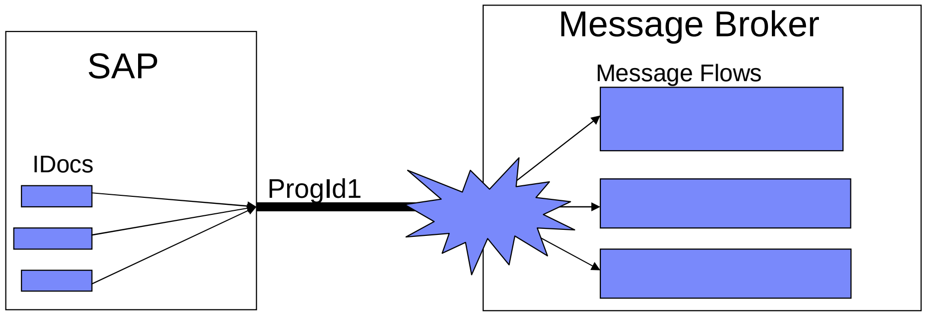
13 Enhancements to the SAP adapter © 2010 IBM Corporation

This section describes the new generic IDOC routing function in Message Broker version 7. This is also known as the “single program ID” function, since it uses this SAP function.



Background

- Message Broker applications handle multiple different IDOCs
- SAP system provides only one program ID for broker to use
- The processing for each IDOC is significantly different and belongs in separate flows
- This gives ease of maintenance. One flow can be changed, without needing to re-test the other flows
- A new IDOC type can be added without the need to change/re-test any of the existing flows



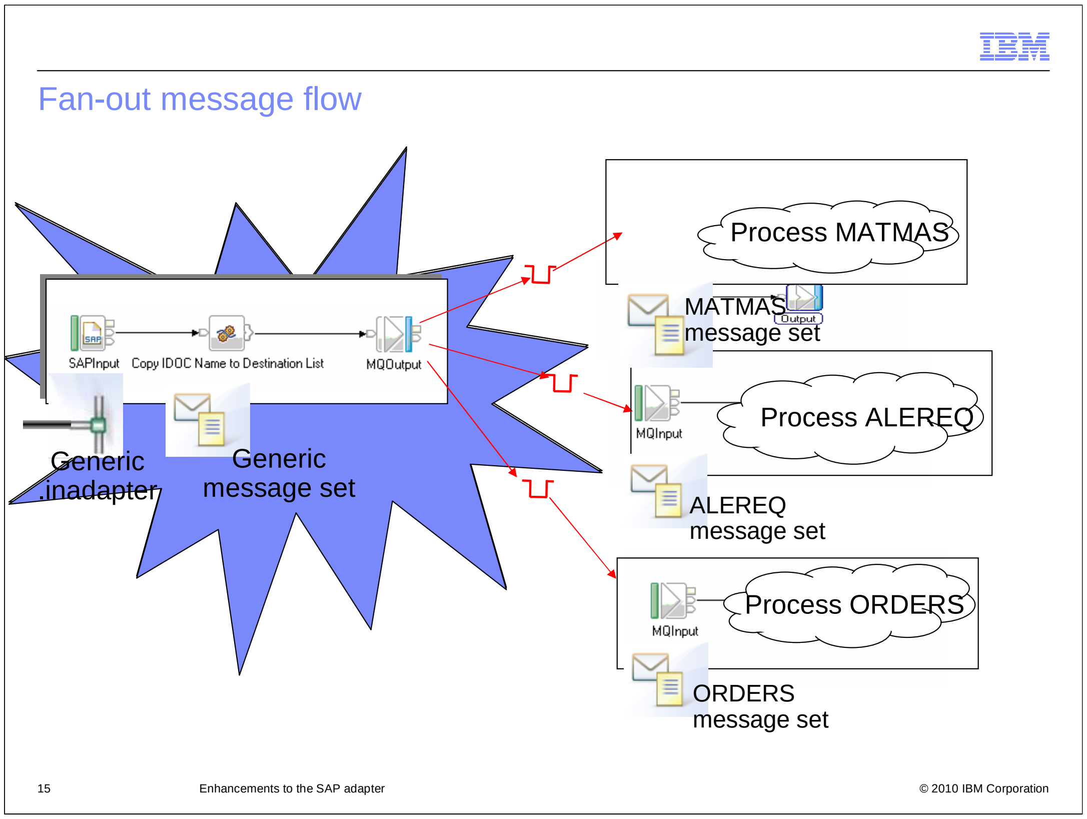
14 Enhancements to the SAP adapter © 2010 IBM Corporation

This slide describes the scenario that the single program ID function addresses. Most SAP systems have many IDOCs, perhaps as many as several hundred different IDOC definitions. In previous versions of Message Broker, for a message flow to process these different IDOCs, each defined adapter had to have a separate program ID destination. This is analogous to an MQ channel in an MQ environment.

