

MQSeries Integrator V2

Change management and naming standards examples

Version: 1.0

February 20, 2001

Mark Cox
Andrew Humphreys
Emir Garza

IBM Software Group Services

Property of IBM

Take Note!

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

First Edition, February 2001

This edition applies to Version 1.0 of *MQSeries Integrator V2 - Change management and naming standards examples* 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

1.0	Overview	1
1.1.	Workspace And Export Files.....	2
2.0	Software Change Management Process.....	3
2.1.	Message Flows Promotion Procedure	3
2.1.1.	Export From Development.....	3
2.1.2.	Import Into System Test.....	4
2.1.3.	Import Into Production.....	4
2.1.4.	Message Flow Migration - Sub-Flows.....	5
2.1.5.	Message Flows Promotion Procedure Diagram	5
2.2.	Message Sets Promotion Procedure	6
2.2.1.	Export From Development.....	6
2.2.2.	Import Into System Test.....	6
2.2.3.	Import Into Production.....	7
2.2.4.	Message Sets Promotion Procedure Diagram	7
2.3.	NEON Formats & Rules	8
2.3.1.	Export From Development.....	8
2.3.2.	Import Into System Test.....	8
2.3.3.	Import Into Production.....	9
2.4.	Publish/Subscribe Topic Changes	10
2.4.1.	Export From Development.....	10
2.4.2.	Import Into System Test.....	10
2.4.3.	Import Into Production.....	10
2.5.	Plug-in Nodes and Parsers	11
2.6.	Back-out Procedure	12
2.7.	Changes to Configuration	13
3.0	Hardware and Product Software Change Management Process.....	14
4.0	Naming Standards	15
4.1.	MQSeries Integrator Naming Standards.....	15
4.2.	Message Flow Naming Example.....	18

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 been 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:

- CMVC
- IBM
- MQSeries
- MQSeries Integrator
- MQSI

The following terms are trademarks of other companies:

- NEON New Era of Networks, Inc
- PVCS Merant

Summary of Amendments

Date	Changes
20 February, 2001	Initial release

Bibliography

- *MQSeries Integrator v2.0.1 Programming Guide*, IBM Corporation. SC34-5603-01.

1.0 Overview

This document provides suggested procedures for version management and change control of MQSeries Integrator V2.0.1 (MQSIv201) objects. This comes as the result of a number of requests from customers who are going into production using MQSIv201 and wanted some ideas as to ways that this could be achieved with the currently available product.

An assumption is made in this document that there are three environments between which MQSIv2 objects need to be promoted. These are Development, System Test (including stress and performance testing) and Production.

The change control process should meet the following standards:

- There should be separation of environments
- All source code should be versioned and controlled by a version control system
- All prior versions should be available for backout purposes in the event of a problem
- All changes to Test and Production should be documented for auditing purposes
- Change control procedures should be automated as much as possible to avoid errors

It is assumed that you already have in your infrastructure some form of version control system (for example CMVC, PVCS etc). This is a key component of the suggested methodology as it provides not only a safe storage area for current and previous object versions but also allows detailed tracking and alteration information to be stored.

It is also assumed that you have downloaded and installed the MQSIv201 SupportPac IC01 entitled "Message Flow versioning utilities".

Although this document provides a guide for implementing a Change Management process it should be stressed that this may not be the correct solution for all customers. It may be appropriate to only implement a number of the steps described here to be consistent with the particular requirements within your environment.

Finally at the end of this document is a brief section discussing MQSeries Integrator object Naming Standards. This was added to provide some guidelines for users about how to consider naming the more common MQSIv201 objects. Again, as above, these are only guidelines and should be used in conjunction with your current processes and procedures.

One final statement to make here is that although it may appear that this document deals only with MQSIv201 the processes and procedures detailed here could also be applicable to both previous and future releases. This is because we have attempted to keep the suggestions both of a general and broad nature. There may be additional functionality present in future releases of the product meaning that some procedures here may change slightly but generally the information contained here could still be used within your organisation.

