WebSphere MQ Integrator – AMiT Detector Nodes
Version 1.1

15 November, 2002

Irena Notkin
Dror Orell

orell@il.ibm.com

Property of IBM

Take Note!

Before using this report be sure to read the general information under "Notices".

Second Edition, November 2002

This edition applies to Version 1.1 of 'WebSphere MQ Integrator – AMiT Detector Nodes' and to all subsequent releases and modifications unless otherwise indicated in new editions.

© Copyright International Business Machines Corporation 2001. All rights reserved. Note to US Government Users -- Documentation related to restricted rights -- Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule contract with IBM Corp.

Table of Contents

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.

Trademarks and service marks

The following terms, used in this publication, are trademarks of the IBM Corporation in the United States or other countries or both:

- IBM
- MQSeries
- MQSeries Integrator
- WMQI
- MQSI

The following terms are trademarks of other companies:

- Windows NT, Microsoft Corporation

Summary of Amendments

Date	Changes
19 July 2002	Initial release
15 November 2002	Version 1.1

Preface

This SupportPac includes two processing nodes, which are capable of detecting complex *situations* rather than single events. With the use of AMiT (Active Middleware Technology), the nodes monitor the message flow allowing the detection of (and reaction to) situations which may be defined over multiple messages. The situations that are to be monitored are defined by applying a combination of logical, arithmetical and temporal operators on events (messages).

Incorporating an AMiT based node into a WMQI message flow allows harnessing the power of active behavior to the messaging application. Consider for example, the option to react to situations where a message did not arrive. The ability to react to complex situations as opposed to reaction to specific messages is a powerful extension which is now available as a standard WMQI processing node.

Bibliography

- *IBM MQSeries Integrator for Windows NT Version 2 Installation Guide,* IBM Corporation. SC34-5600.
- *IBM MQSeries Integrator Version 2 Using the Control Center,* IBM Corporation. SC34-5602
- Amit Manual – provided as part of this SupportPac.

Chapter 1. Overview

Introduction

Active technology is used in applications in which actions are performed as a result of a transition in the application domain and not as a result of an explicit request. Traditionally, active applications were used in active databases but nowadays active technology is used in a multitude of domains. Combining active technology with messaging applications will result in a powerful mechanism that is capable of functionalities such as load balancing, intelligent routing and alert handling. But active technology is by no means limited to these domains, in fact active technology could contribute to almost any messaging application. It seems there is hardly any application that will not benefit from the option to analyze the messages and monitor certain relationships between messages in the flow.

This SupportPac includes the ServiceControl and SituationManager processing nodes, which are capable of detecting complex situations. With the use of AMiT (Active Middleware Technology), the nodes monitor the message flow allowing the detection of (and reaction to) situations which may be defined over multiple messages. Monitored situations are defined by applying a combination of logical, arithmetical and temporal operators on events.

Incorporating an AMiT based node into a WMQI message flow allows harnessing the power of active behavior to the messaging application. Consider for example, the option to react to situations where a message did not arrive. The ability to react to complex situations as opposed to reaction to specific messages is a powerful extension which is made available as a standard WMQI processing node.

Both nodes that are provided in this SupportPac make use of the AMiT technology, however they differ in the fact that the ServiceControl node is predefined to a specific domain. The node monitors situations in which a service message should arrive on a given periodical basis. While many parameters can be configured in this node, the general structure of the situations it monitors is predefined. In contrast, the SituationManager can monitor situations that are freely defined and are independent of any domain specific constrains.

This document describes both the ServiceControl and SituationManager nodes. Chapter 2 details the content of the SupportPac and installation instructions for both of the nodes. Chapter 3 details usage of the nodes, Chapter 4 covers compilation issues and Chapter 5 includes examples regarding both nodes.

AMiT

AMiT (Active Middleware Technology) is a runtime control tool that enables fast and reliable development of reactive applications, without requiring programming skills. Reactive applications are based on reactions to an event's occurrence. AMiT enables reactions to situations rather than reactions to single events. A situation is a condition that is satisfied by an operator over an event history. Situations include composite events (e.g., conjunction), counting operators on events (e.g., at least 3), absence operators, and temporal operators. AMiT performs as a runtime monitor that receives information on the occurrence of events, detects situations, and reports the detected situations.

AMiT can be configured to monitor events of virtually any domain since the definitions of events and situations to be monitored are loaded dynamically. Configuration utilities are provided to allow wizard guided creation of such definitions. The configuration process of AMiT enables domain specific definition sets to be created without the use of programming skills. Be it a business process, stock market, server farm monitoring or any other domain, AMiT can be integrated to the systems to provide functions like intelligent message routing, alert handling, load balancing and more.

SituationManager Node

The SituationManager node is a generic monitor node, which encapsulates a fully featured AMiT tool. With proper configuration, this node can be integrated into any WMQI message flow to provide a wide set of functions. As will be detailed in chapter 3, the parameters of the specific domain (message format) should be introduced to the node, customizing it to the specific message flow. The definitions of the situations that are to be detected determine the node's functionality, where load balancing, message analysis and alert handling are just a few examples. At runtime the SituationManager node registers each message and propagates it unchanged. The detected situations are posted on an output queue to be consumed by the same message flow, a separate message flow or by another application.

ServiceControl Node

The ServiceControl node is designed to monitor if a service message was received within its time constraints. Configuration of this node includes designating the service message and the characteristics of the time constraints for the monitored service. The node is a domain specific monitoring node, which makes use of AMiT in a transparent way. The general structure of the situations it detects is predefined, simplifying the configuration process. As opposed to the SituationManager, the use of AMiT is done 'under the hood' without the need to expose any of the AMiT specific configurations. Chapter 3 details the process of configuring this node.

Chapter 2. Installing the plug-in node

SupportPac contents

The supplied zip file should be unzipped in a temporary directory. The following files and sub-directories will be created:

- /classpath.txt - instructions for configuring system classpath
- /amit - standalone AMiT Wizard for situation definition, including development environment (workspaces).
- /bin - all files used by nodes at run-time (data, configuration, etc.):
- /bin/amit.dtd - DTD for Amit-compliant XML definitions
- /bin/ eepDefinitions.txt - AMiT configuration file
- /bin/data/workspaces/default – SituationManager node default workspace
- /bin/data/workspace/netsales – SituationManager node example workspace (includes definitions used in example detailed in chapter 5)
- /bin/data/workspaces/serviceControl – ServiceControl node workspace
- /classes - all supplementary jar files used by the nodes (AMiT, graphics, XML parsers, etc.)
- /classes/java/util/ - Timer class files to be used if nodes are run on JDK versions earlier that JDK1.3
- /config - header and XML files for integrating nodes into the WMQI Control Center:
- /config/situation.h – message definition for the message posted by the nodes.
- /config/netsales.h, /config/stockinfo.h, /config/situation.h –message definitions used in the examples that are detailed in chapter 5.
- /config/ComIbmAmitnodesSituationManager.wdp, /config/ComIbmAmitnodesSituationManager – SituationManager node definitions
- /config/ComIbmAmitnodesServiceControl.wdp, /config/ComIbmAmitnodesServiceControl – ServiceControl node definitions
- /doc/Amit_Manual.doc – document describing the process of creating AMiT definitions
- /gif - nodes icons for WMQI Control Center
- /jplugin/amit.jar - The jar file containing implementation classes of both nodes and implementation classes of ServiceControl node customizer

Prerequisites

This SupportPac provides two plug-in nodes to be used with the IBM MQSeries Integrator Version 2.1 or compatible versions. For normal use, there are no other prerequisite products other than those required by IBM MQSeries Integrator Version 2.1 itself.

Supported Platforms

This SupportPac has been developed and tested in a Microsoft Windows NT environment. Since the nodes have been implemented in pure Java, they should be runnable on any platform, provided by the JDK 1.3.

Installing the plug-in nodes on broker system

The supplied zip file contains the files that are required for the installation of both the ServiceControl and SituationManager nodes. Several files are used by both nodes and some are relevant only to one of the nodes. It will be stated in this section which files are required for the installation of each node.

The plug-in '**jar**' file should be installed by copying or moving the '**amit.jar**' file from the temporary '**/jplugin**' directory to the directory '**<wmqi_2.1_root>\jplugin**'.

In addition, the following files/directories should be copied/moved or created to provide the complete nodes installation in the broker domain:

For both nodes:

- ❑ copy/move '**/bin/amit.dtd**' to the directory '**<wmqi_2.1_root >\bin**'
- ❑ copy/move '**/bin/eepDefinitions.txt**' to the directory '**<wmqi_2.1_root >\bin**'
- ❑ copy/move '**/classes/amitapi.jar**' to the directory '**<wmqi_2.1_root >\classes**'
- ❑ create directory '**<wmqi_2.1_root >\bin\data\workspaces**'

For ServiceControl node:

- ❑ copy/move '**/bin/data/workspaces/serviceControl**' to the directory '**<wmqi_2.1_root >\bin\data\workspaces**'
- ❑ copy/move **'/classes/java/util/*'** to the directory '**<wmqi_2.1_root >\classes\java\util'**

For SituationManager node:

- ❑ copy/move '**/bin/data/workspaces/default**' to the directory '**<wmqi_2.1_root >\bin\data\workspaces**'
- ❑ copy/move '**/bin/data/workspaces/netsales**' to the directory '**<wmqi_2.1_root >\bin\data\workspaces**' (to be used only for running the example application detailed in chapter 5).

Check that your system classpath is set according to the classpath.txt instructions and reboot in order for the system to detect the new settings. You must stop and restart the broker to enable it to detect the existence of the new plug-in nodes.

