WebSphere MQ Workflow

Best Practices Guide

Version 1.0

July 22, 2003

Marc Fasbinder

Senior I/T Specialist

IBM Technical Sales Support

mfasbind@us.ibm.com

Contents

About this book

IBM WebSphere MQ Workflow (referred to as MQ Workflow in this book) is a powerful tool for business integration. However, MQ Workflow is a complex product that interacts with a variety of IBM and third party software, and has literally hundreds of configuration options. The published manuals often do not contain all the information you need to get the product up and running.

For these reasons, and at the request of several customers, this Best Practices Guide has been put together. It covers a variety of topics, ranging from planning and installation to programming and usage. Some topics cannot be covered in a book like this because they vary between customer implementations. For example, your company policies might require you to use naming standards other than the ones recommended here.

It is important to remember that this book does not replace the product manuals. This book is meant to enhance the documentation set.

This is the first edition of this guide. Over time, further best practices will emerge and it is intended that these will be incorporated into updates of this guide. Feel free to send suggestions for items that you'd like to see included in later editions.

Who should read this book

This book is intended for technical users of MQ Workflow as well as internal IBM resources. Developers, process modelers, administrators, and I/T support staff will also find this information helpful.

Acknowledgments

Thanks for ideas and input to Dave Paskach, Larry Stetler, Ron Daniels, Michael Snare, Larry Williamson, Elke Painke, Werner Fuehrich, Ed Johnson, Rick Marcuson, Erich Fussi, and Richard Metzger. Special thanks to Wolfgang Kulhanek for his help and input on the document.

1 Planning

Planning is very important when you are considering business process automation. The first consideration is the choice of tool. MQ Workflow? WebSphere Interchange Server? Object Oriented programming? Lotus Workflow?

The bottom line is to make sure that you use the right tool for the job. When you have decided on your tool, you can begin to address other issues, such as system sizing and topology.

Refer to *SupportPac WD02 - "MQ Workflow - Considerations for production roll out"* for information on different architectures, then read *SupportPac WP01 - "MQ Workflow - Performance estimates for solution and capacity assessments*" for guidance on system sizing.

1.1 Is MQ Workflow the right tool for the job?

Before you even start a project, you must decide whether MQ Workflow is the right tool for the job. As with any software product, there are certain functions and capabilities that come with the product, and others which can be added through customization.

Based on the features and functions available in MQ Workflow, you can decide if a given project is a good candidate for the tool. The following list contains some of these decision criteria.

- **Long running workflows**: MQ Workflow is a tool that is best used to implement long running workflows, where the state is maintained. If, for example, the server box on which the system is running were to go down, the workflow system would be able to resume work where it left off, because data is made persistent in the DB2 or Oracle database. However, if the flows needed are short lived, MQ Workflow might not be the best tool. For example, if data is received, one or two manipulations are done on the data, and then the data is sent off, MQ Workflow would not be a good fit.

- **Workflows involving people and automated steps**: MQ Workflow supports both automated workflow steps, and ones involving users. A workflow can contain both. If the workflow envisioned is completely person-to-person, MQ Workflow is still a good fit, because of the flexible work assignment functionality. In addition, over time, you can add automated steps to improve user efficiency.

  If, however, you envision a workflow that is completely automated with no human intervention, MQ Workflow is also a good candidate for the system. It provides features that are often needed in process automation:

- If there is a problem with a workflow, a person can be designated for exception handling.
- MQ Workflow maintains the state of the long running business process, and allows for monitoring and reporting.

- **Repeatable business processes**: Some business processes are repeatable; they run the same way every time, with the exception of handling for errors and branching based on business data. However, sometimes a 'process' does not follow predetermined rules. Such a process is much more difficult to model, and might not be a good candidate for MQ Workflow. However, do not overlook capabilities, such as dynamic subprocess invocations, when considering if the workflow engine is a good fit for the business process.

  Sometimes the requirements state that users need to be able to jump to any step in the workflow at any time, bypassing business rules. This can be accomplished by a series of bypass flags, allowing users to go to the desired step. However, a need to jump around in a process and to bypass rules often points to a process that is not well understood, and is not ready for workflow of any kind. It is recommended that you consider reengineering such a business process before attempting to automate it.

- **Platforms**: It is important to select a workflow engine that runs on your platform of choice. MQ Workflow supports Windows NT, Windows 2000, AIX, Solaris, HP/UX and z/OS (OS/390). It is even more important that the solution can take advantage of features of the platform. For example, on z/OS MQ Workflow takes advantage of the Work Load Manager (WLM), Automatic Restart Manager (ARM), and parallel sysplex architecture.

- **Scalability**: It is important to select a solution that is scalable both horizontally (by moving to a bigger box) and vertically (by having multiple smaller nodes in a cluster or system group). Because MQ Workflow is designed for scalability, it is an excellent candidate. If a workflow engine cannot scale up to the size needed, a project will not succeed. A workflow engine should also be able to scale beyond current needs, because over time, more workflow processes and users might be added, causing a need for a larger system.

- **Topology**: When you select a workflow system, it is critical to make sure that the system can work in configurations ranging from "everything on a single box", up to n-tier architectures. If, for example, a given solution cannot support a 3-tier topology, you would be constrained by what you can do on a single server. MQ Workflow is very flexible in terms of topology.

- **Thick or thin client**: MQ Workflow has both a thick and a thin client. Most customers use a thin client implementation. The thin client is also available as a portlet in WebSphere Portal Server. This was released as *SupportPac WA85: MQ Workflow Portal Client*

- **Worklists:** The typical interaction between a user and the workflow system is a "worklist", which shows the work items a user can act on. For a given work item, one or more users can take ownership and perform the work. The worklist can be rendered in the thick client, thin client, or through WebSphere Portal Server using a portlet. If user interaction with the system requires a metaphor other than worklists, it could mean that MQ Workflow is not the best fit for the job. However, be sure to check whether the extensions to the Web client (*SupportPac WA84,* for example) or a custom client can be used to add the required functionality.

  Some workflows have all straight through processing (STP) with no human interaction. If you have an STP process, it does not mean that you can't use MQ Workflow. See 'Functionality' for more information.

- **Functionality:** You should consider functionality when deciding which tool to use. For example, MQ Workflow supports parallelism within a workflow with splits and joins. Some process engines do not have this feature. Some process engines are targeted for short running business processes, or short to medium-length processes. This should be taken into account when selecting a workflow tool. For example, if a process is waiting on a response from a service, is it persistent? Would it have to restart if there is a problem, or if an administrator takes the system down for a database backup?

- **Workflow engine drives tasks:** In a workflow metaphor, the workflow engine drives the process, deciding what step needs to be done after a given step has completed. Events can also be incorporated using *SupportPac WA06,* for example. However, there are some systems where instead of applying business rules, determining what needs to be done, and bringing work to users, the design requires almost the opposite: users perform work, and determine the rules on the fly. Such a 'process' is not a good candidate for workflow, regardless of the product selected. It would require more of a 'work support' system than a 'workflow system'.

- **Monitoring.** A workflow system needs monitoring to gain the greatest benefit from process automation. A monitor can look at real-time data to show the status of the workflow, problems in the workflow, jobs that are taking longer than desired, and find other potential operational problems. Monitors can also build reports based on historic data. Either off-the-shelf monitors can be used, or the workflow engine's APIs can be used to create custom monitors. MQ Workflow has both the WebSphere Business Integration Monitor as an off-the-shelf solution, and the audit trail and API interface for custom monitoring solutions.

1.2 Users and system sizing

The number of users who will eventually use the system must be taken into consideration when planning your installation. MQ Workflow does not have an absolute limit on the number of users. Most customers use the thin Web client, which does not impose any limits on users, other than the load placed on the system. *See SupportPac WP01* for details on how this load impacts performance.

If you use the thick client, there is a limit to the number of connections a queue manager can handle. This limit depends on the platform you are using. The limits are around 100 users for Windows NT, and around 1000 users for UNIX systems. If you need more connections, you must configure a client concentrator as shown in Figure 1.

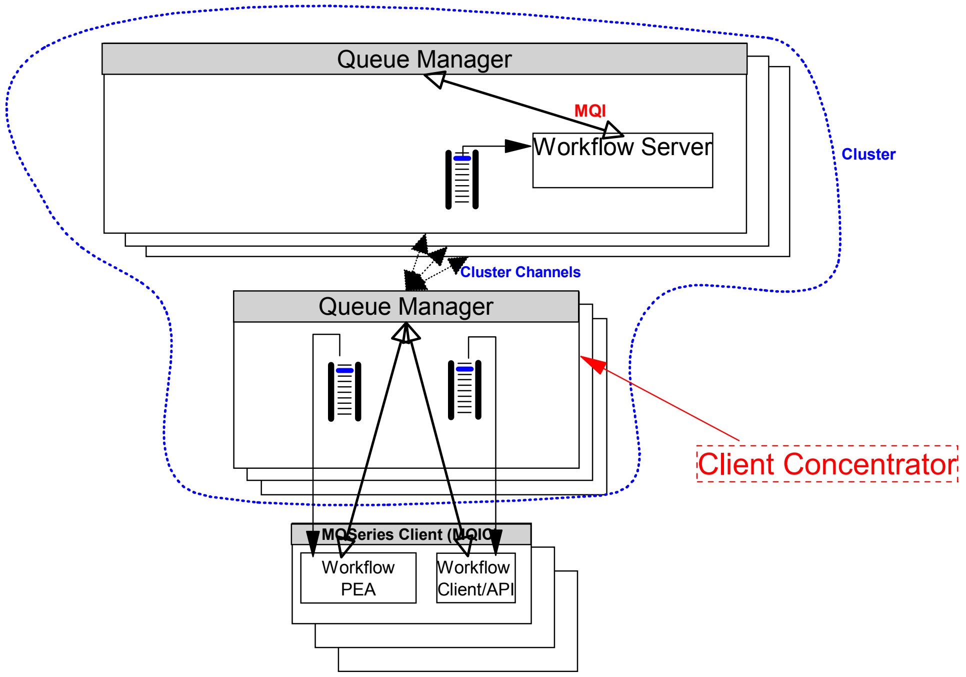

Figure 1: Workflow Client Concentrator

The Client Concentrator can connect to a large number of clients; the workflow queue manager now only needs to connect to the client concentrator.

As for the server which runs the workflow system, for Silvernodes (and comparable machines) 600 users are possible without problems. The maximum number of users recorded is 4000. On Windows NT boxes, 100 users are OK, 150 are still possible, and 200 are probably the limit (200 users would mean about 800 amqcrsta processes in the system—Windows NT is much less capable than UNIX of handling a large number of operating system processes).

For fully automated processes, the number of users is irrelevant. Only the number of steps in the process, and the frequency at which the process runs are factors. For most systems, the business process will be a mix of steps involving people, and fully automated steps.

1.3 Architecture

There are many architecture options for an MQ Workflow system. This section covers some of the major considerations. Refer to *SupportPac WD02 - "MQ Workflow - considerations for production roll out"* for more information on different architectures.

1.3.1 2-tier vs. 3-tier installations

If the MQ Workflow servers run on the same node as the workflow database, this is known as a 2-tier installation (clients are tier 1, servers are tier 2). You can move the database to a separate physical box. This is called a 3-tier installation (client/server/database). In a 3-tier installation, the database server must be on the same platform as the workflow server.

Tests have indicated that for an MQ Workflow Basic Work Unit (or BWU, as defined in *SupportPac WP01*), 60% of the load is for the server, and 40% for the database. As the overall system load grows, you might consider moving the database work to another node to allow for greater throughput. However, if you move the database, you pay a performance penalty of about 10%. When you move to a 3-tier installation, you might experience worse performance at first. However, a 3-tier installation allows you to increase the load on the MQ Workflow server, without having to upgrade your hardware. In addition, it allows you to later increase the size of the system group. See section 1.3.2 for a definition of a system group.

If you think you might need to move to a system group in the future, it is a best practice to configure your installation as a 3-tier system at the initial installation, because it is difficult to move your Runtime database to another server later on.

1.3.2 System groups

If you use a 3-tier installation, multiple servers can share a single database server. Incoming requests are spread across the systems, allowing for greater throughput. A system group allows you to run high volumes of workflow transactions, and to spread the requests across multiple machines. However, a system group (or an MQSeries cluster) does **not** give you failover. Consider the following scenario.

A message comes in to the Client Concentrator Queue Manager, CCQM. CCQM passes the message to the queue manager on one of the servers in the system group called FMCQM. The message is now on the queue on FMCQM1. Assume that the moment after the message is safely queued, the hardware server where FMCQM1 is running goes down. Where does the request stand? When the node for FMCQM1 comes back up, it will process the message, because it was persistent. However, while the node is down, the message is "marooned". The other servers in the system group are still up and processing requests, but they can't process the marooned message on FMCQM1. If there were failover, the message would not have to wait to be processed. If you need high availability, it is a best practice to use hardware failover, such as HACMP in an AIX environment.

In a system group, if one physical server goes down, others can continue. However, if the database server goes down, none of the MQ Workflow servers can operate. Therefore, it is a best practice to use hardware redundancy on the database server. This eliminates the possible single point of failure.

If high availability is needed, the fee-based *SupportPac WC01 - "Integration of MQ Workflow in High Availability Environments"* is available. This offering provides assistance in configuring and installing your environment for AIX HACMP and Sun Clusters.

Currently the database server for a 3-tier installation must be of the same platform as the workflow server. This means that all servers in the system group must be on the same platform. A Windows NT server and an AIX server cannot both be in the same system group.

1.3.3 Recommendations

To determine which type of installation, you need to consider the factors covered in *SupportPac WD02*, such as high availability. The information derived from *SupportPac WP01* will also help you decide if you need to move to a 3-tier installation for performance reasons, or if you need a system group cluster.

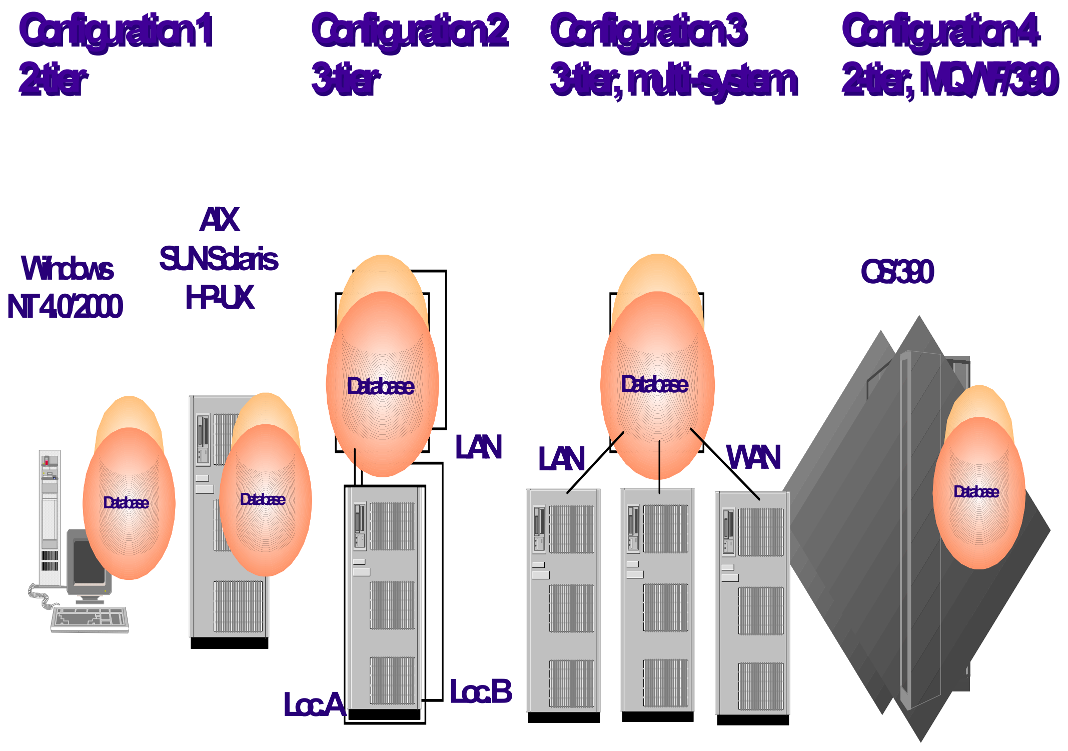

Figure 2: Configuration options

In Figure 2, configuration 1 is a 2-tier Windows installation. The MQ Workflow servers and the database all run on the same machine. Configuration 2 is a 3-tier installation, with the database on a separate node on the network. Configuration 3 expands to a system group with three MQ Workflow nodes sharing a single database node. Configuration 4 is a 2-tier installation running on OS/390 (z/OS).

If you are using a 3-tier installation, it is a best practice to make sure that the clocks on the servers are synchronized as closely as possible. If the clocks are more than a few seconds out of sync, there is a risk of an out-of-sync timestamp being misinterpreted.

1.3.3.1 Development, test, production

For development systems, most customers select configuration 1, because it can run on a laptop or desktop machine by itself. For the first level of testing, it can be advantageous if the developers can use their own machines. When the workflow has passed basic testing, it can be moved on to the test server. You could use configuration 2 for development, but that would require a connection to the server. This means that the connection must be available for testing to take place.

After basic debugging has been completed, testing needs to take place, such as unit test, string test, and stress test. For this, the test server should match the final production environment. If an AIX server is used for production, then the test server should be AIX as well. Because it will require less processing power, the test server can be a small, low-end system.

1.3.3.2 Scalability

Often, when MQ Workflow is first installed, only one or two processes are implemented. Over time, more processes and users are added to the system, so scaling is important. If you know that the system will grow considerably in the near future, it makes sense to start with a 3-tier installation, so that growth into a system group is easier as demand increases.