1.1. Workspace And Export Files

An example has been given in this document for a naming standard for the MQSeries Integrator Workspace and Export files. This is to allow you to easily recognise the type of file you have either on your hard disk or stored safely in your chosen version control system. The example used for these is as follows:

Saved Workspace	ProjectName.DDMMYYYY.Version.WKS
Exported Message Flows	ProjectName.DDMMYYYY.Version.IMBEXP
Exported Pub/Sub Topics	ProjectName.DDMMYYYY.Version.TPSEXP
Exported Message Sets	MessageSetName.DDMMYYYY.Version.MRPEXP
Exported NEON Formats	FormatName.DDMMYYYY.Version.NNFIE
Exported NEON Rules	ApplicationGroup.DDMMYYYY.Version.NNRIE

- ProjectName – This is a unique identifier prefix given to the name to relate it to either a particular department and/or project (for example InternetStockProject)
- DDMMYYYY – Saved or exported date of the file (for example 25122000)
- Version – The version number of this set of objects (for example 001)

The last part of the name specified is an extension declaring where the file came from originally and where it should be imported.

WKS	Workspace File
IMBEXP	IBM Message Broker Export File
MRPEXP	Message Repository Export File
TPSEXP	Pub/Sub Topics Export File
NNFIE	NEON Formats Export File
NNRIE	NEON Rules Export File

2.0 Software Change Management Process

There are various components in the MQSiv201 environment that need to be considered when looking at version control and change management. This document provides suggested procedures for:

- Message set changes and deployment
- Message flow changes and deployment
- Publish/Subscribe topic changes and deployment
- NEON Format changes and deployment
- NEON Rules changes and deployment
- Plug in Node/Parser changes and deployment
- Execution group configuration changes and deployment
- Broker configuration changes and deployment

THE MQSiv201 release requires separate procedures for each component. In a future product release, however, it is envisaged that the procedures should merge closer together and perhaps some of the additional steps mentioned here could be omitted.

2.1. Message Flows Promotion Procedure

2.1.1. Export From Development

When developing Message Flows you should:

- Create a new Workspace to contain new or duplicated message flows during development. This Workspace should be named according to your naming standards. A suggested naming standard, for example, could follow *ProjectName.DMMYYYY.Version.WKS*
- Ensure all message flow development is carried out within the Development environment until you are happy that the message flows are ready to go into System Test

Once development is complete the message flow should then be locked by development so that no further changes can be made to that particular version whilst it is in System Test. To export message flows from the development environment perform the following steps:

1. Create a new workspace in the Control Center. Add all message flows to be exported to the newly created workspace
2. Export the Workspace to a flat file using the Control Center GUI File->Export option and give the export file the same name as the Workspace
3. Use the *mqsfiltermsgflows* utility from the IC01 support pack to create a separate export file for each message flow. The *mqsfiltermsgflows* utility takes an export file and creates a 'filtered' export file containing a single message flow, without modifying the original export file. The utility requires two parameters, the name of the export file and the message flow to be filtered. As this is a command line process it would be possible to build a batch utility to automate this process so all the message flows in an export file are filtered into individual files with a single command. The individual files are automatically named *MessageFlowName.XML*

4. Check the files created in the previous step into the chosen version control system
5. Update the Version Control tracking document to hold the current status along with any information considered necessary at this stage

2.1.2. Import Into System Test

The team responsible for importing the message flow into the System Test environment should also be responsible for assigning it to the necessary broker(s) and deploying it. This would also involve deleting any previous editions of the message flow, which have had to go back into Development if System Test fails. To import message flows into System Test:

1. Copy all required message flow export files from the version control system onto the target Configuration Manager server file system
2. Merge all message flow export files into one using the *mqsicombinmsgflows* utility from the IC01 SupportPac. The *mqsicombinmsgflows* utility takes a list of 'filtered' export files, as produced by the *mqsifiltermsgflows* utility, and combines them into a single export file. Copy all the filtered export files into a single directory and run the utility passing as parameters the directory and name of the new combined export file to be created
3. Use the *mqsdeletemsgflows* utility to delete all message flows contained in the export file created by *mqsicombinmsgflows* from the System Test environment Configuration Manager database. This removes the need to use the Control Center to check out all message flows to be updated prior to importing the new versions. The *mqsdeletemsgflows* utility reads through an export file and deletes all unlocked matching message flows directly from the shared configuration in the Configuration Manager database. Note that when deleting, a message flow is considered to match if it has a label the same or a UUID the same. So the net result is that any message flow with either the same label or the same UUID will be deleted. To prevent message flows being deleted by the *mqsdeletemsgflows* utility they should be locked using the Control Center
4. Use the Control Center GUI File->Import option to import the new message flows from the file to the Control Center workspace. Specify "Message Flows Only" on the Import Options panel
5. Issue a local saved to shared in the Control Center GUI to check the new message flows into the Configuration Manager database
6. If new message flows have been added or removed from the environment (but not if an existing version of a message flow is just being updated), check out all affected execution groups, remove deleted message flows and updated message flows from all affected execution groups. Assign all new message flows to the required execution groups and check in all affected execution groups
7. Deploy the changes to all affected Brokers
8. Update the version control tracking document to hold the current status along with any information considered necessary at this stage