Integrating the plug-in node into the Windows Control Center

Use the following table to copy the files to their correct location. These locations should already exist providing you have deployed at least one message flow. Prepped your **<wmqi_2.1_root >** to the **Copy to location** value.

Use the following to replace the placeholders:

<hostname>	-	TCP/IP hostname
<CM QMName>-		Configuration Manager's queue manager name

Copy from location	Filename	Copy to location
	For SituationManager node:	
/config	ComIbmAmitnodesSituationManager.wdp	\Tool\repository\private\<hostname>\<CMQMName>\MessageProcessingNodeType
/config	ComIbmAmitnodesSituationManager	\Tool\repository\private\<hostname>\<CMQMName>\MessageProcessingNodeType
/gif	SituationManager.gif	\Tool\images
/gif	SituationManager 30.gif	\Tool\images
/gif	SituationManager 42.gif	\Tool\images
/gif	SituationManager 58.gif	\Tool\images
/gif	SituationManager 84.gif	\Tool\images
	For ServiceControl node:	
/config	ComIbmAmitnodesServiceControl.wdp	\Tool\repository\private\<hostname>\<CMQMName>\MessageProcessingNodeType
/config	ComIbmAmitnodesServiceControl	\Tool\repository\private\<hostname>\<CMQMName>\MessageProcessingNodeType
/gif	ServiceControl.gif	\Tool\images
/gif	ServiceControl30.gif	\Tool\images
/gif	ServiceControl42.gif	\Tool\images
/gif	ServiceControl58.gif	\Tool\images
/gif	ServiceContro84l.gif	\Tool\images
/classes	spinner.jar	\classes

Defining the node to the configuration repository

Once you have installed the files in the appropriate directories, as described in the previous section, you must make these definitions available to the Control Center.

1. For ServiceControl node ensure that your system classpath includes the following jar files: '**<wmqi_2.1_root>\jplugin\amit.jar**' , which contains the ServiceControl node customizer classes;
   '**<wmqi_2.1_root>\classes\spinner.jar**', which contains GUI classes used by the customizer. Reminder – in order for changes in system classpath to take effect, the system should be rebooted.

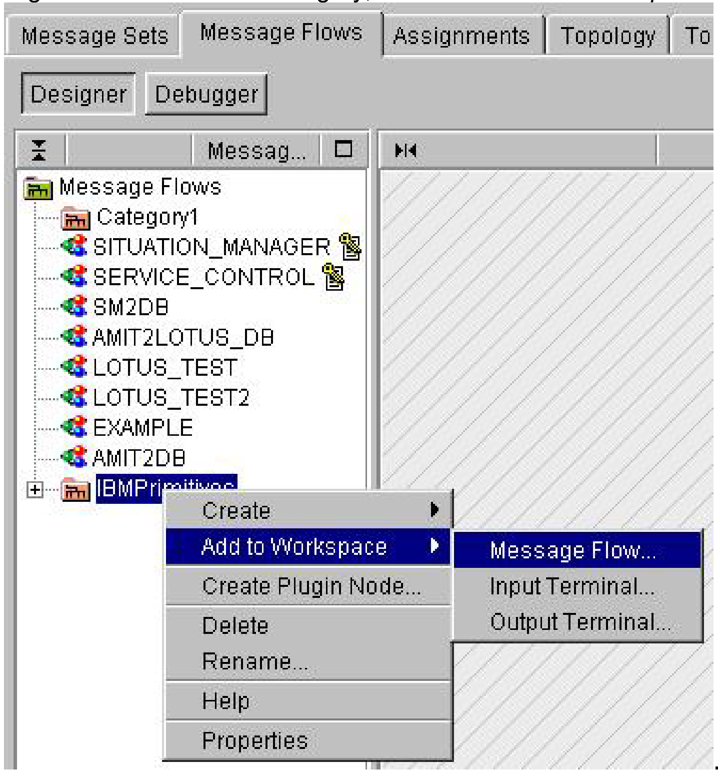
2. Start the Control Center. The user ID you are using must be a member of the MQSeries Integrator group ***mqbrdevt***. You are recommended to use the superuser ***IBMMQSI2*** to

complete this task[1]. This causes your new node to be locked under the same user ID as all the supplied IBM primitive nodes. If you do not use this user ID, the definition files in the configuration repository might be accidentally locked, and therefore open to unauthorized update.

3. Select the Message Flows view.

4. Select an existing Message Flow Category, or create a new one.

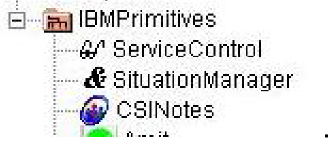
5. Right-click the selected category, and select *Add to Workspace->Message Flow:*



A list box is displayed showing all existing IBM-supplied primitive nodes and any defined message flows you have installed following the instructions provided.

6. Select the message flow (the node) – SituationManager, ServiceControl or both.

This node now appears within the message flow category you selected in the tree view in the left-hand pane:



7. Select your new node, and right-click. Select *Check In*.

[1] You must take care if you change logon IDs to complete this task. Changing logon IDs can effect the operation of the Configuration Manager's queue manager if it is on this system, but not running as a Windows NT service. See the *MQSeries Integrator Administration Guide* for more information about queue manager operation (Chapter 2) and the superuser *IBMMQSI2* (Chapter 4).

8. Right-click again, and select Lock. Then right-click again and select Check In for a second time. After this check, the interface and ***.wdp** definition files disappear from the local directory and go into the shared repository, where they are available to all users of the Control Center. However, user can only use this new node if they have installed the additional files (icons, properties files, and so on) on their own system.

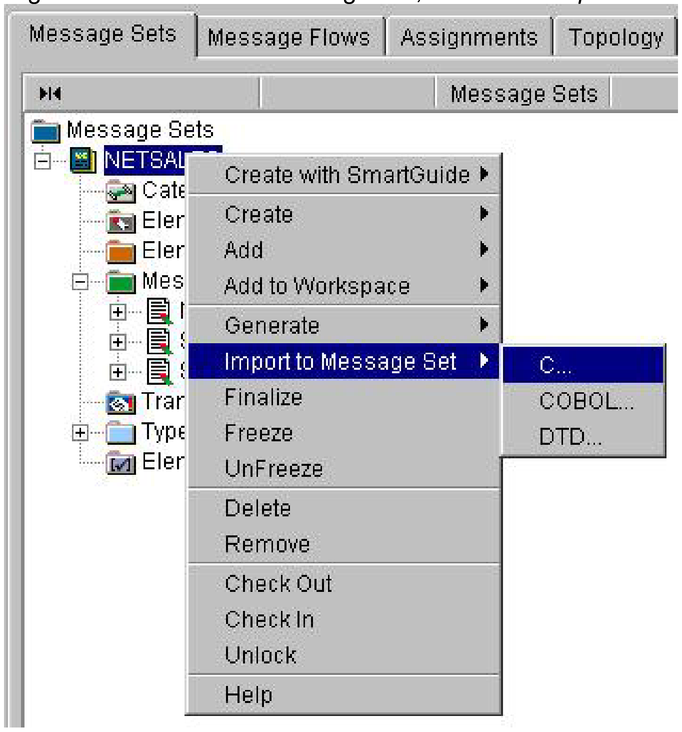
Defining messages to the configuration repository

Both nodes report the detected situations by posting the SITUATION (MRM) message to the output queue(s), specified in the nodes configuration. The situations are also logged to the report files. If you design a message flow that reads the SITUATION messages that were written by an AMiT node, you'll need to define the SITUATION message in your message set. The simplest way to do so is to import the message definition from the C header file:

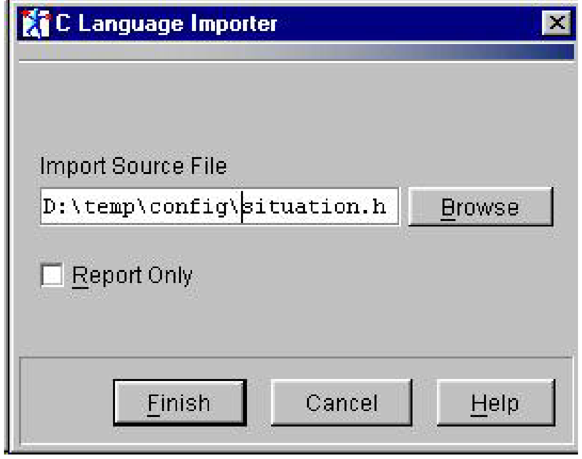
1. Select the Message Sets view.

