WebSphere Business Integration for Financial Networks

Sample routing message flow using AMiT

Readme

Note

Before using this information and the product it supports, be sure to read the general information under “Notices” on page v.

First Edition, August 2004

Readme for SupportPac: WebSphere BI for FN Sample Routing Message Flow using AMiT.

Contents

Preface

What's new in this release

This is the initial release.

Notices

The following paragraph does not apply in any country where such provisions are inconsistent with local law.

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you. References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM licensed program or other IBM product in this publication is not intended to state or imply that only IBM's program or other product may be used. Any functionally equivalent program that does not infringe any of the intellectual property rights may be used instead of the IBM product. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility. IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594, USA.

The information contained in this document has not be submitted to any formal IBM test and is distributed AS IS. The use of the information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item has been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

- WebSphere MQ
- IBM

Introduction

The message flow contained in this support pac routes messages to different queues depending on various conditions. You can modify these conditions and the names of the queues to meet your particular message routing needs.

Note: This manual assumes you are already familiar with:

- Active Middleware Technology (AMiT) concepts such as *situation*, *event*, *situation attribute*, and *threshold condition* (for more information, see *AMiT User's Guide*)
- Use of the SWIFT message set

Routing situations

A *routing situation* is a situation that determines to which destination queue a message is routed, for example:

```
if situationX.amount ≥ 100000 then queueName = 'HIGHVALUE'
   -------------------------      ----------------------
      threshold condition            situation attribute
```

A situation is triggered (in AMiT terminology, it is said to have been *detected* and *notified*) when the threshold conditions of the events that comprise its condition are evaluated and found to be true. When a situation is triggered, it sets a *situation attribute*. In the previous example, the situation attribute **queueName = 'HIGHVALUE'** is set when the threshold condition **amount ≥ 100000** is evaluated and found to be true.

A situation can have, as its condition, any number of events, and these events can be arbitrarily complex; for example:

```
if
Message_Type is one of ('103','202','205','910','920','940','950')
and
Currency = 'EUR'
and
Amount ≥ 100000
then
queueName = 'HIGHVALUE'
```

Routing situations typically have a routing order; that is, they are to be triggered only if certain other routing situations, which were evaluated earlier, were not triggered. For example:

```
if EventA=true
 then route to RED
if EventB=true
 then route to BLUE
if EventC=true
 then route to GREEN
if EventD=true
 then route to YELLOW
```

Figure 1. Example of situations with a processing order

In this example, the routing situations have an order, so that if condition A is evaluated and found to be true, the message is routed to RED, evaluation stops, and the other situations are ignored.

However, AMiT always evaluates all situations, and unless there are dependencies among them, the order in which situations are evaluated is arbitrary. As a result, when using AMiT, you must do one of the following:

- Implement all routing situations so that they are mutually exclusive; that is, each routing situation must be defined in such a way that it is not triggered if any other routing situation is triggered. This can result in situations becoming complex and unwieldy. For example, in AMiT, the situations shown in Figure 1 on page 1 would have to be implemented like this:

```
if EventA=true and EventB=false and EventC=false and EventD=false
 then route to RED
if EventB=true and EventA=false and EventC=false and EventD=false
 then route to BLUE
if EventC=true and EventA=false and EventB=false and EventD=false
 then route to GREEN
if EventD=true and EventA=false and EventB=false and EventC=false
 then route to YELLOW
```

Figure 2. Example of situations without a processing order

- Implement all routing situations in such a way that each situation is dependent on at least one other situation. You can use *internal situations* (that is, situations that are only recognized within the AMiT engine and not reported externally) to create chains of situations that are triggered only if other situations have already been evaluated. This is the method employed by the sample routing message flow.

  In the following example, the internal situations S1, S2, S3, and S4 are defined, and each is used as part of the condition of the situation that follows it in the processing order:

```
S1: if EventA=true
 then route to RED
S2: if EventB=true and situation S1=false
 then route to BLUE
S3: if EventC=true and situation S2=false
 then route to GREEN
S4: if EventD=true and situation S3=false
 then route to YELLOW
```

Figure 3. Example of situations without a processing order using internal situations

  Note that, for example:

  – S2 can be triggered only if S1 is not triggered (this is part of the condition of S2)
  – S2 triggering precludes S3 from triggering (this is part of the condition of S3)
  – S2 triggering precludes S4 from triggering (S3 not triggering is part of the condition of S4, and S2 triggering precludes S3 from triggering)

The structure of the sample routing message flow

The sample routing message flow is shown in Figure 4 on page 3.