System Test procedures should now be performed, including stress and performance tests. If System Test fails then the failing message flow(s) should be checked out of the version control system, unlocked in the development environment and imported to the development environment to be amended. Note though, at this point, the version number of the message flow should not be increased (if your naming standard includes the version number in the message flow name).

2.1.3. Import Into Production

If the message flow passes System Test then the appropriate team will then be responsible for promoting the message flow into Production.

The same process should be followed to import message flows into production as is used in System Test. The version control system should be used as the point of record for all versions of the objects so the imports into production should be from the files held in the source control system. The version control tracking document should be updated to hold the current status along with any information considered necessary at this stage.

2.1.4. Message Flow Migration - Sub-Flows

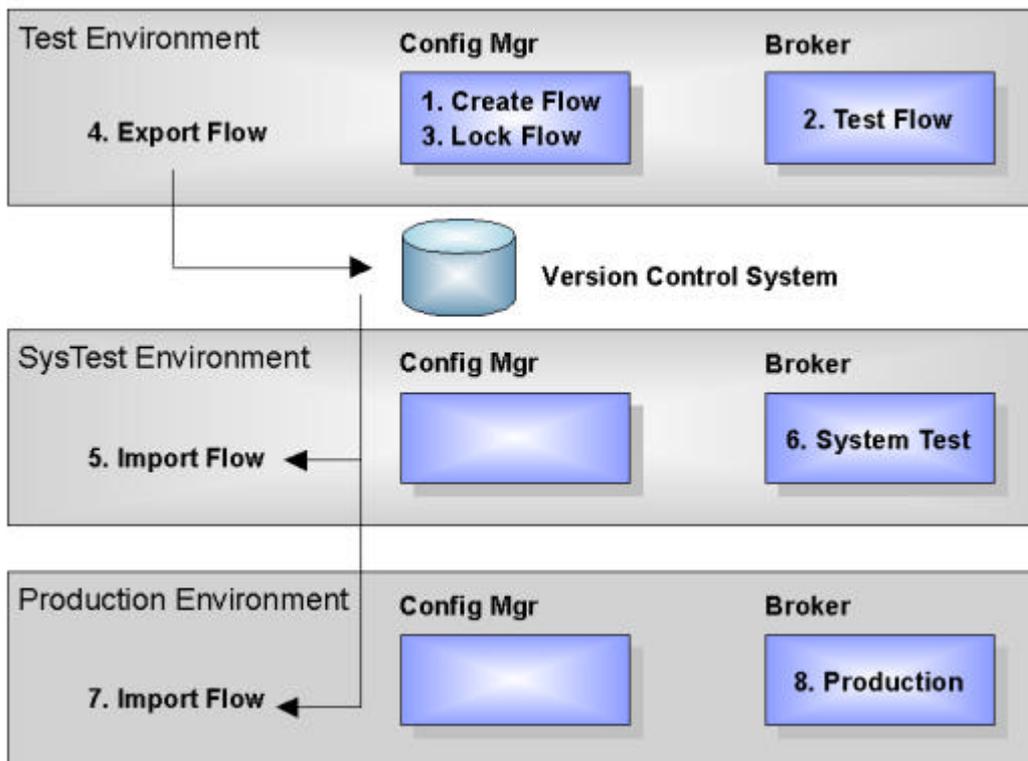
In terms of migrating between environments sub-flows should be considered to be message flows in their own right. The IC01 support pack utilities treat sub-flows in the same way as the Control Center message flow view. That is they are considered to be separate message flows so you need to treat them as such during your promotion procedures.