2. Select an existing Message Set, or create a new one.

3. Right-click the selected Message Set, and select *Import to Message Set->C…:*

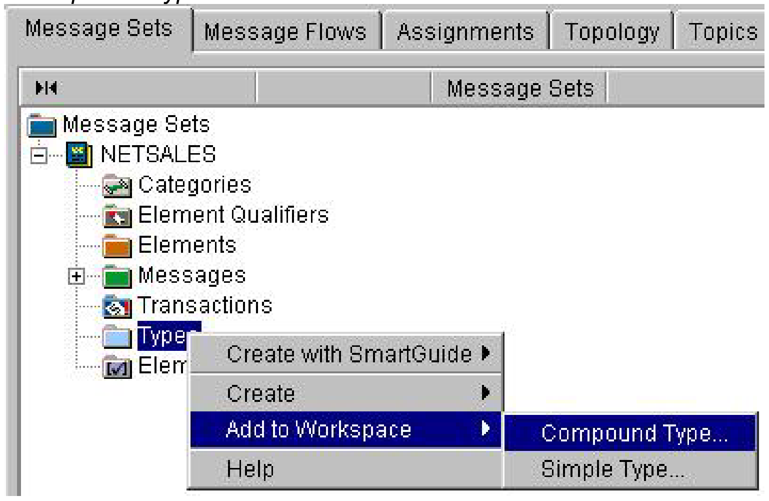


4. Locate the '**situation.h**' in your temporary directory and click Finish:

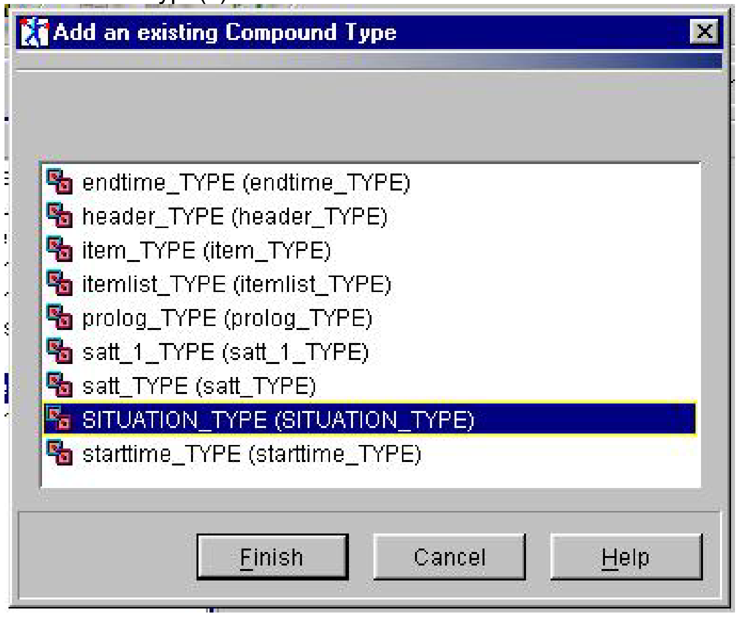


5. If you are going to run the SituationManager node example, supplied in this SupportPac, you will need to import netsales.h and stockinfo.h definitions of messages NETSALES and STOCKINFO, correspondingly.

6. Once importing definition(s) has been successfully completed, you need to add the newly created types to your Workspace. Select Types, right-click and select *Add toWorkspace->Compound Type*.

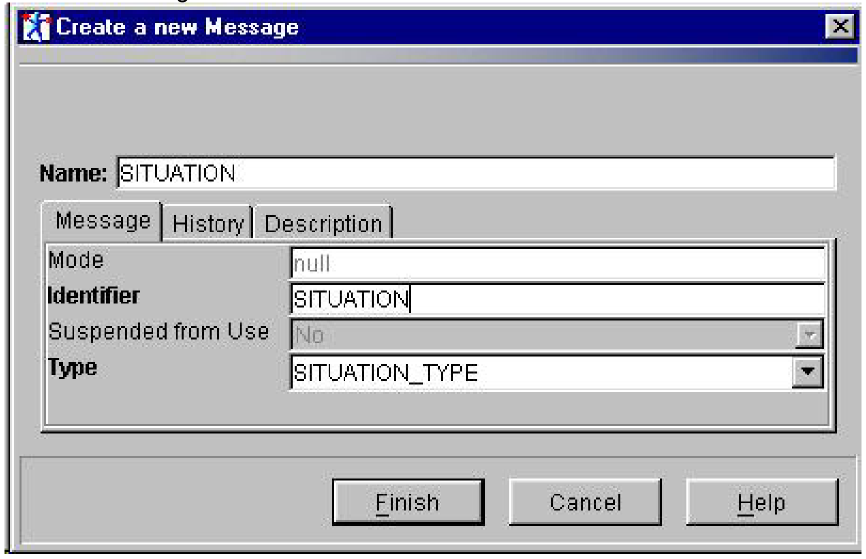


Choose new type(s) and click Finish:



endtime_TYPE (endtime_TYPE)
header_TYPE (header_TYPE)
item_TYPE (item_TYPE)
itemlist_TYPE (itemlist_TYPE)
prolog_TYPE (prolog_TYPE)
satt_1_TYPE (satt_1_TYPE)
satt_TYPE (satt_TYPE)
SITUATION_TYPE (SITUATION_TYPE)
starttime_TYPE (starttime_TYPE)

7. Next, create the SITUATION message by selecting Messages, right-clicking and choosing *Create->Message*.



Name:	SITUATION
Mode	null
Identifier	SITUATION
Suspended from Use	No
Type	SITUATION_TYPE

Chapter 3. Using the plug-in node

Description

As opposed to most processing nodes that receive an incoming message, process it and send an outgoing message, the nodes provided in this SupportPac perform an asynchronous task. The messages produced by the node to notify of a detected situation are not necessarily created as a response to a specific incoming message (for instance a notification that less than 5 purchase requests were made during the first hour of trade). Therefore, the notification messages are posted on a dedicated queue rather than being propagated to the next node in the message flow. Messages from this dedicated queue can be used by the same message flow (as demonstrated in chapter 5), a separate message flow or as input to a different application. Messages are also added to a report file that is placed at the workspace folder under **<wmqi_2.1_root>\bin\data\workspaces\.**

As will be detailed later in this chapter the name of the message queue, on which the nodes should post the notifications, is provided to the nodes as a configuration parameter in the customizer.

SituationManger Node

The SituationManager node, being a generic monitor node, has to be adapted to the specific domain it should perform in. Therefore, the configuration of the SituationManager node requires a preliminary phase of setting the AMiT related configuration. The messages in the message flow should be mapped to AMiT events and the situations we wish to detect should be defined in reference to these events. Prior to the node configuration, with use of a dedicated tool, a set of AMiT definitions should be created to specify the format of events and the situations. This section will provide an overview of the AMiT configuration process. The AMiT Manual document, which is included in this supportPac, describes this configuration process in detail.

The AMiT definitions are packed in an XML file that should be placed in the workspace folder . This definition file is created with use of AMiTGUI, a wizard-based utility that is provided as part of this SupportPac.

The following types of definitions should be created

- Events – a representation of the messages in the message flow
- Lifespans – define time spans (the duration between two specified events, an hour after a certain event, the duration between two dates, etc.)
- Keys – a key is a set of event attributes, within several events, that are semantically equivalent.
- Situations – The situation definition specifies the events that are relevant to this situation, the keys and their value, the lifespan and the operators that should be applied to the events (i.e. what is the expression over events that should be satisfied).

Important note: The AmitGUI utility, for internal reasons, creates directory structure in addition to the definition file. The SituationManager node only requires this single XML file. This file, which is created in the ‘export’ process, should be named AmitDefinitions.xml and it should be placed in the <WMQI_2.1_root>\bin\data\workspaces\<workspace name>\ folder.

During run time, the node will automatically create an AMiT event instance corresponding to each incoming message. Situation detection will be performed on these events and not on the MRM messages. AMiT should therefore be configured to recognize the format of the automatically generated events. In this version of the SituationManager node, the process of defining the AMiT events that correspond to the messages in the message flow should be performed manually. Future - versions of the SituationManager node are planned to include a mechanism that will automatically define the AMiT events, i.e. when given a set of MRM messages, this mechanism will automatically create the definitions of compatible AMiT events.

In contrary to MRM messages that may include hierarchical attributes, composite attributes of AMiT events are still not supported. Therefore, the run time translation mechanism includes a ‘flattening’ process that translates composite MRM message attributes to equivalent primitive attributes of an AMiT event. This implies that the definition of AMiT events should be created in a way that for composite MRM attributes, each leaf attribute is mapped to an equivalent AMiT event primitive attribute. A fully qualified name is set to such event attributes in a way that includes the names of all the MRM attributes in the hierarchy. The underscore character (‘_’) is used to denote hierarchy. Primitive MRM attributes should be mapped to event attributes with an identical name. The following table provides an example that details a message with composite attributes and the corresponding attribute names in an AMiT event. It is important that this naming scheme will be strictly followed to ensure that manually created definitions correspond to the events that are automatically created in run time. Failing to follow this naming scheme is likely to cause MRM messages to be ignored by the SituationManager node.

MRM Message	AMiT Event
NETSALES	
HEADER	
RETAIL_STORE_ID	HEADER_REATAIL_STORE_ID
ITEMLIST	
ITEM	
ITEM_ID	ITEMLIST_ITEM_ITEM_ID
QUANTITY	ITEMLIST_ITEM_QUANTITY
TOTAL_TRANSACTIONS	ITEMLIST_ITEM_TOTAL_TRANSACTIOS

Please note that when defining the AMiT events, it is not necessary for the event to include all the attributes of the MRM message. Only attributes that are used in the definition of situations must be included in the event definition, the rest of the attributes can be ignored. Furthermore, it should be noted that the use of underscore to denote hierarchy does not exclude the use of attribute names that include the underscore character (as shown in the example above).

A log file is created by the node. The file is created in the **<wmqi_2.1_root>\bin\** folder and has the following naming convention: <message flow name>_SituationManager-<instance_number>.log

ServiceControl Node

The ServiceControl node is designed to detect situations in a predefined domain – it monitors periodical service messages. However, the situations to be monitored are not predefined and the final tuning that sets the node’s specific functionality is set by the user during the configuration stage. The user defines which of the messages in the message set should be monitored, the possibly repetitive time span (the service period) in which this message should arrive, and preferences regarding the required notifications. This configuration process is performed using the node’s customizer in the Control Center and does not involve any additional tools. A detailed specification of the configuration process and the specific fields in the node customizer is provided later on in this chapter.

