Infrastructure Series

TechDoc

WebSphere Message Broker / IBM Integration Bus

Broker Archive (BAR) File Management (Distributed Systems)

February 2015 - Infrastructure – IBM Message Broker – BAR File Management

Author(s): Glen Brumbaugh

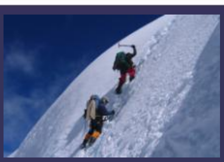


Table of Contents

Introduction

Document Version

This document describes how to manage and deploy WebSphere Message Broker (WMB) / IBM Integration Bus (IIB) Source code and Broker Archive (BAR) files. A necessary part of this deployment process is the maintaining the relationship of the BAR file to the version of the Message Flow software that the BAR file reflects. While this document does not deal with specific Source Code Control (SCC) software programs, it does describe how an SCC product could integrate into the deployment process. This document also describes a solution if no SCC product is in place.

This document should apply to most versions of the WMB/IIB products. The contents of this document have been specified verified on the following production versions:

• WebSphere Message Broker	v7.0.0.2
• IBM Integration Bus	v9.0.0.0

This documentation has been created and maintained by:

- Glen Brumbaugh

This documentation has been reviewed by:

- Chris Crampton

This documentation was last updated:

- Version 1.1 February 2015

WebSphere Message Broker Overview

Product Name

The product currently known as IBM Integration Bus has been through a number of different product names during its several decade long evolution. The product was originally developed by the New Era of Networks (NEON) Corporation and was marketed and resold by IBM. IBM completely redesigned and rebuilt the product and released their own in-house developed product beginning with version 2.0. The product has had the following names and version numbers:

• MQSeries Integrator (MQSI)	Version 1.0 – 2.0
• WebSphere MQSeries Integrator	Version 2.1
• WebSphere Business Integration Message Broker (WBIMB)	V5.0
• WebSphere Message Broker (WMB)	Version 6.0 - 8.x
• IBM Integration Bus (IIB)	Version 9.0 - 10.0

For the remainder of this document, the product will be referred to as "*Message Broker*". This is both for historical reasons and to signify that this documentation applies to both the WMB and IIB product versions.

Terminology

With the Version 9.0 product rename (to IBM Integration Bus), several key product architectural components were given new names; while continuing to fill virtually the same role they had previously filled. This documentation will continue to refer to the "old" names because the steps documented here refer to both old and new product versions.

The old and corresponding new names are as follows:

- Message Broker → Now called "Integration Node" (Beginning with v9.0)
- Execution Group → Now called "Integration Server" (Beginning with v9.0)
- Message Flow → Still called "Message Flow"

Product Architecture

While some Message Broker architectural elements have come and gone over the years, there are two fundamental components of the Message Broker that have not changed. These two components are:

- Message Broker Runtime (e.g. Message Broker)
- Message Broker Toolkit

Applications are <u>developed</u> using the Eclipse based Toolkit. This standalone product is typically deployed on an Application Developer's or Administrator's personal computer (desktop or laptop; typically Windows). Message Broker applications, called Message Flows, are developed in this component and all of the necessary files for a given application deployment are gathered together into a Broker Archive (BAR) file for deployment and execution in an Execution Group.

Applications (e.g. Message Flows) are executed inside of a Message Broker. Within a Message Broker, Message Flows are contained within an Execution Group. Message Flows are therefore deployed to a specific Execution Group as a Broker Archive (BAR) file.

Message Flow Deployment

Introduction

Message Broker development involves the creation of Message Flows using the Message Broker Toolkit. The deployment of these Message Flows involves the building, migration, and configuration of Broker Archive (BAR) files. BAR files containing one or more Message Flows and are deployed into a specific Broker and Execution Group. As with any other programming language component, a BAR file should be created in the Development environment and migrated through the Test and Production environments.

While a BAR file is the runtime "package" for Message Broker, it does not contain the source code. That code consists of a number of different types of files. These files are created or imported and then stored within the Message Broker Toolkit during the development process. The file types include, but are not limited to, the following:

• Message Flow files	(".msgflow", ".subflow", "cmf", etc. files)
• Program files	(".class", ".esql", etc. files)
• Data Definition files	(".mset", ".mxsd", ".wsdl", ".xsd", "xml", etc. files)
• Resource Adapter Archive files	(".rar", etc. files)
• Test Files	(".mbtest", etc. files)

As can be seen, there are a number of different types of files that can be involved in building a BAR file. In this respect, a BAR file is quite similar to an Enterprise Archive (EAR) file, containing all of the elements necessary to support the runtime functions provided by code within the file.

The Message Broker Toolkit does not provide any automated Message Flow Source Code versioning. All of the necessary source code files to deploy an application are packaged together in the Toolkit and used to build a Broker Archive (BAR) file. Message Broker assumes that any required Source Code Control (SCC) will be provided by an external SCC software package.

Source Code Control software will generally not be able either to backup Message Broker Source Code or to create and deploy BAR files without some custom modifications. This deployment process can be scripted and may be added to the SCC system. In this case, both the source code and the "properties" files to configure the BAR files for specific environments need to be loaded into the Source Code Control system. Scripts can then be developed to build and deploy the BAR files using both the Source Code and the Properties files.

The remainder of this document describes the source code versioning process, the BAR file build process, and the BAR file configuration and deployment process. The commands used in those processes are also described. These processes can be either manual or automated and may or may not include external Source Code Control software. This document should contain enough information to allow the deployment process to be scripted, if desired.

Lifecycle Considerations

There is a complex relationship between how the Toolkit is used during the development process, how Source Code is managed, how BAR files are built, how BAR files are deployed, and the Message Broker Execution Groups that the BAR files are deployed into. These relationships are highlighted in Table 1.