If you migrate message flows that contain sub-flows, any new message flows that contain new sub-flows will need both the message flow and sub-flows to be migrated. When just the sub-flow has changed you need not migrate all message flows that use the sub-flow, you just need to migrate the new version of the sub-flow (Note: you will need to re-deploy all Execution Groups that are running message flows that contain the updated sub-flows). If a message flow has changed but a sub-flow has not, only the new version of the message flow need be migrated. If both a message flow and sub-flow has changed them both must be migrated.

If you use a sub-flow extensively within your message flows then care should be taken to ensure that when a sub-flow is updated, all of the message flows using it are also updated.

This may need some form of tracking mechanism to be held in your version control system.

2.1.5. Message Flows Promotion Procedure Diagram



2.2. Message Sets Promotion Procedure

2.2.1. Export From Development

When developing message sets you should ensure:

- All message set development is carried out within the Development environment
- The message set is not finalised in Development until it is migrated to production as there may be future changes necessary depending on the outcome of the System Test procedures

To export message sets from development:

1. The message set export command line utility should be used (*mqsimrmimpexp*) to selectively export the required message set. The naming standard should be in line with your company naming standards. A suggested naming standard, for example, could be *ProjectName.DDMMYYYY.Version.MRPEXP*
2. This message set export file should be entered into the chosen version control system along with any additional information considered necessary for tracking

2.2.2. Import Into System Test

The team responsible for importing the message set into the System Test environment should also be responsible for assigning it to the necessary broker(s) and deploying it. This would also involve deleting any previous editions of the message set which have had to go back into Development if System Test fails. To import message sets into System Test:

1. Delete existing versions of the message sets to be imported from the System Test Configuration Manager. This is required since the message set import utility does not have an overwrite facility. To achieve this it is recommended to delete the message set using the delete option of the Control Center GUI. It would be possible to write a utility to delete the entries relating to the message set directly from the MRM database, which would enable the process to be automated, however the database schema may change in future MQSI releases so any utility may become obsolete
2. Import the message sets into the System Test environment using the command line *mqsimrmimpexp* utility
3. If either new message sets have been added or existing message sets have been removed from the environment (but not if just new versions of existing message sets are being updated), check out all affected Brokers, remove deleted message sets from all affected brokers, assign all new message sets to the required Brokers and Save all to Shared
4. Deploy the changes to all affected Brokers
5. System Test procedures should now be performed, including stress and performance testing
6. The version control tracking document should be updated to hold the current status along with any information considered necessary at this stage.
7. If System Test fails then the message sets should be updated in development not system test. Note though, at this point, the version number of the message set should not be increased.
8. If System Test passes then the appropriate team will then be responsible for importing the Message Set into Production.