At runtime the node will perform the following functionality. At a specified time it will begin monitoring messages of the type that was defined as the service message. Monitoring will consist of a repetitive cycle that is divided into two equal time periods: a service period and a delayed message period. Messages arriving during the service period will be reported as successful, messaged arriving during the delayed message period will be reported as delayed service and if no service message was received till the end of the cycle, a failure notification will be spawned. A new cycle begins immediately after the delayed message period is completed and additional cycles are then repeated. For example, if the node was configured to start monitoring at startup and the time span was set for 5 minutes, the node will wait 5 minutes for “successful delivery” then it will wait an additional 5 minutes for “delayed delivery” after which a new cycle will begin every 10 minutes.

The user can set if the service period starts at startup, at a given time or at the arrival of a certain message. The repetitive cycle monitoring that is described above will be performed only for the first two options, i.e. if the service period begins with the arrival of an initiator message, the monitoring cycle will not be repeated.

Following is the list of situations that the ServiceControl detects:

- Successful Service Delivery - A message signifying service delivery arrived within specified service period.
- Service Delivery Failure - No message signifying service delivery had arrived within service period or delayed period.
- Delayed Delivery - A message signifying service delivery arrived after the specified service period and during the delayed service period.
- Growing Delivery Delay - Three Delayed Deliveries had occurred so that the delivery delay grew from the first to the third message.
- Repetitive Failure - Service Delivery Failure situation happened several times within specified observation period. This is an optional situation, which replaces the ”Service Delivery Failure” message and is enabled or disabled in the node customizer.

A log file is created by the node. The file is created in the **<wmqi_2.1_root>\bin\** folder and has the following naming convention: <*message flow name*>_ServiceControl-<*instance_number*>.log

Note – Configuration of this node is performed using only the node customizer. All the required files which hold the AMiT definitions are found in the service control workspace directory and there is no need change them or to use the AMiTGUI utility in the configuration of this node.

Plug-in nodes terminals

SituationManager and ServiceControl nodes have very similar architecture and in particular, the same terminals:

Terminal	Description
In	The input terminal that accepts a message. Each accepted message is first of all propagated to the Out terminal and besides that, can also be delivered to AMiT engine for situations processing. Whether the message is delivered to AMiT or not, depends on the node configuration.
Out	The output terminal that outputs the original message. Pay attention that SITUATION messages which represent the result of nodes processing, are not propagated to the Out terminal. Instead, SITUATION messages are written to the queue(s) specified in the configuration and can be read by an Input node or by a different application.

Plug-in nodes properties

SituationManager properties

The SituationManager node has two properties which are common to all AMiT-based nodes, such as ServiceControl node and any other node which might be developed in the future:

#	Name	Description	Default Value	Comments
1	OutputQ	List of queues to which SITUATION message should be sent.	ECE_FALSE	List may contain one name or any number of names, separated by colon, semicolon, space or tab.
2	Workspace	Workspace/Directory name (under '**<wmqi_2.1_root>\bin\data\workspaces**', where the corresponding AMiT metadata (event & situation definitions) resides.	default	Default workspace contains the 'netsales' example metadata.

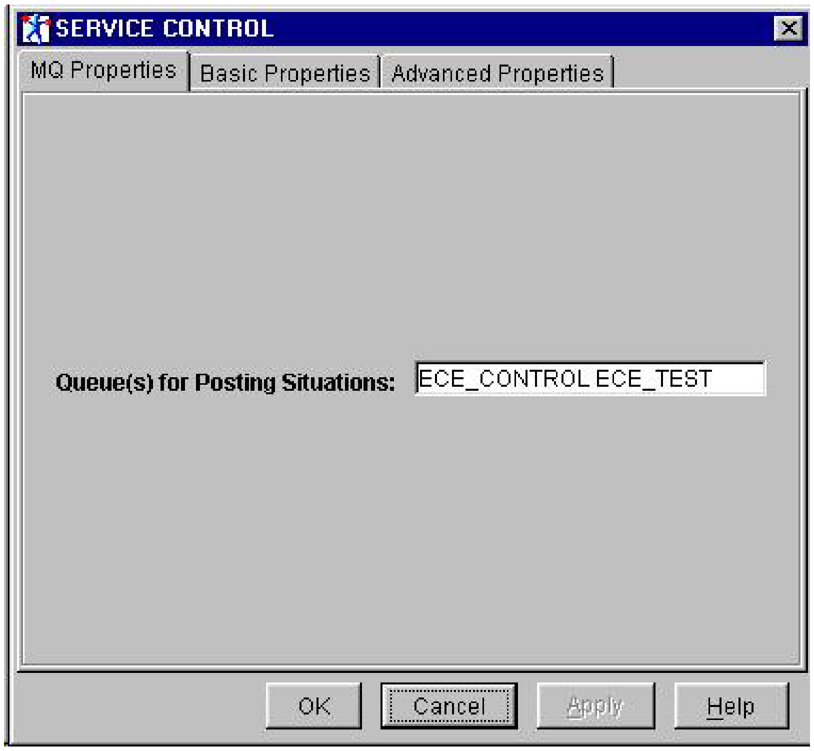
ServiceControl properties

The ServiceControl node properties may be divided into the three logical groups:

General Properties

#	Name	Description	Default Value	Comments
1	OutputQ	List of queues to which SITUATION message should be sent.	ECE_FALSE	List may contain one name or any number of names, separated by colon, semicolon, space or tab.
2	Workspace	Workspace/Directory name (under '**<wmqi_2.1_root>\bin\data\workspaces**', where the corresponding AMiT metadata (event & situation definitions) resides.	serviceControl	Currently could not be changed by user via the Properties.

Corresponding configuration page:

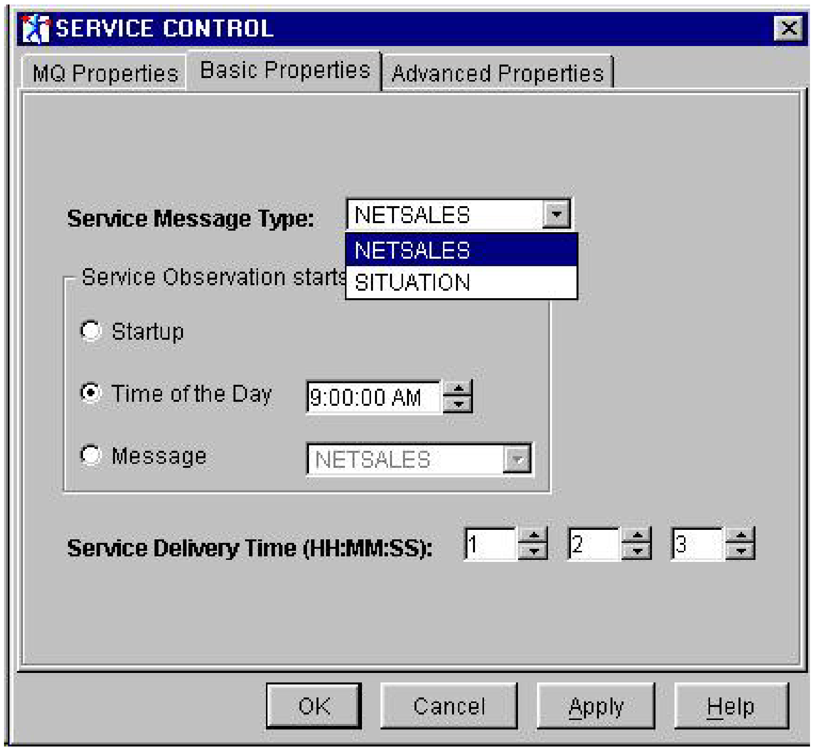


Queue(s) for Posting Situations: ECE_CONTROL ECE_TEST

__Basic Properties__

#	Name	Description	Default Value	Comments
3	MessageType	The name of MRM message, designating service delivery.	NETSALES	
4	StartOn	When to begin service control: 1. upon system startup or 2. at a specified time or 3. upon arrival of some message	startup	Options 1 and 2 will cause service control to happen repeatedly, while in case of option 3 service control will be initiated each time the specified message arrives.
5	serviceDeliveryTime	Period of time within which service should be delivered.	15 sec	
6	messagesNumber	Currently not used.		

Corresponding configuration page:



Service Message Type: NETSALES (NETSALES, SITUATION)

Service Observation starts:

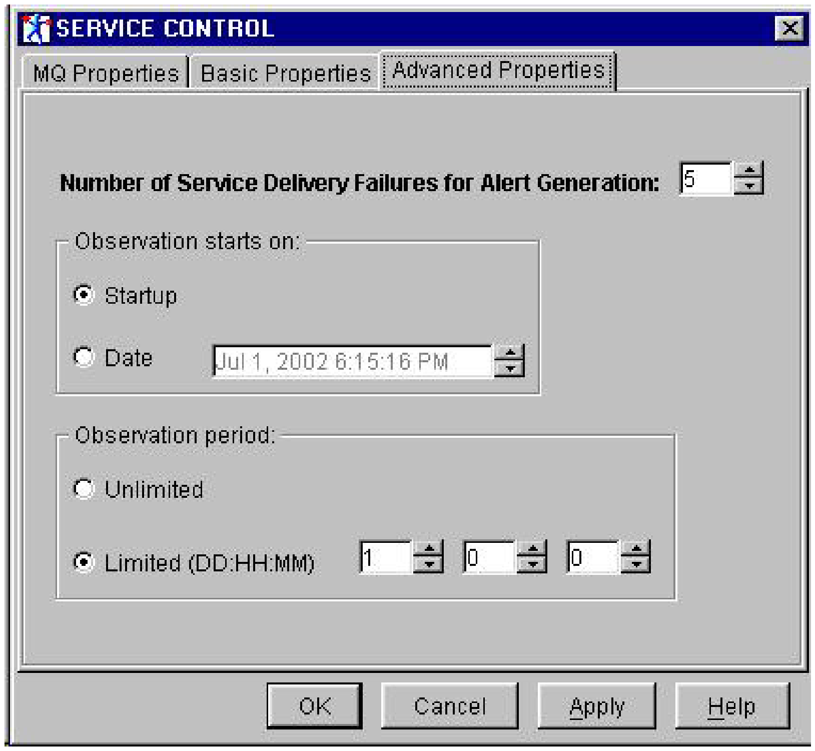
- Startup
- [✓ Time of the Day] 9:00:00 AM
- Message NETSALES

Service Delivery Time (HH:MM:SS): 1 2 3

<u>Advanced Properties</u>

#	Name	Description	Default Value	Comments
7	FailuresNumber	Number of sequential service delivery failures to generate the alert (SITUATION message).	2	If value 1 is set the advanced service control is not initiated and the SITUATION message is sent each time the service delivery failure is detected.
8	startObservationOn	When to begin control of several sequential failures: 1. upon system startup or 2. at a specified time	startup	
9	observationPeriod	Period of time within which control of sequential failures should be done: 1. unlimited 2. limited to time period	Limited to 10 min	If unlimited option is chosen, the advanced service control will last till the end of run. If limit is set, the advanced service control will happen repeatedly, starting the next period when the previous one ends up.