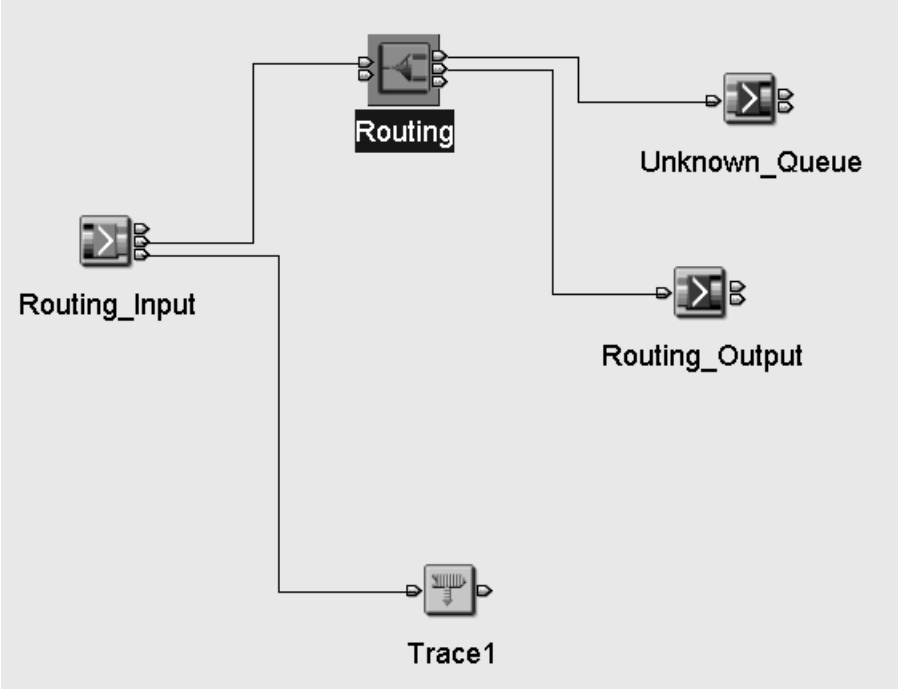

Figure 4. Sample routing message flow

The heart of the sample routing message flow is the AMiT IntelligentFilter node, which propagates messages to one of two output terminals:

- If AMiT detects one or more external situations, the AMiT IntelligentFilter node inserts the name of the destination queue into the LocalEnvironment folder of the current message, and propagates the message to the **conditional out** terminal.
- If AMiT does not detect an external situation, the AMiT IntelligentFilter node propagates the message to the **out** terminal.

If the message flow receives a message that is not defined in the SWIFT message set, or if there is an error during processing, its Trace node discards the message. If more sophisticated error handling is required, replace the Trace node, which is connected to the catch terminal of the MQInput node, with an error-handling subflow.

The decision tree shown in Figure 5 on page 4 illustrates the processing order of the routing situations used by the sample routing message flow.