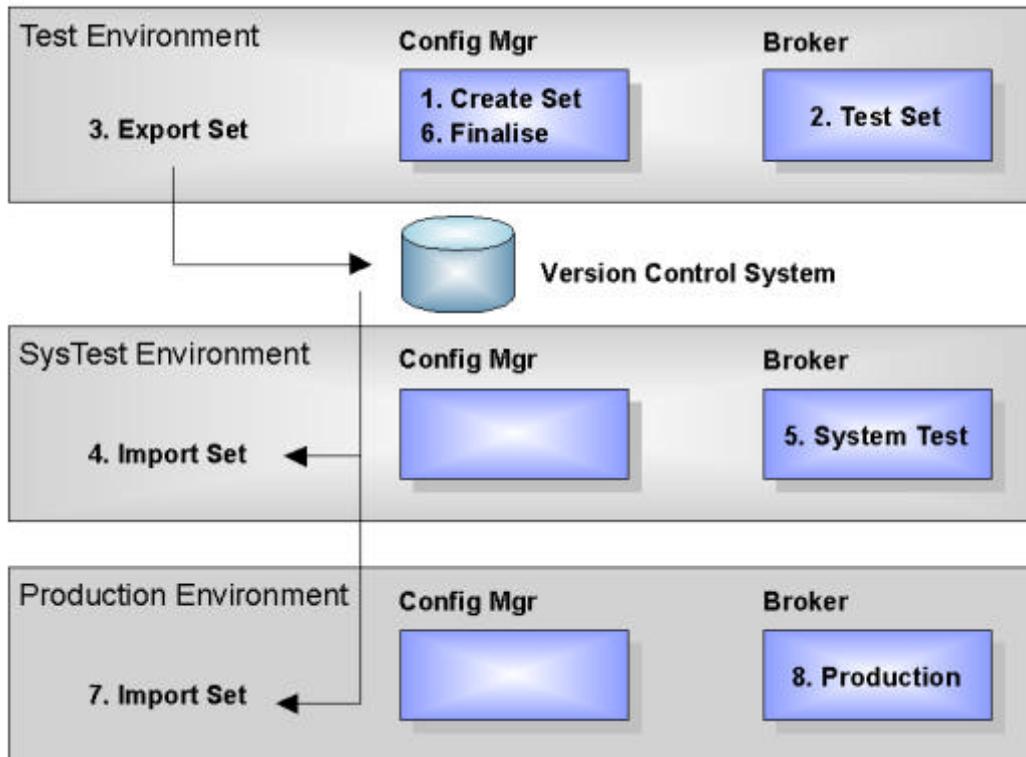
2.2.3. Import Into Production

The same process will be followed to import message sets into production as is used in System Test. The version control system should be used as the point of record for all versions of the objects so the imports into production should be from the files held in the source control system. The Version Control tracking document should be updated to hold the current status along with any information considered necessary at this stage.

At this stage there may be a future requirement to update the message set because of changes in message formats etc. In this case the message set should be copied from the version control system to the development environment, the message set duplicated (based on the existing message set if appropriate) and the version number incremented to reflect the difference.

Note: After using the *mqsimrmimpexp* utility to import a message set you should currently be aware that you are required to stop and start both the Configuration Manager and the Control Centre. This information is detailed on Page 136 of the Administration Guide.

2.2.4. Message Sets Promotion Procedure Diagram



2.3. NEON Formats & Rules

2.3.1. Export From Development

If you have designed message flows that use NEONFormatter or NEONRules nodes you need to migrate the formats and rules defined in the NEON database between environments.

To export NEON formats:

1. Use the NEON NNFie utility to export a single NEON Format to a flat file. The naming standard should be in line with your company naming standards. A suggested naming standard, for example, could be *FormatName.DDMMYYYY.Version.NNFIE*
2. Check the export file into the source control system
3. Update the version control tracking document to hold the current status along with any information considered necessary at this stage

To export NEON rules:

1. Use the NEON NNRie utility to export a single Application Group's NEON rules to a flat file. The naming standard should be in line with your company naming standards. A suggested naming standard, for example, could be *ApplicationGroup.DDMMYYYY.Version.NNRIE*
2. Check the export file into the source control system.
3. Update the version control tracking document to hold the current status along with any information considered necessary at this stage

2.3.2. Import Into System Test

The team responsible for importing NEON formats and rules into the System Test environment will be responsible for assigning it to the necessary broker and deploying it. This would also involve deleting any previous versions, which have had to go back into Development if System Test had failed. To import NEON formats and rules:

1. Check the required NEON Formats and Rules export files out of the version control system
2. Import the NEON Format using the NNFie command line utility with the overwrite option to delete previous versions of the Format
3. Import the NEON Rules using the NNRie command line utility with the overwrite option to delete previous versions of the Rules
4. Update the version control tracking document to hold the current status along with any information considered necessary at this stage
5. Run the command line utility *mqsinfreload* against the affected Brokers to re-cache the NEON formats if necessary

2.3.3. Import Into Production

The same process will be followed to import NEON formats and rules into production as is used in System Test. The source control system should be used as the point of record for all versions of the objects so the imports into production should be from the files held in the source control system. The Version Control tracking document should be updated to hold the current status along with any information considered necessary at this stage.

2.4. Publish/Subscribe Topic Changes

2.4.1. Export From Development

Publish/Subscribe topics can be exported from a Configuration Manager to a flat file in the same way as message flows by using the File->Export option on the Control Center GUI. However the IC01 SupportPac does not work with topics so it is not possible to use the SupportPac to create export files for individual topics. To export topics:

1. Open a new Workspace and add the topics to be exported to the workspace. If it is required to store each topic individually in the version control system you will need to have just the individual topic you wish to work with on the workspace when you do the export
2. Export the workspace using the Control Center File->Export option. The naming standard should be in line with your company naming standards. A suggested naming standard, for example, could be *ProjectName.DDMMYYYY.Version.WKS*
3. Check the export file into the version control system
4. Update the version control tracking document with the current status along and any information considered necessary at this stage