Corresponding configuration page:



Number of Service Delivery Failures for Alert Generation: 5

Observation starts on:

(●) Startup

( ) Date Jul 1, 2002 6:15:16 PM

Observation period:

( ) Unlimited

(●) Limited (DD:HH:MM) 1 0 0

Runtime constraints

In this version of the supportPac the following constraints should be noted:

- All messages that are to be monitored by AMiT nodes should be MRM messages.
- The log file created by the ServiceControl node may become quite big. When repetitive service durations are monitored, notifications of a successful service or service failure are added to the log for every service duration.

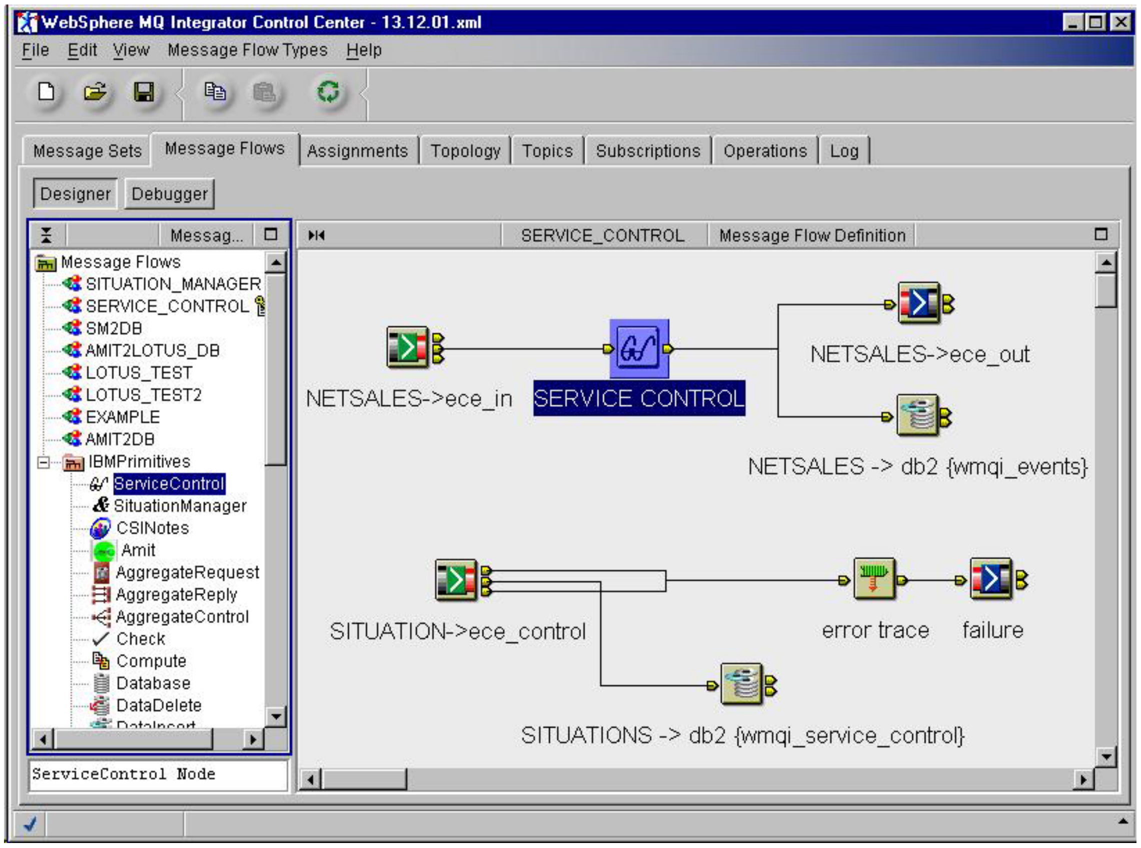
Chapter 4. Compiling the plug-in node

This version of the SupportPac does not include source code and therefore, no compilation is required. The supplied jar file can be used on all platforms that support JDK1.3.

Chapter 5. Example using the plug-in node

ServiceControl node

The following figure details a Message Flow which makes use of a ServiceControl node:



Message Flow SERVICE_CONTROL reflects the following scenario:

1. MQInput node “NETSALES->ece_in” reads NETSALES messages from the ece_in queue and transfers them to the “SERVICE CONTROL” node;

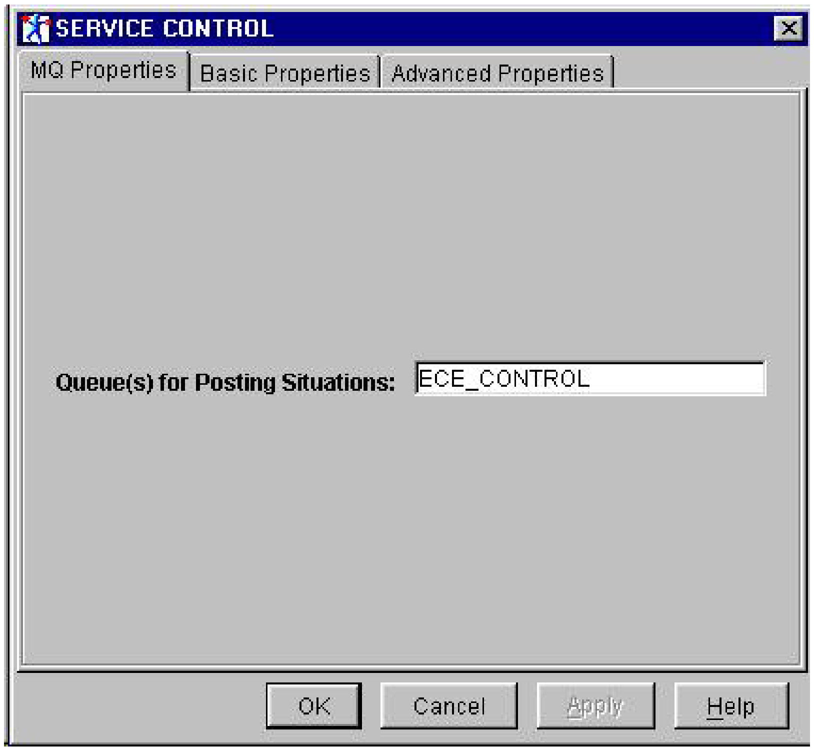
2. “SERVICE CONTROL” node propagates input messages as is to the:

    a. NETSALES->ece_out” MQOutput node which puts them to the ece_out queue.

    b. NETSALES->db2 {wmqi_events}” DataInsert node which stores them in the wmqi_events db2 table.

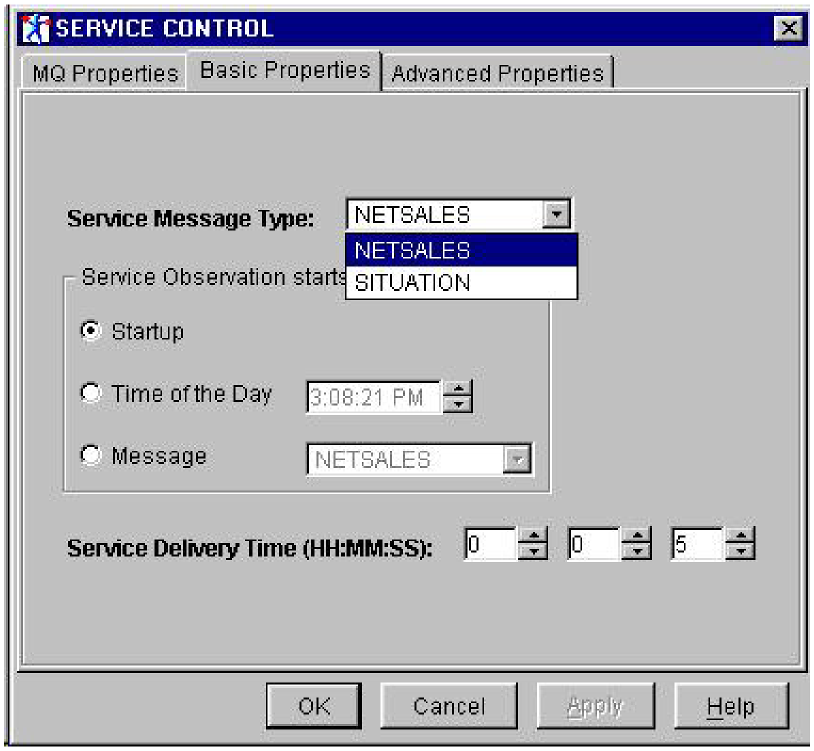
3. Upon situation detection SITUATION message is written by the “SERVICE CONTROL” node to the ece_control queue, from where it is read by the MQInput node “SITUATION->ece_control”. In case of any problem (failure, exception), the message is propagated to the MQOutput node “failure” via the Trace node “error trace”. Otherwise, the SITUATION message is propagated to the DataInsert node SITUATIONS->db2 {wmqi_service_control}” which stores it in the wmqi_service_control db2 table.