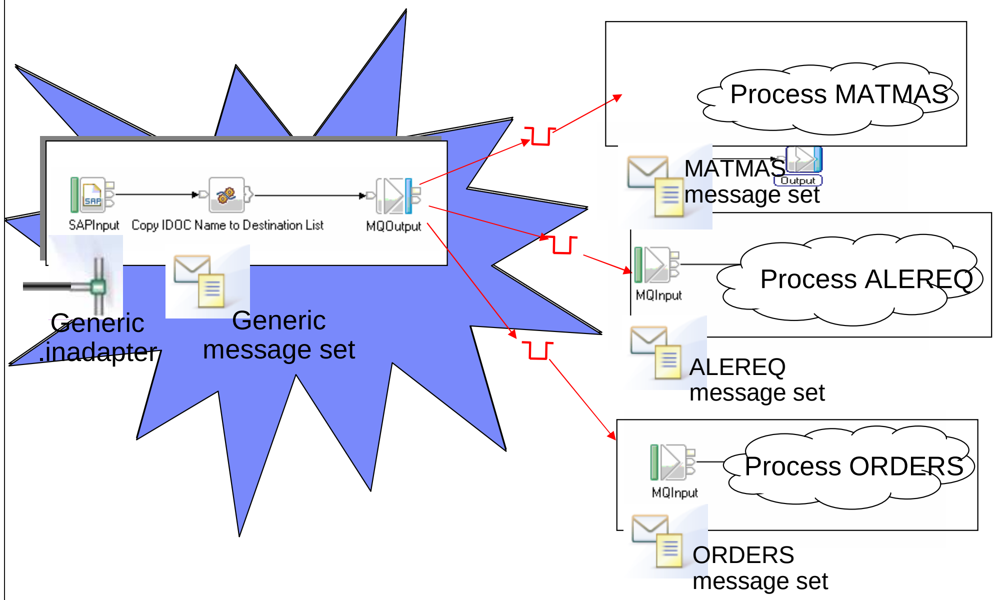
Alternatively, a single message flow must be provided which can handle all possible IDOCs from SAP. This means that the message set must contain all possible IDOC message definitions, and a single message flow must provide the processing logic for all IDOCs.

In most cases, it is desirable to be able to add new message flows, to accommodate new IDOCs, without having to define additional program IDs in SAP. This is addressed with the single program ID function in Message Broker version 7.

This means that new IDOCs can be added easily, and a corresponding new message flow can be written to receive this IDOC. Development and testing can therefore be isolated, and changes made in one message flow will not affect the other message flows.



Fan-out message flow



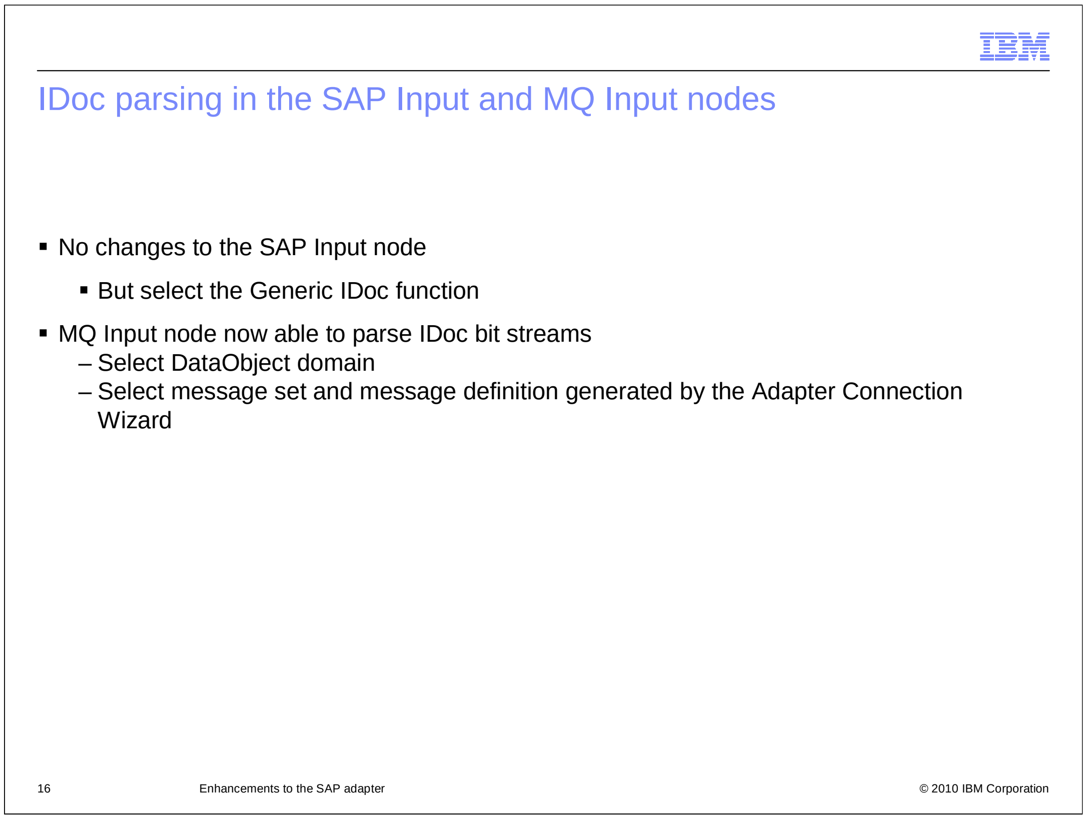
15 Enhancements to the SAP adapter © 2010 IBM Corporation