Also, keep in mind the scalability of the platform itself. If you install on a UNIX server, you can add more CPUs or memory later. Z/OS also can scale by adding hardware or Sysplex nodes.

1.3.3.3 Operating system

Because production workflows tend to be implemented for mission-critical business processes, make sure that the selected platform is well supported internally. The operating system must also be capable of managing an application, such as MQ Workflow.

MQ Workflow functions are the same across all platforms, with the exception of z/OS. If MQ Workflow runs on z/OS, it can take advantage of the Automatic Restart Manager (ARM) if the application is stopped for any reason. The Workload Manager (WLM) can be used to start and stop server instances on the fly. This helps to keep performance consistent as the load on the system changes during the day. There is also a security exit for RACF, and a component called a Program Execution Server (PES), which allows the safe invocation of CICS and IMS transactions through the CPIC and EXCI interfaces.

CICS and IMS transactions can be executed from the distributed platforms, via the MQSeries Bridge. However, if a majority of the workflow is going to be CICS transactions, then z/OS might be a better choice of platform.

1.3.3.4 Centralized or distributed

Some customers with multiple sites prefer servers at every location, while others prefer large central servers. Follow your current standards in making the decision. This should take into account the effort of maintenance of multiple remote servers compared to a large central server. However, if there are locations across the world (say in Zurich, New York, Sao Paulo, Singapore and London), this might play a role in the architecture decision.

1.3.3.5 High availability

If high availability is required, consider using an HA cluster, such as HACMP for AIX, or the highly reliable z/OS platform.

1.3.3.6 Cost

Cost is always a factor in the choice of architecture. Ideally, everyone would have a cluster of the very largest server on their platform of choice, fully loaded with the maximum number of CPUs and RAM. However, in the real world, you must live with a budget. Larger systems have higher costs for both hardware and software. Often, your budget will determine the final target platform. However, make sure not to compromise on critical aspects, such as high availability.

1.3.4 MQ Workflow and WebSphere Application Server

If you are using the thin client, you will need a Web application server, such as WebSphere. The question arises as to whether you should put WebSphere Application Server on the same box as MQ Workflow

The biggest limitation of having all the software on one box is that as the system grows, the box will reach its performance limitations sooner. Separate boxes spread the workload, and allow the servers to last longer before they need to be upgraded. Having all the software on one system is easier to configure, but you also have a single point of failure. For very large systems, such as z/OS installations, or large Regatta-class servers, this will be less of a problem. For Windows systems or small UNIX boxes, it will be of greater concern. Make sure to plan accordingly.

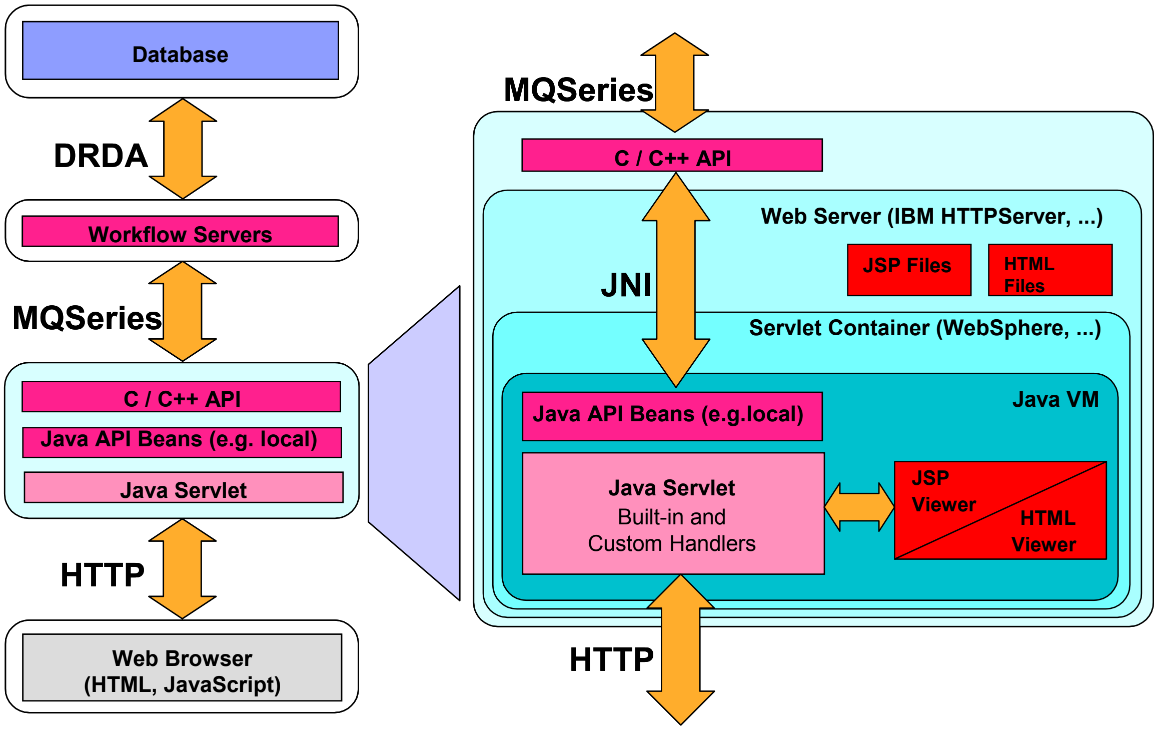

Figure 3: Web client architecture

1.3.5 MQSeries Queue Manager and WebSphere Application Server

When WebSphere Application Server is on a different node than MQ Workflow, a connection must exist between the WebSphere Application Server node and the MQ Workflow node. An MQ Server connection must be used.

1.3.6 Architecture for development and deployment

It is also important to think about the architecture for the development and test environment. For this example, assume that the production system is AIX.

If developers have stand-alone systems with all the software installed, they can build and test, even when they are offline. If they have only Buildtime or WBI Workbench (formerly called Holosofx), they need an active connection to the AIX server to be able to test.

It is also a best practice to use DB2 for the Buildtime database. While MS Jet Engine does have a smaller memory footprint than DB2, it is not nearly as stable, does not have the utilities, and is not an “industrial strength” solution. DB2 comes free with MQ Workflow, so cost is not an issue. If you use DB2, it is easier to extend if a remote database is to be used.

After the developers have built the workflows and perhaps tested locally on Windows, the next logical step is to test them in an environment similar to the production system. If production is on Windows NT or Windows 2000, then the development box could potentially be used. However, it is a best practice to use a dedicated server, because you can then do things, such as stress tests against the server. For these tests to be effective, the platform must closely match the production environment. To root out errors before deployment, ensure that the test box has the same software levels as the production system. Because the production system is AIX in this example, a small AIX box should be used for test. Another advantage to this approach is that you can upgrade the software on the test box, and run through a test suite before attempting to do so on the production box.

After completing all test phases, the FDL can go to the production server. You could even implement a workflow model in MQ Workflow to follow through this process to get the appropriate signoffs, promote the FDL, and so on.

Another issue is the methodology and system setup for multiple developers. MQ Workflow does not include versioning. Many customers want to manage workflows with versioning, just as they do their other applications. However, because you are creating Flow Definition Language (FDL), you can use your current tool set to manage the FDL files, just as you might manage a Java file.

MQ Workflow Buildtime does not have a checkin/checkout function, although the WBI Workbench does if you use the WBI Workbench Server. If you have a central database with MQ Workflow and developer A checks out a workflow, the record is locked. So, if developer B tries to check it out, they will get a message that the model is being edited in read only mode.

Another common methodology is to use the checkin/checkout facilities of the configuration or library management system you already have. Before you change the FDL for a process, check it out from the central code repository using the standard procedures already in use by the developers. Then after completion, check it back in. In this case, each developer can have a unique database instead of a shared database. You can also manage the programs that go into creating the complete solution. If you use this approach, then you should also consider the following:

- Keep all of the topology settings in one FDL. This can be imported to each machine.
- Keep all the people in a separate FDL.
- Keep each process in a separate FDL.

1.3.7 Naming standards for UPES queues

When deciding on the names for the MQSeries queues for a UPES implementation, keep the following in mind:

- Because many customers already have MQSeries installed, there might be a predefined set of naming standards. Talk with the MQSeries system administrator first to find out if any standards are in place.

- MQSeries is case sensitive. For example, if you define a UPES in MQ Workflow that points to a queue named MyUPESQueue, but you define the queue in MQSeries as MYUPESQUEUE, you will get error messages telling you that the queue does not exist. Make sure to be consistent on capitalization.
- UPES definitions will change from the development environment, through test, to production.
- MQ Workflow does not create the queue when you import the definition of your UPES into Runtime. You have to go out through the standard MQSeries interface to do this. Many customers have an administrative process to follow to request a new queue, which can sometimes take weeks. Plan the creation of the queue well in advance!

1.3.8 Architecture and naming standards

When you create the different constructs for MQ Workflow, such as system groups and systems, you should set standards and keep to them. There are several different approaches to this, and you should consider not just the current installation, but also how things might look in the future.

One common way of setting up the system is to have the MQ Workflow system group name exactly match the MQSeries cluster name. Even if there is only one system at first, an MQSeries cluster is still used, which makes it easier if the system ever expands to multiple boxes. The system is often named the same as the host name of the box it runs on, since for most installations there is one system per physical server box. However, you can have multiple systems running on one physical server box, so this method might not meet your needs. Some customers also name the configuration after the box where MQ Workflow will run.

However, what happens if you move the system to a different physical server? If you've tied your names to the current box, you cannot just migrate the current system to the new box; you will have to re-create everything in a new configuration with new names. Therefore, you might want to use names that are more neutral and not tie them to the server name.

You should also consider defining a consistent naming standard for the development, test, and production environments. It is also a best practice to use unique names, rather than the defaults: Group FMCGRP, System FMCSYS, Queue Manager FMCQM, Runtime database FMCDB, and Buildtime database FMCBTDB. If you use unique names, you reduce the chance of conflicts due to duplicate names.

2 Installation and Configuration

This chapter introduces concepts relevant to installing and configuring the workflow system. You should read *IBM WebSphere MQ Workflow Installation Guide,* SH12-6288. The classes MW150 – *MQ Workflow Basic Tailoring* and MW200 – *MQ Workflow Web client and API* offer further information and training on this topic.

The first and most important best practice for installation and configuration is to PLAN AHEAD. This may sound obvious, but consider the following typical scenario:

- The customer wants a two-tier installation, with a Web client. MQ Workflow will be installed on the AIX box named PROD15. The WebSphere Application Server instance to be used will be on PROD22.
- The customer has a DB2 support group, which has standards for DB2 database names and who can access them. They have an MQSeries admin group with standards for Queue Manager names.
- The developers want to run Buildtime on Windows 2000 with a shared database.
- They want to use a Queue Manager on the WebSphere Application Server box, rather than using RMI over TCP/IP.
- The UNIX administrators have tight standards on creating users and groups. A request needs to be sent one week in advance.

Some configuration steps on AIX will require root authority. Some steps will require DB2 administrator authority. Only certain users at the customer sites are authorized for these functions.

For this scenario, the installation is going to involve the following groups:

- UNIX administration
- MQSeries administration
- DB2 administration
- WebSphere Application Server administration
- Development/workstation administration

All of these groups need to be involved for the installation to succeed. If the UNIX group is not contacted ahead of time, you cannot create the IDs and groups needed for the various products. Additionally, if you don’t work with the DB2 administration group, you might create a database with a name that deviates from the internal standards, and will have to be changed later. It is critical to coordinate ahead of time.

A second best practice is to read the *MQ Workflow Installation Guide* before attempting the installation. This is very long, but you can just keep to the relevant sections. For example, if you are installing on AIX, you can ignore all the sections on Windows.

You should also read the *MQ Workflow Administration Guide,* SH12-6289, and *SupportPac WD02 MQ Workflow - Consideration for Product Rollout* before attempting to install. The lab also has a Systems Capacity Assessment (SCA), which helps you plan and prepare your installation.

Also, given that you are installing multiple products, make sure you install at least the **minimum** maintenance levels of MQSeries, DB2, and Workflow, and test each component with their IVP before progressing. Check the Web site for the latest information on minimum levels, download and apply the patches as necessary.

When configuring MQ Workflow, there are many options. This section discusses some of these options and their ramifications, and outlines some little-known options that are buried in the *MQ Workflow Installation Guide*.

2.1 Overview of the installation

An MQ Workflow installation is a complex process. This means that planning is critical to success. To gain an understanding of what needs to be accomplished, here are the major steps:

1. Install MQSeries (including creation of a group and user). A queue manager is created and configured as well.

2. Install DB2 (including creation of a group and user. A database will be created and configured as well). Oracle can be used instead of DB2, but DB2 comes with MQ Workflow at no additional cost. If DB2 is located on a different machine (3-tier installation), then this is a more complex step. In this case, DB2 is installed on the second box; DB2 Connect is installed on the MQ Workflow box. Then DB2 Connect must be configured to work with the remote database server.

3. Install MQ Workflow (including creation of a group and user). After the MQ Workflow code is on the server, it is a best practice to install the latest maintenance level (fix pack) before you do the next step.

4. Configure MQ Workflow. The configuration includes setting up general system settings, and setting up the queue manager and the database.

5. Install MQSeries and MQ Workflow on the WebSphere Application Server box (including creation of groups and users).

6. Configure MQ Workflow on the WebSphere Application Server box. This includes configuring and deploying the MQ Workflow servlet on the WebSphere Application Server through its XML interface. Some applications such as WebSphere Portal Server might turn on security to disable this interface. The WebSphere Application Server administrator can grant your ID the proper rights or turn security off temporarily for this step to work. In addition, the channels and connections between the workflow server and

WebSphere Application Server need to be configured. If you are using RMI over IIOP, the Java Agent needs to be configured as well.

2.2 Web client configuration

You cannot use the fat client configuration for the Web client. You must have a separate configuration for the Web client, although you can reuse some of the fat-client settings, such as the queue manager. What can be confusing, in version 3.3.2 on the Windows platforms, where there is a GUI tool for the standard configuration, but the command line tool, `fmczutil` had to be used to create the configuration for the Web client. In version 3.4, there is also a GUI for the Web client configuration.

MQ Workflow supports multiple configurations running on the same physical box. If you have enough resources, you can even run two separate MQ Workflow systems on the same computer.

Web servers are rapidly evolving. When you configure MQ Workflow, you can select which "skeleton file" to use for different versions of WebSphere Application Server (3.5, 3.5.2, 3.5.4, and 4.0). If you upgrade your WebSphere Application Server box from version 3.5.4 to 4.0, you would then want to update MQ Workflow to take advantage of the latest WebSphere Application Server 4.0 features. You wouldn't want to have to delete your entire configuration. Therefore, if you have a separate configuration for just the Web client, you can then delete the old Web client configuration, and create a new one using the new version of the schedule. Now WebSphere Application Server will be reconfigured for MQ Workflow, using the right setup.

It is a best practice to select a different name for the second configuration, so that you can tell at a glance, which is which. For example, if your main configuration is FMC1, your Web client configuration might be FMC1WEB.

See the Web client online documentation for more information. For z/OS installations, see the documentation included with the Web client, located in the file: `{MQ_Workflow_directory}\DOC\WebClient\installation.html`.

2.3 Web client and JDK

When you configure MQ Workflow and the Web client, ensure that you always refer to the same JDK in all parts of the installation. It is a best practice to use the JDK in WebSphere to ensure that applications are running on the same levels. If, for example, you have the JDK in WebSphere and a Sun JDK installed on the same box, and you point different parts of the configuration to different JDKs, you can get errors.

If you are using JNDI for your configuration, the Java Agent needs the WebSphere Application Server JDK, because a required class is not supplied in the Sun JDK.

2.4 Web client and HTTPS

In some installations, MQ Workflow might be required to work with HTTPS for higher levels of security. The Web client comes with a file: `{MQ_Workflow_dir}\DOC\WebClient\unsuppd.html`. This "unsupported" file shows how you can make MQ Workflow work with HTTPS, and other configurations that use the IBM VisualAge for Java 3.5 WebSphere Test Environment with the Web client, instead of needing a full WebSphere Application Server. The Web application servers listed in this file are officially supported, but this file is provided on an as-is basis.

Also, the readme file for Service Pack 1 for version 3.4 mentions:

```
If you want to use https in WebSphere 4.0.2, set the system properties
under jvm settings.  Set the property 'java.protocol.handler.pkgs' to
the value 'com.ibm.net.ssl.internal.www.protocol'.  For further update
details, see the WebSphere manuals.
```

2.5 Installation error codes

When you configure MQ Workflow, if you receive an error when you run `fmczchk` (the configuration checking utility), the message is not listed in the MQ Workflow Messages and Codes manual. You can find a full list of errors and explanations in the file `fmczchk.htm`. This file is located on Windows in:

```
C:\Program Files\MQ Workflow\bin\fmczchk.htm
```

2.6 Installation and tuning considerations

There are several other items that should be planned for in advance of an installation. Some deal with standards, such as naming. Others deal with tuning the various components of the MQ Workflow system for maximum performance. Consider the following:

- Plan the disk space and partitioning needed when creating file systems for the various components. If, for example, not enough disk space is allocated for DB2, when MQ Workflow tries to create the database, there won't be enough available resources, and the configuration will fail.
- If you are using the Web client, you will need to configure the base URL. The default is http://hostname/MQWFClient-ConfigName/RTC.html. The section `MQWFClient-ConfigName` is configurable. Make sure to pick a name that meets the customer naming standards.
- If you are using a 3-tier setup or the Web client on a separate server from the workflow server, make sure to use the same fix levels on all servers. If the workflow server is on Fix Pack 1, but the Web server is on Fix Pack 2, problems may result.