The following figures show three snapshots of the “SERVICE CONTROL” node configuration:



Queue(s) for Posting Situations: ECE_CONTROL

User specifies the name of the queue, where the SITUATION messages should be posted.



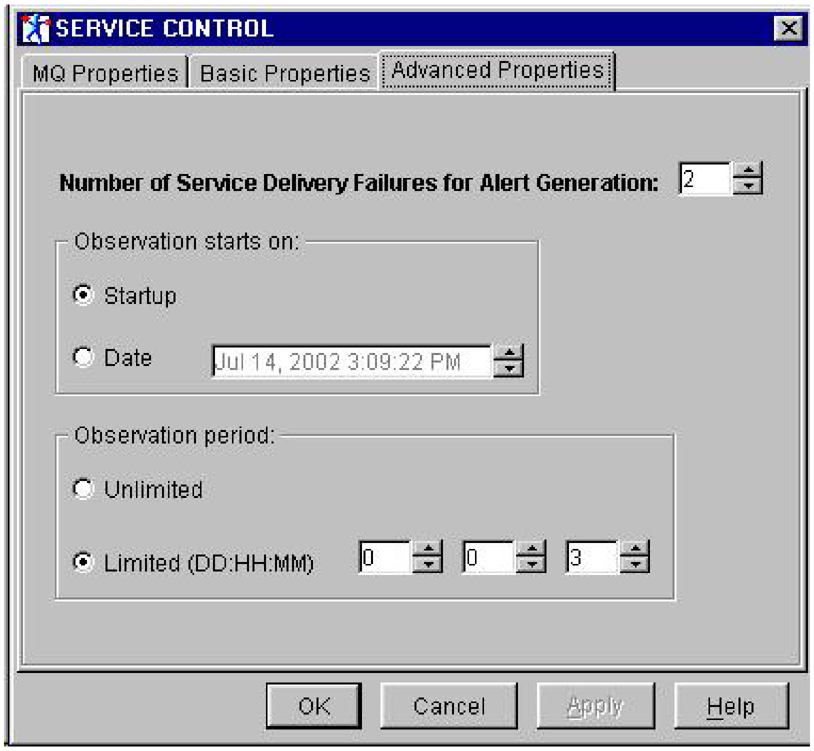
Service Message Type: NETSALES

Service Observation starts

- [✓ Startup]
- Time of the Day 3:08:21 PM
- Message NETSALES

Service Delivery Time (HH:MM:SS): 0 0 5

User specifies the NETSALES message as a service message which should be controlled and the Startup option for the service control starting point. The duration of service period is set to 5 seconds.



Number of Service Delivery Failures for Alert Generation: 2

Observation starts on:

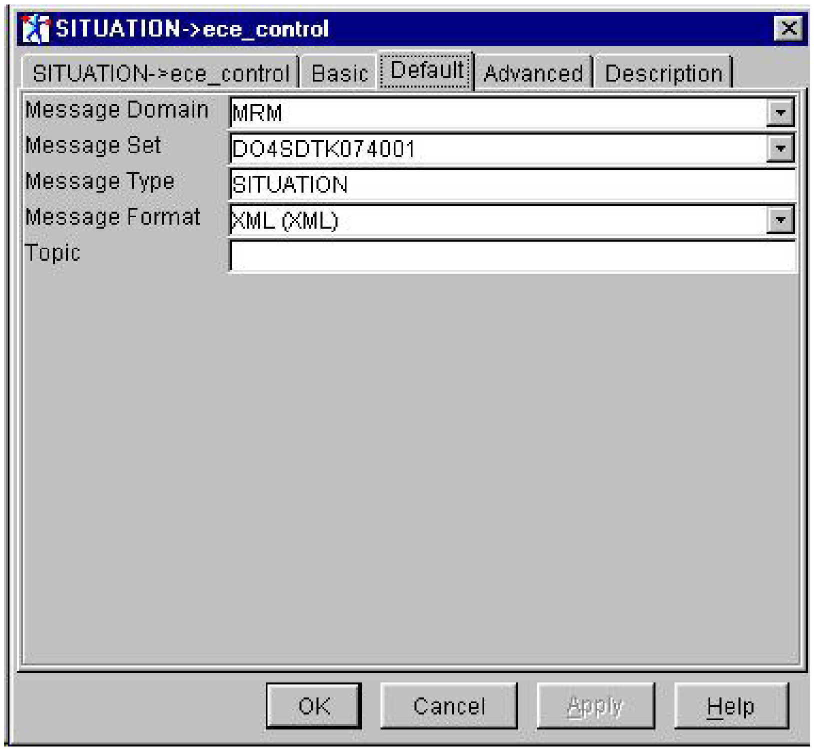
- [✓ Startup]
- Date Jul 14, 2002 3:09:22 PM

Observation period:

- Unlimited
- [✓ Limited (DD:HH:MM)] 0 0 3

User specifies the number of service failures for alert generation as 2 and the Startup and period of observation.

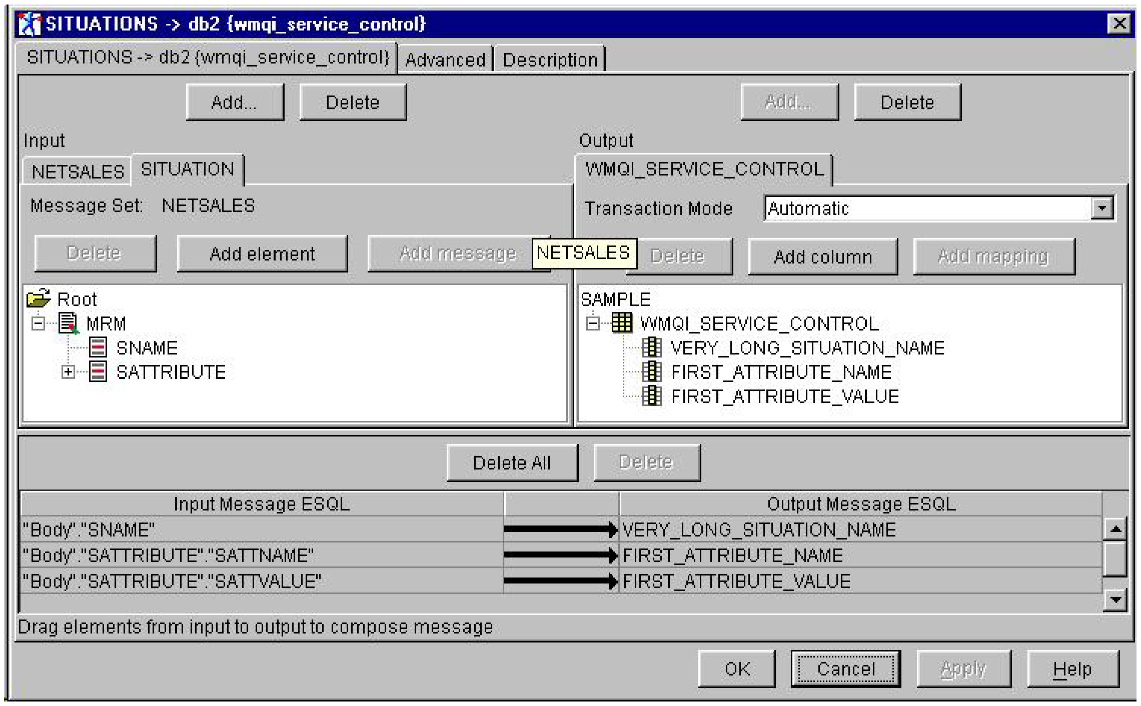
The next figure shows the configuration of “SITUATION->ece_control” MQInput node:



Message Domain	MRM
Message Set	DO4SDTK074001
Message Type	SITUATION
Message Format	XML (XML)
Topic	

User specifies the message parameters of the SITUATION message.