In Message Broker version 7, you can create a single message flow which provides the connection to SAP. This flow will contain the SAP Input node, and will use the defined adapter to connect to SAP. However, this message flow will not be concerned about the specific format of the received IDOC. The generic IDOC message set will allow this message flow to determine the specific IDOC type. The rest of the message data will be represented as a binary string, and will be passed to another message flow for specific processing.

In the example shown here, the first message flow will examine the IDOC message, and determine the name of the specific IDOC that it has received. This information should be written to the destination list in the local environment. Using this information, the flow will then place the output message onto the specific MQ queue, as determined by the IDOC type. A second message flow, specifically designed to process the particular IDOC, will then receive that message, and process it as required.

The second message flow is started by a message arriving on its MQ input queue. On the MQ input node, this message flow applies the message definition that is appropriate to the specific IDOC that it designed for. This message definition is defined by the adapter wizard. When you run the adapter wizard for this flow, you should just generate the message definition. The adapter definition that is also created can be removed. When you specify the parsing for the MQ Input node, select the DataObject domain, and select the specific message definition for this flow which was created by the wizard. When this flow runs, it will parse the incoming message with this IDOC message definition. The resulting barfile for this flow will contain only the message flow and the message set. The SAP adapter is not required for this flow.

Note that since the first message flow and the second message flow are connected using MQ, these flows can be deployed in separate execution groups, or even in separate broker instances. Using the Message Broker Remote Adapter Deployment, you can deploy the message flow that connects to SAP in the Remote Adapter broker instance, and deploy the main processing flow in a separate larger broker instance. You can also place these main processing flows in several broker instances, to enable the processing workload to be split across multiple systems.



IDoc parsing in the SAP Input and MQ Input nodes

- No changes to the SAP Input node
  - But select the Generic IDoc function
- MQ Input node now able to parse IDoc bit streams
  - Select DataObject domain
  - Select message set and message definition generated by the Adapter Connection Wizard

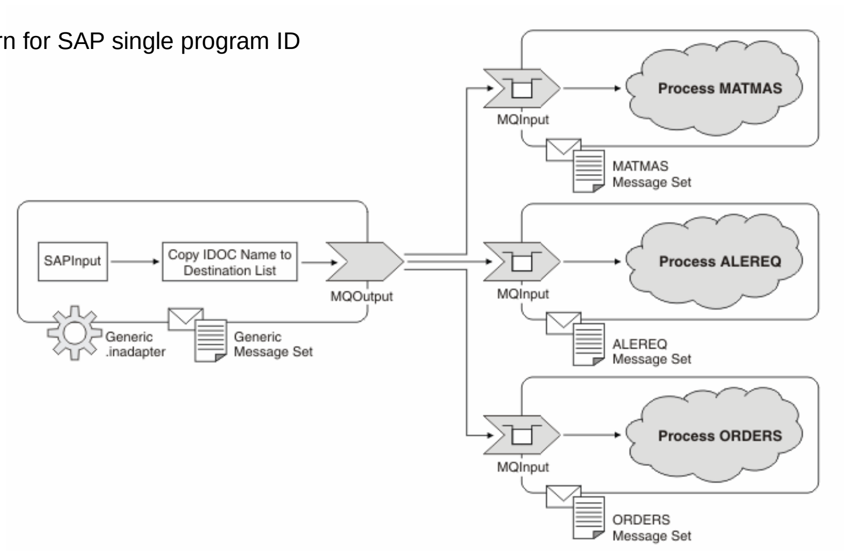
16 Enhancements to the SAP adapter © 2010 IBM Corporation

This new capability has not changed the external implementation. The SAP Input node works in exactly the same way as in previous versions. However, to take advantage of this capability, you should use the generic IDoc parsing, rather than parsing a specific IDoc. This means that any received IDOC will be passed through this node unparsed.