- On UNIX systems, be very careful about file permissions. Unlike Windows, where all files are visible to all users, a file (or directory) in UNIX may only be accessible by certain users. For example, if the directory for the workflow JSPs is owned by the user 'webadmin', but the user 'mqwf' is the owner of the JSPs, the Web client might not be able to see them, because it does not have the right permissions. Use the UNIX command `chmod` to set the files so that other users can view them as well.
  - Another example of this is where MQSeries is installed, but the user trying to start the workflow system does not have the proper authority to see the shared libraries, so it cannot start workflow.
  - It is a best practice to plan the IDs that will be involved in the system ahead of time, and make sure that they are all in the correct groups, with the correct permissions for files.
- Tune DB2
  - You might want to change the default DB2 installation to make some performance improvements. For example, you can put the DB2 logs on a different physical volume from the data:
    ```
    db2 update db cfg for <dbname> using newlogpath <existing new
    path>
    ```
  - Distribute tablespaces on separate disks. The tablespaces that require high performance are Indexes (INSTINDX), Audit Trail (ADTTRAIL) and Work Items (WORKITEM). If you audit to MQ instead of the database, then you don't need to optimize the Audit Trail.
  - The tablespaces requiring medium performance are: Processes and Activities (ACTWI, BLOCK, BLOCKACT, PROCACT, PROGACT, PROCESS, NTFYITEM), Container (CONTAINR), and Sessions (ADMIN).
  - The tablespaces requiring low performance are: Model, Staff and Topology Data (MODEL, TEMPLATE, LIST, TOPOLOGY, STAFF)
  - Tune the DB2 memory usage. The defaults for DBHEAP and SORTHEAP are too low for most cases.
  - Create separate bufferpools for Indexes and Audit Trail
  - Increase bufferpool sizes as long as bufferpool hit ratios increase (try to reach > 95% hit ratio)
  - Set NUMIOSERVERS >= the number of physical disks used +2
- Tune MQSeries
  - Physical separation of MQ queues from MQ log files
  - Use MQServer interface for MQ Workflow API whenever possible. This saves operating system processes for MQClient connections (amqcrsta). Do **not** use the MQClient API on the WebSphere Application Server box.
  - Response time improvements of 10% - 35% can be achieved through reduced pathlength on the client side
- If you will be dealing with large amounts of data, or very large worklists, you might exceed the default maximum message size. This has to be adjusted in several places:
  - MAX_ MESSAGE_ SIZE in the FDL
  - MAX_ QUERY_ MESSAGE_ SIZE in the FDL

    - `alter qmgr maxmsgl(...)` in runmqsc
    - `alter channel (...) chltype( clntconn) maxmsgl(...)` and `chltype( svrconn)` in runmqsc (to allow the channels to handle larger chunks)
    - `alter qlocal( EXCINPUTQ) maxmsgl(...)` to allow the queue to store these large messages
- Create shell scripts to replace the manual start and stop of the MQ Workflow servers on a UNIX system.
    - Create a start script to start DB2 (if needed), the queue manager, the trigger monitor, and the admin server.
    - Create a similar script for shutting down the system.
    - Set LD_LIBRARY_PATH and LIBPATH to include the `/usr/lpp/fmc/lib` directory in your startupserver.sh. Make sure to do it before WebSphere Application Server launches.

It is important that the MQ Workflow system be brought down in a proper manner (as opposed to using `kill` in UNIX or the task manager in Windows). If you do not do this, shared resources might be left in a state where they are locked. It is also a good practice to run `fmcclean` after the system is brought down.

2.7 Runtime recycle threshold

The runtime recycle threshold is a parameter that can be set by using `fmczchk`. By default, it is not activated; the MQ Workflow servers remain running until the system is shut down. If you enable this parameter, the servers will be recycled every *n* transactions. If the servers are holding resources, such as memory, or database locks, these are released. The following is an except from the *MQ Workflow Installation Guide*:

> Setting this variable to a value greater than zero, activates the server recycle feature. The value specifies the maximum number of server transactions that a server instance performs before it frees its resources, by shutting down and restarting. The actual threshold is determined by a random threshold based on this value. This feature applies to the execution server, cleanup server and scheduling servers. It can help if you have problems with memory consumption increasing, server performance decreasing, or keeping servers running for weeks.
>
> Not setting the profile variable or setting it to 0 (or a negative number) disables this feature. For example, if you want to shutdown and restart fmcemain every 1000 transactions, you can modify this variable by entering:
>
> ```
> fmczchk -c inst:m,RTRecycleThreshold,1000
> ```
>
> Now, any new fmcemain that you start will shutdown after 1000 transactions and a new one will be started.

2.8 Separation of work

In some organizations, there is a requirement to have one configuration for the workflow system, but to keep work separate for different lines of business. For example, a bank has retail, commercial, and investment banking groups. Each line of business might want their workflows kept separate from the others.

The best approach for this situation is to use naming standards and categories. For example, all workflows for the retail group might have names starting with "Retail_", and all of their workflows in a category named "Retail". Users can be authorized for just one category, so that a user from Retail would not see any work items for the Commercial category.

3 Process modeling

This chapter covers best practices for using the MQ Workflow toolset. It is not intended as a complete modeling guide. For further information, see the following Redbooks at www.redbooks.ibm.com:

- *MQ Workflow for Windows NT for Beginners*, SG24-5848-00,
- *Continuous Business Process Management with HOLOSOFX BMP Suite and IBM MQ Workflow*, SG24-6590-00.

Also, see the manual *IBM WeSphere MQ Workflow Getting Started with Buildtime*, SH12-6286 and the SupportPac WD01 - *Business Process Modeling with MQ Workflow*.

In addition, IBM Learning Services offers the class MW301: *MQ Workflow Modeling*. See the IBM Learning Services Web site for course roadmaps and schedules:
 http://www-3.ibm.com/services/learning/index.html

3.1 Naming standards

Each activity step in a process is given a name on the General page in Buildtime. If this name is too short, it might not be descriptive enough, for example, bnklnchk) However, long names can also cause problems. For example, BankingApplicationLoanProcessingCheckoutProcedure will be displayed in the visual representation of the diagram making the diagram hard to read.

To keep track better of which program is associated with which process, you can make the name a combination of the process name and the activity name. Because these names are a property of an activity, they no longer affect the graphics. For example, LoanExamineData and LoanApprovalRouting might be program names for the Loan process, whereas CreditExamineData and CreditSendApproval are the program names for the Credit process.

Although program names and process template names can contain blanks, it is a best practice not to use them, because of the problems they can cause with XML messages and the Web client. Instead of “Final Approval Step”, use “FinalApprovalStep”, for example. In addition, if you use capital letters for just the first letter of each word, this makes the name easier to read (FinalApprovalStep vs. finalapprovalstep or FINALAPPROVALSTEP, for example) and should be used as a standard format for names.

Some other general recommendations are:

- Use active verbs for activities
- Avoid the words “process” or “task”
- Begin words with uppercase letters

- Keep names short
- Use business terminology (not MQ Workflow terminology)
- Do not use underscores
- Use generally accepted names
- Keep names and abbreviations consistent
- Use descriptive and informative names

3.2 Container use

Each activity in a process has an input container and an output container. With MQ Workflow 3.3, the maximum size of the container was increased to 4MB. However, the size of the container has a direct impact on performance. For this reason, do not use the container as a data store. It should be used to pass process relevant data from one activity to subsequent ones. Instead of passing a large chunk of data, pass a pointer to an external data store.

The number of elements in a container also has a performance impact. The *SupportPac WP01* states that every additional container item costs 0.02 Basic Work Units (BWUs). (A BWU is a unit of measure for system load.) The larger the load measured in BWUs, the larger a system you will need to run on. See *SupportPac WP01* for a full explanation.

Therefore, if you pass 50 items in the container, it will cost you 1 BWU of overhead. For maximum performance, keep this number as low as possible.

3.3 Container reuse

Workflow can save some of the BWU cost of activity navigation when it can reuse the output data container from a previous activity as the input container of the current activity. Where this is possible, the server does not need to be recalculated and no new data structure needs to be created in the database. If you are evaluating container reuse for a process, consider the following:

- Container reuse saves time for the navigation between activities, not for the activity execution itself.
- You can take advantage of container reuse only where straight sequential control flow is used. If you model activities with merging control flows, the activity where they merge cannot profit from container reuse.
- Data loop connectors prevent container reuse, because the input container needs to be created from the data mappings of the incoming containers, and possibly modified in the loop.
- The mapping between the output container of an activity and the input container of the following activity must be a Struct-to-Struct mapping. You cannot map to fields, not even if you are mapping to all fields.
- Nested container structures are allowed, but these must be present in all the containers you intend to reuse.

- Container members that are set for navigation mean that the containers must be checked and prevent container reuse.
- You cannot specify default values for the input container of a subsequent activity, because this means that the output container of the previous activity must be checked against the input container of the following activity. This is not allowed for container reuse.

3.4 Blocks

Blocks were designed for iterative sets of activities. For example, if an employee submits a request to a manager, the manager can approve it, reject it, or send it back for more information. In this case, a block can be used so that the manager can keep sending the request back until he approves or rejects it.

Blocks incur an overhead of 0.8 BWU (compared to 3.9 BWU for a subprocess), but can be very effective in process modeling.

- Do not use a block for high speed or high volume loops. For example, it would be very poor process design to have a "polling" activity, and then use a block for a loop to wait until the poll came back as true. The polling should be done within the activity itself, and the activity be called in the process rather than a block loop. You can use an expiration if you do not want to potentially wait forever.
- Do not use a block for purely cosmetic purposes, unless you have a low volume application, due to overhead incurred.
- Run the calculations using *SupportPac WP01* to gain an understanding of how blocks affect the load.

Note: if you use the WBI Workbench, you need to create a block as if it were a subprocess, then go to the Fields tab of the settings, and select **Block** for the export type.

3.5 Subprocesses

A powerful tool in process modeling is the use of subprocesses. Any process can be reused as a subprocess. It is a best practice to always use a data source and sink for your process, so that it can pass inputs and outputs if it needs to be reused.

Subprocesses are late binding. When a process starts in MQ Workflow, it is *instantiated*. This means that the process template is cloned in the process instance database table. So, even if a new version of the process template is imported, the previously started instances run to completion using the model from which they were cloned. The exception to this is that a subprocess is not bound until that step in the process is reached.

For example, if Process A is started, and its subprocess S is at version 1.0. Before the subprocess activity is reached, version 2.0 of S is imported to MQ Workflow. When

Process A comes to the step where it calls the subprocess, version 2.0 of S will be used. For this reason, it is important to be careful with new versions of subprocesses; they will work with previously instantiated processes.

If you need to make major changes to a subprocess and need mixed versions to coexist, one way to avoid this problem is to use the dynamic subprocess capability. Instead of hard coding the name of the subprocess to call, store the name in a variable. You could name your subprocess "CheckCredit_v1". When you update it, change the name to "CheckCredit_v2" before importing. Update the main process to call the new version. Then when you import the new versions of the process and subprocess, any old instances will still use the old subprocess, while new instances will use the new one.

Subprocesses add overhead because they are instantiated when they are called. *SupportPac WP01* states that the process start cost is 1.3 BWU, plus 2.6 BWU to create the process instance (or 3.9 BWU overall to call a subprocess). Therefore, if you were to have a subprocess within a subprocess within a subprocess, you would have a high overhead for the main process.

3.6 When should you use a subprocess?

- Don't use a subprocess just for cosmetic reasons.

- If you are using subprocesses to make the main high level process diagram look simpler, keep in mind that the overhead for a block is just 0.8 BWUs vs. 3.9 BWUs for a subprocess. Thus, a block might be a better choice.

- If you need to write something complex that will be reused by several processes, a subprocess is a better choice, because you only have to make updates once. If you use a block, you have to update it everywhere you use it.

- If you have a programming problem that lends itself to recursion, then you can use a subprocess, although the process start cost should be kept in mind. In addition, recursive algorithms are very hard for non-technical people to understand. If recursion is used, it can be very confusing to someone looking at the process monitor, or to a business user wanting to make updates to the process later. For this reason, unless recursion is really needed, it is best to avoid it.

- If extremely complex logic is needed to decide what to do next, you can use an external program, such as a rules engine, which holds the complex business rules to make the decision. Then use a *dynamic subprocess* to decide where to go next. This keeps the process diagram from getting overly complex. This makes most sense when there is a repository of business rules in use, but it means that you now have business rules stored in two different places.

- If you are implementing a business process where you know that there will be frequent changes and you want to avoid the early binding behavior of MQ

Workflow, a subprocess might be the solution. Normally, when a process is started, it will run end-to-end with the version of the template that was in use at the time the process was started. If, for example, one of the last parts of the process is the "final verification", and you know that you will be updating it constantly, you can put that part of the overall process into a subprocess. When your main process reaches this last phase, even if it is a year after you originally started, you will still get the latest version of final verification.

If the `_STATE()` function in an exit or transition condition is used in a subprocess, you can only refer to the process steps within the scope of the subprocess. If you attempt to use as an exit condition:

```
_STATE("StepFromMainProcess") = _Finished
```

You will get the error: `FMC22551E Activity StepInSubProcess: Activity StepFromMainProcess not found.`

Because a subprocess can be invoked by any other process, it is not a good idea to refer to a step in the main process. This limits the subprocess to use with just the one main process and goes against the best practice of code reuse.

3.7 Global data container

The global data container (GDC) was introduced in MQ Workflow 3.3.2. The GDC is a container that can only be set, and is used only for logging the process relevant data to the audit trail. You cannot read from the GDC (as the name might imply) in your process model. However, you can access the data with the API.

If you define a GDC for the process, whenever you map data to it, the data will be logged in the database table FMC.AUDIT_TRAIL. With this in mind, it is best to keep the amount of data in the GDC to a minimum. If you have the maximum container size of 4MB, every time you map to the GDC, you write 4MB to the database. This adds a lot of overhead and disk I/O, so be selective in what you write to the GDC.

After you have set up a GDC, you cannot remove any of the fields you defined in the data structure. Also, remember to try to keep it as small as possible, for efficiency's sake.

Another important aspect of the GDC is that you can use the values in the GDC for worklist sorts and filters. This allows you, for example, to show only work items related to a particular customer. However, sorts and filters add overhead. If you have a worklist with 100 items from 100 different processes, it requires 100 queries to the database to retrieve the GDC information.

If you use GDC values for sorts and filters, there are a couple of options for improving performance. The GDC data is stored in a database table. You can specify your own table, or use the system default table. If you expect a large number of sorts or filters, it is a best practice to specify your own table when you define the GDC, and add indexes where appropriate for better performance. If you have filters or sorts that do not use an index, you might have a performance hit, particularly on systems with large numbers of work items.

Unfortunately, the data in the GDC does not display in the Web client as a column in the worklist, unless you create a custom viewer. You can however put the input container data into the description. Use percent signs around the variable name, such as: `Customer %CustIDNum% Credit Request`. It is a best practice to use descriptions with dynamic data as in this example, because descriptions can be made more informative and you can sort on them.

3.8 Use of descriptions

When you create an activity in a process model, the description field will be seen by the Runtime user, unless you have created a custom client that does not use it. Nevertheless, for the majority of installations, it will be available for viewing. While the description can be static, it is much more useful to have dynamic content.

You can use both static text and variables to describe the activity. Any variable from the container, for example, can be used in the description. So, instead of the description "Review Customer Data", you can have "Review Customer Data for Jane Smith" by using percent signs (%) around the names of the variables. For example:

```
Review customer data for %CustomerData.FirstName% %CustomerData.LastName%
```

This is not only more helpful to the end user, it also allows you to sort descriptions, for example, to see different customer names.

For this reason, it is a best practice to use descriptions with variable information.

3.9 Default connectors

Only use default connectors when you use transition conditions. A default connector is drawn as a solid line with a circle through it. If none of the other conditions is true, then the default path is taken. If one or more of the conditions are true, then the default is not taken. In other words, it functions like an "else" clause.

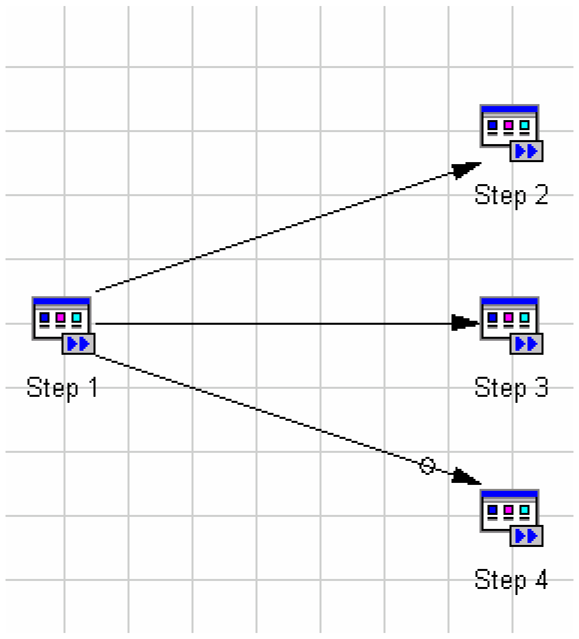

Figure 4: Default connector

In figure 4, if the conditions for moving from Step 1 to Step 2 and from Step 1 to Step 3 are both false, then the default is taken and the process moves to Step 4. If there are no transition conditions for connecting Step 1 to Steps 2 and 3, the default connector is never traversed, and Step 4 is executed.

It is a best practice to always use a default connector when branching for the following reasons.