In general, the following considerations apply:

- Toolkit Workspaces are used to separate Applications /Projects that have little overlap.
    - Separation by Organization / Line of Business / Application / Development Project
- Applications and Projects are used to Group Message Flows
    - Basic unit of Source Code Control
    - Message Flows that are closely linked operationally
    - Message Flows that share common resources (e.g. XML, XSL, etc.)
    - Message Flows that will be built and deployed together in a common BAR file.
    - Message Flows that will be deployed to the same Execution Group
- Execution Groups are used to group Applications/Projects for execution
    - Message Flows that are part of a common business solution (e.g. Application)
    - Message Flows that can have common operational controls (e.g. Start/Stop)
    - Message Flows that share common resources (e.g. XML, XSL, etc.)

The key to efficiently using the Toolkit throughout the build/deploy lifecycle are the Applications and Projects used during Message Flow development. The size and breadth of these objects defines the granularity of the SCC source code easily backed-up as well as the breadth and scope of any BAR Files built from those Applications or Projects.

It should be noted that the Development and Build environments can be separated. In this case, objects will need to be exported and imported between the environments. If this is done, keep in mind that the build process requires a ".project" file. This means that a "shadow" project will need to be maintained and that individual components will need to be imported in the shadow project for incremental builds. This adds to the overall complexity of the solution, but effectively separates the individual developer environments from the build environment.

Table 1 - Message Broker Object Relationships

Object	Contains	BAR File Impact	Comments
Toolkit	One or more Workspaces		Only one Workspace in use at a time
Workspace	One or more Applications / Projects	Limits scope of BAR File	BAR Files cannot span Workspaces
Application or Project	One or more Message Flows	Directly referenced by BAR File build command	Basic Source Code unit. Contains Message Flows. Contains Programs. Contains Data Definitions. Contains Resources.
BAR File	One or more Message Flows		Basic unit of deployment. Contains executables Contains Data Definitions. Contains resources.
Execution Group	One or more Message Flows	Built by one or more BAR Files	Contains Message Flows and their resources.

Deployment Considerations

A BAR file is created by a Message Broker "Deploy" process, either from the command line or from the Message Broker Toolkit. Thus, at the time the BAR file is created, the BAR file and all of the source files it represents are synchronized. If the BAR file was created from the command line, via an mqsicreatebar command, then those source files may either have been contained within the Message Broker Toolkit workspace or copied from that workspace.

Since the Toolkit keeps only the current version of the Message Flow, any subsequent changes made to the Message Flow in the Toolkit cause a break in the linkage between the BAR file and the source code files that were used to create the BAR file. This situation is, of course, identical to that of any other programming languages. Once development is complete, the deployment process thus consists of three separate steps:

1) The creation of a software "version"; identifying the version and saving the version's files.
2) The creation of an executable (in this case a BAR file) corresponding to the version.
3) The deployment and migration of the executable through the necessary environments.

Across all of these steps, both Source Code and the related BAR files need to be preserved and managed. In fact, the linkage between these two different assets (Source Code and BAR files) is a very essential part of this process.

Source Code Versioning

Once a "version" of the software is ready to be deployed, the first step is to create an official "version" of the software. While the mechanics of this step will vary depending upon the SCC solution, any solution must achieve some common goals:

- The "scoping" of the version (single message flow, multiple flows, resources, etc.)
- The naming of the version (date, version number, label, etc.)
- The saving of the "source code" related to the named version.

If SCC software is used, one of the primary functions of that software will be to save a copy of the software version "source code". The SCC used will also determine the naming or "labeling" of that version.

A quick look at the "scope" of the version will identify a number of potential challenges. These challenges include:

- What Message Flows should be backed up?
  - A single message flow?
  - Multiple message flows that work together to create a single business function?
- What ESQL should be backed up?
  - Is there an ESQL file that matches in scope the related Message Flow(s)?
- What other resources related to the Message Flow(s) need to be backed up?
  - Java classes?
  - Libraries/data definitions used by one or more message flows?

It is important to keep in mind the goal of the software "version". The goal is not to create a perfect change history for each individual component (".msgflow", ".esql", etc.). The goal is to capture all of the files that are involved in building a BAR file and to be able to recover the Message Broker Toolkit state associated with that Message Flow. This means that some files (e.g. ".xsd" files) will be

captured many times and that there may not be any changes in those files across the versions. If the SCC can manage to create a change history across these versions then that is all to the better.

Most SCC software will not, out of the box, support Message Broker deployments from the Toolkit. This means that files must be extracted from the Toolkit Workspace and then imported into the SCC process. A key decision in this process is the scope and mechanism of the source files to be backed up. There are several possible approaches to this issue. The pros and cons of these approaches include:

- Selecting individual files for backup:
    - Is by far the most complicated solution (identifying the files that have changed).
    - Is by far the most difficult to restore.
    - Has the most comprehensive change history capability.
    - Requires the least amount of disk space.
    - Is by the most error prone solution (it is easy to miss backing up a file).
- Backup up a container (Project or Application):
    - Is a fairly simple solution.
    - Also requires backing up libraries/resources (e.g. ".xsd" files).
    - Is fairly easy to restore (but requires care to avoid mixing old and new).
    - Has limited change history capability.
    - Is <u>not</u> guaranteed to restore a working Toolkit environment (resources missing).
- Backup up a Workspace:
    - Is the least complicated solution.
    - Is by far the easiest to restore.
    - Has the least comprehensive change history capability.
    - Requires the most amount of disk space.
    - Is guaranteed to restore a working Toolkit environment.