The next figure shows the configuration of the “SITUATIONS->db2 {wmqi_service_control}” DataInsert node:



Message Set: NETSALES

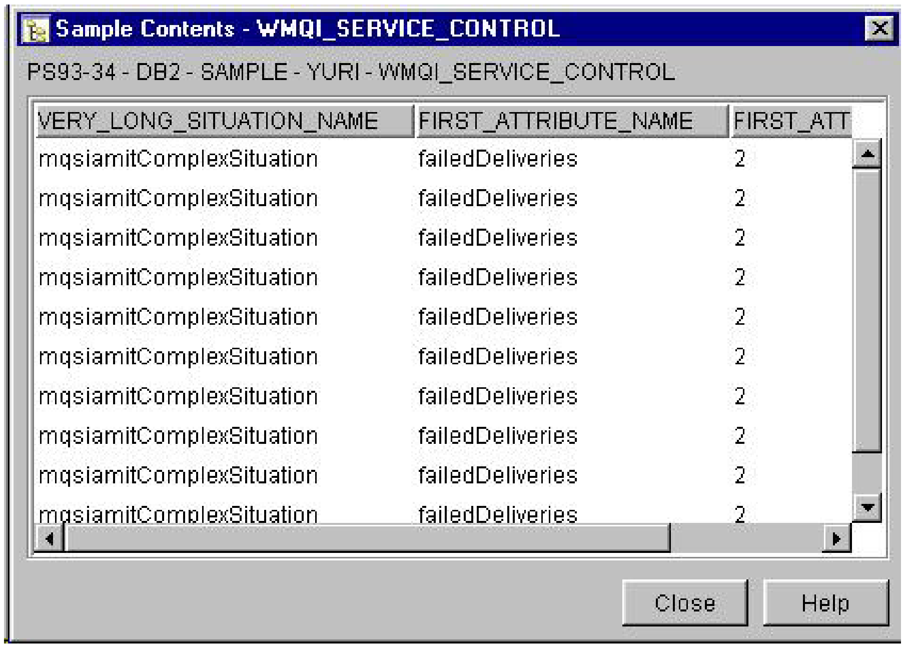
Transaction Mode: Automatic

Input Message ESQL		Output Message ESQL
"Body"."SNAME"	→	VERY_LONG_SITUATION_NAME
"Body"."SATTRIBUTE"."SATTNAME"	→	FIRST_ATTRIBUTE_NAME
"Body"."SATTRIBUTE"."SATTVALUE"	→	FIRST_ATTRIBUTE_VALUE

Drag elements from input to output to compose message

User specifies the mapping scheme for storing the SITUATION message attributes in the fields of the wmqi_service_control table.

As a result of running this Message Flow, each time a NETSALES message arrives to the “SERVICE CONTROL” node, it will be propagated to the ece_out queue and to the wmqi_events table, where it could be checked. Each time one of the ServiceControl node predefined situations is detected, it will be propagated to the wmqi_service_control table, where it could be checked. Situations will also be written to the report file in the corresponding workspace. The following figure shows the contents of the wmqi_service_control table after a run of SERVICE_CONTROL Message Flow:

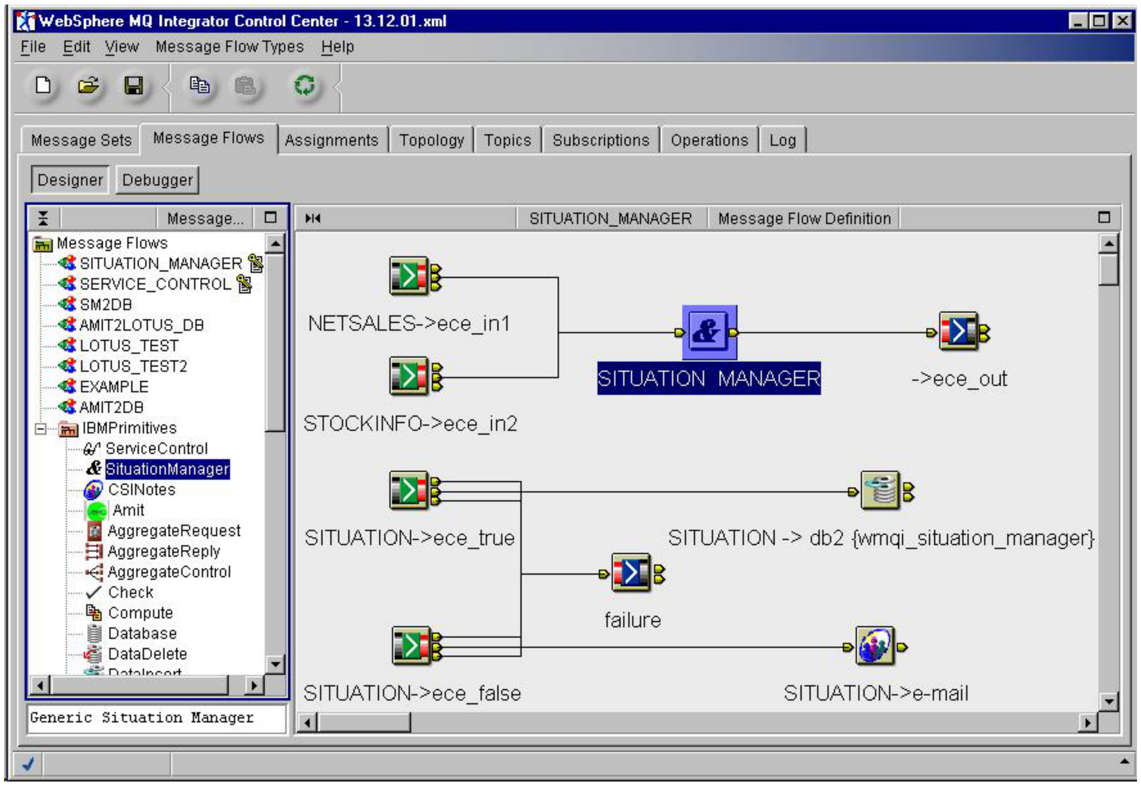

Sample Contents - WMQI_SERVICE_CONTROL

PS93-34 - DB2 - SAMPLE - YURI - WMQI_SERVICE_CONTROL

VERY_LONG_SITUATION_NAME	FIRST_ATTRIBUTE_NAME	FIRST_ATT
mqsiamitComplexSituation	failedDeliveries	2
mqsiamitComplexSituation	failedDeliveries	2
mqsiamitComplexSituation	failedDeliveries	2
mqsiamitComplexSituation	failedDeliveries	2
mqsiamitComplexSituation	failedDeliveries	2
mqsiamitComplexSituation	failedDeliveries	2
mqsiamitComplexSituation	failedDeliveries	2
mqsiamitComplexSituation	failedDeliveries	2
mqsiamitComplexSituation	failedDeliveries	2
mqsiamitComplexSituation	failedDeliveries	2

SituationManager node

The following is the figure of a Message Flow which makes use of a SituationManager node:



Message Flow SITUATION_MANAGER reflects the following scenario:

1. MQInput node “NETSALES->ece_in1” reads NETSALES messages from the ece_in1 queue and transfers them to the “SITUATION_MANAGER” node;

2. MQInput node “STOCKINFO->ece_in2” reads STOCKINFO messages from the ece_in2 queue and transfers them to the “SITUATION_MANAGER” node;

3. “SITUATION_MANAGER” node propagates input messages as is to the “->ece_out” MQOutput node which puts them to the ece_out queue.

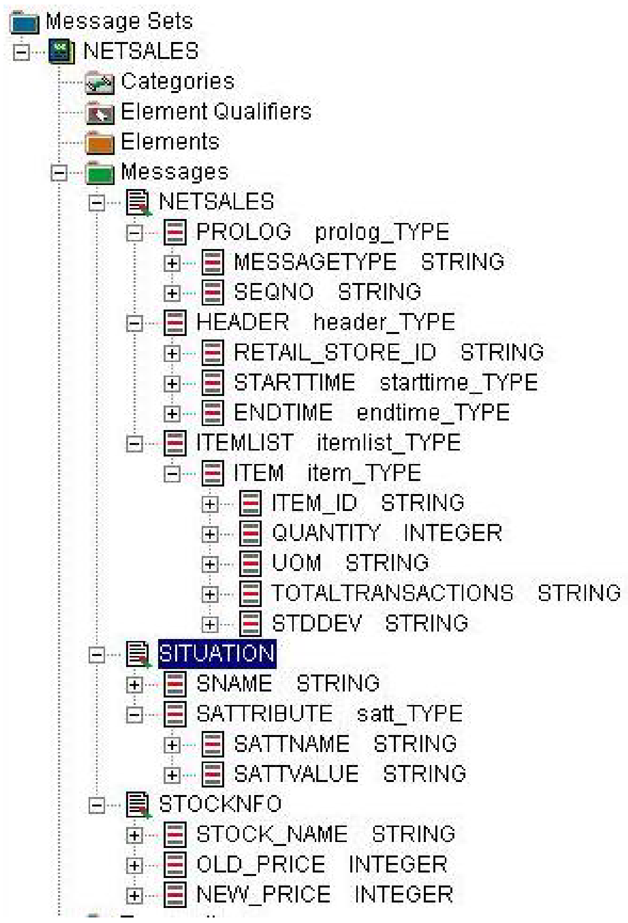
4. Upon situation detection SITUATION message is written by the “SITUATION_MANAGER” node to the:

    a. ece_true queue, from where it is read by the MQInput node “SITUATION->ece_true” and propagated to the DataInsert node “SITUATION->db2 {wmqi_situation_manager}” which stores it in the wmqi_ situation_manager db2 table.

    b. ece_false queue, from where it is read by the MQInput node “SITUATION->ece_false” and propagated to the CSNotes node “SITUATION->e-mail” which sends it in the form of e-mail to the specified e-mail address;

5. MQOutput node “failure” receives the messages in case of any problem (failure, exception).

Message Set NETSALES used in this Message Flow is comprised of three messages, NETSALES, STOCKINFO and SITUATION:



```
Message Sets
NETSALES
    Categories
    Element Qualifiers
    Elements
    Messages
        NETSALES
            PROLOG   prolog_TYPE
                MESSAGETYPE   STRING
                SEQNO   STRING
            HEADER   header_TYPE
                RETAIL_STORE_ID   STRING
                STARTTIME   starttime_TYPE
                ENDTIME   endtime_TYPE
            ITEMLIST   itemlist_TYPE
                ITEM   item_TYPE
                    ITEM_ID   STRING
                    QUANTITY   INTEGER
                    UOM   STRING
                    TOTALTRANSACTIONS   STRING
                    STDDEV   STRING
        SITUATION
            SNAME   STRING
            SATTRIBUTE   satt_TYPE
                SATTNAME   STRING
                SATTVALUE   STRING
        STOCKNFO
            STOCK_NAME   STRING
            OLD_PRICE   INTEGER
            NEW_PRICE   INTEGER
```

As detailed in chapter 3, MRM messages should be mapped to AMiT events. Examples of the AMiT definitions that correspond to the MRM messages above can be found as part of the workspace of the

AmitGui utility that is provided as part of this SupportPac. The NETSALES and STOCKINFO messages have been mapped to the AMiT events in the following way:

<u>NETSALES:</u>
PROLOG
    MESSAGETYPE   // Run-time value can be either “Order” or “Deal” or “Delivery”
    SEQNO   // Order id
HEADER
    RETAIL_STORE_ID
ITEMLIST
    ITEM
        ITEM_ID
        QUANTITY
        TOTALTRANSACTIONS

Not all the original MRM NETSALES message attributes have been mapped (e.g., the following have not been mapped - STARTTIME, ENDTIME, etc.). Among mapped attributes only the bold-faced attributes have been used in situations.