- If you do not have a default connector and none of the transition conditions is met, the process ends because there is no further path to navigate. If you cannot think of a scenario where this might happen, you will at least need some sort of exception handling, rather than have the process just end.
- Some transition conditions can be very complex. It is better to have a default connector, than to use a long condition, such as: NOT Amount>5000 and AccountType <> "Commercial" AND….
- For maintainability, instead of having to edit a long condition, you have the equivalent of an "else" clause to use.

3.10 Expiration times

You can hard code an expiration time in an activity, or set one from a container variable. If you use a container variable, it is not obvious how you specify the time. The trick is that the time needs to be specified in seconds.

When the activity expires, the process model moves on to the next step. If there are no transition conditions, it moves to the next steps connected. However, if you use expiration times, it is a best practice to also use a transition condition that checks the state of the activity. If `_state()=_Expired`, then you can do something to take the expiration into account. Otherwise, you can continue with normal processing.

If an activity expires, a data mapping is not performed. This means that you should always use a default data connector with an activity that expires, so that you have some data to pass to the next activity.

If you use expirations (or notifications), the process modelers need to coordinate with the workflow administrator to make sure that the Scheduling Server is properly configured. For example, if the Scheduling Server is configured to check every 30 minutes and you have an expiration time of 5 minutes, the server still checks only every 30 minutes. If the Scheduling Server is not started, expirations (and notifications) will not work because there is nothing monitoring them.

For this reason, it is a best practice for the process modelers and the workflow administrator to coordinate the minimum check interval required for the Scheduling Server. Too frequent an interval wastes resources; too infrequent an interval impacts the functionality of the process models.

3.11 Branch at first step

Sometimes the first thing you need to do in a process is to branch based on some of the data from the input container. However, you cannot have a transition condition until you have gone through an activity first. The solution to this is to have a “dummy” activity at the start of the process, which doesn’t actually do anything. Before MQ Workflow 3.3.2 Service Pack 3, you could use an asynchronous UPES activity to do this. Service Pack 3 introduced the NOOP activity for this.

3.12 No operation activity (NOOP)

MQ Workflow 3.3.2 Service Pack 3 introduced a construct for no operation activities (also known as *empty activities*).

3.12.1 Empty activity at the start of a process

These activities can be used to branch at the first step. It is more efficient than a dummy activity because an XML message does not have to be built and sent. The step will not undergo staffing, and only a minimal amount of work will actually be done. Another use for NOOP might be to test the logic of a workflow and its data mapping.

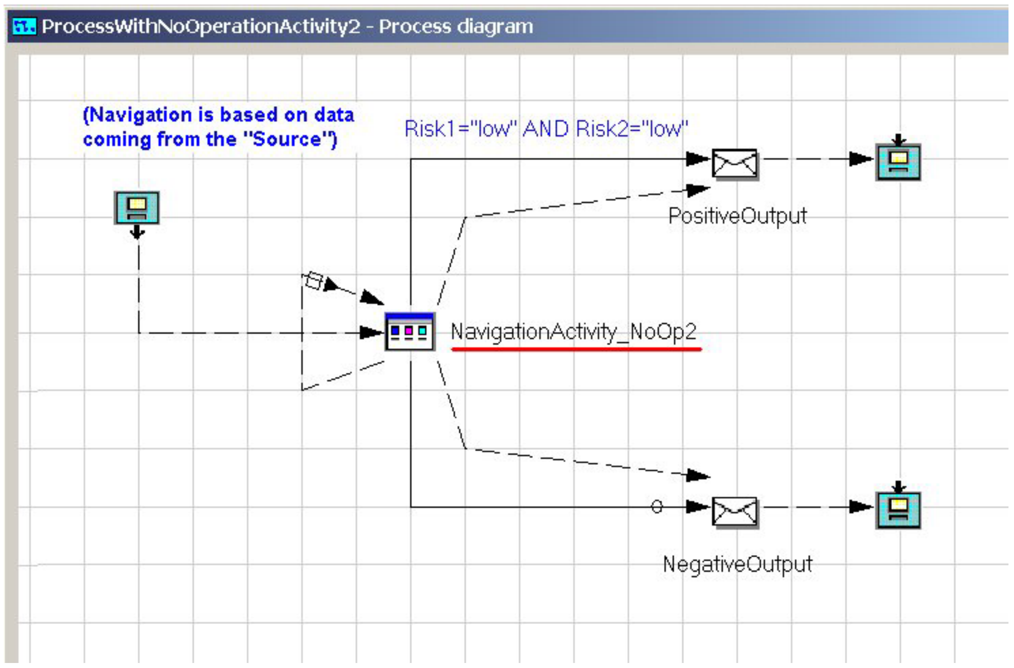
ProcessWithNoOperationActivity2 - Process diagram
(Navigation is based on data coming from the "Source")
Risk1="low" AND Risk2="low"
PositiveOutput
NavigationActivity_NoOp2
NegativeOutput

Figure 5: Empty activity at the start of a process

To use the empty activity, do the following:

1. Create a new program.
2. For the name on the General tab, use FMCINTERNALNOOP.
3. On the Data tab, select **Program can handle any data structures** and **Program can run unattended.**

When you define an empty activity in a workflow, you can use this program by doing the following:

1. Select FMCINTERNALNOOP as the program
2. On the Execution tab, deselect **User program execution agent**, as if you were defining a UPES activity.
3. Select **Asynchronous**.
4. Reselect **User program execution agent**, or type NONE for the server name.
5. On the Start tab, select **Automatic**.
6. On the Data tab, select identical data structure names for the input and output data structures.

When this step is reached, it will not perform any work. If you need to have the process pass data to the next activity, make sure that you use the default data connector. Also, note that when you run error checking on the process, you will receive the warning

FMC21730W because there is no executable specified. This is expected, and can be ignored.

3.12.2 Empty activity as a synch point

Empty activities can also be used where you have two parallel steps that need to synchronize before making a branching decision. For example, if you assign two different users to assess the risk of a loan, and then need to make a decision based on input from both of them, this would give you a sync point.

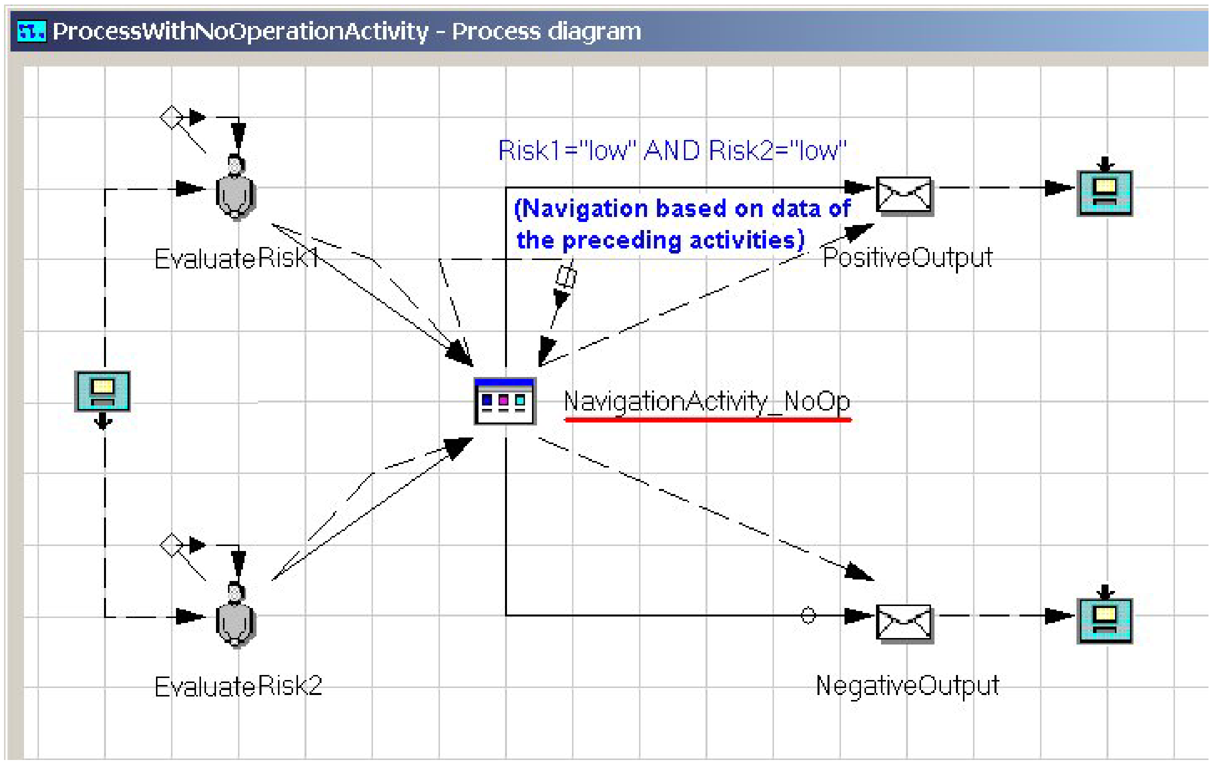
ProcessWithNoOperationActivity - Process diagram
Risk1="low" AND Risk2="low"
(Navigation based on data of the preceding activities)
EvaluateRisk1
PositiveOutput
NavigationActivity_NoOp
EvaluateRisk2
NegativeOutput

Figure 6: Empty activity as a sync point in a process

3.12.3 Empty activities with multiple parallel activities

If you have multiple parallel activities that have transition conditions to branch to follow-on steps, an empty activity drastically reduces the number of control paths. For example, if you have four parallel activities, and four follow-on activities that you might branch to, you would need 4 x 4 = 16 control connectors (and the same number of data connectors). If you use the empty activity, you need only eight of each. It also makes the process model easier to update, because you don't have to repeat the logic for the transition conditions multiple times.

3.13 Wait for a time

A frequent requirement in a workflow model is to be able to wait for a given time after an activity before moving on to the next one. While there is no predefined icon for this, you can implement the functionality in one of the following ways:

- Use an activity with an expiration time. Use the technique described in section 3.12. Put a message to a "dummy" queue, make it a synchronous message, and set an expiration time on the Exit tab. As an alternative, you could create a work item for a dummy user, and again, use an expiration time. This eliminates the need to clean the message from a queue, but it creates a work item in the database, which must be removed later, causing some overhead.

  The process model now waits before moving on to the next activity or until the expiration time is reached. However, because the dummy UPES (or dummy user) never sends back a reply message, the activity always expires.

- Use a dummy activity with a data structure containing just one field, an integer called TimeOut. Set the expiration time to come from the container instead of hard coding it. Now you can set the timeout by mapping a value to the TimeOut field, either manually, or from some existing data in the process model. So, for example, if you wanted a user to be able to specify "go to sleep for *n* minutes after this step", they could specify a value for *n*, and the model would wait that long. If they specify 0, the step would not wait at all. This gives you additional flexibility.

  A best practice is to keep a dummy process model that consists of all of your custom constructs, such as the wait activity, or the dummy UPES. You can then copy constructs from your collection, and paste it into your actual model. This helps you to eliminate errors in your model.

- Sometimes you want to wait for a time, but if an event, such as a customer response occurs, processing should be resumed. Here, it is better to use the "suspend process" functionality. You can suspend a process until a given time and date, at which point it will resume. However, an administrator can resume the process. With the dummy UPES, you would need to find the right message on the queue, and create a valid reply. With the dummy work item method, you would have to find the right one, transfer it to yourself, and then complete the work item. Both of which are more complex than suspend/resume.

3.14 UPES and data mapping

If you use a UPES, ensure that you do not use default values in your process model, because they are all overwritten. Make sure to map all fields within your UPES implementation. You will need to coordinate the process modeling with the UPES programmer.

3.15 Data structure limits

Data structures can have a maximum of 512 user-defined data members, including members and child members. In other words, nested structures do not help you to get around the limit. However, it is desirable to keep the container size as small as possible, so you should consider whether you really need 512 data members, or if you can use just a subset, and point to some external resource like a database to hold the rest of the data.

3.16 Compensating transactions

In a long running workflow process, an event, such as a cancellation, can cause the need to back out previous steps. At first glance, it would seem that holding a group of steps as a unit of work, then doing a single commit or rollback would be a good solution.

However, because a business process is often long running, it can span hours, days, weeks, or even years. If you try to hold a number of steps under a unit of work for a single commit, you can run into a problem. While waiting for a commit, a record in a database or a message sent to MQSeries is locked in a temporary hold until the commit or rollback is received. Imagine that the resources for five steps running over the course of a week are each waiting for their commits. Multiply this by the several hundred times a day that the process runs. Soon, there will be so many locked records that the system will stop. Alternatively, if a server goes down, all the locked resources roll back, and you have to start the whole process over again.

Therefore, you have to issue a commit for the resources after each step in the process. A long running process cannot be rolled back—it must be compensated for. You can create a compensating transaction to undo the work from the previous steps. Typically, you will not have to undo all of your previous steps. For example, a process for a cable company to install a new hookup is:

1. Schedule a time for the installer to call on the customer.
2. Customer Service confirms the time and date with the customer by letter.
3. Inform the installer of the schedule, along with a checklist of what to do.
4. Notify the warehouse of the equipment to be loaded on the truck.
5. And so on.

If the customer calls to cancel after step 4, what needs to be undone? Step 4 requires a cancellation of the notice to the warehouse. Step 3 doesn't need any action (you can't "un-notify" the installer, or "un-send" an e-mail). Steps 1 and 2 also do not need to be undone (you can't unsend a letter). There might be a need for a new step for quality reasons to confirm everything and close out the account. Therefore, in this example, there is only one step that needs compensation, plus one new step.

As this example shows, a compensating transaction is not always as complex as it might appear. As a rule-of-thumb, most steps that involve people do not need to be undone, while steps that involve back-end programs are more likely to need compensating. There will also be workflows that do not even need compensating transactions.

If you need a compensating transaction, it is often a good idea to put the compensating transaction into a subprocess (or a block) in order to keep the high level process less complex. Because the compensation is not needed most of the time, the overhead incurred is not significant.

If you have parallel paths in your process, take care when using compensation. If the error causing the need for compensation occurs in one of the paths, you need to use transition conditions or start conditions in the other parallel paths so that they do not continue processing after the error has occurred.

When the compensating subprocess has completed, you can restart the main process (calling itself as a subprocess), or you can end it. You can also restart a block if required. For highly sensitive processes, you can have multiple blocks acting as ‘compensation spheres’, with their own compensating subprocesses. How you implement compensation depends on the functionality needed by your process.

3.17 Design for monitoring

When programmers write a workflow, they like to take advantage of concepts, such as blocks and subprocesses. However, if too many of these are used in the process, the process monitor becomes very hard to use and understand. If, for example, your process has a block step that contains a subprocess that calls another block that uses a subprocess, and so on, the instance monitor would not be very helpful, because you would have to drill down too far to figure out what is really going on.

For this reason and performance considerations, you should try to avoid excessive nesting of blocks and subprocesses. Also, be sure to take advantage of the process context construct that was introduced with MQ Workflow version 3.3.2. This can be used to track processes through subprocesses and other products, such as MQSI. However, this does not help with visual monitoring.

3.18 Program definitions

When you define a program under the Implementations in Buildtime, you need to address the following:

- If you are using the thick client, the program to be run is specified on the tab for the appropriate platform (Windows 9x, Windows NT, etc.).
- If you are using the thin client, the logical name of the program is picked up as the JSP name (not the physical program name).

If you are using the thin client, always specify a program name under the Windows NT tab. Most customers use `fmcnshow.exe`. This example program comes with MQ

Workflow for viewing and setting container data. If you need to test your system but the Web server is not available, having the programs set up to use `fmcnshow` allows you to still be able to test. For this reason, it is a best practice to always use `fmcnshow` in your program settings.

If you are using the thick client, specify the actual program or DLL that you are using. However, it is a best practice to always use just the name of the file, rather than specify the full path.

3.19 CrossWorlds Connector for MQ Workflow

If you use the CrossWorlds Connector for MQ Workflow, do not name any of your fields "response" when you design your data container. This will confuse the connector, and cause problems that are difficult to debug. The connector does not currently support the messages for `ActivityExpired` and `TerminateProgram`. To avoid problems, set your UPES to be MQ Workflow version 3.3.0 (instead of the current version), because these messages were not included with 3.3.0, so they will not be sent.

WBI Workbench 4.2.3 has a specific object for an ICS collaboration. What makes this different from a normal activity pointing to a UPES (which sends messages to the connector), is that this WBI Workbench can now attach to the ICS System Manager, and return information on existing collaborations. Instead of typing in things like collaboration names, you simply select the one you want from a list. This reduces errors and reduces work. For this reason, it is a best practice to use WBI Workbench 4.2.3, or later, if you use the ICS connector.

3.20 WBI Workbench modeling recommendations

If you are using the WBI Workbench to model your process, here are some tips and best practices, in addition to those for Buildtime.

- Use decision names phrased as questions
- Phi types reflect the kind of information or materials
- Follow established modeling standards
- Model details iteratively
- Take advantage of drill-down capabilities
- Take advantage of documentation capabilities
- Use decisions only when transfer of information is involved to reduce the number of cases
- Use multiple choice decisions where possible
- Avoid endless loops
- Keep the model and repository "clean" (that is, delete items that are no longer used)
- Make the highest percentage path a straight line
- Standardize and use relevant bitmaps for the phis (create your own)

- Use colors for organizational units or functions (manual/automatic)
- Avoid crossing lines

4 Staffing and worklist handling

MQ Workflow provides a variety of techniques for assigning work to users. This chapter discusses several techniques that are commonly used, the performance impact that comes with using them, and any advantages or disadvantages of the techniques.

You also need to assess what effect a workflow implementation will have on the current structure of the organization. Make sure to take into account the way the organization will be after the workflow project has been implemented, not just how it currently is.