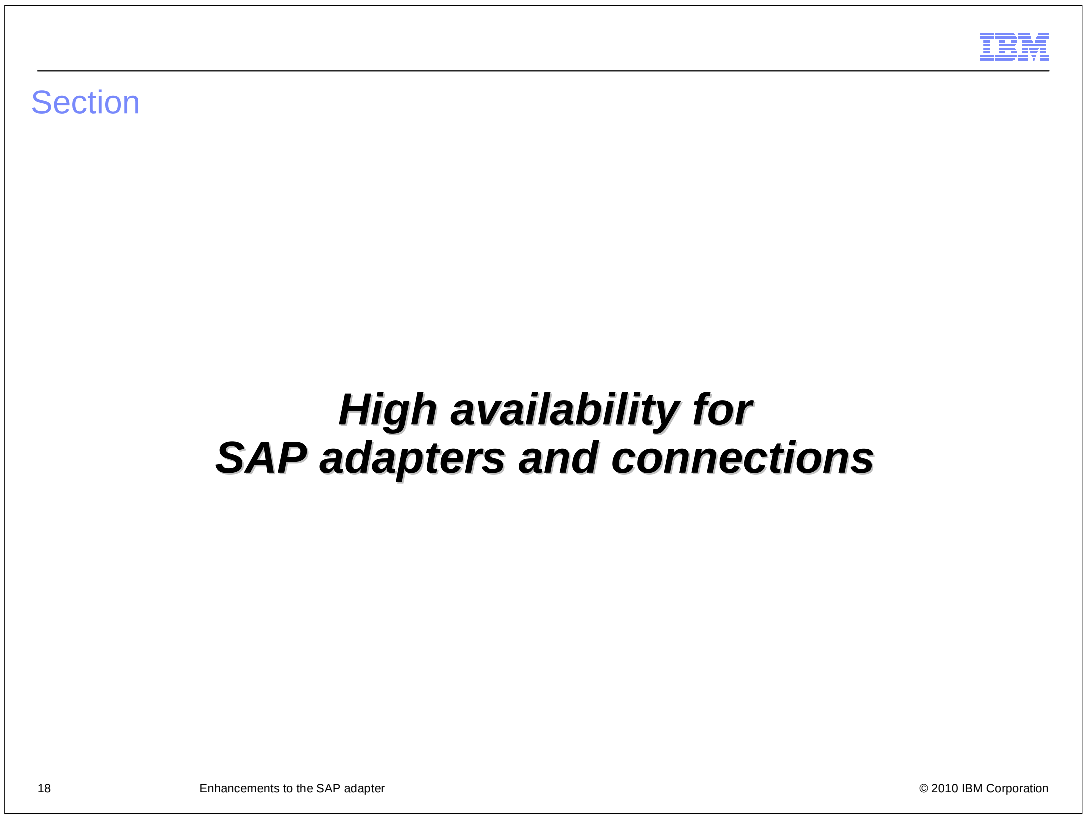
On the MQ input node in the second message flow, select “DataObject” domain, and select the specific message set and message definition required for the specific IDoc. This will have been created by the SAP Adapter connection wizard.

Supplied pattern

- Pattern for SAP single program ID



Message Broker version 7 provides several patterns, which are templates for creating specific instances of message flows using a standard design approach. The pattern for the SAP single program ID is shown in this screen capture, and provides a starting point for your own applications.