As can be seen, the approach decided upon involves the weighing of various tradeoffs. The decision as to the scope of the source code to backup and version, however, cannot be made without also considering:

- How BAR files will be created from the backed-up version of the source code.
- How the backup-up source code version will be reloaded, if necessary, in the Toolkit.
- How developers will interact with the deployment process:
    - Scope of Workspace.
    - Use and scope of Applications.
    - Scope of Projects.

BAR File Creation

Broker Archive (BAR) files are created either from the Toolkit or by using the *mqsicreatebar* command. A close look at this command will show how intimately connected BAR file creation is with the Toolkit (Workbench) file system. The *mqsicreatebar* parameters are as follows:

- **mqsicreatebar**

-data *WorkspacePath*	(Workspace from which to create the BAR file)
-bar *Path/BARFileName*	(Path and BAR file name)
-a *ApplicationName*	(List of Applications to include in the BAR file)
-p *ProjectList*	(List of Projects to include in the BAR file)

- -l *LibraryName* (List of Libraries to include in the BAR file)
- -o *ProjectName/FileName* (List of objects to include in the BAR file)

Workspaces ("-data"), Projects ("-p"), Applications ("-a"), and Libraries ("-l") are all represented within the filesystem as directories. In the *mqsicreatebar* command, the Workspace is specified with a complete path, while Projects, Applications, and Libraries are specified as paths relative to the Workspace. Objects ("-o") are specified as paths beginning with the Project/Application they are included in.

NOTE: Applications and Libraries were introduced in Message Broker Version 8.0.

All *mqsicreatebar* build commands must specify the "-data" and "-bar" parameters. If development was done using "Applications" as containers, then only the "-a" parameter also needs to be added (the "-p", "-l", and "-o" parameters should NOT be used). All necessary Message Broker objects and libraries will automatically be included in the BAR file. If, however, development was done using "Projects" as containers, the "-l" and "-o" flags may also be needed.

The exact command parameters used will vary depending upon the organization of the Workspace. If the Message Flows were developed using a Message Broker "Application" container, then only the "Application" ('-a') parameter must be specified. The "Application" container contains all necessary objects and references to all necessary libraries. All of the "Application" objects will automatically be included in the BAR file. In fact, when using "Applications", the "-p", "-l", and "-0" parameters may not be used.

If the Message Flows were developed using a Message Broker "Project" container, then the "Project" ('-p') parameter must be specified. The "Project" container contains the Message Flows and their related files (".esql", ".class", etc.) to be deployed. Libraries and other resources can be included using the "-l" and "-o" parameters.

Message Broker Toolkit Implications

As can be seen from both the discussion regarding the scope of the Message Flow files to be backed up and the discussion on how BAR files are created, there is a close relationship between the structures created in the Toolkit and the deployment process. Toolkit structures include:

- **Workspaces:** Since Workspaces define the scope of the software loaded into the Toolkit at Toolkit startup time and are referenced during BAR file creation, they are a very important structure. The size of the Workspace should be kept in mind. Separate Workspaces for separate Business Applications and/or projects, for example, might work nicely to meet the needs of developers, administrators, and SCC software.

- **Applications:** The use of Application containers provides a convenient way to group related resources into a BAR file. However, all resources in the "Application" will be included in the BAR file. Thus, the Message Flows and Libraries included in the Application need to be considered.

- **Projects:** The use of Project containers also provides a convenient way to group related Message Flows into a BAR file. All of the Message Flows within a Project, and their related program files (".esql", ".class") will be included in the BAR file. Libraries and Message Flows not included in the Project container will have to be specifically identified using the "-l" and "-o" parameters.

Some standards as to the use of these containers must be identified so that developers can use the Toolkit in the manner that is most consistent with the deployment process.

Linking Source Code Versions and BAR Files

When the need for a new version of the source code is identified, the necessary Toolkit files will be backed up into the SCC software or libraries. A label for this version will be identified to differentiate the files. Ideally, the BAR file should be built from this backup library. This ensures that the BAR file exactly corresponds to the backup source code and that all necessary source code modules have been included in the backup.

The mechanism for generating these "labels" will vary by SCC software product and by the enterprise change control processes in place. Regardless of the format of this label, any enterprise label used should be reflected in the BAR file name. This makes the correlation between source code and BAR file obvious.

For the remainder of this document, a simple date format will be described for use as a label. This solution will prove satisfactory in situations where neither SCC software nor enterprise change control processes generate labels.

Migrating BAR Files

When the need for a new version of the Message Flow source code is identified, the source code is backed up and labeled. A Broker Archive (BAR) file is then created to match this source code version. It is this BAR file, the runtime component of the Message Flow that will be migrated through the various environments (Development, Test, QA, Production, etc.).

While the Message Flow business logic will not change during this migration, some properties of the BAR file may need to be modified for each environment. These properties include resource names that may change for each environment, performance settings, etc. This process is identical to other programming languages, which might have an externalized ".properties" file that could change across environments.

1) For Message Broker, the **mqsiapplybaroverride** command is the mechanism through which these properties can be modified. The changes in properties can be entered either directly from the command line or contained within a file. This document describes the process of putting these "override" commands into a file. The command itself is described in the "mqsiapplybaroverride" Section. The "override" input file to this command is described in the '**Environment Tailoring:** If the Message Flow contains properties that need to be overridden for a particular environment, a "Properties" file will need to be created for each environment. See Section '".properties" File Content' for details on these files. Since these override files may contain sensitive information, they are not normally stored in a public Source Code Control system. They are important files, however, for both Backup/Recovery purposes and for "Falling Back" or "Restoring To" a different Message Flow version.