2.4.2. Import Into System Test

The team responsible for importing the topics into the System Test environment will be responsible for deploying the changes. This would also involve deleting any previous versions of the topic, which have had to go back into Development if System Test had failed. To import topics:

1. Open a new Control Center workspace and add all topics to the workspace.
2. Check out all topics that are to be updated
3. Use the Control Center GUI File->Import function to import from the file. Specify Topics only on the import options panel
4. Issue a local saved to shared to check in all topics
5. Deploy topic changes
6. Update the version control tracking document with the current status along and any information considered necessary at this stage

2.4.3. Import Into Production

The same process will be followed to import topics into production as is used in System Test. The source control system should be used as the point of record for all versions of the objects so the imports into production should be from the files held in the source control system. The Version Control tracking document should be updated to hold the current status along with any information considered necessary at this stage.

2.5. Plug-in Nodes and Parsers

Plug-in nodes and parsers consist of a number of files that need to be managed by the version and change control process. They are:

- The source C code and header files for the plug in
- The compiled LIL file

If a plug-in node is also integrated into the Control Center the following files may also be required:

- The XML interface definition file (mandatory)
- The WDP file (mandatory)
- The icon files (optional)
- The help file (optional)
- The properties file (optional)
- A property editor (optional)
- A customiser (optional)

Copies of all these files should be maintained in the Version Control system. When moving between environments the Version Control tracking document should be update with the current status along and any other information considered necessary.

To install the plug-in an a new environment the files will need to be copied from the version control system to the appropriate location on the file system on the machines in the new environment. Refer to the *MQSeries Integrator v2.0.1 Programming Guide Chapter 7 – Installing a Plug-in Node or Parser* for details on this.

Note that a Broker must be stopped and restarted to detect the new LIL.

2.6. Back-out Procedure

It is suggested that at least two previous versions of all objects are stored on the System Test and Production machines. This would allow the appropriate team to quickly re-assign the old Versions to the broker and re-deploy efficiently in the event of any problems occurring. If any previous versions prior to this are required for any reason then these can be restored from the Version Control system. Note that this is dependent on your naming standards including the version number in the name of each MQSIv201 object.

It is highly recommended that at all times, if problems occur, then, as much information as possible pertaining to the problem should be recorded into the Version Control System. This is not only for historical data retention but also to aid new members to projects to quickly understand the previous issues and resolutions.

2.7. Changes to Configuration

Changes to Broker configurations, i.e. the addition or deletion of Brokers and Execution Groups and changes to the Message Flows assigned to Execution Groups and number of instances of each Message Flow to be run should be made using the Control Center for each environment. This is because these settings are likely to be configured differently for each environment.

3.0 Hardware and Product Software Change Management Process

Along with MQSeries Integrator object change management you also have to consider changes to system hardware and product software. Product software changes not only include new versions of software (for example upgrading MQSeries V5.1 to V5.2) but also software maintenance and corrective service. Some of the more major points to consider are highlighted below:

- Time should be given for any product maintenance to mature in the field (approx. two months) if the maintenance is not needed to fix an urgent problem
- Any maintenance should go through regression testing in your corporate System Test environment
- Applying the product maintenance should go through the set procedure so as to ensure that a back-out option is always available. Also the back-out procedure should be tested in System Test before the change is applied to Production to make sure it works.
- Any changes made to Production should be made out of hours to minimise impact.

For hardware and machine upgrades time should be allocated out of hours for this to take place. For Test and System Test this should not cause any problems but for Production this needs to be carefully managed. For the Production environment there may be a failover capability so that if one machine goes off-line then there is sufficient capacity for other machines in the domain to continue operation and assume the workload of the off-line machine.

Therefore if an upgrade is necessary to a Production machine these failover procedures should take over and the machine hardware upgrades can take place without any disruption to any live production applications (if any as this should be done out of hours). When the machine is brought back on line then normal service should resume and the upgraded machine can assume its normal workload.

If any hardware components for a particular machine are hot swappable and do not require the machine to be taken off line then this is a preferable solution. Although as per the previous paragraph these changes should still be made out of hours in case of any unforeseen circumstances.