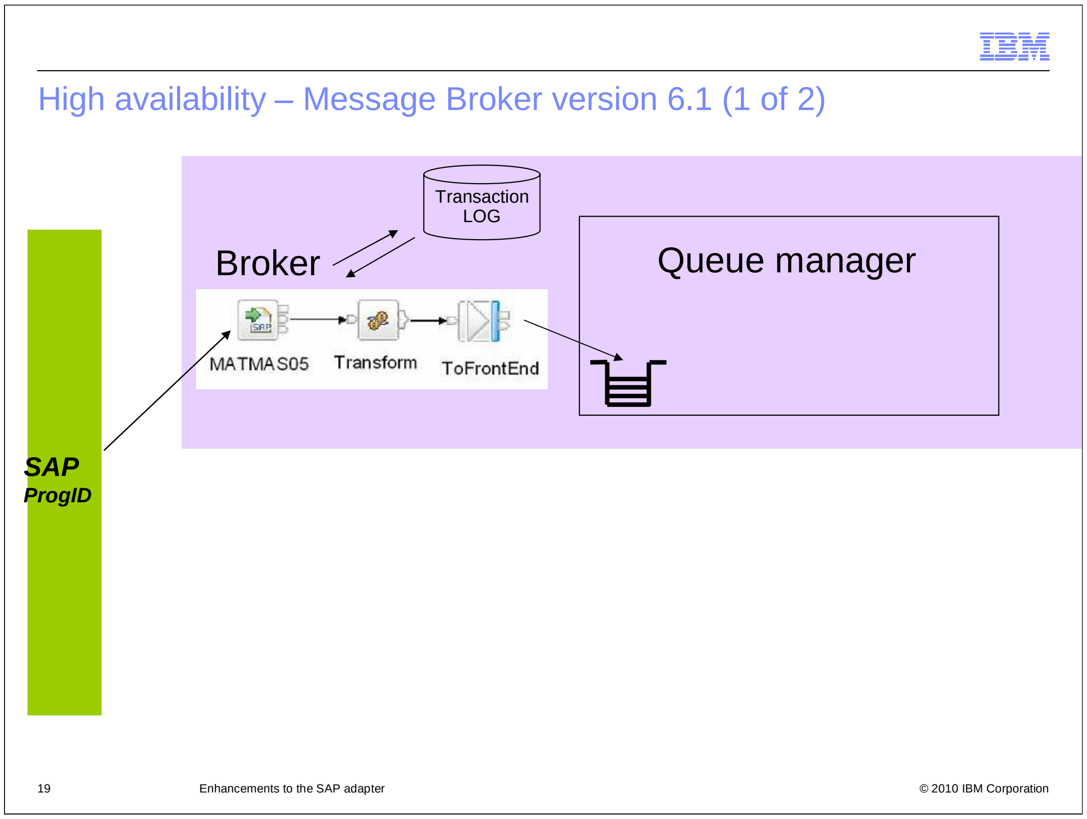


Section

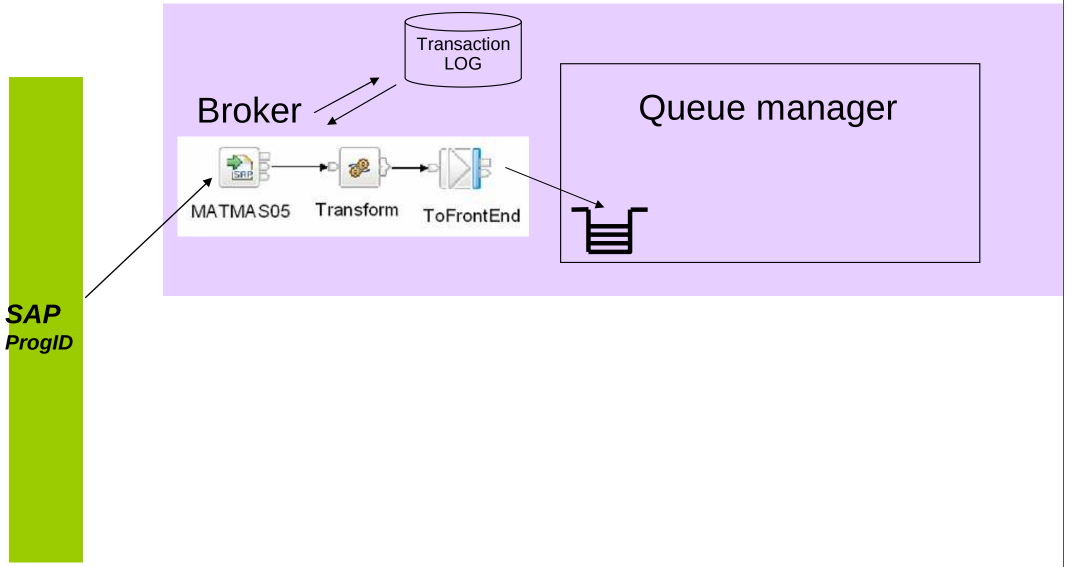
High availability for SAP adapters and connections

18 Enhancements to the SAP adapter © 2010 IBM Corporation

This section covers high availability for SAP systems. Many SAP systems are configured for high availability, and the corresponding integration components, such as Message Broker, should be able to provide the same level of availability. This has now been provided in Message Broker version 7.



High availability – Message Broker version 6.1 (1 of 2)



19 Enhancements to the SAP adapter © 2010 IBM Corporation

The ”tRFC” protocol between SAP and the RFC Server, WebSphere Message Broker in this case, ensures “exactly once” delivery of IDOCs or “tRFC BAPI” calls.

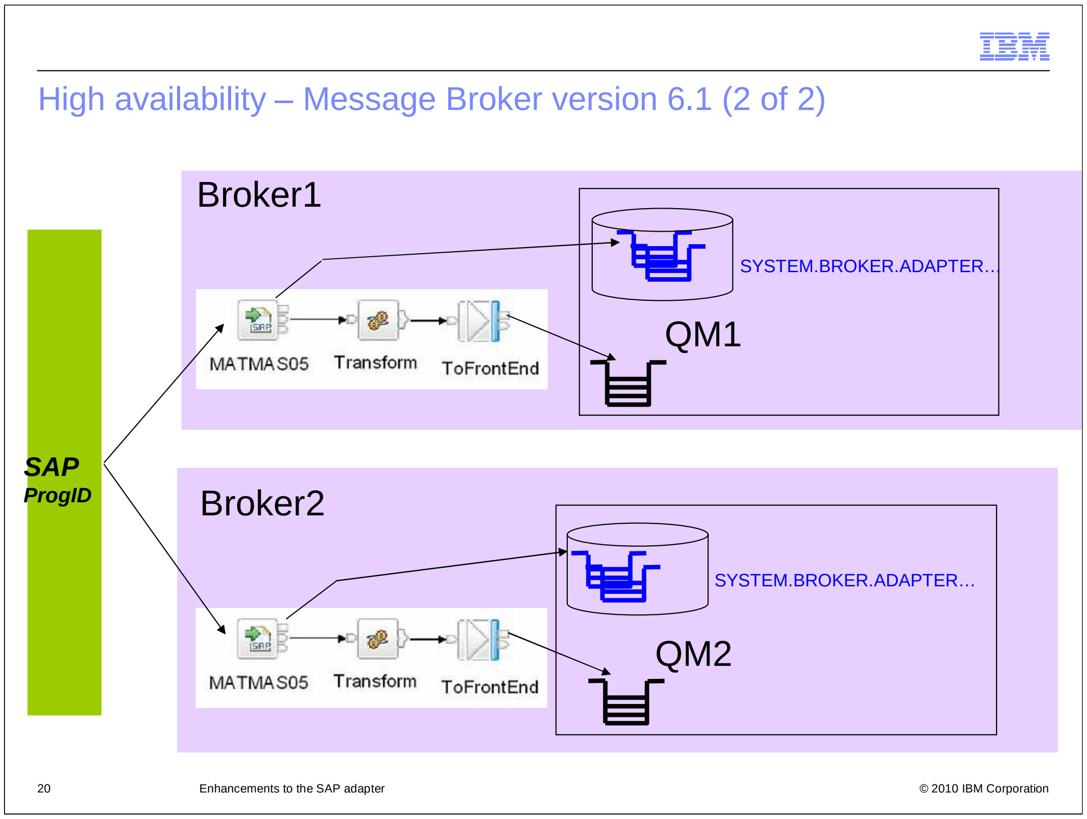
This is possible because each delivery has an associated Transaction ID, which is stored on the transaction log. The transaction log is implemented as an MQ queue, defined on the local queue manager of the broker.