    - Process is controlled by the Message Broker Administrator. If BAR files are also being centrally stored, then these "Property" files can be stored in the same directory.
    - Property files are stored centrally and can be reused.
    - End result is a ".properties" file for each environment the Message Flow will be deployed into.

2) **Deployment:** To deploy a Message Flow, the Message Broker Administrator must copy the BAR file and the appropriate Properties file to the target Message Broker server. This

should be stored in a BAR file directory on that server. See the "Best Practices" Section for a recommended directory.

- Process is controlled by the Message Broker Administrator.
- Files (".bar" and ".properties") copied onto the Message Broker server.
- New Broker Archive file is created using the "*mqsiapplybaroverride*" command.
- Message Flow (new BAR file) is deployed using the "*mqsideploy*" command (See the "Message Broker Command Reference" Section for complete details).
- End result is a deployed Message Flow that can be redeployed whenever necessary.

Broker Archive (BAR) ".properties" files' Section.

BAR File Deployment - Options

Message Flows, and their associated BAR files, may be created and deployed to a Broker directly from the Toolkit. This is generally only considered acceptable in a Development environment. Regardless of how they are deployed, BAR files frequently require some minimal modification as they are promoted through various environments (e.g. Development to Test to QA to Production). A common requirement is to change a Database User ID and Password to the appropriate values for each environment.

One possible solution for this need to configure the BAR file is to create a new BAR file, with the proper values, for each environment. This process is unacceptable for a number of reasons. These include:

- The process violates the normal "Separation of Duties" requirement as it ultimately gives developers direct access to a production environment.
- The process does not have an audit trail.
- Multiple copies of the same Message Flow exist simultaneously making unclear which is the authoritative version.
- The process creates difficulty in ensuring the integrity of all of the environments.
- The process cannot ensure whether the source code in an environment matches the code currently executing in that environment.

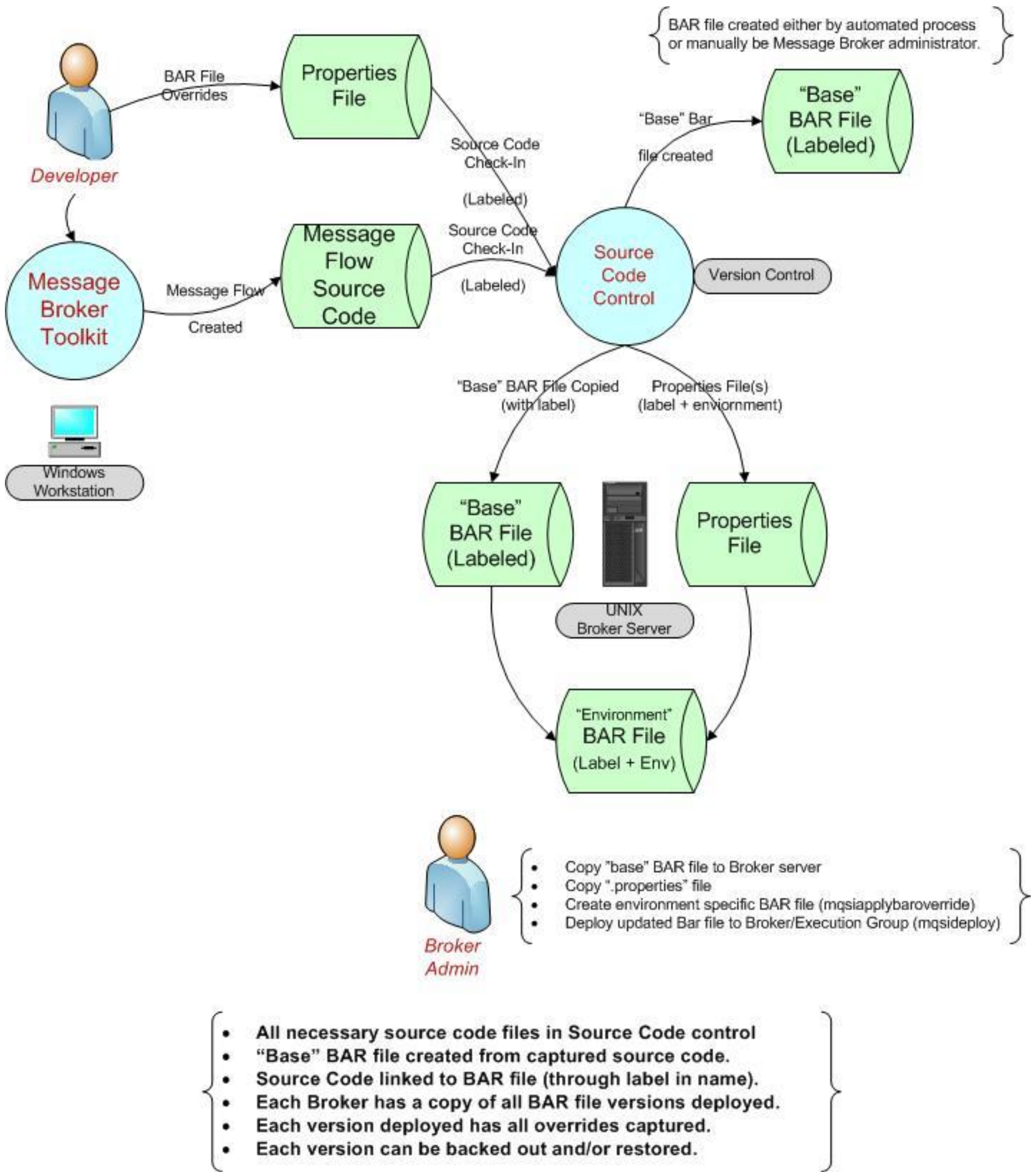
The only alternative to the above process is to institute a Message Flow deployment process for Message Broker. This process should have the following characteristics:

- Message Flows should be capable of having multiple named versions.
- For each version of a Message Flow, there should be a corresponding Source Code version.
- For each version of a Message Flow, there should be exactly one BAR file.
- The Source Code and BAR files should be maintained separately from the Message Broker Toolkit; ideally in a Source Code Control system.
- BAR files should be promoted to other environments in a manner similar to other software. This ensures that the BAR file being deployed is the same BAR file that was tested in the previous environment.
- Ideally, the promotion process should be as automated as possible.