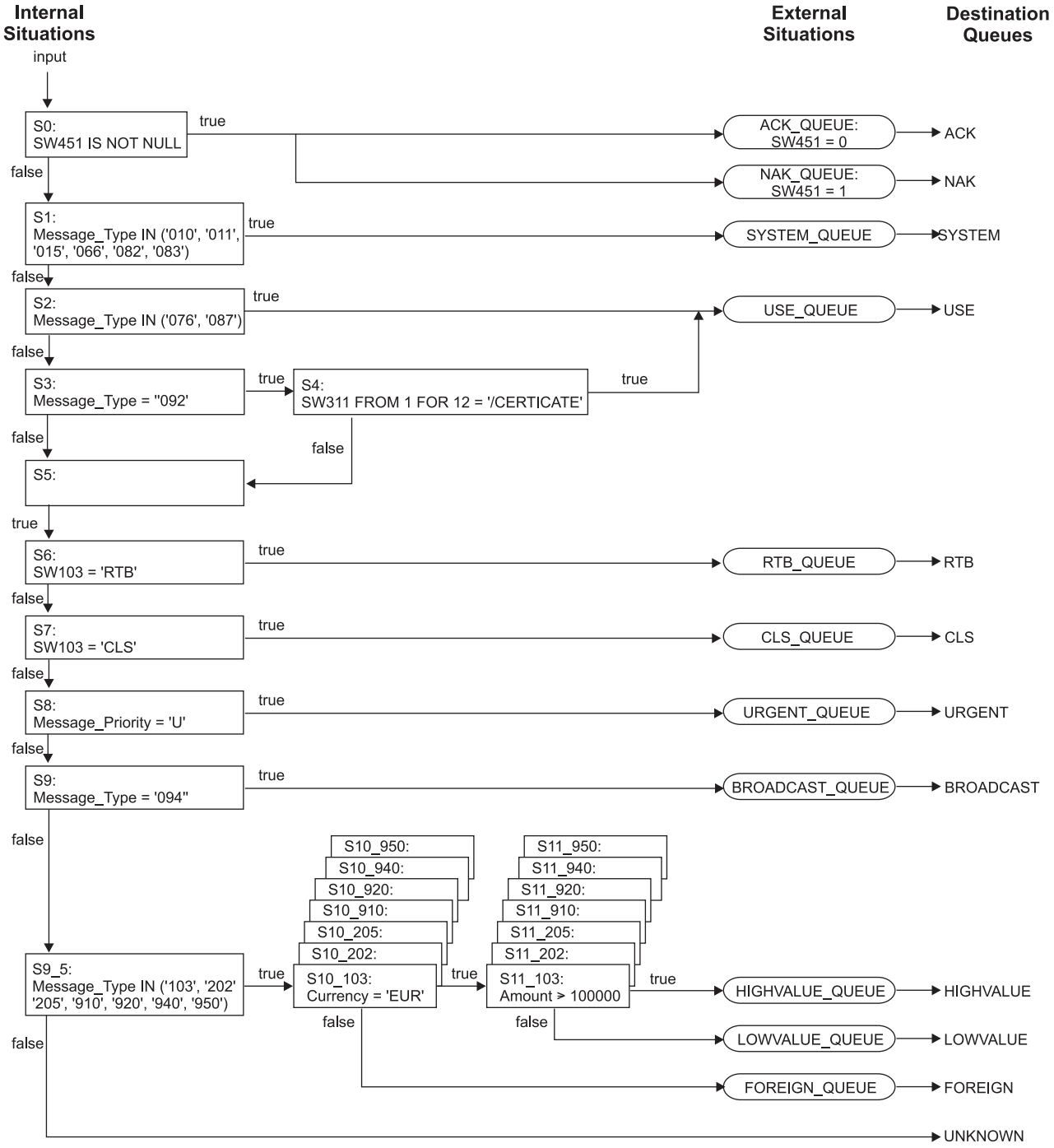

Figure 5. Decision tree showing processing order of routing situations

Each situation that assigns messages to a destination queue is implemented as an external situation, and its name corresponds to the name of the destination queue. The conditions of these situations are described in Table 1 on page 5.

Each decision that does not end in a queue assignment is implemented as an internal situation. Internal situations have names of the form **S*x***, for example S0, S9, S9_5, and S10_102. The conditions of these situations are described in Table 2 on page 5.

Table 1. External situations and their conditions

External Situation	Conditions
ACK_QUEUE	Any of the following: • Situation attribute **result == true** for event S0 • Threshold condition: SW451 = 0
NAK_QUEUE	Any of the following: • Situation attribute **result == true** for event S0 • Threshold condition: SW451 = 1
SYSTEM_QUEUE	• Situation attribute **result == true** for event S1
USE_QUEUE	Any of the following: • Situation attribute **result == true** for event S2 • Situation attribute **result == true** for event S4
RTB_QUEUE	• Situation attribute **result == true** for event S6
CLS_QUEUE	• Situation attribute **result == true** for event S7
URGENT_QUEUE	• Situation attribute **result == true** for event S8
BROADCAST_QUEUE	• Situation attribute **result == true** for event S9
HIGHVALUE_QUEUE	Any of: • Situation attribute **result == true** for event S11_103 • Situation attribute **result == true** for event S11_202 • Situation attribute **result == true** for event S11_205 • Situation attribute **result == true** for event S11_910 • Situation attribute **result == true** for event S11_920 • Situation attribute **result == true** for event S11_940 • Situation attribute **result == true** for event S11_950
LOWVALUE_QUEUE	Any of: • Situation attribute **result == false** for event S11_103 • Situation attribute **result == false** for event S11_202 • Situation attribute **result == false** for event S11_205 • Situation attribute **result == false** for event S11_910 • Situation attribute **result == false** for event S11_920 • Situation attribute **result == false** for event S11_940 • Situation attribute **result == false** for event S11_950
FOREIGN_QUEUE	Any of: • Situation attribute **result == false** for event S10_103 • Situation attribute **result == false** for event S10_202 • Situation attribute **result == false** for event S10_205 • Situation attribute **result == false** for event S10_910 • Situation attribute **result == false** for event S10_920 • Situation attribute **result == false** for event S10_940 • Situation attribute **result == false** for event S10_950
UNKNOWN_QUEUE	• Situation attribute **result == false** for event S9_5