<u>STOCKINFO</u>**:**
STOCK_NAME   **//** Semantically equals to the ITEM_ID attribute in the NETSALES message
OLD_PRICE
NEW_PRICE

Original MRM STOCKINFO message has been mapped to the AMiT event as is.

Based on the above events the following situations have been defined for the node:

1. **Failed Order -** No Delivery message followed the Order message within specified time (30 sec).

2. **Delayed Delivery -** Delivery message followed the Order message after more than 30 sec.

3. **Wrong Delivery -** Wrong item quantity delivered (Delivery quantity != Order quantity).

4. **Full Cycle -** A sequence of Order, Deal & Delivery messages for the same SEQNO.

5. **Popular Item -** At least 3 orders were made for a certain item.

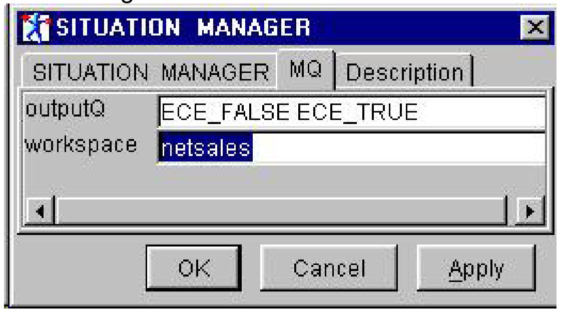
6. **Item Quota -** Item quota reached – more than 25 units of a certain item have been delivered.

7. **Big Order -** The quantity of items ordered in one Order request is more than 5.

8. **Big Day -** At least 2 Big Orders were made during one day (24 sec in the test).

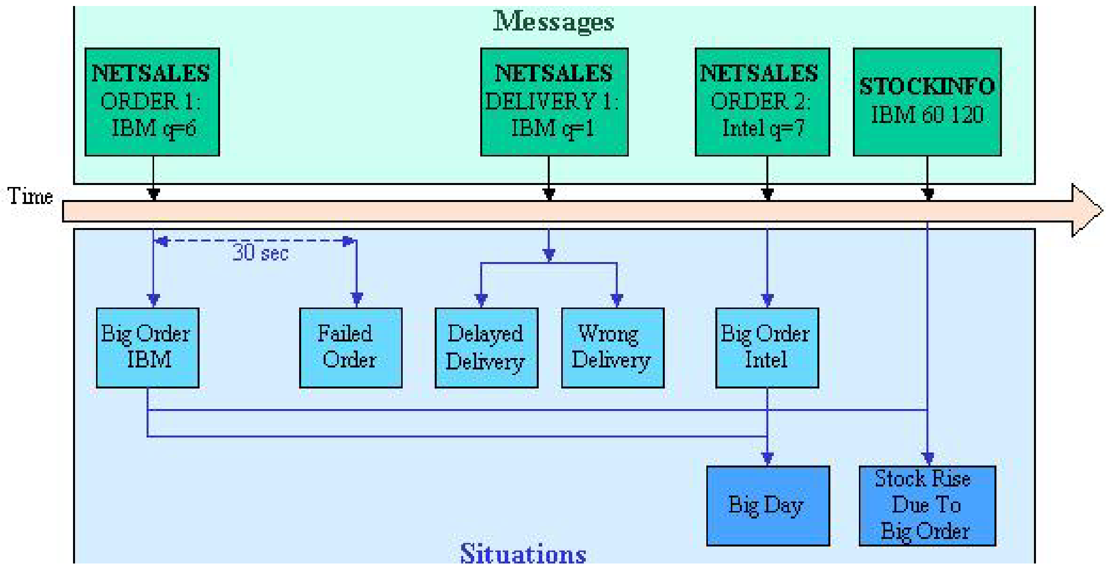
9. **Stock Rise Due Big Order -** Stock price grew following a Big Order for this stock.

Please note that situations 1-7 are defined functions of basic (NETSALES) events, while situation 8 is a function of situation 7, and situation 9 is a function of situation 7 and basic (STOCKINFO) event. The XML definitions of the above events, situations and other relevant meta-data (lifespans, keys) are supplied as part of this SupportPac under the **netsales** workspace. Corresponding SituationManager node configuration looks as follows:

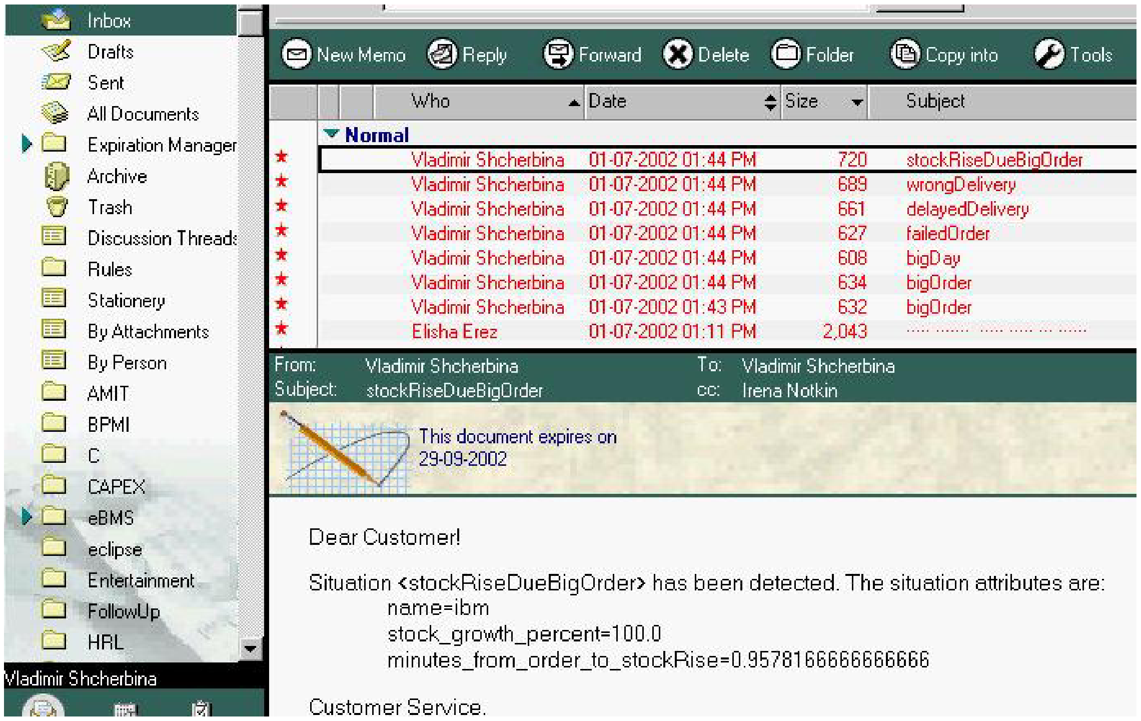


outputQ: ECE_FALSE ECE_TRUE
workspace: netsales

The configuration shown above means that the output SITUATION message should be written to ECE_FALSE and ECE_TRUE queues (see Message Flow scenario above) and the AMiT configuration data should be read from the **netsales** workspace. The following figure illustrates a run of this Message Flow:



SITUATION messages written to the ECE_FALSE queue and propagated to the CSNotes node which sent them as e-mails, can be seen on the following figure:



Who	Date	Size	Subject
Vladimir Shcherbina	01-07-2002 01:44 PM	720	stockRiseDueBigOrder
Vladimir Shcherbina	01-07-2002 01:44 PM	689	wrongDelivery
Vladimir Shcherbina	01-07-2002 01:44 PM	661	delayedDelivery
Vladimir Shcherbina	01-07-2002 01:44 PM	627	failedOrder
Vladimir Shcherbina	01-07-2002 01:44 PM	608	bigDay
Vladimir Shcherbina	01-07-2002 01:44 PM	634	bigOrder
Vladimir Shcherbina	01-07-2002 01:43 PM	632	bigOrder
Elisha Erez	01-07-2002 01:11 PM	2,043	[illegible]

From: Vladimir Shcherbina To: Vladimir Shcherbina
Subject: stockRiseDueBigOrder cc: Irena Notkin

```
Dear Customer!

Situation <stockRiseDueBigOrder> has been detected. The situation attributes are:
        name=ibm
        stock_growth_percent=100.0
        minutes_from_order_to_stockRise=0.9578166666666666

Customer Service.
```

(Note: according to the CSNotes node configuration, e-mails are sent from the e-mail address, which is currently active on the machine, running the Message Flow.)

--- End of Document ---