Message Flow Deployment - Recommended Process

The recommended Message Flow deployment process is illustrated in Figure 1. This process can be summarized as follows:

3) **Message Flow Creation/Modification:** Application Developer creates/modifies a Message Flow or Flows in his Toolkit. The result of this development is a set of Message Flow and related ".properties" files.
    - Process controlled by the Application Developer.
    - Process results in a new "version" of a Message Flow(s).
    - Process results in ".properties" files to configure each environment.
4) **Source Code Control:** The source code files are "Checked Into" the Source Code Control system. This can either be an external software package such as Subversion, or it can be a centrally controlled directory managed and secured by the Message Broker administrator. In both cases, a label must be assigned for the new version. If it is an administrator managed directory, the Workspace directory name has a date included in the name. See the "Best Practices" Section for recommended naming standards and directories.
    - For external Source Code Control software, this process is controlled by the Application Developer, who must "Check In" the source code files.
    - For Message Broker Administrator controlled Source Code Control, this Process is controlled by the Message Broker Administrator. Administrator will rename the Application/Project directory with a date to implement version control.
    - End result is Source Code controlled Workspace directory version.
5) **BAR File Creation:** A BAR file is created from the source code checked in.
    - "Base" BAR file created without using any overrides.
    - BAR file has the label included in name and "BASE" for an environment.


BAR file created either by automated process or manually be Message Broker administrator.
Developer
BAR File Overrides
Properties File
Source Code Check-In (Labeled)
"Base" Bar file created
"Base" BAR File (Labeled)
Message Broker Toolkit
Message Flow Created
Message Flow Source Code
Source Code Check-In (Labeled)
Source Code Control
Version Control
Windows Workstation
"Base" BAR File Copied (with label)
Properties File(s) (label + enviornment)
"Base" BAR File (Labeled)
UNIX Broker Server
Properties File
"Environment" BAR File (Label + Env)
Broker Admin

- Copy "base" BAR file to Broker server
- Copy ".properties" file
- Create environment specific BAR file (mqsiapplybaroverride)
- Deploy updated Bar file to Broker/Execution Group (mqsideploy)

- **All necessary source code files in Source Code control**
- **"Base" BAR file created from captured source code.**
- **Source Code linked to BAR file (through label in name).**
- **Each Broker has a copy of all BAR file versions deployed.**
- **Each version deployed has all overrides captured.**
- **Each version can be backed out and/or restored.**

Figure 1 - BAR File Deployment Process

6) **Environment Tailoring:** If the Message Flow contains properties that need to be overridden for a particular environment, a "Properties" file will need to be created for each environment. See Section '".properties" File Content' for details on these files. Since these override files may contain sensitive information, they are not normally stored in a public Source Code Control system. They are important files, however, for both Backup/Recovery purposes and for "Falling Back" or "Restoring To" a different Message Flow version.

- Process is controlled by the Message Broker Administrator. If BAR files are also being centrally stored, then these “Property” files can be stored in the same directory.
- Property files are stored centrally and can be reused.
- End result is a “.properties” file for each environment the Message Flow will be deployed into.

7) **Deployment:** To deploy a Message Flow, the Message Broker Administrator must copy the BAR file and the appropriate Properties file to the target Message Broker server. This should be stored in a BAR file directory on that server. See the “Best Practices” Section for a recommended directory.

- Process is controlled by the Message Broker Administrator.
- Files (“.bar” and “.properties”) copied onto the Message Broker server.
- New Broker Archive file is created using the “*mqsiapplybaroverride*” command.
- Message Flow (new BAR file) is deployed using the “*mqsideploy*” command (See the “Message Broker Command Reference” Section for complete details).
- End result is a deployed Message Flow that can be redeployed whenever necessary.

Broker Archive (BAR) ".properties" files

Overview

BAR file "override" files may take several different formats. These formats are as follows:

- An individual Deployment Descriptor ("META-INF\broker.xml") file.
- Another BAR file that contains a Deployment Descriptor to override this BAR file.
- A properties file with "tag=value" pairs.

The remainder of this document describes the processing of a "properties" file.

".properties" File Content

The "properties" files used in the "*mqsiapplybaroverride*" command has a strict set of format requirements. The following points must be kept in mind:

- Comment lines are not allowed.
- Blank lines are not allowed.
- White space is not allowed.
- Variables cannot be used, only literals.
- A "Tag" without a corresponding value removes any existing setting for that Tag.
- Tags have a scope. This scope can be:
  - Message Flow/ Node specific.
  - Message Flow (All nodes).
  - All nodes in all Message Flows.

The basic format for each line is as follows:

- *MessageFlowName***#***NodeName***.***PropertyName***=***Value*

For each of these lines

- The Message Flow name and "#" are optional and do not contain the ".cmf" file extension. It is the name of the Message Flow that would be used in a Message Broker command. The Message Flow name is terminated by the "#" character.