Table 2. Internal situations and their conditions

Internal Situation	Conditions
S0	• Threshold condition: SW451 IS NOT NULL
S1	Any of the following: • Situation attribute **result == false** for event S0 • Threshold condition: Message_Type IN ('010', '011', '015', '066', '082', '083')
S2	Any of the following: • Situation attribute **result == false** for event S1 • Threshold condition: Message_Type IN ('076', '087')

Table 2. Internal situations and their conditions (continued)

Internal Situation	Conditions
S3	Any of the following: • Situation attribute **result == false** for event S2 • Threshold condition: Message_Type IN ('076', '087')
S4	Any of the following: • Situation attribute **result == false** for event S3 • Threshold condition: Message_Type = '092'
S5	Any of the following: • Situation attribute **result == false** for event S3 • Situation attribute **result == false** for event S4
S6	Any of the following: • Situation attribute **result == false** for event S5 • Threshold condition: SW103 = 'RTB'
S7	Any of the following: • Situation attribute **result == false** for event S6 • Threshold condition: SW103 = 'CLS'
S8	Any of the following: • Situation attribute **result == false** for event S7 • Threshold condition: Message_Priority = 'U'
S9	Any of the following: • Situation attribute **result == false** for event S8 • Threshold condition: Message_Type = '094'
S9_5	Any of the following: • Situation attribute **result == false** for event S9 • Threshold condition: Message_Type IN ('103', '202', '205', '910', '920', '940', '950')
S10_103 to S10_950	Any of the following: • Situation attribute **result == true** for event S9_5 • Threshold condition: Currency = 'EUR'
S11_103 to S11_950	Any of the following: • Situation attribute **result == true** for event S10_nnn • Threshold condition: Amount ≥ 100000

How the sample routing message flow works

When the IntelligentFilter node of the sample routing message flow receives a message, it retrieves the following information from the message:

- The message type specified in the message content descriptor (for a SWIFT FIN message, this will always be the string 'SwiftEnvelope')
- The value of the first message element that follows the elements specified in the messageNaming property of the IntelligentFilter node (this is the SWIFT message type, for example, 'MT103'[1])

The IntelligentFilter node then searches for an event with a name created by concatenating these two strings with an underscore character (_). For example, when the node receives a SWIFT MT103 message, it searches for an event with the name **SwiftEnvelope_MT103**.

1. The SWIFT message type, which has a name of the form **MT*nnn***, is not to be confused with the message type specified in the message content descriptor of the message.

The sample routing message flow contains one SwiftEnvelope_MT*nnn* event for each of the following SWIFT message types:

- MT103
- MT202
- MT205
- MT910
- MT920
- MT940
- MT950

Each SwiftEnvelope_MT*nnn* event:

- Contains, as its attributes, all the fields needed to trigger the situations that result in routing decisions
- Triggers a reference event with the name SwiftEnvelope, which:
  - Sets the required attributes that are common to all messages, regardless of their SWIFT message type
  - Triggers the internal situation S0

Situation S0, like all the other internal situations used by the sample routing message flow, sets a Boolean situation attribute with the name **result**. This attribute indicates whether the conditions of the situation were satisfied, and is used as a condition for the next situation in the processing order. For example, the result attribute of S0 is part of the threshold condition for S1; S1 is triggered only if **result == false** for S0.

Depending on the outcome of each internal situation, either an external situation or a subsequent internal situation is triggered. After S0, each situation refers to some other, previous situation, as shown in Figure 5 on page 4. This decision chain is constructed in such a way that, as soon as an external situation is triggered for a message, no other internal or external situation can be triggered for that message.