4.1 Creation of work items

Sometimes work is assigned to a single fixed user from the Staff 1 page, instead of using a dynamic assignment. In this case, just one work item is created, so system performance is not an issue.

Usually, multiple users can act upon a given work item. This is where the greatest performance impacts can occur, because multiple work items can be created in the database table. Because each work item corresponds to a unique entry in the Runtime database table, adding 1000 work items adds 1000 records in the table FMC.WORK_ITEM. Thus, as work items are created, checked out, checked in, etc., there is some load placed on the database. *SupportPac WP01* states that it costs 0.02 Basic Work Units (BWU) for each work item created. If you create 1000 work items, it would cost 20 BWUs. Therefore, a design goal is to have as few work items as possible.

When too many work items are created for one unique task, multiple users try to start the same work item. For the previous example, 1000 people can see the work item; one person can start to work on it, yet it remains on the other 999 user's worklists until they next refresh their worklists. This can result in someone trying to check out a work item that has already been processed by someone else. Other than keeping the number of users who can see a work item low, you can also address this problem by using the Group Worklist Handler. See the Web client documentation in the file `MQ_Workflow_ dir)\DOC\WebClient\index.html` for details, and section 4.2 for an overview.

When people interact with a computer system, they can only process a certain amount of information before they get overwhelmed. In a workflow system, if a user sees 500 work items, it will be too much to process at once. Therefore, it is a best practice to use thresholds on worklists to keep the number of work items the user sees down to a reasonable level. This is also better for system performance. See section 4.4 on filters and thresholds for more information. Custom viewers, such as the Group Worklist, can also perform a "get next n work items" to improve on this even more.

4.2 Work assignment scenarios

4.2.1 Work assigned to one user

You can assign work to just one user from the Staff 1 page**.** This is the most efficient scenario in terms of database load. However, it is very inflexible, and is rarely used. In most cases, multiple people are assigned the work item, so that whoever is not busy can act upon it. This allows for the most efficient processes.

Assigning work to one user from the Staff 1 page is best used in situations where you want to assign work to a manager or exclude the starter of the process or of a particular activity (perhaps when a second approval is required). Also, you might need to assign work to a specific user, or to notify someone (perhaps the starter of a previous step) that the request was fulfilled.

4.2.2 Work assigned to All People

You can select **All People** from the Staff 1 page. MQ Workflow creates one work item for each user in the system. Therefore, if you have 200 users, 200 work items are created. When one user checks out (or starts) their work item, the other 199 are deleted from the work item database table. As the number of users grows, this becomes a less and less desirable scenario. If there were 2000 users in the system for example, there would be a very large load placed on the system. This is the least efficient method, and it should only be used in small systems.

If you specify an organization on the Staff tab of the process properties, then only users in that organization are assigned the work if you select **All People**. For this reason, it is a best practice to always select an organization if you intend to use this option, to limit the number of users who are assigned the work.

4.2.3 Work assigned to multiple users

Another way to assign work is to use the dynamic assignment from the Staff 2 page (by role, organization, level, or a combination of these criteria). If there are 10 users out of the 2000 in the system that fit the assignment criteria, then only 10 work items are created. If the number of users who would fit the criteria is reasonably small, then this technique works quite well. However, if there are a large number of people who fit the criteria, then there is less of a saving.

It is a best practice to use role/organization/level based work assignment when possible. In addition, keep the roles specific enough so that there are fewer users who can be assigned a particular work item. A given user can have multiple roles; so keeping the number small is not an issue.

4.2.4 Work assigned to virtual IDs

If a large number of users need to see a work item, virtual IDs (also called *ghost IDs*) can be used to maximize performance. For this example, there are 2000 users in department 123, any of which can act upon a work item. The following technique could be used:

1. Create an ID called DEPT123.
2. Use the Staff 1 page to assign the work to this ID.
3. Allow the users in department 123 to view this ID's work items in their settings.

Advantages:

- There is only one work item created, even though there are 2000 users.
- All 2000 users can see the work item.
- Work is automatically load balanced (see below for details).
- Performance is better: The overhead from transferring a work item is 0.3 BWUs. However, if there is only one work item created instead of 2000, you gain a saving of the system not having to invalidate and delete the 1999 that did not actually get run. The overhead from this technique is more than negated by the savings, even for systems with fewer users.

Disadvantages:

- The work item must be transferred to the user before they can check it out or start it.
- You have to train the users to transfer the work item to themselves before working on it, or you have to modify the Web client or thick client application to do both the transfer and then the check out/start the work item for them.
- Multiple users can still attempt to check out the same work item (see below for details).

One other consideration is load balancing. It is important to make sure that you don't have the situation where Employee A has many items to work, while Employee B has none. If you use the virtual ID concept, then the work is automatically load balanced! Because the multiple users can see all of the work assigned to that ID, you never run into the situation where Employee B has nothing on their worklist, while Employee A is overly busy.

If there are a large number of users, two or more users might click on the same work item at nearly the same time. The first one will get the work item, but the second will be told that the item is already checked out. As the number of users grows, the probability that someone has already checked out a work item on the user's list also grows. This drawback means you have to refresh your worklist more often, which adds a higher load to the system. The solution to this problem is the group worklist, which is discussed in section 4.2.5.

This approach also works well for the case where the work assignment in MQ Workflow is not fine-grained enough for the customer's requirements. This puts all of the work into a pool. End users can have individual filters to pick up only the work items they are interested in.

You can also use this technique for adding more users to the pool to handle increased load. If, for example, a department normally processes 1000 orders per day, but because of an upcoming holiday, there are suddenly 3000 orders per day. More users can be authorized to see work items for the virtual ID, allowing more people to work on the orders, and complete them in time.

4.2.5 Group worklist

The MQ Workflow Web client comes with a Group Worklist Handler. The Web client documentation describes this as a sample program. It solves the problem with the virtual ID technique, where two users can try to check out the same work item at the same time. Work items are still assigned to a virtual ID. However, when a user logs on to the system, they see their own work items rather than seeing the entire worklist of the virtual ID. The handler goes to the virtual ID, takes a configurable number of items from the virtual ID, and transfers them to the user's own worklist. Therefore, if you configure the handler to get 10 work items, it will transfer 10 items from the virtual ID to the user.

As the user consumes items on their worklist, they will at some point drop below a configurable threshold. If this is set to 3, for example, when the user gets below 3 items on their list, the handler will go to the virtual ID and transfer more work items to fill the list back up to 10 items.

When the user logs off, any items on their list that they did not complete are transferred back to the virtual ID.

When using the group worklist, be careful if the number of work items is low, and the number of users is high. If there are only 30 items on the virtual ID worklist, and there are five users, the first three to log on get the 30 items. The other two users do not get any work items until more are created. Therefore, this is not a good technique for low volume applications. See the documentation for the Group Worklist Handler for further information.

4.2.6 Peak load handling

Often in a workflow environment, different departments normally do their own work, but at peak times, a particular department might be asked to help.

One simple way to do this is to use the sorting and filtering capability of the worklists, taking advantage of the fact that a user can switch between multiple worklists. You can set a worklist filter to show all work items, or just ones belonging to the user. You can also set up a sort based on the owner of the work item. A user's normal worklist might just show them their own work items or just the first 10 for a virtual ID. At peak times, their manager might direct them to use a different worklist. They would then see the work items from all users, sorted so that the other department's items are listed first, for example. This gives the required functionality, with a minimum amount of customization. See section 4.4 for more information on filters and sorts for worklists.

4.2.7 Staffless activities

The best performance can be achieved when automatic, unattended activities are used. These use the XML interface to invoke applications. If you assign a user, it will only be used if the activity is in error.

4.2.8 Staff assignments via container values

You can use values in the data container for any of the values on the Staff pages, instead of hard coding the values. For example, if you have an external program that goes through a very complex algorithm to determine the staffing for an activity, you can have one step where you call that program, then pass the input to the next step. The value from the input container can then be used to assign the work item.

This technique can be used when you cannot determine the staffing at build time, but you are able to do it based on process relevant data. This makes it a very flexible, dynamic approach.

The downside to this technique is that you have something external to MQ Workflow, and you must maintain your staffing rules there as well. However, the upside is that you can use more complex rules than MQ Workflow is able to implement on its own.

Also, don't forget to look through the information in the "predefined members". Open the data mapping for any step, and you will see both _PROCESS_INFO and _ACTIVITY_INFO. These fields contain information ranging from the ID of the process administrator, to the ID of whom to notify if the activity takes too long. If you want to have a step at the end of a workflow process to let the person who received a notification in step 2 know that the process is complete, you can map from the predefined member of step 2 into the current step, and use that ID in the container data to assign the work.

4.3 Worklist refresh policy

When a worklist is refreshed, whether on the thick or thin client, a request is sent to the MQ Workflow server, which returns the work items. This action costs 1 BWU from the server. Thus, the more often a worklist is refreshed; the more load there is on the server. However, if you have a refresh rate that is too slow, it might mean that the data displayed is no longer valid, such as when work items on a list have already been processed.

In the thin client, you can set autorefresh in the WebClient.Properties file. If AutoRefresh is true, when you complete a work item, the Web client goes to the server and brings back an updated worklist. If you set it to false (or comment it out), only the work item you just acted upon is refreshed. As the number of users in a system grows, you might want to consider keeping it set to false to keep the load on the system lower.

If you use the Windows client, you can set up a “push” worklist, where work is sent to the client, instead of the client asking for work. It is a best practice to keep worklist refreshes to a minimum to keep workload down.

4.4 Worklist filters and thresholds

Another important aspect of worklists is their size. If a system has 10 000 work items, you would not show them all to a user in a worklist for the following reasons:

- A person cannot make sense of a list of 10 000 items. It is a best practice to show users a manageable number of items in their worklists. For most users, this is 20 items or less.
- If all 10 000 items are displayed, then 10 000 records must be read from the database, and all 10 000 must be sent over the network to the worklist, decreasing performance.

To keep the number of items on a worklist down, you can use filters and thresholds. If you use a threshold of 20, then at most 20 work items on the list, even if there are 10 000 items in the database that fit the criteria for the worklist. The database load does not change, but the size of the returned message is smaller. (A threshold is not a limit on the number of records returned from the database; it is applied at the workflow server level.)

If you have a business reason for not using a threshold, you can use a filter. You can create a filter on different fields, such as state, description, name, or date. Instead of seeing all 10 000 items, you can limit the list to the items in the Ready state, with a particular process model name. This reduces the amount of data sent to the client when a worklist is refreshed.

If you use a custom client instead of a default client that comes with MQ Workflow, filters and sorts are applied by MQ Workflow as it makes SQL calls to access the work item database. This is more efficient than if you try to sort and filter the work items yourself. Note that there is a 4096 byte limit on the length of a filter statement.

Also see section 3.7 on the global data container for information on how it can be used for sorts and filters.

There is a well-known example in the workflow world that should be kept in mind when deciding what work items a user needs to see. A customer implemented a workflow system, where the users could see a number of work items needing their attention. Because they knew they were being monitored for productivity, they started to “cherry pick”, and just worked on the easy work items, leaving the harder ones for someone else.

This customer decided to change their interface to present the users with only one task, the next one they needed to do. Now hard work items no longer fell through the system.

This may be an extreme example, but it demonstrates a key concept, in that you must keep in mind the ways the users will interact with the system, and perform their work. Make sure to think out these issues before deploying the production system.

4.5 Private and public lists

Lists in MQ Workflow can be public or private. A public list can be used by any user, whereas a private list is only for the user who created it. So, if the user ADMIN creates a private list called AdminPrivateTemplateList, then only the user ADMIN can even see that the list exists.

You can take advantage of this feature in your implementation. For example, you can make your process template lists all private, so that only administrators can see them. Normal users would have no process template lists, so they cannot even see the templates in the system. You might also want to limit access to process instances. For example, you might want CSRs to be able to see the list, so they can pull up the monitor for a particular process instance, yet you might want to keep data processing staff from being able to see them. Use the public and private list concept to keep any particular user focused on just the work you want them to be able to see. However, if a user has process category authorization, they can create their own lists unless you have modified the default client.

For a public list, it is a best practice to include the filter `Owner=CURRENT_USER`, so that only the current user's work items are displayed. Otherwise, the user might see many work items (including duplicates) which belong to other users.

5 Clients, APIs, and interfaces

MQ Workflow has both a Windows-based “fat” client and a thin client. This chapter discusses some of the issues with each, and addresses topics such as customization. Also discussed is the XML interface to MQ Workflow. This section is intended to give guidance on several issues, but it is not a replacement for the programming manual, *IBM MQ Workflow Programming Guide,* SH12-6291.

5.1 Web client or thick client?

When you select the architecture for your workflow application, you must decide which client you want to use. Most modern applications use a thin client, because it frees the end user from needing special software on their workstation. This makes the system cheaper and easier to maintain. It also means that if you upgrade the version of workflow on the server, you do not have to go out to all of the client workstations and upgrade them as well. As an additional benefit, a company can allow external users (such as, agents for an insurance company) to access the system, without requiring them to have any special software on their systems.

If a company is primarily a Microsoft shop with lots of Visual Basic programs and Microsoft tools, such as Word, then the fat client might be a good fit. The fat client has a feature called a Program Execution Agent (PEA), which does not have an exact equivalent in the thin client. The PEA is used to launch EXEs and DLLs. The servlet of the thin client is similar in that it ‘launches’ the user screens in the browser, but the PEA can also start other programs external to workflow. For example, it can launch Word with a specific document, as part of a user step.

The fat client is Tivoli Inventory Enabled. This means that you can use Tivoli to help you automate your installation. An installation script can also be captured and used, so that the installer does not have to go through all the options from the install program.

However, for reasons, such as the ability to more easily migrate, usually the Web client will be the better choice.

5.2 Web client

This section discusses some best practices for using the Web client. You need to modify the default Web client as it comes “out-of-the-box” before you use it in production.

There is a servlet, which is deployed on the Web server. This servlet receives commands from the browser, communicates to MQ Workflow, and builds a reply. What the user sees depends on a “viewer” which works with the servlet. Different viewers are provided with the Web client, or you can create a custom viewer of your own. The following lines are taken from the WebClient.properties file:

```
# This setting specifies a Java class that is used to create the
# response pages for the commands handled by the Web client.

#DefaultViewer=com.ibm.workflow.servlet.client.DefaultViewer

# JSPViewer uses JSPs instead of HTML template files

DefaultViewer=com.ibm.workflow.servlet.client.JSPViewer

# InternetConnectionViewer mimics the look & feel of the old internet
# connection for IBM FlowMark

#DefaultViewer=com.ibm.workflow.servlet.sample.InternetConnectionViewer
```

By default, the JSPViewer is selected, although the other viewers can be enabled. The presentation of Web pages displayed by the default Web client is controlled by the ListViewer.jsp and other JSPs and HTML files served by the JSPViewer.

5.2.1 General customization

For serviceability reasons, do not change any of the JSPs or HTML pages shipped as part of the Web client. Make your changes to copies of the files as in the following example. When you apply Web client updates, they will not interfere with your customizations. If you just change ListViewer.jsp, you would have to merge your customization with updated versions of ListViewer.jsp that are part of a service pack.

1. Copy ListViewer.jsp to MyListViewer.jsp and make your changes there.
2. Copy JSPViewer.java to MyJSPViewer.java.
3. Change MyJSPViewer's base class from DefaultViewer to JSPViewer.
4. Change all methods forwarding to ListViewer.jsp to forward to MyListViewer.jsp and remove all other methods.
5. Register the MyListViewer class in WebClient.properties.

If you add a custom command to the Web client, it is a best practice to preface the name with `x-`. For example, instead of using `transferAndCheckout` for the name, use `x-TransferAndCheckout`. This is because new commands might be added to the Web client over time, and there is a chance that your custom command might conflict with a new one. The lab has stated that they will not add any commands that begin with `x`.

A new chapter on customizing the Web client was added to the *MQ Workflow 3.4 Programming Guide*, in Chapter 69. It is recommended that you read this chapter for more information.

5.2.2 UI design

When you create your workflow system, the way in which the user interacts with the system is important. There are many ways to design the user interface (UI) for the system. However, you should keep the following in mind:

- The Web client supports JSPs for activity implementations and presenting the worklist to the user. If you use the standard Web client, or use it with minor modifications, you can rapidly deploy your system.
- If you design a look and feel for the system first without taking into account how the Web client works, then you will need to do extensive programming with the APIs.

You should consider whether the default client would be enough for your needs, or if minor customizations to the provided client can provide the needed functionality.

It is a best practice to consider the Web client's look and feel **before** you design the user interface. This allows you to take advantage of the Web client's functions and features, reduces customization, and reduces the time to implement the system

5.2.3 Error handling

The error messages for errors, such as "not authorized" or "incorrect password", often contain the entire Java error path, which can often be very long. You should consider modifying the default Web client to handle at least the common errors, and show messages that are more user friendly.

The file ViewError.jsp is used to display the errors. This file is located in the `{MQWFInstDir}\Configname\WebClient\WebPages\forms` directory. For a list of exceptions that can occur, look in the file `{MQWF Directory}\api\fmcmretc.h` or the Programming Guide. It is a best practice to update ViewError.jsp, so that a more informative error is displayed. Include information on who to contact if the problem persists.

5.2.4 Force logon

By default, the Web client creates a session in PRESENT mode. This enables users to log on to multiple computers on the network. There is an option on the logon screen for force logon. This makes the session mode PRESENT_HERE. If a user is logged on in PRESENT_HERE mode, then logs onto another system in PRESENT_HERE mode, the session at the first logon is invalidated.

An organization can make it a standard practice to ask all users to use force logon whenever they logon to the system. If you want to enforce this, it is a best practice to modify Logon.jsp to have the session always set to PRESENT_HERE automatically. That way, a user cannot forget to do it. Make sure to use the guidelines in section 5.2.1 for modifying files for the Web client.