- The Node Name and "." are optional unless the Message Flow name has not been supplied. If the Message Flow name has been omitted, then the Node Name may not be specified. The Node name is terminated by a period.

- The Property Name is required.

- The "=" and Value are optional. If they are not supplied, then existing values are removed.

These rules therefore allow the following formats to be used:

• *MessageFlowName***#***NodeName***.***PropertyName***=***Value*	(Specific Node replace)
• *MessageFlowName***#***PropertyName***=***Value*	(All Nodes replace)
• *PropertyName***=***Value*	(Global replace)
• *MessageFlowName***#***NodeName***.***PropertyName*	(Specific Node remove)

Warning: Any typographical errors will result in a property NOT being overridden! Instead, a new property will be added! Test these commands and verify ("*mqsireadbar*") prior to deployment!

Message Broker Command Reference

Background

Message Flows can be deployed either directly from the Message Broker Toolkit or via the command line. Message Flows are often deployed directly into the Development environment by the developers using the Toolkit. This Section describes the Message Broker commands than can be used from the command line to modify and deploy BAR files.

mqsilist

The “*mqslist”* command can be used to locate the BAR file that is currently associated with a Message Flow. When a “Detail Level” of 2 is specified with an Execution Group, the location of the BAR file associated with each Message Flow is displayed.

To display detailed information about a Message Flow, use the following command:

- **mqsilist** *BrokerName* **-e** *ExecutionGroup* **-d2**

mqsideploy

The “*mqsideploy*” command is used both to deploy a BAR file to a specific Message Broker Execution Group or to remove previously deployed files from an Execution Group. This command can be used to deploy either to a local or a remote Queue Manager. If deployments are made to a local Queue Manager & Broker, the Broker Name is specified in the command (as illustrated below). For remote deployments (especially useful in scripts), the Message Broker name can be replaced by the following three parameters:

- **-i** *IPAddress*
- **-p** *PortNumberofQueueManager*
- **-q** *QueueManagerName*

To incrementally deploy a BAR file locally, use the following command:

- **mqsideploy** *BrokerName* **-e** *ExecutionGroupName* **-a** *BarFileName*

The Message Flow files included in a BAR file can be displayed using the *mqsireadbar* command. These files can be removed later from the Execution Group using the following command:

- **mqsideploy** *BrokerName* **-e** *ExecutionGroupName* **-d** *FileName*

mqsireadbar

The “*mqsireadbar*” command is used to read a BAR file. This command displays the Message Broker toolkit files that are contained in the BAR file. The command also displays all of the Deployment Descriptors and their attributes in the BAR file. These Deployment Descriptor attributes can be modified using the “*mqsiapplybaroverride*” command

To read a Bar file, use the following command:

- **mqsireadbar** **-b** *BarFileName*

See the Section “Sample Output - mqsireadbar

```
BIP1052I: Reading Bar file using runtime mqsireadbar...
Message_Flow_Name.2015-01-01.bar:
  Message_Flow_Name.cmf (11/25/14 10:03 AM):
  Message_Flow_Name_MessageSet.xsdzip (11/25/14 10:03 AM):
  Message_Flow_Name_MessageSet.dictionary (11/25/14 10:03 AM):
  Message_Flow_Name_2_MessageSet.xsdzip (11/25/14 10:04 AM):
  Deployment descriptor:
    Message_Flow_Name#additionalInstances
    Message_Flow_Name#commitCount
    Message_Flow_Name#commitInterval
    Message_Flow_Name#coordinatedTransaction
    Message_Flow_Name#consumerPolicySet
    Message_Flow_Name#providerPolicySet
    Message_Flow_Name#consumerPolicySetBindings
    Message_Flow_Name#providerPolicySetBindings
    Message_Flow_Name#securityProfileName
    Message_Flow_Name#monitoringProfile
    Message_Flow_Name#.Message_Flow_URL....URLSpecifier = /Message_Flow_Test
    Message_Flow_Name#.Message_Flow_URL....faultFormat = HTML
    Message_Flow_Name#.Message_Flow_URL....securityProfileName
    Message_Flow_Name#.Message_Flow_URL....timeoutForClient = 360
    Message_Flow_Name#.Message_Flow_URL....useHTTPS = true
    Message_Flow_Name#.Message_Flow_URL....validateMaster
    Message_Flow_Name#OUTPUT.QUEUE.01.queueManagerName
    Message_Flow_Name#OUTPUT.QUEUE.01.queueName = OUTPUT.QUEUE.01
    Message_Flow_Name#OUTPUT.QUEUE.01.replyToQ
    Message_Flow_Name#OUTPUT.QUEUE.01.replyToQMgr
    Message_Flow_Name#OUTPUT.QUEUE.01.securityProfileName
    Message_Flow_Name#OUTPUT.QUEUE.01.validateMaster
    Message_Flow_Name#Fail.txt.filePath = /tmp/Fail.txt
    Message_Flow_Name#OUTPUT.QUEUE.FAIL.01.queueManagerName
    Message_Flow_Name#OUTPUT.QUEUE.FAIL.01.queueName = OUTPUT.QUEUE.FAIL.01
    Message_Flow_Name#OUTPUT.QUEUE.FAIL.01.replyToQ
    Message_Flow_Name#OUTPUT.QUEUE.FAIL.01.replyToQMgr
    Message_Flow_Name#OUTPUT.QUEUE.FAIL.01.securityProfileName
    Message_Flow_Name#OUTPUT.QUEUE.FAIL.01.validateMaster
    Message_Flow_Name#HTTP Reply.validateMaster
    Message_Flow_Name#MQ Header.mqdlhDestQMgrName
    Message_Flow_Name#MQ Header.mqdlhDestQName
    Message_Flow_Name#MQ Header.mqmdReplyToQ
    Message_Flow_Name#MQ Header.mqmdReplyToQMgr
    Message_Flow_Name#Transform XSD.dataSource
    Message_Flow_Name#Transform XSD.validateMaster
    Message_Flow_Name#Remove HTTP Header.dataSource
    Message_Flow_Name#Remove HTTP Header.validateMaster
    Message_Flow_Name#Setting GLogXMLElement.dataSource
    Message_Flow_Name#Setting GLogXMLElement.validateMaster
    Message_Flow_Name#XSD to MRM.dataSource
    Message_Flow_Name#XSD to MRM.validateMaster

BIP8071I: Successful command completion.
```