Some attributes required for routing decisions, such as amount and currency, are located in fields whose fully-qualified names are dependent on the SWIFT message type. For example, the amount of a MT103 message is located in the field:

```
Root.MRM.FIN.TEXT_BLOCK.MT103.SW32A.AMOUNT
```

AMiT does not support the use of wildcard characters in field names; for example, you cannot specify:

```
Root.MRM.FIN.TEXT_BLOCK.*.SW32A.AMOUNT
```

Therefore, you must specify each fully-qualified field name once per condition, for example:

```
Root.MRM.FIN.TEXT_BLOCK.MT103.SW32A.AMOUNT >= 100000
Root.MRM.FIN.TEXT_BLOCK.MT202.SW32A.AMOUNT >= 100000
Root.MRM.FIN.TEXT_BLOCK.MT205.SW32A.AMOUNT >= 100000
Root.MRM.FIN.TEXT_BLOCK.MT910.SW32A.AMOUNT >= 100000
Root.MRM.FIN.TEXT_BLOCK.MT920.SW32A.AMOUNT >= 100000
Root.MRM.FIN.TEXT_BLOCK.MT940.SW32A.AMOUNT >= 100000
```

You can specify such conditions either:

- As one of several conditions for a single situation
- As a single condition for one of several situations

The sample routing message flow uses the latter method. It uses one version of situations S10 and S11 for each of the message types that can result from the check

made by situation S9_5. These situations have names of the form **S10**_*nnn* and **S11**_*nnn*, where *nnn* represents the SWIFT message type.

Installing and using the message flow

Package content

The sample routing message flow package contains the following files:

DNI_AMIT_ROUTING_SAMPLE.xml
The sample routing message flow.

dni_routdef_amit.zip
A zip file containing the AMiT workspace for the sample routing message flow.

dni_routdef_amit.xml
The compiled and ready to use routing definition.

LA_en.txt
International License Agreement for Non-Warranted Programs.

license2.txt
IBM LICENSE AGREEMENT FOR CATEGORY 2 SUPPORTPACS. Please read this file before installing the package on a WebServer.

ReadMe.pdf
Information about the WebSphere BI for FN sample routing message flow in a format suitable for printing.

ReadMe.htm
Information about the WebSphere BI for FN sample routing message flow in a format suitable for viewing online.

Prerequisites

The sample routing message flow requires the following software:

- WebSphere® MQ Integrator Windows® 2000 Version 2.1 with APAR IC34773
- Version 1.3.1 of the WebSphere MQ Integrator support pac IAOS, which contains the Active Middleware Technology (AMiT) nodes and tooling
- The SWIFT message set (included with the Bridge for SWIFTNet feature of WebSphere BI for FN; otherwise, available from IBM)

Installation

If you plan to:

- Modify the sample routing definition, follow the procedure described in "Installing and customizing the message flow"
- Implement the routing definition described in Figure 5 on page 4 in a way that prevents you from ever changing it, follow the procedure described in "Installing the message flow for demonstration purposes" on page 11

Installing and customizing the message flow

To install and customize the sample routing message flow:

1. In the WebSphere MQ Integrator Control Center, click on the Message Sets tab. If the SWIFT Message Set is not among those listed, import it into the

WebSphere MQ Integrator MRM using the **mqsiimpexpmsgset** command, which is described in *WebSphere MQ Integrator: Administration Guide*.

2. The file DNI_AMIT_ROUTING_SAMPLE.xml is the sample routing message flow. Import this file into the WebSphere MQ Integrator Control Center.
3. Unzip the file dni_routdef_amit.zip. This creates a subdirectory with the name dni_routdef_amit.
4. Open the AMiT GUI, select **File > Import workspace...** from the menu bar, and import the workspace contained in the dni_routdef_amit subdirectory.
5. Close and then reopen the AMiT GUI.
6. Create a decision tree similar to that shown in Figure 5 on page 4 that describes your routing. Based on this tree, edit the events and situations as described in the AMiT User's Guide.