5.2.5 Applying updates

If you apply a fix pack on the workflow server, any updates to the Web client are not automatically sent to the Web server. To update the Web server, you must reconfigure your Web client. Thus, it is a best practice to keep your Web client configuration separate from the main configuration (normally called FMC).

When a new fix is released and applied to the workflow server, you can re-create your Web client configuration, which will redeploy all the needed files to the Web server. You will need to copy all your JSPs and images after applying a fix pack and re-creating the Web client. Make sure to back up these files before you upgrade the Web client. However, do **not** back up the system files, such as logo.gif. Otherwise, you might restore an older system file on top of a new one.

Another reason for using a separate configuration for the Web client is that if you get a failure during the installation, previous steps are backed out. If, for example, you created your MQSeries queues and your DB2 database then had a problem with the Web client, you would have to redo everything. However, if use two separate configurations, any failures in the Web client configuration will not make you need to redo the rest of it again.

If you put everything into a single configuration, then you would have to redo the entire system configuration to get the updates onto the Web server. It is a best practice to keep separate configurations for each.

5.3 Fat Client

5.3.1 Program Execution Agent (PEA) and the Web client

On the fat client, there is a component called the Program Execution Agent (PEA), which is used to launch programs. With a Web client, there is just a Web browser on the client side. This means that any of the PEA-related APIs do not apply to Web client programs as of the current release. The PEA also launches the programs for activity implementations.

The PEA can be used to launch "tools", or related programs. For example, when a work item is started on a thick client, the PEA can launch Microsoft Word with a document that describes the procedures. You can simulate this behavior on a Web client by launching another browser window, and pointing that window to the Word document (use `target="_blank"` in the HREF statement to specify a new window).

The path to the document you use (for example, `e:\info\procedures.doc`) is evaluated on the client side. Make sure that you have a standard network drive that all users are connected to, so that the path is the same no matter which machine you are on. In addition, plug-ins might be needed for your browser to view documents in a particular format. For example, MS Word installs a plug-in to Explorer, so that .doc files can be viewed.

Another use of the PEA is to launch a back-end program for an automatic step. Again, this cannot be done with a Web client. However, there is a way to start a program on the server. To set this up, do the following:

1. For the activity in the workflow, in the properties on the Control page, uncheck Inherited for "Program activities can be checked out", and make sure this option is not checked. This prevents the program from being executed as a Web client activity, and will then fall to the PEA on the server.
2. To manually start a PEA on the server, use the command `fmcxspea -u=admin -p=password`.
3. To manually stop a PEA on the server, use the command `fmcxpsd`.

5.3.2 PEA and future considerations

The future direction of Workflow is increasingly into the WebSphere space, with thin clients, and even workflows running within the application server space. For these reasons, the use of a PEA should be discouraged whenever possible. A solution today that uses the PEA will work. However, if you want a strategic solution for the next 5-10 years, you should consider designing the solution without this component. This will make migration easier in the future.

5.4 The MQ Workflow XML interface

MQ Workflow has an XML interface to communicate with external systems. Incoming messages can make requests to the MQ Workflow engine, while MQ Workflow can send messages to request a program to perform an action in a user-defined program execution server (UPES) implementation. Each of these will be examined in this section.

All XML messages for MQ Workflow are carried by MQSeries. Therefore, the message is really the MQSeries message header (MQMD) followed by the payload (XML formatted data.

5.4.1 UPES implementations

You can use the MQ Workflow API within an external application to interface from MQ Workflow to external systems. However, it is often not desirable to change an existing application. In addition, MQSeries has become pervasive in the industry, which means that many applications are already MQ enabled, or have MQSeries adapters available "off-the-shelf".

Because of this, the MQ Workflow XML interface is a good way to let MQ Workflow interface to external systems. If MQ Workflow wants to request an external program to perform work as a step of a process, it can send out an XML message to it. However, it is unlikely that the receiving application understands the MQ Workflow format of XML. In this case, you can use a tool, such as MQSeries Integrator to transform the message on-the-fly into the proper format so that you don't have to modify the application. If the request was synchronous, a reply message will be sent back. Again, MQSI can transform the reply message back into the MQ Workflow XML format.

MQ Workflow calls such an interface a UPES. From an MQ Workflow standpoint, a UPES is simply a queue where the messages are sent. It is the job of the UPES implementation to read in the XML messages from MQ Workflow, and deal with them. There are three main messages that MQ Workflow can send to a UPES:

- ActivityImplInvoke
- ActivityExpired
- TerminateProgram

A best practice is to ensure that the UPES can handle all three message types. If an `ActivityImplInvoke` is issued to the UPES, but it takes too long (based on the expiration time defined in the process model), the process model will move on to the next activity. When this happens, the `ActivityExpired` message will be sent to the UPES. Your UPES should be able to handle the incoming message, because it implies that a reply is no longer needed and there is no need to do further processing. Usually, if the activity was expired, you should roll back any work you did, because the process model is not expecting the step to have completed. Likewise, it is a best practice for the UPES to handle the `TerminateProgram` message. If an activity is terminated or it expires, MQ Workflow ignores messages sent back from the UPES for that activity. This makes the UPES transactional.

If your UPES sends an incorrect reply to MQ Workflow, you will receive a "general error message". To have this message sent, you must fill in the ReplyToQ and ReplyToQMgr fields of the MQMD. If you use MQ Workflow 3.3.2 with Service Pack 2 or later, the original XML message is included as a comment in the reply message. With MQ Workflow 3.4, there is a new tag set:

```
<RequestAsHexString>

 …

</ RequestAsHexString>
```

This returns the request as a hex string, so that even hidden characters can be examined. It is a best practice to take advantage of these error handling capabilities.

Another major design point for UPES implementations is to decide whether all UPES messages for all your UPESs should go through one queue, or multiple queues. Here are some advantages and disadvantages for each design.

5.4.1.1 Multiple queues

- Because each UPES message goes to a different queue, the queues can be spread across multiple systems in the enterprise to get better throughput.
- You can use MQSeries clustering so that there are multiple instances of the queue, and multiple instances of the program processing.

- Because there are multiple UPES implementations, each one needs their own copy of the base code to handle the messages. If you make a change to the framework, you need to ensure that each implementation is updated. Therefore, you should use a construct, such as a class if you use Java, or a subflow if you use MQSI. The *XML SupportPac WA05* has an extensible UPES that you can use for this.
- With multiple queues and multiple implementations, if a program goes down, the others are not affected. Thus, this approach does not have the possible single point of failure problem of the single queue.

5.4.1.2 Single queue

- Because all the UPES messages come through a single queue, this queue can become a bottleneck if the volume grows too high. Consider using MQSeries clustering to spread the messages for better throughput.
- Because all messages are coming in to one queue, you can write the logic for parsing and handling the messages once, then fan out the requests to the appropriate resource. This listener approach in the *XML SupportPac WA05* can be used for this.
- Because one program handles all messages, if that program goes down, all messages stop flowing. To avoid having a single point of failure, consider using MQSeries clustering on that queue, with multiple instances of the UPES implementation, or use hardware redundancy, such as an AIX HACMP cluster.

5.4.2 UPES and load balancing

After you decide how to distribute the messages, you can also decide how to deploy your UPES implementation. You can run it on the same box as MQ Workflow, or elsewhere on the network. You can run one instance of it, or multiple instances. Here are a few of the key configuration options:

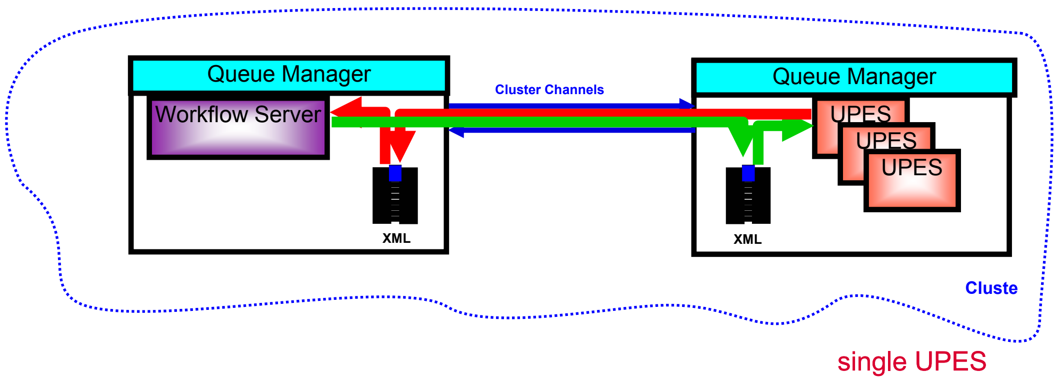

Figure 7: Single UPES – multiple instances on single box

This option allows multiple instances of the UPES to process the message, improving performance. You can control how many instances are started to control the load on the system and throughput. The UPES runs on a remote server, so that the workflow server has less system load.

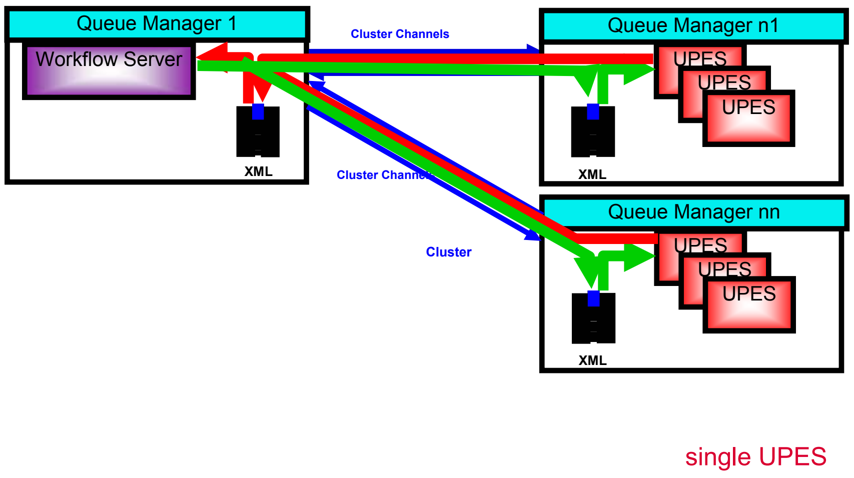

Figure 8: Single UPES – multiple instances on multiple boxes

Now there are two (or more) servers where the UPES implementations run. This provides load balancing, because the requests are performed by multiple servers. The queue managers are all in the same cluster.

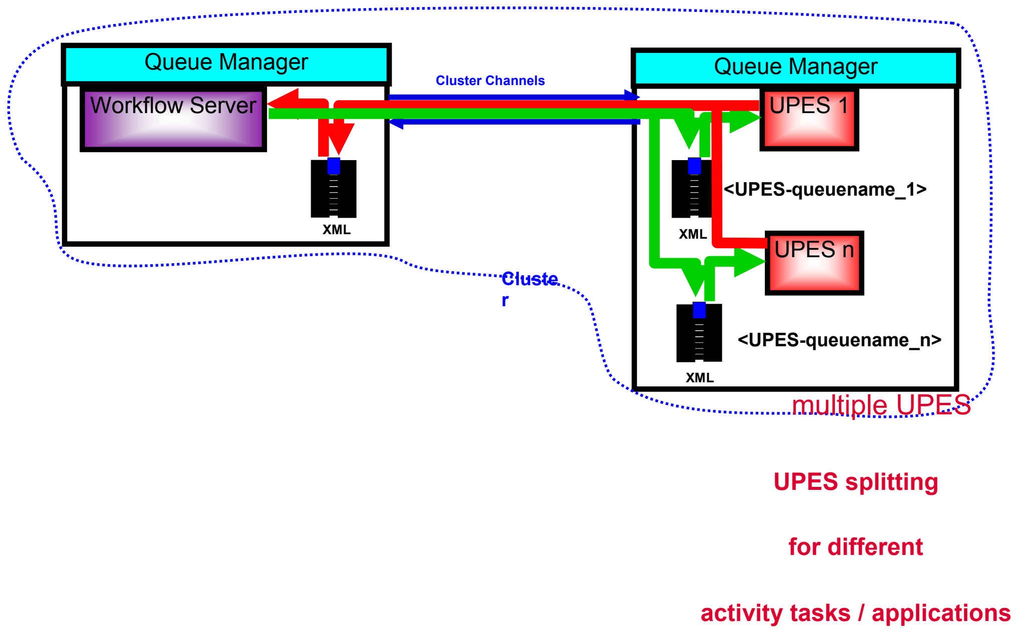

Figure 9: Multiple UPESs on a single box

In this configuration there is only one queue manager, but with multiple queues for the UPES implementations to read their messages from. Different activities use the different queues.

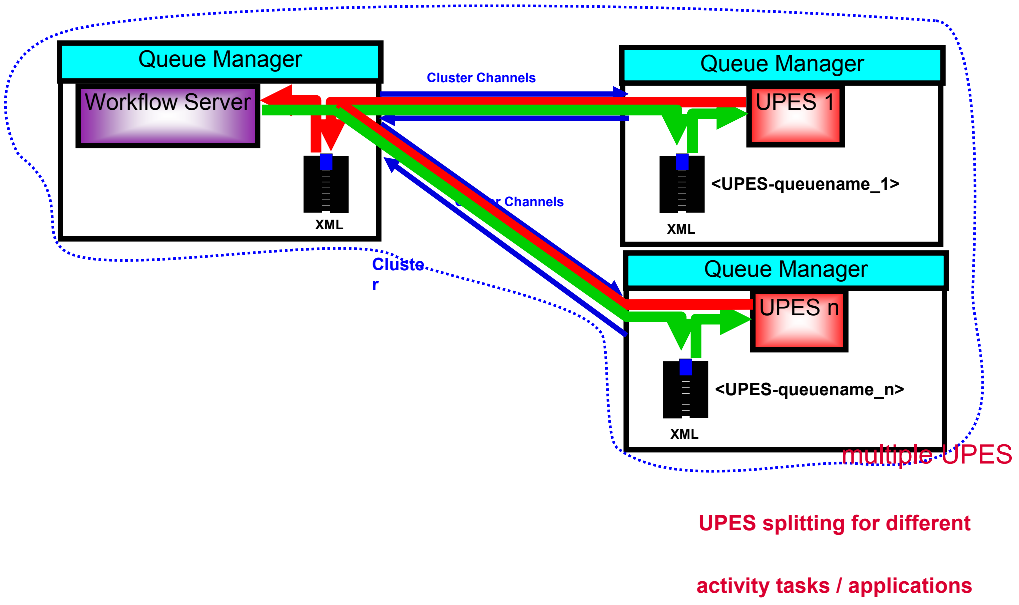

Figure 10: Multiple UPESs on multiple boxes

This configuration has each UPES implementation with it's own unique queue, and each UPES implementation on a different physical server.

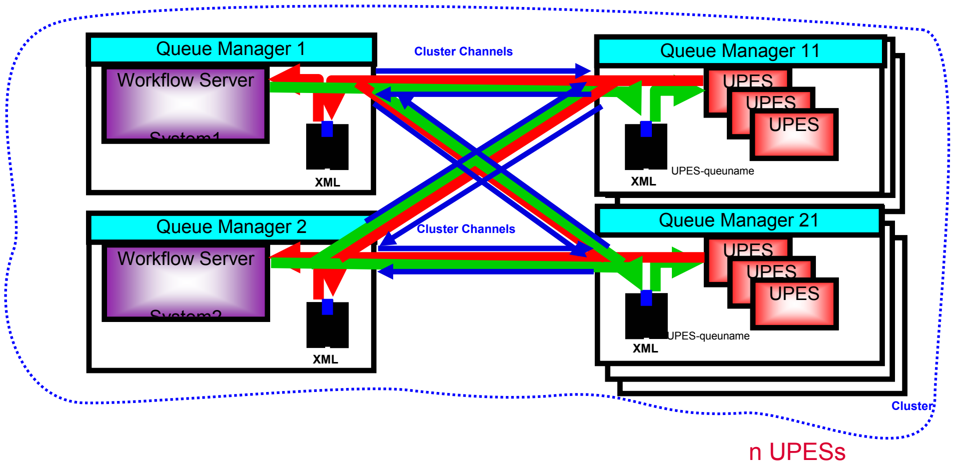

Figure 11: UPES–load balancing within a Workflow system group

Finally, in this configuration MQ Workflow uses a system group, with multiple physical servers, each with their own queue managers in an MQSeries cluster. There are also multiple physical servers for the UPES implementations, also in the same MQSeries cluster.

Now a workflow can run on any of the servers in the system group. The message goes into the cluster where it can be processed by any of the UPES implementations on any of the boxes. For a large system with high throughput rates, this is the most desirable configuration.

5.4.3 Incoming XML messages

MQ Workflow can accept the following incoming XML message types, which request the workflow engine to provide a service:

- ProcessTemplateCreateInstance
- ProcessTemplateCreateAndStartInstance
- ProcessTemplateExecute
- ProcessInstanceDelete
- ProcessInstanceRestart
- ProcessInstanceResume
- ProcessInstanceStart
- ProcessInstanceSuspend
- ProcessInstanceTerminate

This is just a subset of the MQ Workflow API set, although it now has the complete process instance API. The XML interface is not intended for doing things, such as querying worklists or monitoring. It is a subset of functions for dealing only with process instances and process templates. The interface allows an external system or workflow engine to request a service from MQ Workflow. One MQ Workflow system can even request another MQ Workflow system to run a process if they are completely separate from each other.

5.4.4 Security: who can make the requests

The MQ Workflow Programming guide states that:

> The value of the UserIdentifier field in the MQMD of the incoming message is used as the MQ Workflow user on whose behalf the request is to be performed. Authorization checks for that user are performed as usual.