The RFC Server is responsible for keeping track of the progress of a delivery until such time that SAP signals confirmation of a successful delivery.

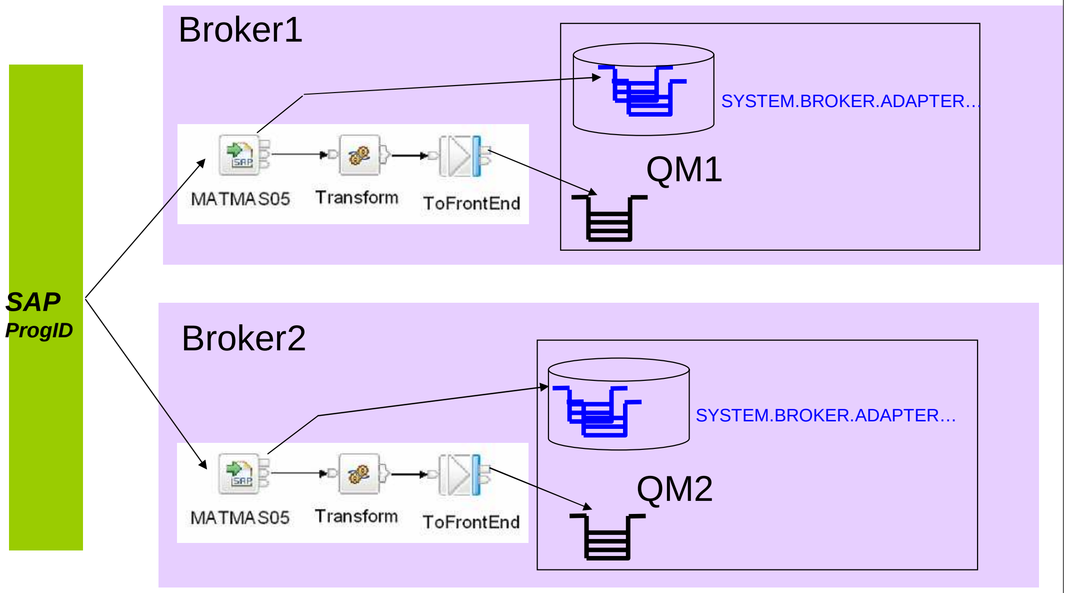
If the connection is lost or the RFC Server terminates before that confirmation, then SAP will attempt to re-deliver on another connection.

By keeping a persistent record transaction log, the broker can ensure integrity and avoid duplicate delivery. Even if SAP attempts to resend the same information that has been committed in the message flow, Message Broker will reject this resent data, because it has been written in the transaction log.

However, this scenario does not provide a highly available solution. There is a only a single broker instance, and if this broker system fails for any reason, then no SAP requests can be processed.



High availability – Message Broker version 6.1 (2 of 2)