Sample Properties File” for a sample of the output of this command.

mqsiapplybaroverride

The “*mqsiapplybaroverride*” command is used to modify the Attributes of Deployment Descriptors within a BAR file. This may be necessary as BAR files are promoted across environments (e.g. Test to Production). These kinds of changes should be done with this command rather than creating a new BAR file in the toolkit. Using this command ensure the integrity of the code promoted across the environments.

To update the Deployment Descriptors in a Bar file, use the following command:

- **mqsiapplybaroverride -b** *InputBarFile* **-o** *OutputBarFile* **-p** *OverridesFile*

Sample Output - mqsireadbar

```
BIP1052I: Reading Bar file using runtime mqsireadbar...
Message_Flow_Name.2015-01-01.bar:
  Message_Flow_Name.cmf (11/25/14 10:03 AM):
  Message_Flow_Name_MessageSet.xsdzip (11/25/14 10:03 AM):
  Message_Flow_Name_MessageSet.dictionary (11/25/14 10:03 AM):
  Message_Flow_Name_2_MessageSet.xsdzip (11/25/14 10:04 AM):
  Deployment descriptor:
    Message_Flow_Name#additionalInstances
    Message_Flow_Name#commitCount
    Message_Flow_Name#commitInterval
    Message_Flow_Name#coordinatedTransaction
    Message_Flow_Name#consumerPolicySet
    Message_Flow_Name#providerPolicySet
    Message_Flow_Name#consumerPolicySetBindings
    Message_Flow_Name#providerPolicySetBindings
    Message_Flow_Name#securityProfileName
    Message_Flow_Name#monitoringProfile
    Message_Flow_Name#.Message_Flow_URL....URLSpecifier = /Message_Flow_Test
    Message_Flow_Name#.Message_Flow_URL....faultFormat = HTML
    Message_Flow_Name#.Message_Flow_URL....securityProfileName
    Message_Flow_Name#.Message_Flow_URL....timeoutForClient = 360
    Message_Flow_Name#.Message_Flow_URL....useHTTPS = true
    Message_Flow_Name#.Message_Flow_URL....validateMaster
    Message_Flow_Name#OUTPUT.QUEUE.01.queueManagerName
    Message_Flow_Name#OUTPUT.QUEUE.01.queueName = OUTPUT.QUEUE.01
    Message_Flow_Name#OUTPUT.QUEUE.01.replyToQ
    Message_Flow_Name#OUTPUT.QUEUE.01.replyToQMgr
    Message_Flow_Name#OUTPUT.QUEUE.01.securityProfileName
    Message_Flow_Name#OUTPUT.QUEUE.01.validateMaster
    Message_Flow_Name#Fail.txt.filePath = /tmp/Fail.txt
    Message_Flow_Name#OUTPUT.QUEUE.FAIL.01.queueManagerName
    Message_Flow_Name#OUTPUT.QUEUE.FAIL.01.queueName = OUTPUT.QUEUE.FAIL.01
    Message_Flow_Name#OUTPUT.QUEUE.FAIL.01.replyToQ
    Message_Flow_Name#OUTPUT.QUEUE.FAIL.01.replyToQMgr
    Message_Flow_Name#OUTPUT.QUEUE.FAIL.01.securityProfileName
    Message_Flow_Name#OUTPUT.QUEUE.FAIL.01.validateMaster
    Message_Flow_Name#HTTP Reply.validateMaster
    Message_Flow_Name#MQ Header.mqdlhDestQMgrName
    Message_Flow_Name#MQ Header.mqdlhDestQName
    Message_Flow_Name#MQ Header.mqmdReplyToQ
    Message_Flow_Name#MQ Header.mqmdReplyToQMgr
    Message_Flow_Name#Transform XSD.dataSource
    Message_Flow_Name#Transform XSD.validateMaster
    Message_Flow_Name#Remove HTTP Header.dataSource
    Message_Flow_Name#Remove HTTP Header.validateMaster
    Message_Flow_Name#Setting GLogXMLElement.dataSource
    Message_Flow_Name#Setting GLogXMLElement.validateMaster
    Message_Flow_Name#XSD to MRM.dataSource
    Message_Flow_Name#XSD to MRM.validateMaster

BIP8071I: Successful command completion.
```

Sample Properties File

The following ".properties" file commands make two changes. These changes are made to a Message Flow named "Message_Flow_Name". The two changes are:

- Node: "Fail.txt";
  - Set the "filePath" attribute to "/var/mqsi/apps/Message_Flow_Name/Fail.txt".
- Node: "OUTPUT.QUEUE.FAIL.01";
  - Set the "queueName" attribute to "newQueueName".