If any new software is added to System Test or Production machines then installation documentation should be produced as well as backup and recovery documentation. This is to allow any machines to be rebuilt to current specifications in case of disaster recovery, or any new capacity machines to be included into the current domain with the same specifications as the current machines.

4.0 Naming Standards

As a general rule naming standards for MQSeries and MQSeries Integrator components should closely follow organisational rules for naming objects but there may be a few subtle differences required.

4.1. MQSeries Integrator Naming Standards

A naming convention for all components of the MQSeries Integrator network is required to ensure that all names are unique and that users have clear guidance as to how to name new objects. This section of the document aims to lay out a suggested naming convention for an MQSeries Integrator network.

Brokers

The name used at creation time must be unique within the broker domain. Since Queue Manager names are normally unique within a given organisations environment it is suggested that brokers should have the same name as the Queue Managers they are associated with. For example a Queue Manager called "MQSI" would have a broker "MQSI" associated with it.

Execution Groups

The name for each Execution Group must be unique within a broker. It would be recommended to separate Execution Groups by department name. This means that all related departmental message flows could be grouped easily within the broker.

There may, however, be a situation where message flows within a department may need to be kept apart from the others (for example if one is particularly large). In this instance the name of the execution group could be the department name with a suffix of the relevant project name or identifier.

One further thing you may wish to consider when working within the development environment is having separate Execution Groups per developer. The benefit of this is that it will avoid interference at deploy time with other developers and also keep each persons work autonomous within system processes for testing purposes. One way of achieving this would be to use a prefix or suffix on the Execution Group name to identify a particular developer. However, you could only use this naming standard in a development environment, as the Execution Group configuration in System Test needs to exactly match that of Production.

Message Flows

The name for a message flow must be unique within a broker domain. Any reference to that name within the domain must use that one message flow. However, several brokers or execution groups may use a single message flow. A suggested naming standard for message flows could be:

ProjectCode_MessageFlowDescription_VersionNumber

Change management and naming standards examples

1. Project Code is an alphanumeric code representing the project that owns the message flow, for example "BillingProject"
2. Message Flow Description is a short description that is concise, clear and understandable as to the function of the message flow, for example "AccountValidation"
3. Version Number is the current version of the message flow being used. Previous versions and version information should be stored in the chosen version control system.
4. All sections of the message flow name should be separated by an underscore

Message Sets

The name for a message set must be unique within a broker domain. Any reference to that name within the domain must use that one message set. However, several brokers may use a single message set. A suggested naming standard for message sets could be:

ProjectCode_MessageSetDescription_VersionNumber

1. Project Code is an alphanumeric code representing the project that owns the message set, for example "BillingProject"
2. Message Set Description is a short description that is concise, clear and understandable as to the function of the message set, for example "AccountingMessages"
3. Version Number is the current version of the message set being used. Previous versions and version information should be stored in the chosen version control system.
4. All sections of the message set name should be separated by an underscore

Messages

The name for a message must be unique within a message set. Any reference to that name within the message set must use that one message. However, several message sets may use a single message. A suggested naming standard for message could be:

ProjectCode_MessageDescription_InputOrOutput_VersionNumber

1. Project Code is an alphanumeric code representing the project that owns the message, for example "BillingProject"
2. Message Description is a short description that is concise, clear and understandable as to the function of the message set, for example "Account Validation"
3. Input or Output describes whether the message is an "Input" or "Output" message type. If a message is of both Input and Output type then this could be set to "Generic" or left blank depending on your preference.
4. Version Number is the current version of the message being used. Previous versions and version information should be stored in the chosen version control system.
5. All sections of the message name should be separated by an underscore

Going down into lower levels from messages you get to elements, element lengths etc. Naming standards are normally not needed at this point as element names are normally generated automatically from C structure components or COBOL copybook elements if an importer is used. If you are not using the importer then the message element names should be given something meaningful. It should also be noted that message element identifiers will be generated automatically for you and it is recommended to change these to the same names as the elements themselves. The reason for this is that when you use the messages within your message flows you refer to the specific element using the element identifiers and not the element names. Therefore it will make you E-SQL easier to write and understand if your element identifiers are the same as your element names.