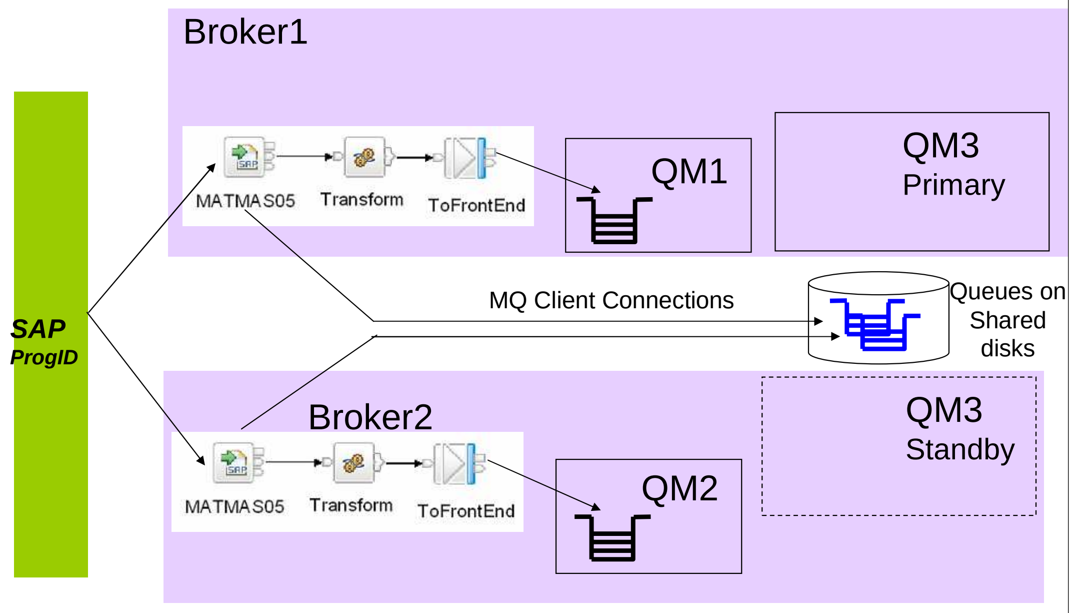
20 Enhancements to the SAP adapter © 2010 IBM Corporation

When two “in adapters” with the same Program ID are deployed to two brokers, these appear to SAP as two connections to the same RFC Server. They are regarded by SAP as two threads on the same logical connection.

In the example shown here, if Broker1 fails, and the connection from SAP to Broker1 is lost, then SAP might attempt to re-deliver the message to the Broker2 system.

As Broker1 and Broker2 have separate transaction stores, then Broker2 will accept re-delivery even though the Broker1 might have processed some, or even all, the IDOCs in the packet. In this scenario, there is a data integrity issue.

High availability – Message Broker version 7



21 Enhancements to the SAP adapter © 2010 IBM Corporation

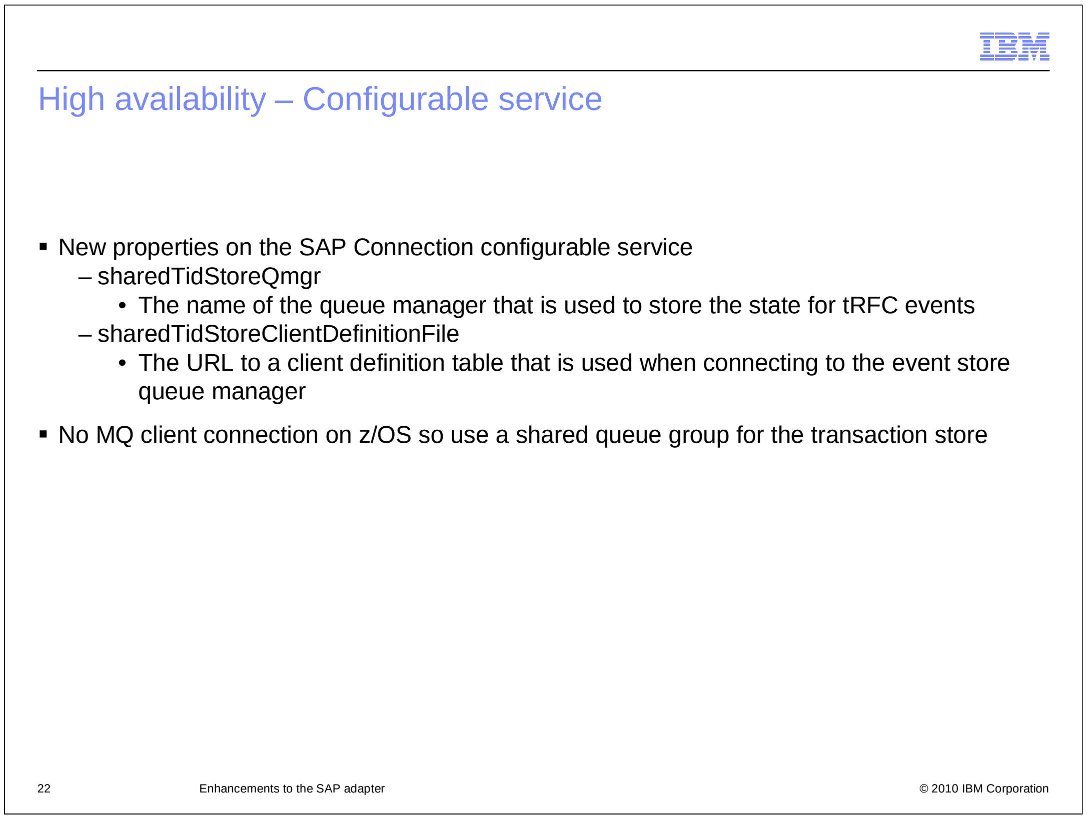
In Message Broker version 7, the transaction log can be moved to a separate queue manager, which can be shared between two brokers.