When you build an MQSeries message, you fill in the MQ Message Descriptor (MQMD) with information, such as what the destination queue is, persistent or non-persistent, or the user ID. You also need to set the flag MQC.MQOO_SET_IDENTITY_CONTEXT on the openQueue options.

5.4.5 Setting up UPES activities

When you set up an activity in a process model for a UPES, it is a best practice to use the following recommendations:

- If the UPES goes in error, a work item is created, and is assigned to a random, qualified ID. If you leave **All Users** selected, any user can get the work item. Therefore, use a dedicated ID as the user for this step (even though it will normally not involve any users).
- Make sure that **Program can run unattended** is selected for the program used by the activity.
- Make sure that start and exit are both set to **Automatic** for the UPES activity.
- To optimize the execution server performance, that is, to bypass the staff resolution, it is recommended that you specify the process administrator as the work item owner and deselect on a process level **Assign Substitute if user is absent** (on the Activity tab of the process properties). This also optimizes the staff resolution for the non-error case.
- Deselect **Program activities can be checked out** (on the Control tab) to prepare the unattended automatic activity for an activity repair with the Web client if there are UPES errors or problems. This causes the redefinition of the Checkout button to a Start button for this activity.

6 Users and APIs

MQ Workflow does not provide APIs for defining users. The only way to define a user or to change their attributes is through Flow Definition Language.

Therefore, if you want to add a user to the system, you cannot call an API: you must build an FDL file, and use `fmcibie` to import the file into the workflow system.

There are some examples available that will help you to get users into the system. Based on these examples, you can build a program to read from your data repository where you store user information, and to create FDL for import.

- *SupportPac WO02 People Administration* has an example workflow application to create and manage users.
- *SupportPac WA08 SYSCOM Enterprise Tools and Services* documents the SYSCOM tool for user management.
- MQ Workflow 3.4 provides an LDAP Bridge. Many organizations use LDAP to store information on users. The bridge is a program, which can query data on users from the LDAP system, map those attributes to the user attributes in MQ Workflow, and build an FDL file based on the results. The FDL file can be imported, bringing in the latest information on users each night, eliminating the need to do it manually.

7 Testing and roll out

This chapter covers some of the important issues to consider when you test and roll out a workflow system, and ways to make testing easier for the developers.

7.1 Testing considerations

When you test a workflow, you can start the process through either the fat client or the thin client. However, there is an important difference between the two methods:

- If you start the process through the thin client and you map a data source in your first step, any fields you do not fill in with values will come through as Null. Thus, if you use default values in your mapping, the default replaces the null.
- If you start the process through the thick client, any empty fields will receive 0 values for numbers, and blanks for strings. Thus, if your data mapping uses default values, the fields are already 0 or blank, so the defaults will **not** be mapped in.

7.2 Web client testing environment

When you set up a test environment for the Web client, there are two options:

- Install a full instance of WebSphere Application Server, so that developers can use the Rapid Deployment Wizard to deploy the JSPs created in WebSphere Studio Application Developer. The Rapid Deployment Wizard is a tool for Studio Application Developer that takes your FDL and creates the JSPs for an activity implementation automatically.

- Studio Application Developer comes with a fully functional 'single user' version of WebSphere Application Server. You can set up Studio Application Developer to run a small footprint version WebSphere Application Server, on the same box. Then you can test your JSPs on the same computer on which you developed them. When you are satisfied that they are ready, you can then deploy them to a full WebSphere Application Server instance.

  To do this, you need to set up WebSphere Application Server inside of Studio Application Developer to use the Web client Servlet. The instructions for this are in:

  ```
  C:\Program Files\IBM WebSphere MQ Workflow\DOC\WebClient\upsupptd.html
  ```

  The document uses the old name for Studio Application Developer. Look for the line: *IBM VisualAge for Java V3.5 WebSphere Test Environment.*

7.3 Test with Web client

A feature of the Web client is that if you create an activity but don't create a JSP for it, the Web client automatically generates a screen where you can see all of the input container values, and set the output container. Because of this, you can exploit the Web client for testing proposes.

The following is a method for using the Web client to debug workflows:

1. Test the process logic and data mapping with the Web client.
When you create your workflow, keep all steps, even ones that will be automatic steps with a UPES, as manual steps. You can step through your entire process, test your transition conditions, and ensure that your logic and data mapping are correct.

2. Create and test JSPs for the manual activities.
When you are sure the process works properly, you can begin to create the JSPs for the manual steps, and test them. Because you know that the process model works correctly, any missing data or incorrect results must come from a JSP.

3. Test each UPES in a separate flow.
When all of the manual steps are correct, you can then work on the automated steps. You might want to create a small flow where you can set values, send the data to your UPES, and examine the results.

4. Change the main flow to use the UPES instead of a manual activity.
When you are satisfied with the UPES, you can go into your main process, change the step that uses that UPES to be automatic, and point to the right queue.

7.4 Testing environments

The optimal environment for most customers will be:

- Windows server for development, simple tests. This is the least expensive system, but the lowest powered one.

- Small server on the same platform as the production server for system test, string test, etc. This allows tests in a "clone" of the actual production environment, but without for a server as large as the production system. If you are considering using a partition on a machine, such as a Regatta class AIX server, the standard licensing agreement states that you still have to pay for the entire server. However, you can use a special bid process to purchase a license for just the partition you are going to use. Your IBM sales specialist will be able to help you with this option.

- Large server, UNIX or z/OS for production. The larger server allows for future growth. It is recommended that the production server run at no more than 60% of capacity. This allows for peak load times, and room for future growth. Most customers deploy to UNIX or z/OS because of the increased bandwidth, manageability, scalability, security, I/O capacity, configuration options, and other various related reasons.

8 Administration and Support

The administration and support of a workflow system is critical. After the workflow system has been installed, and the applications have been put into production, the systems administrators still need to watch the system, and keep its various components in tune. This chapter covers some of the considerations for administering a workflow system. Also, see the manual *IBM MQ Workflow Administration Guide,* SH12-6289.

8.1 DB2 administration and tuning

The database used by MQ Workflow requires only a few small administrative tasks from time to time.

8.1.1 Pruning the Audit Trail table

As discussed elsewhere in this document, you can use various techniques such as filters to limit the amount of data logged to this table. Nevertheless, over time, it continues to grow, and can become overly large. You can use manual SQL, or the Audit Trail Cleanup Utility: `fmcsclad`. See *MQ Workflow Administration Guide* for more information.

You can also use tools, such as DB2 propagator, to move data to another DB2 database for long-term storage. What you want to keep long-term and how you want to keep it will determine what procedures you put into place.

Implement a backup strategy, which includes a time-synchronized backup of the MQ Workflow DB2 database and MQSeries, since part of the process data of in-flight processes might be kept in queues. For disaster recovery, adhere to the data resilience rules described in the DB2 documentation.

8.1.2 Tuning the database

The DB2 database has default settings for optimization. However, the usage pattern for your system will most likely not match this exactly. In addition, MQ Workflow uses static SQL statements, but since there is no runstats data initially when you install, the initial configuration is non-optimized. Therefore, you need to use the DB2 runstats/rebind commands to optimize the database from time to time. After you've started using your system and a usage pattern is established, running these commands can cause as much as a 7 to 10 times throughput boost! If you are satisfied with the current performance of the system, it is not necessary to tune the database. However, it might be a good idea to do it from time to time to ensure optimum performance.

Run this procedure regularly or at least whenever the MQ Workflow usage pattern (and with it the DB2 access pattern) changes. For example, if you add new users, or a new process. However, if there are no significant changes, there is little effect. Because this procedure runs through the entire table, the tables are locked during this process. However, DB2 8.0 will do sampling.

You can create a script to perform this function. Consider the following DB2 commands:

```
db2 select tabname, stats_time from syscat.tables where tabschema=″
FMC″

db2 select pkgname, LAST_ BIND_ TIME from syscat.packages where
pkgschema=″ FMC″

db2 reorgchk update statistics on table all

db2 rebind <pkgname> (for all packages)
```

You can do all of these steps in a script, or you can also rebind all packages from `fmczutil`, under the database menu. Select the following:

1. fmczutil
2. s (select)
3. <enter> (using default FMC)
4. r (runtime database)
5. b (bind packages)

8.2 Occasional administration tasks

Review FMCSYS.LOG, DB2DIAG.LOG, and FDC files to look for messages that might indicate a problem is beginning to occur, even if no one report has reported a problem. Periodically scan for large files on the system, especially trace files. Sometimes people forget and leave Workflow or MQSeries trace turned on. Check for core files as well. If you have too many large files, you might run out of disk space. If you've just installed a fix pack, erase the temporary files that were uncompressed. These can accumulate over time, and take up significant amounts of disk space.

Periodically erase the FDC files in `/var/mqm/errors` if you haven't seen any recent problems (the same is true for FDCs on Windows platforms). They accumulate in this directory and make new problem detection harder. If you don't want to immediately erase them, move them to a temporary location before you erase them.

On Windows NT or Windows 2000, there is an additional task that is important. Because the MQ Workflow servers are Windows EXE files, they call DLL files (dynamic link libraries), including system DLLs in the `c:\WINNT\system32` directory. However, these DLLs are also used by other programs. In fact, when a program is installed in Windows, it often overwrites the system DLLs it uses with its own versions. However, these versions might be older than what MQ Workflow needs.

Windows does not manage the versions of installed system DLLs. Sometimes, when you install an unrelated software package, it breaks MQ Workflow (or other software). Because of this, it is a best practice to use appropriate software to do system backups, so that if there is a problem, the working levels of software can be restored.

Take a "snapshot" of the information in the Runtime database that you import through FDL. You can use `fmcibie` to both import an FDL file into the Runtime database and export from the database into a file. For example:

```
fmcibie –u userid –p password –e ExportAll.FDL
```

This creates a file called ExportAll.FDL, that includes all users, process models, data structures, and UPES definitions. This can be used to sync up Buildtime with Runtime, or to back up the current process templates in the database.

8.3 Performance problems

If a system has been working normally, but suddenly performance problems are reported, check the following:

- Check if MQ Workflow, MQSeries, or system tracing has been left on.
- Check if a runstats/rebind has been done recently. (For new installations when basic benchmarks are done, you must do a runstats/rebind after the database is loaded with a typical amount of data, even if that data amount is very small).
- db2diag.log for any DB2-related problems.
- The file systems sizes.
- Check for errors in the AMQERR01.LOG(s) or if FDCs are being produced.

8.4 Considerations for backup-restore of databases

When you put a strategy in place for backing up and restoring the Workflow database, take into consideration business data that needs to be backed up and restored in synch with the Workflow database.

For example, a Workflow activity implementation does a work item check out. It then manipulates business data outside the control of Workflow, such as updating customer data on another database or sending an e-mail to a client. Make sure you have a procedure in place for what happens to this data when the Workflow database is backed up and restored. At the very minimum, the backup and restore of the associated databases needs to happen at a synchronized point in time.

8.5 MQSeries considerations after a backup/restore

When you restore MQSeries, make sure you refresh the cluster to get it to the most current state. The REFRESH CLUSTER command discards local cluster information, including any auto-defined channels that are not in doubt, and forces the repository to be rebuilt. To run the command, enter `runmqsc` from a command window to enter the MQSeries command environment.

8.5.1 REFRESH CLUSTER command

Use this command only if you want your queue manager to make a fresh start in a cluster. For example, you might use it if you think your repository is not up-to-date, perhaps because you have accidentally restored an out-of-date backup. The format of the command is:

```
REFRESH CLUSTER(clustername)
```

The queue manager from which you issue this command loses all the information about the cluster from its repository. It also loses any auto-defined channels that are in doubt and which are not attached to a repository queue manager. The queue manager has to make a cold-start in that cluster. It must reissue all information about itself and must renew its requests for updates to other information that it is interested in. (It does this automatically.)

Because you are unlikely to need to use this command, except in exceptional circumstances, you may choose to avoid the danger of issuing it accidentally. On MQSeries for z/OS, you can use a security profile to protect the command and prevent it from being issued.

8.6 Instance names

When you start a process, you can specify the name of the instance. If you do not specify a name, the system will assign one for you. This name will be the process template name appended with a long random string. You can choose a name that is more descriptive, so that if you sort on the instance name field, you find useful information. It is therefore a best practice to assign your own instance name.

There are a few considerations to keep in mind. It is a best practice to always choose a unique name for each process you create. If you use the same name as a previous instance of a process, you run the risk of database deadlocks if some resources from the previous instance are not yet cleaned up (perhaps your cleanup server doesn’t run until later). Alternatively, if you are monitoring or creating reports from audit trail data, using the same name will cause confusion.

A good example for a process instance name is:

```
ProcessTemplateName + "_" + TimeDateString + "_" + RandomFiveDigitNumber
```

This example has some descriptive information (date and time started) and a random number so that you can avoid problems with database deadlocks and processes being started at the exact same time.

9 System administration

This chapter discusses some important system administration topics, and highlights best practices.

9.1 Log files

In MQ Workflow, there are a number of log files:

- The error log holds information on system errors, such as SQL exceptions.
- The system log records events such as servers starting and stopping.
- If the Web client is used, a servlet log can be enabled.

You can use the administration utility, `fmcautil`, to access, view, and purge the error and system logs. One option to consider is the "Message retain period" for these logs. If this period is set too short, then messages might be automatically purged before you can examine them. However, if it is too long and messages are not purged, the log grows so large that it becomes slower and harder to use.

The servlet.log file holds a large amount of detailed data, which can help in debugging. However, it is a best practice not to enable this log during normal use, because the file grows quickly. For example, for a logon, you will get the following lines:

```
2002-02-27 21:57:17.49 Worker#24: logon called.
2002-02-27 21:57:17.58 Worker#24:   userID: ADMIN
2002-02-27 21:57:17.59 Worker#24:   mode:   null
2002-02-27 21:57:17.60 Worker#24:   SessionTable: key
com.ibm.workflow.servlet.client.SessionContext not found in HttpSession
JGMQIRQAAAAAA5YAAAAQ0YQ
2002-02-27 21:57:17.60 Worker#24:   Locating system '' in group ''.
2002-02-27 21:57:17.60 Worker#24:   Binding agent 'MQWFAGENT' for FMCW
2002-02-27 21:57:20.89 Worker#24:   Agent bound.
2002-02-27 21:57:21.77 Worker#24:   Logging on for userid ADMIN
2002-02-27 21:57:23.14 Worker#24:   Logon: ADMIN OK
2002-02-27 21:57:23.14 Worker#24:   User agent: Mozilla/4.0 (compatible; MSIE 5.01;
Windows NT 5.0)
2002-02-27 21:57:23.14 Worker#24:   Session timeout: 1800s
2002-02-27 21:57:23.24 Worker#24:   Timezone: server offset=-300[min], client offset=-
300[min]
2002-02-27 21:57:23.24 Worker#24:   SessionTable: Storing
com.ibm.workflow.servlet.client.SessionContextImpl@86c71cc7 in HttpSession
JGMQIRQAAAAAA5YAAAAQ0YQ
2002-02-27 21:57:23.24 Worker#24: valueBound called.
2002-02-27 21:57:23.24 Worker#24:   SessionTable: adding UwAAAAEAIgAMAAAAAAAAABT =
JGMQIRQAAAAAA5YAAAAQ0YQ
2002-02-27 21:57:23.24 Worker#24: valueBound done.
2002-02-27 21:57:23.25 Worker#24:   Accept-Language: en-us,de;q=0.5
2002-02-27 21:57:23.25 Worker#24:   Locale: en_US
2002-02-27 21:57:23.26 Worker#24: logon done.
2002-02-27 21:57:23.27 Worker#24: queryProcessTemplateLists called.
2002-02-27 21:57:24.01 Worker#24: queryProcessTemplateLists done.
2002-02-27 21:57:24.01 Worker#24: queryProcessInstanceLists called.
2002-02-27 21:57:24.32 Worker#24: queryProcessInstanceLists done.
2002-02-27 21:57:24.32 Worker#24: queryWorkLists called.
2002-02-27 21:57:24.77 Worker#24: queryWorkLists done.
2002-02-27 21:57:24.78 Worker#24: queryWorkItems called (useCache=false).
2002-02-27 21:57:25.40 Worker#24: queryWorkItems done.
```

```
2002-02-27 21:57:26.12 Worker#24: sendResponse for 'logon' called.
2002-02-27 21:57:26.14 Worker#24: sendResponse done.
```

In a production system, this file can grow quickly to a very large size. In addition, while the servlet is active, you cannot erase the file, since it is open for I/O. Therefore, only have logging active for debugging purposes.

9.2 Using command-line utilities

When you use command-line utilities, such as `fmczutil` or `fmcautil`, after you learn the menus and their shortcut keys, you can take advantage of a feature for entering multiple commands. For example, if you were in `fmcautil` Select Server Menu (m), Admin Server Commands Menu (a), and you wanted to go to the error log, you could:

1. Type `x` for exit, then press enter.
2. Type `x` for exit again, then press enter.
3. Type `e` for error log, then press enter.

Instead, you can issue all of these commands by typing `xxe`, and then enter. The system processes them one-by-one. This can save time if you are familiar with the menu structure.

Another related feature is the ability to record your entries, and play them back to automate tasks such as system configuration. You can automate common tasks to save time and effort on the administrator's behalf. See Appendix D of *MQ Workflow Administration Guide* for details.

9.3 Passwords and authorizations

One of the first things done when MQ Workflow is installed is to import the default objects into the Runtime database. For example, the user ID "admin" is created, with a default password of "password". To avoid a possible security hole, it is a best practice to change this password to something non-obvious.