   Modify the situations, events, situation attributes, and threshold conditions as required to achieve the desired routing behavior. The naming scheme of event attributes is similar to ESQL. For example, to include the amount field of a MT103 message, define the following event attribute for the event SwiftEnvelope_MT103:

   ```
   Root.MRM.FIN.TEXT_BLOCK.MT103.SW32A.AMOUNT
   ```

7. Compile and export the AMiT definition file (dni_routdef_AMiT.xml).
8. Copy or move the exported AMiT definition file to a directory for which the user ID of the message broker's started task has read permission.
9. Create the following queues:
   - An input queue for the sample routing message flow.
   - A destination queue for each external situation you defined in step 6. Give each destination queue a name that indicates the situation to which it applies; for example ACK_QUEUE.
10. In the Control Center, select the imported sample routing message flow. Open the properties editor of the MQInput node by right clicking the node and selecting **Properties** from the popup menu.
11. Select the **Basic** notebook tab.
12. Change the value specified for queue name to the input queue created in step 9.
13. Select the **Default** notebook tab.
14. Press the OK button.
15. Right click on the IntelligentFilter node with the name **Routing** and select **Properties** from its popup menu. This opens the properties editor of that node.
16. Select the **Basic** notebook tab and set the following properties:

    AMiTDefinitionFile
    The fully-qualified name (that is, the path and filename) of the AMiT definition file (dni_routdef_AMiT.xml) specified in step 8

    AMiTDefinitionUpdateFile
    The fully-qualified name of the AMiT update file

    reportFile
    The fully-qualified name of the file to which report entries are to be written

    logLevel
    The level of detail of the messages written to the log

For more information about these properties, refer to the documentation provided with support pac IAOS.

17. Press the OK button.
18. Select the **Routing** tab and ensure that the enrichRouteDestination checkbox is enabled.
19. Right click on the MQOutput node **Unknown_Queue** and select **Properties** from the popup menu. This opens the properties editor of that node.
20. Select the **Basic** tab and change the values as required by your environment.
21. Press the OK button.
22. Right click in the message flow pane and select **Check-in** from the popup menu. This checks in the sample routing message flow.
23. Select to the **Assignments** tab and assign the sample routing message flow to the appropriate execution group.
24. Perform a delta deploy, or deploy the execution group that contains the sample routing message flow. The sample routing message flow is now ready for use.

Installing the message flow for demonstration purposes

To install the message flow for demonstration purposes:

1. In the WebSphere MQ Integrator Control Center, click on the Message Sets tab. If the SWIFT Message Set is not among those listed, import it into the WebSphere MQ Integrator MRM using the **mqsiimpexpmsgset** command, which is described in *WebSphere MQ Integrator: Administration Guide*.
2. The file DNI_AMIT_ROUTING_SAMPLE.xml is the sample routing message flow. Import this file into the WebSphere MQ Integrator Control Center.
3. The file dni_routdef_amit.xml is the compiled and ready to use routing definition. Transfer this file from the AMiT GUI to the system on which your message broker runs, and store it in a directory for which the user ID of the message broker's started task has read permission.
4. Create the following queues:
   - An input queue for the sample routing message flow
   - A destination queue for each of the external situations described in Figure 5 on page 4
5. In the Control Center, select the imported sample routing message flow. Open the properties editor of the MQInput node by right clicking the node and selecting **Properties** from the popup menu.
6. Select the **Basic** notebook tab.
7. Change the value specified for queue name to the input queue created in step 4.
8. Select the **Default** notebook tab.
9. Press the OK button.
10. Right click on the IntelligentFilter node with the name **Routing** and select **Properties** from its popup menu. This opens the properties editor of that node.
11. Select the **Basic** notebook tab and set the following properties:

    AMiTDefinitionFile
    The fully-qualified name (that is, the path and filename) of the AMiT definition file (dni_routdef_AMiT.xml)

AMiTDefinitionUpdateFile
The fully-qualified name of the AMiT update file.

reportFile
The fully-qualified name of the file to which report entries are to be written

logLevel
The level of detail of the messages written to the log

For more information about these properties, refer to the documentation provided with support pac IAOS.

12. Press the OK button.
13. Select the **Routing** tab and ensure that the enrichRouteDestination checkbox is enabled.
14. Right click on the MQOutput node **Unknown_Queue** and select **Properties** from the popup menu. This opens the properties editor of that node.
15. Select the **Basic** tab and change the values as required by your environment.
16. Press the OK button.
17. Right click in the message flow pane and select **Check-in** from the popup menu. This checks in the sample routing message flow.
18. Select to the **Assignments** tab and assign the sample routing message flow to the appropriate execution group.
19. Perform a delta deploy, or deploy the execution group that contains the sample routing message flow. The sample routing message flow is now ready for use.

Troubleshooting

In case of an error, use the troubleshooting facilities provided by AMiT and WebSphere MQ Integrator:

- For AMiT:
  - Check the report file specified during installation
  - Change the log level to **debug**
- For WebSphere MQ Integrator, set the user trace.