Message_Flow_Name.2015-01-01.properties file:

```
Message_Flow_Name#Fail.txt.filePath=/var/mqsi/apps/Message_Flow_Name/Fail.txt
Message_Flow_Name#OUTPUT.QUEUE.FAIL.01.queueName=newQueueName
```

Best Practices

- **Source Code Control:** Establish a formal mechanism/procedure for controlling both Message Broker Source Code, ".properties", and Archive (BAR) files as you would with any other in-house developed software. Use existing Enterprise Source Code Control software (Git, Subversion (SVN), Concurrent Versions System (CVS), ClearCase, etc.) where available. Source Code should be checked into the Source Code control process by the developer. BAR files should be built by the Message Broker administrator (ensuring the relationship between the two).

  Lacking Source Code Control software, establish a secure directory controlled by the Message Broker administrator to check-in Source Code, ".properties" files and to build BAR files into. Since versioning will not be externally provided, insert the date (.yyyy-mm-dd) just before the file suffix (e.g. ".msgflow", ".bar") to each Application/Project or ".properties" imported and to each BAR file that is built. This will maintain a version history of the files and will also allow roll-back to a known version. If BAR file overrides will be generated through the "*mqsiapplybaroverride*" command, insert an environment identifier (e.g. ".Prod") following the date to identify the specific configuration of this BAR file. This Environment Identifier should be the same tag that is included in the corresponding ".properties" file.

  This will allow the different versions of a BAR file in each environment to be easily compared through the following steps:

  - "*mqsireadbar*" against each BAR file, saving the output of each command in a file.
  - A comparison of these two output files (i.e. "diff" file1 file2) will display the differences.

- **Separate Build Environment:** There should be a separate build environment for the building and deployment of BAR files. In the interests of productivity, this process can begin in the environment after the Development environment unless the requisite builds require coordination across multiple developers. Responsibility for the "check-in" of the proper files lies with the Message Broker developers. Responsibility for executing the builds lies with the Message Broker administrators.

- **BAR File Promotion:** Message Flows files should be developed/modified once, ideally in the Development Environment, and then promoted into the succeeding environments (Test, Production, etc.) through a normal life-cycle. This ensures the integrity of the Message Flow code. Custom BAR Files should not be needed for each environment! Note that this means that some BAR file properties (e.g. Database names, User IDs & Passwords, etc.) may need to be changed for each environment.

- **BAR File Parameters:** BAR files should be created with appropriate attributes so that only a few attributes should need to be overridden in each environment. Any values that are common across most environments should be placed in the BAR file when it is created. This will minimize the amount of overrides needed. Likewise, values that are modified in each environment should not be set in the base BAR file. This will allow the necessary ".properties" file to be created and tested.

- **".properties" File Naming Standards:** Properties files should be named as follows:

MessageFlowName.*yyyy-mm-dd*.*EnvironmentIdentifier*.**properties**

○ *MessageFlowName*	= Message Flow name (From Toolkit)
○ *yyyy-mm-dd*	= Source Code check-in date.
○ *EnvironmentIdentifier*	= Identifies Bar File Overrides used (e.g. ".Prod")

This will allow the overrides used to both be reused in any subsequent deployments (e.g. *mqsideploy*) and to be easily identified with the underlying base BAR file.

- **BAR File Naming Standards:** Broker Archive (BAR) related files should use the following naming standards. BAR files should be named as follows:

MessageFlowName.*yyyy-mm-dd*.*EnvironmentIdentifier*.**bar**

○ *MessageFlowName*	= Message Flow name (From Toolkit)
○ *yyyy-mm-dd*	= Original BAR file creation date.
○ *EnvironmentIdentifier*	= Identifies Bar File Overrides used (e.g. ".Prod")

- **Message Broker Directories:** The Broker Archive (BAR) files and their related property files should be stored on the server to which they are deployed. This eases administration, provides an audit trail of what was deployed, and simplifies fallback processing. It is recommended to create the following directory to store the ".bar" and ".properties" files:

  - /var/mqsi/BAR

References

- IBM – WebSphere Message Broker – Knowledge center (v7.0, 8.0)
  http://www-01.ibm.com/support/knowledgecenter/#!/SSKM8N/mapfiles/product_welcome.html

- IBM – IBM Integration Bus – Knowledge center (v9.0, 10.0)
  http://www-01.ibm.com/support/knowledgecenter/#!/SSMKHH/mapfiles/product_welcome.html

- IBM – WebSphere Message Broker (V 7.0) – Configurable Properties of a Broker Archive
  http://www-01.ibm.com/support/knowledgecenter/?lang=en#!/SSKM8N_7.0.0/com.ibm.etools.mft.doc/af14620_.htm

- IBM – WebSphere Message Broker (V 7.0) – *mqsireadbar* command
  http://www-01.ibm.com/support/knowledgecenter/?lang=en#!/SSKM8N_7.0.0/com.ibm.etools.mft.doc/an26160_.htm

- IBM – WebSphere Message Broker (V 7.0) – *mqsiapplybarorverride* command
  http://www-01.ibm.com/support/knowledgecenter/?lang=en#!/SSKM8N_7.0.0/com.ibm.etools.mft.doc/an19545_.htm

- IBM – WebSphere Message Broker (V 7.0) – *mqsideploy* command
  http://www-01.ibm.com/support/knowledgecenter/?lang=en#!/SSKM8N_7.0.0/com.ibm.etools.mft.doc/ap08682_.htm

- IBM – developerWorks – Automating Message Broker Build and Deployment using Ant
  http://www.ibm.com/developerworks/websphere/library/techarticles/1302_chatterjee/1302_chatterjee.html