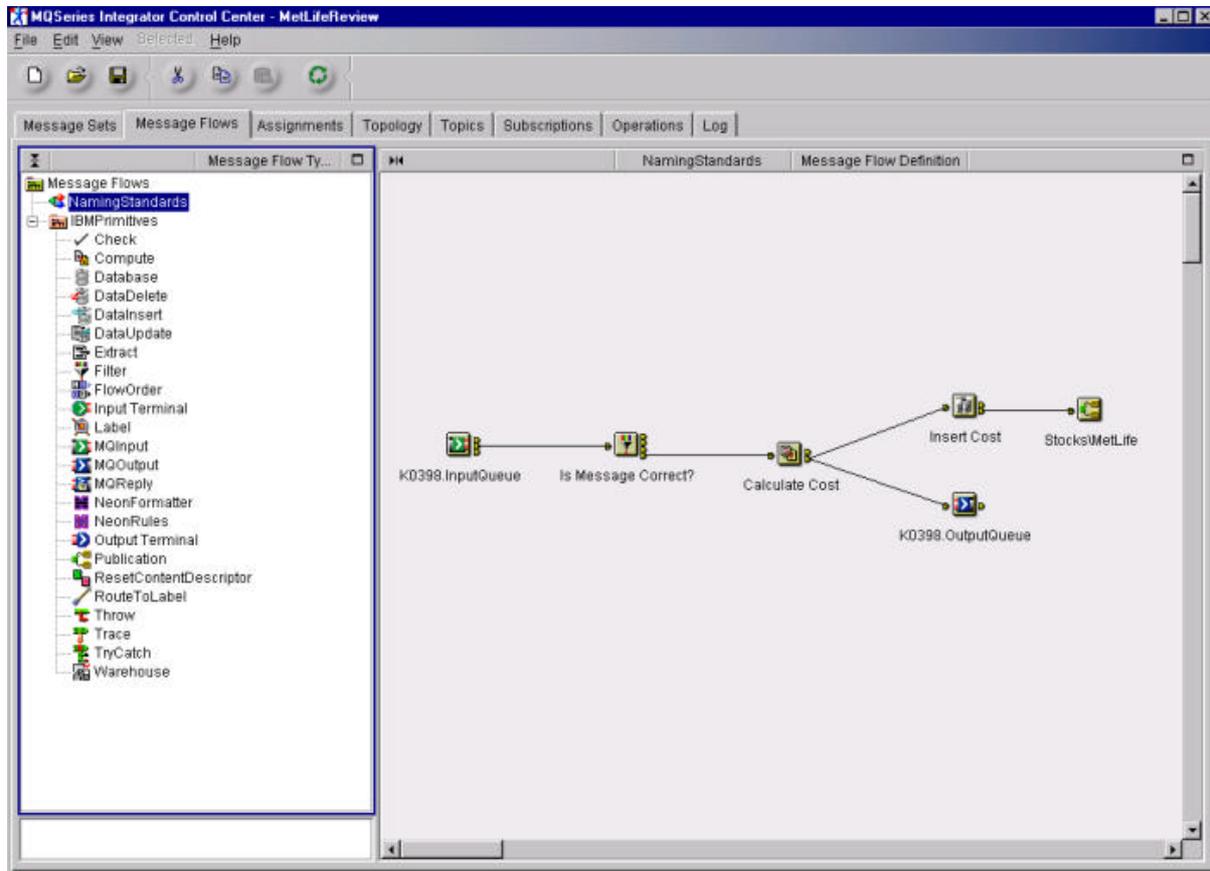
For element lengths, these will be generated automatically in the event of using the importer. If you have to create these manually then you should give them a meaningful name. For example if you are creating an element length for a string of length 20 the call the element length "Length_20". This means that you could re-use this component if another message element required the same element length.

Message Flow Nodes

Generally speaking there is no hard and fast way that message flow nodes should be named. However there are some guidelines that can be presented here in order to make the message flow clearer for people trying to understand its business purpose.

- When naming MQInput and MQOutput nodes always name them the same as the underlying MQSeries queue for clarity
- When naming Filter nodes give them a question title, for example "Is Name NULL?"
- When naming Compute nodes give them a meaningful name to concisely explain their business purpose
- If you are using Database nodes try and give them a meaningful name to concisely explain their business purpose
- When using Publication nodes name the node after the Subscription Point name, or possibly a Business Description of what is being published, or maybe even a combination of both

4.2. Message Flow Naming Example



End of Document