The two brokers connect to the separate queue manager using MQ client connections.

In the example shown here, the system running Broker1 might fail after the message flow has successfully committed its output, and the transaction log has been written. SAP might then resend the same IDOC, and this would be received by Broker2. Broker2 has access to the same transaction log, hosted by the shared queue manager. Broker2 will reject this resent request, because it knows that the transaction has already been processed by Broker1.

The separate queue manager, QM3 in this example, can be a local or remote queue manager. The brokers connect to this queue manager using MQ client connections, not local bindings, so you can place this queue manager in the most appropriate location for your requirements.

Using the MQ support for Active and Standby queue managers, QM3 in this example is not a single point of failure..



High availability – Configurable service

- New properties on the SAP Connection configurable service
  - sharedTidStoreQmgr
    - The name of the queue manager that is used to store the state for tRFC events
  - sharedTidStoreClientDefinitionFile
    - The URL to a client definition table that is used when connecting to the event store queue manager
- No MQ client connection on z/OS so use a shared queue group for the transaction store

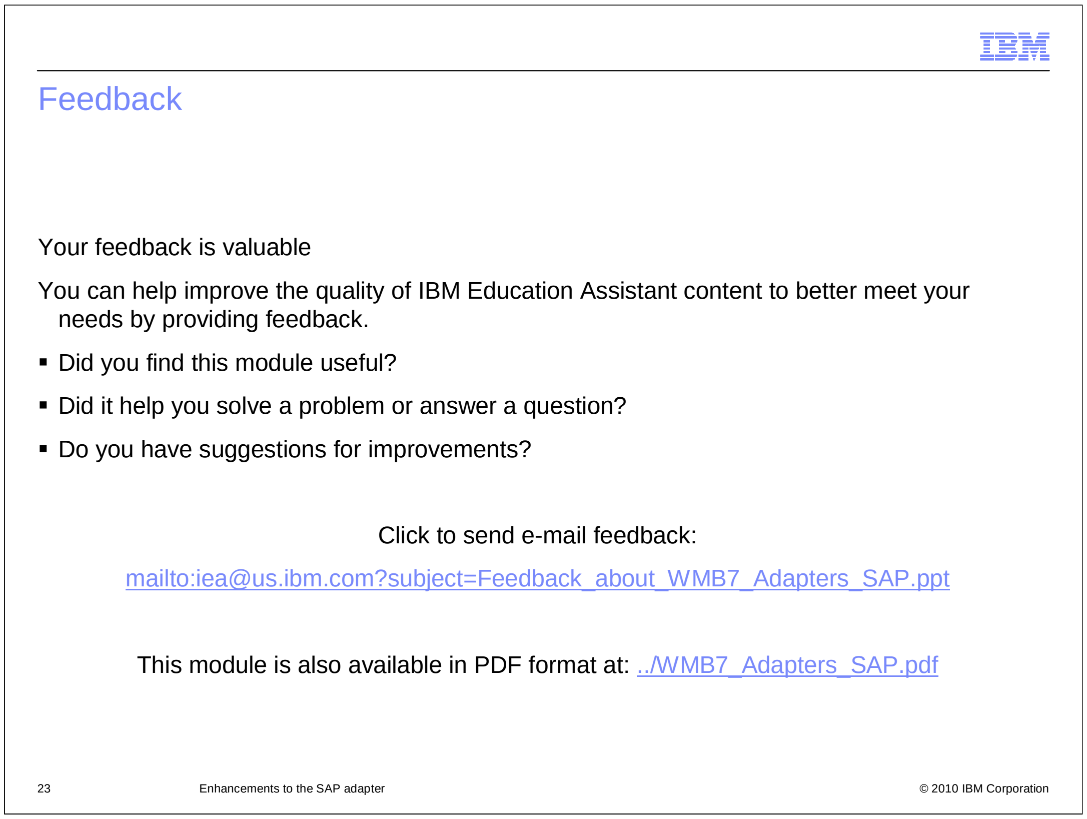
22 Enhancements to the SAP adapter © 2010 IBM Corporation

Two new configurable services have been introduced for these high availability configurations.

The first specifies the name of the queue manager that is used to host the transaction log store. Setting this property means that brokers on separate systems can use the same RFC program ID to process outbound requests from SAP, whilst maintaining data integrity. If this configurable service is not set, the transaction log will be defined on the local queue manager of the broker.

The second specifies the location of an MQ client connection file that can be used to enable the broker to connect to the remote queue manager.

If you are deploying Message Broker on z/OS, then MQ client connections are not available. In this case, the same approach is used, but is implemented with an MQ shared queue group.



Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WMB7_Adapters_SAP.ppt

This module is also available in PDF format at: ../WMB7_Adapters_SAP.pdf

23 Enhancements to the SAP adapter © 2010 IBM Corporation

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, disclaimer, and copyright information