Likewise, an ID called "starter" is often created for starting automated processes from the Web client. Because this name is also common, it is a best practice to create and use a different ID that is not known outside your administrative group.

One Runtime function of MQ Workflow that you might want to restrict is the use of the Process Instance Monitor. You might want a particular user to be able to monitor only a certain class of processes, or none at all.

- If a user is the starter of a process, they can always see the instance and monitor it, no matter what the category.

- If a user is authorized for only selected categories, they can only see the process templates in those categories. This implies that the user can only start processes in those categories.
- If a user is authorized for all categories, they can see all process templates and instances, and monitor instances. If you are not authorized for a category, you cannot see the process instances, even if you are assigned the work item. You cannot monitor the instance either. You will get a "not authorized" message in this case.
- If you are authorized for work items for just yourself or a small list of users, you will only see those work items, no matter what category they belong to. If you can see all persons work items, if there is a work item assigned to multiple people, you will see each of the copies (that is, you will see multiple copies of the same work item).

If, for example, you want to restrict the user 'CSR123' to see only their own work items, and to monitor only service request processes, you can set their work item authorization to just their ID, and set them as authorized just for the category ServiceRequest. If a manager transfers a work item to them from another category, they can work on that item, but not monitor it.

This implies that it is a best practice to use categories for processes, to be able to limit what users can see. It is also a best practice in most cases to allow ordinary users to see only their own work items (or perhaps theirs, plus ones for the people they back up).

9.4 Changing passwords

MQ Workflow has automatic user IDs, such as FMC that is used to start processes. You create these IDs at the start of the installation. Some customers let these IDs keep the same password forever. However, some customers have standards in place that require passwords, even for automatic IDs, to change from time to time.

When you change the password for an ID, you must update the MQ Workflow profiles to reflect that change. This can be done through `fmczutil`. See the chapter *"Changing your configuration on (operating system)"* in *MQ Workflow Administration Guide*.

It is a best practice to set up a procedure so that the administrator who updates the passwords for the IDs also updates the MQ Workflow profiles to reflect these changes. Doing them both at the same time avoids problems.

Likewise, anything else that changes on the system, such as a new hostname or a different port for the queue manager, can affect MQ Workflow, and this change will also need to be updated in the configuration. Make sure to work closely with your systems administrators, so that when they make changes, you can keep the workflow configurations in accord.

9.5 Getting support

When you call the 800 number for support, be prepared to send at least the following documentation:

- fmcsys.log
- fmcerr.log
- AMQERR0x.logs (/var/mqm/erros, in /var/mqm/qmgrs/@system/errors, and in /var/mqm/qmgrs/<qmgrname>/errors and equivalent places on Windows platforms)
- MQSeries FDC files (in /var/mqm/errors)
- db2diag.log file (a must! do a "find" command from root to find the most current file and review to make sure it's the right DB2 database (`find . -name db2diag.log`)
- error message ID (if any) and exact text of message

9.6 Where can you find the Messages and Codes manual?

You can use the online site to download the Messages and Codes manual for MQ Workflow:

http://www-3.ibm.com/software/ts/mqseries/library/manualsa/index.html

The Messages and Codes Manual for **all** platforms is listed under the books available for z/OS. The messages and codes for the zSeries Server platform (found in Chapter 7) are special to that platform. However, all of the other messages and codes are the same for all platforms.

The manuals are also available from the IBM Publications Center, at:

http://www.elink.ibmlink.ibm.com/public/applications/publications/cgibin/pbi.cgi

9.7 Optimize clean-up time

- Under Execution Server settings, use "NO IMMEDIATE CLEANUP" (which is the default). There is a check box under **Domain Properties > Server > Complete execution server settings > Server specifics** that should remain unchecked.

  NO IMMEDIATE CLEANUP causes the execution server to end a process faster, because items to be deleted are "logically" deleted, and the cleanup server deletes them later.

- For each process model under **Process Properties > Control**, uncheck "Inherited" for "Keep finished work items" and "Keep finished processes" and then set work items to "Forever" and processes to "For 8 hour(s)" (or some length of time that will defer this to off-shift).

This option (keep finished work items forever) should have a major impact on cleanup times. When the cleanup server starts, it first looks for work items to be deleted, then for process instances. While it processes finished process instances, it also looks whether associated work items have to be deleted. And if those work items would have been deleted already (because they are *not* "KEEP FOREVER" and have already been deleted with the first pass of the cleanup server run) then it is doing some of the work twice.

9.8 Deleting users from Runtime

To delete a person from runtime, you can create an FDL file with the following format:

```
/*
 FDL definitions generated by MQ Workflow at 2001-xx-xx, 03:03:05 PM.
*
*/
CODEPAGE 1252
FM_RELEASE V3R2 1
/*
* PERSONs
*/
DELETE PERSON 'USER'
```

9.9 Migrating hardware systems

During the lifetime of the workflow system, you might need to move from one hardware server box to another one. You should try to plan far enough into the future when you initially implement your system, so you can avoid this scenario if possible. However, if you must do it, there are a few considerations.

You will most likely have processes which are "in flight", still running. If the new database is empty, you can back up from the old database and restore to the new one. The new database must use the same topology information as the old one, because some constructs use system group, system information, or the queue manager. When you back up the old database, make sure that no activities are in a running state (that is, a UPES waiting for a response message, or an activity checked out by a user).

If you loaded the new system and started to run processes, there are additional problems in attempting to move over to the new database. You cannot restore to the new database, because it overlays the existing information. You cannot merge the new database with the backup information, because there is the danger of duplicate instance IDs of workflow objects.

Therefore, do not use the new system until you have backed up and restored from the old system.

If you are migrating to a new software release, be sure to read the detailed instructions in the documentation. Some releases might update or change the formats of the database tables, for example, or add new queues.

You should always test a system to perform quality assurance prior to going live. That way, you can clear up any issues before attempting to use the system in production.

10 Audit trail

The audit trail features of MQ Workflow have several options, each of which has implications for system performance. The business needs will decide which auditing approach to use for an application. This chapter discusses some of the major issues.

10.1 Audit trail level

When you use the audit trail to a database, you have the options of Off (no logging), Condensed, or Full. You can also use filters to limit what is logged. The condensed level only logs major events. Often this is not enough information. For example, the GDC data is not logged with the Condensed audit trail level. Thus, the Full level is more useful. However, this level shows so much information that the database table can grow rapidly, especially for a large production system.

The solution is to use a filter on an audit trail. To do this, you will need to edit the FDL by hand because the filter is not available in Buildtime. After you export your FDL, in the section for audit, instead of just `FULL AUDIT_TO_DB`, you can add:

```
FILTER AUDIT_TO_DB list of events
```

This allows you to focus on just the events in which you are interested. If you were interested in process starts and suspends, for example, but not for “activity ended normally”, you can keep these events out of your database. This keeps the size of the database down, and prevents it from growing as fast. If you are using a monitor such as WBI Monitor, be sure to check what events might be required, so that you don’t filter out any data the monitor is expecting.

10.2 Where to log the data

If you put the audit trail table in the workflow database, you are impacting system performance, particularly if you have a 2-tier system (that is, the database on the same node as the MQ Workflow servers). In addition, any queries you run on the audit trail table will also put load on the system. You must use the database for your audit trail if you are using the WBI Monitor (Holosofx).

One option to improve performance is to not log the data to a database table, but to log it to MQSeries. The data is placed in XML format. Here is an example:

```
message <<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<!-- This document is generated by a MQ Workflow Version 3.3.2 server -
->
<WfMessage>
<WfMessageHeader>
<ResponseRequired>No</ResponseRequired>
</WfMessageHeader>
<AuditTrailRecord>
```

```
<Timestamp>2002-02-21 11:38:42</Timestamp>
<AuditEvent>21018</AuditEvent>
<ProcInstName>GlobalcontainerTest</ProcInstName>
<ProcInstID>UAAAAAEAHMAFAAAAAAAAAAAAAAABAB3AAAAAAAAAAAAAUA==</ProcInstI
D>
<ProcInstTopLevelName>GlobalcontainerTest</ProcInstTopLevelName>
<ProcInstTopLevelID>UAAAAAEAHMAFAAAAAAAAAAAAAAABAB3AAAAAAAAAAAAAUA==</P
rocInstTopLevelID>
<ProcTemplName>GlobalcontainerTest</ProcTemplName>
<ProcTemplValidFromDate>2002-02-21 11:37:48</ProcTemplValidFromDate>
<UserID>ADMIN</UserID>
<AssociatedObject>UAAAAAEAHMAFAAAAAAAAAAAAAAABAB3AAAAAAAAAAAAAUA==</Ass
ociatedObject>
<ObjectDescription>Test the global container</ObjectDescription>
</AuditTrailRecord>
</WfMessage>>
```

In this example, the AuditEvent is 21018, which means that a process instance was created and started. The associated data, such as the date and time, is automatically generated if you set up the FDL to point to a particular queue manager and queue. If this were the event for updating GDC data, you would also have a `<Varname>Data</Varname>` line for each data element in the GDC. If the GDC is logged to a database, it is stored as a Binary Large Object (BLOB), which has to be parsed on retrieval from the database.

A benefit of this approach is that you can have the audit message go to an off-node queue for handling. So all of the logging work can take place somewhere else on a less busy computer. You can simply take the incoming messages, and put them into a database table (which might be set up with the same schema as the FMC.AUDIT_TRAIL database). You would have to write the logging code yourself, but this is more flexible. The overhead of placing an XML message on an MQSeries queue is much lower than an insert in a database.

Another way to log the audit information is an extension to the XML logging to a queue; you can set up MQ Workflow so that a JMS program can subscribe to audit messages, based on topic. This is very powerful, because it allows a program using JMS to tie in to the workflow system, so that when an event you are interested in (such as a process being suspended) occurs, your program is notified, and is able to take the necessary actions.

10.3 Audit trail and performance

When you select the method to use for logging the audit trail data, there is the implication of performance. While it may seem intuitive that there is more overhead from auditing to a database, it is actually more costly to build the XML message, and place it in a queue. Performance tests have indicated that audit to a database costs 4-5% for full audit, while audit to a queue costs 10-30% for full audit.

In other words, there is less of a load on a system to put the audit trail data into a database. However, there is also the issue of reading from the database for reports, or monitoring the data, which should be taken into account when you decide where and how to log the data. In other words, if frequent complex queries are to be run against the database, that might more than make up for the savings of using it over auditing to MQSeries.

11 Common project problems

As with any software product, there is the possibility of running into problems during the implementation of an MQ Workflow system. This chapter covers several scenarios that have been seen in implementations, and outlines ways to avoid them in your own system.

Lack of planning: This is the number one cause of problems during implementation. If the installation was not properly planned, for example, then it can take weeks instead of the usual 1-2 days, which throws off the whole project timeline. See chapter 1 on planning for recommendations.

Hardware platform is too small: End user response time is a critical factor to the success of a system. If a Windows NT box is used for development and test, but there are 2000 Runtime users, the same box will not perform well enough for a production system. Make sure that you do a capacity assessment with the *SupportPac WP01* to ensure that you have a large enough system for production.

Did not plan for performance: Related to the previous items, is the case where a process is implemented using many expensive constructs, such as subprocesses, causing performance to be slower than desired. Again, use *SupportPac WP01* to estimate performance before implementing.

Did not test performance until late in project: Sometimes performance is not measured until the final stages of testing have been completed. This means that if the results are not acceptable, then the system has to be changed, and testing repeated. Therefore, it is best to test performance at several points during the implementation cycle, to make sure that it is on track from the start.

Did not get education: In several implementations with problems, the customer did not go to the training classes, and did not bring in an experienced mentor to help with the first implementation. This can lead to problems, because while the customer might have great programmers, a production workflow system is different from a programming language. Learning how the system works and how best to use it is key to a successful implementation.

Did not design for the system: Sometimes, a detailed design is created before a workflow system is selected. This can cause problems, in that it can lead to a less than optimal use of the system. It is better to look at the features and functions of the workflow system, and then create a system design. Otherwise, key features might not be used, or less than optimal performance might result.

Did not use latest features: Sometimes, a customer learns the features of the current version of MQ Workflow. Then as subsequent versions are released, the new features and functions are sometimes not taken advantage of, even though they would help accomplish what the customer needs to do. It is critical to stay current with the features and functions of new releases of the product.

Did not redesign/model business process: When setting out on a business process automation project, you have two choices: automate the process exactly the way it is today, or reengineer and optimize it first. Sometimes the "as-is" business process can be quite messy. Attempting to automate it is not always optimal (automating a mess results in an automated mess). It is not just process automation that should be the end goal, but process improvement as well. This can only be achieved by redesigning the business process first.

12 Future considerations

When you implement a system with MQ Workflow, it is important to keep in mind the long-term plans for the system. A system that goes into production today will need to have a long life span. If a solution is properly architected, it will still fit well within the enterprise infrastructure when the time comes to move to the next technology base, or to interoperate with future technologies. With this in mind, here are some recommendations from the lab on best practices for applications being built today, with an eye on the future.

12.1 Modeling

Exploit the WBI Workbench modeling solution for additional business views of the process, and the ability to perform simulations on the process, even before it is deployed to Runtime.

Also, exploit the test capabilities of the Web client. When you model a process, export it to FDL, and then import it to the Runtime server. You can use the Web client to step through the process, even if you do not have activity implementations written for any of the steps yet. If there is no JSP for a step, the Web client displays all of the input and output data container fields. If the fields match, it shows just the output data container field, prepopulated with the value of the input data container field.

12.2 APIs

Exploit the Java API, rather than any of the other supported APIs. It is fully functional, portable from operating system to operating system, and can be used in servlets, JSPs, EJBs, and so on. You are therefore free to change architectures later, without having to throw away the work you have done on the code. If you were to use the ActiveX API on a Windows machine and you change to a thin client later, most of your coding would be unusable.

12.3 Clients and user activity implementations

Exploit the thin client technology instead of fat clients. This reduces administration costs because you do not have to synchronize all of the client workstations (not only for the client code, but for any programs which might be on the local drive that MQ Workflow needs to launch). It also reduces the cost to implement customized clients, because of the flexibility of the standard Web client.

12.4 Exploit Web client

The Web client, the ultra-thin client for MQ Workflow, comes with almost all the function typically needed, leading to a fast start. It supports server clustering and is highly customizable, something which you would have to take a lot of care to design into a completely customized application.

12.5 Use JSPs for activity implementations

Because the thin client is the preferred architecture, you use JSPs for the activity implementations. Use the Rapid Deployment Wizard for a fast start.

12.6 Back-end integration

Exploit UPES technology for your back-end integration scenarios, instead of custom programs with APIs. A UPES is transaction safe, very flexible, and not intrusive, unlike using the PEA and the container API. It can easily make use of WebSphere MQ based applications and adapters, WebSphere MQI, WebSphere ICS (which also offers compensation capability for non-transactional adapters), and call EJBs or Web Services.

Model data interchange via standards (DTD, XML Schema Definitions (XSD), or WSDL to make things easier in the future.

12.7 Exploit PES technology with PES data mapping

If you use the zSeries Server platform, use the PES with data mapping. It is also transaction safe, and non-intrusive, unlike the PES with the container API.

12.8 BPM monitoring

Get customer data into audit data. The description field can be used, and is easily searchable. Use the global data container as well.

There is a trade off between functionality and performance. You can use filtering to improve performance by reducing the amount of audited events. Then you have less data logged to use in reports. Consider using audit messages if you need pub/sub capabilities.

Exploit the WBI Monitor solution, because it is predefined (faster and less expensive than developing it all from scratch), yet flexible. It gives business analysts customizable data for reports with a ready-to-use interface. It is integrated with the simulation and modeling from the WBI Workbench, which allows for a continuous process improvement cycle.

13 References

13.1 IBM WebSphere MQ Workflow documentation

Installation Guide, SH12-6288
Programming Guide, SH12-6291
Administration Guide, SH12-6289
Concepts and Architecture, GH12-6285
Getting Started with Buildtime, SH12-6286
Getting Started with Runtime, SH12-6287

MQ Workflow for z/OS Messages and Codes, SC33-7032

Redbooks (see www.redbooks.ibm.com for download)

Image and Workflow Library: Getting Started with MQ Workflow for OS/390, SG24-5598
MQ Workflow for Windows NT for Beginners, SG24-5848
Continuous Business Process Management with Holosofx BPM Suite and IBM MQ Workflow, SG24-6590

13.2 Production Workflow: Concepts and Techniques

Frank Leymann, Germany
Dieter Roller, Germany
Published July, 1999 by Prentice Hall PTR (ECS Professional)
Copyright 2000, 500 pp., Paperback
ISBN 0-13-021753-0

13.3 Web Sites

http://www-3.ibm.com/software/ts/mqseries/workflow/

SupportPacs (http://www-3.ibm.com/software/ts/mqseries/txppacs/txpm2.html#cat2)

WA02 - MQ Workflow - MQSeries Integrator interpretability sample scenarioWA03 - MQ Workflow - API examples
WA04 - MQ Workflow - Java Generic API test and prototyping tool
WA05 - MQ Workflow - XML Development Toolkit and Sample Scenario
WA06 - MQ Workflow - Event Server sample using MQSeries Integrator
WA07 - Web Services Process Management Toolkit
WA09 - MQ Workflow - Generic C API test and prototyping tool
WA82 - MQ Workflow - Web credit example
WA83 - MQ Workflow - Rapid deployment wizard
WA84 - MQ Workflow - Web client extensions
WD01 - Business Process Modeling with MQ Workflow

WD02 - MQ Workflow - Considerations for production roll out
WO02 - MQ Workflow - People administration
WP01 - MQ Workflow - Performance estimates for solution and capacity assessments
WP11 - MQ Workflow for OS/390 - A customer specific performance study

14 Trademarks